SIEMENS

# SINUMERIK

# SINUMERIK 828D
# PLC Programming Tool

**Programming Manual**



Valid for:

PLC Programming Tool Version 3.3


**03/2025**
A5E48414369B AD

## Legal information

### Warning notice system

This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

| ⚠ **DANGER** |
| --- |
| indicates that death or severe personal injury **will** result if proper precautions are not taken. |

| ⚠ **WARNING** |
| --- |
| indicates that death or severe personal injury **may** result if proper precautions are not taken. |

| ⚠ **CAUTION** |
| --- |
| indicates that minor personal injury can result if proper precautions are not taken. |

| **NOTICE** |
| --- |
| indicates that property damage can result if proper precautions are not taken. |

If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

### Qualified Personnel

The product/system described in this documentation may be operated only by **personnel qualified** for the specific task in accordance with the relevant documentation, in particular its warning notices and safety instructions. Qualified personnel are those who, based on their training and experience, are capable of identifying risks and avoiding potential hazards when working with these products/systems.

### Proper use of Siemens products

Note the following:

| ⚠ **WARNING** |
| --- |
| Siemens products may only be used for the applications described in the catalog and in the relevant technical documentation. If products and components from other manufacturers are used, these must be recommended or approved by Siemens. Proper transport, storage, installation, assembly, commissioning, operation and maintenance are required to ensure that the products operate safely and without any problems. The permissible ambient conditions must be complied with. The information in the relevant documentation must be observed. |

### Trademarks

All names identified by ® are registered trademarks of Siemens Aktiengesellschaft. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

### Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.

Siemens Aktiengesellschaft
Digital Industries
Postfach 48 48
90026 NÜRNBERG
GERMANY

A5E48414369B AD
Ⓟ 01/2025 Subject to change

# Table of contents

# 1 Introduction

## 1.1 About SINUMERIK

From simple, standardized CNC machines to premium modular machine designs – the SINUMERIK CNCs offer the right solution for all machine concepts. Whether for individual parts or mass production, simple or complex workpieces – SINUMERIK is the highly dynamic automation solution, integrated for all areas of production. From prototype construction and tool design to mold making, all the way to large-scale series production.

Visit our website for more information SINUMERIK (https://www.siemens.com/sinumerik).

## 1.2 About this documentation

### Target group

Programmers and project engineers

### Benefits and benefit phase

This Programming Manual enables the target group to develop, write, and test programs and to correct errors.

Planning and configuration phase, implementation phase, setup and commissioning phase

### Standard scope

This documentation only describes the functionality of the standard version. This may differ from the scope of the functionality of the system that is actually supplied. Please refer to the ordering documentation only for the functionality of the supplied drive system.

It may be possible to execute other functions in the system which are not described in this documentation. This does not, however, represent an obligation to supply such functions with a new control or when servicing.

For reasons of clarity, this documentation cannot include all of the detailed information on all product types. Further, this documentation cannot take into consideration every conceivable type of installation, operation and service/maintenance.

The machine manufacturer must document any additions or modifications they make to the product themselves.

### Websites of third-party companies

This document may contain hyperlinks to third-party websites. Siemens is not responsible for and shall not be liable for these websites and their content. Siemens has no control over the information which appears on these websites and is not responsible for the content and information provided there. The user bears the risk for their use.

## 1.3 Documentation on the internet

### 1.3.1 Documentation overview SINUMERIK 828D

Comprehensive documentation about the functions provided in SINUMERIK 828D Version 4.8 SP4 and higher is provided in the 828D documentation overview (https://support.industry.siemens.com/cs/ww/en/view/109766724).

You can display documents or download them in PDF and HTML5 format.

The documentation is divided into the following categories:

- User: Operating
- User: Programming
- Manufacturer/Service: Configuring
- Manufacturer/Service: Commissioning
- Manufacturer/Service: Functions
- Manufacturer/Service: Safety Integrated
- SINUMERIK Integrate/MindApp
- Info & Training

### 1.3.2 Documentation overview SINUMERIK operator components

Comprehensive documentation about the SINUMERIK operator components is provided in the Documentation overview SINUMERIK operator components (https://support.industry.siemens.com/cs/document/109783841/technische-dokumentation-zu-sinumerik-bedienkomponenten?dti=0&lc=en-WW).

You can display documents or download them in PDF and HTML5 format.

The documentation is divided into the following categories:

- Operator panels
- Handheld unit/Mini handheld devices
- Machine control panels
- Further operator components

An overview of the most important documents, entries and links on the topic of "SINUMERIK" is provided at SINUMERIK Overview - Topic Page (https://support.industry.siemens.com/cs/document/109766201/sinumerik-an-overview-of-the-most-important-documents-and-links?dti=0&lc=en-WW).

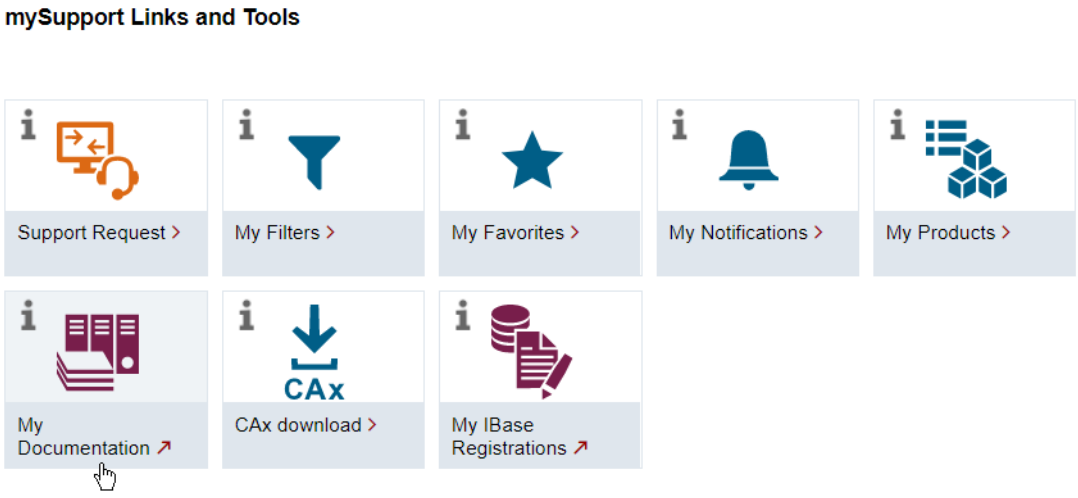
## 1.4 Feedback on the technical documentation

If you have any questions, suggestions, or corrections regarding the technical documentation published in the Siemens Industry Online Support, use the "Provide feedback" link which appears at the end of the entry.

## 1.5 mySupport documentation

With the "mySupport documentation" web-based system you can compile your own individual documentation based on Siemens content, and adapt it for your own machine documentation.

To start the application, click on the "My Documentation" tile on the "mySupport links and tools" (https://support.industry.siemens.com/cs/ww/en/my) portal page:




The configured manual can be exported in RTF, PDF or XML format.

**Note**

Siemens content that supports the mySupport documentation application can be identified by the presence of the "Configure" link.

## 1.6 Service and Support

### Product support

You can find more information about products on the internet:

Product support (https://support.industry.siemens.com/cs/ww/en/)

The following is provided at this address:

- Up-to-date product information (product announcements)
- FAQs (frequently asked questions)
- Manuals
- Downloads
- Newsletters with the latest information about your products
- Global forum for information and best practice sharing between users and specialists
- Local contact persons via our Contacts at Siemens database (→ "Contact")
- Information about field services, repairs, spare parts, and much more (→ "Field Service")

## Technical support

Country-specific telephone numbers for technical support are provided on the internet at address (https://support.industry.siemens.com/cs/ww/en/sc/4868) in the "Contact" area.

If you have any technical questions, please use the online form in the "Support Request" area.

## Training

You can find information on SITRAIN at the following address (https://www.siemens.com/sitrain).
SITRAIN offers training courses for automation and drives products, systems and solutions from Siemens.

## Siemens support on the go


With the award-winning "Industry Online Support" app, you can access more than 300,000 documents for Siemens Industry products – any time and from anywhere. The app can support you in areas including:

- Resolving problems when implementing a project
- Troubleshooting when faults develop
- Expanding a system or planning a new system

Furthermore, you have access to the Technical Forum and other articles from our experts:

- FAQs
- Application examples
- Manuals
- Certificates
- Product announcements and much more

The "Industry Online Support" app is available for Apple iOS and Android.

## 1.7 Using OpenSSL

This product can contain the following software:

- Software developed by the OpenSSL project for use in the OpenSSL toolkit
- Cryptographic software created by Eric Young.
- Software developed by Eric Young

You can find more information on the internet:

- OpenSSL (https://www.openssl.org)
- Cryptsoft (https://www.cryptsoft.com)

## 1.8 Compliance with the General Data Protection Regulation

Siemens observes standard data protection principles, in particular the data minimization rules (privacy by design).

For this product, this means:

The product does not process or store any personal data, only technical function data (e.g. time stamps). If the user links this data with other data (e.g. shift plans) or if he/she stores person-related data on the same data medium (e.g. hard disk), thus personalizing this data, he/she must ensure compliance with the applicable data protection stipulations.

# Fundamental safety instructions 2

## 2.1 General safety instructions

**⚠ WARNING**

**Danger to life if the safety instructions and residual risks are not observed**

If the safety instructions and residual risks in the associated hardware documentation are not observed, accidents involving severe injuries or death can occur.

- Observe the safety instructions given in the hardware documentation.
- Consider the residual risks for the risk evaluation.

**⚠ WARNING**

**Malfunctions of the machine as a result of incorrect or changed parameter settings**

As a result of incorrect or changed parameterization, machines can malfunction, which in turn can lead to injuries or death.

- Protect the parameterization against unauthorized access.
- Handle possible malfunctions by taking suitable measures, e.g. emergency stop or emergency off.

## 2.2 Warranty and liability for application examples

Application examples are not binding and do not claim to be complete regarding configuration, equipment, or any eventuality which may arise. Application examples do not represent customer-specific solutions, but merely serve to provide assistance with typical tasks.

As the user you yourself are responsible for ensuring that the products described are operated correctly. Application examples do not relieve you of your responsibility for safe handling when using, installing, operating and maintaining the equipment.

## 2.3 Cybersecurity information

Siemens provides products and solutions with industrial cybersecurity functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial cybersecurity concept. Siemens' products and solutions constitute one element of such a concept.

Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.

For additional information on industrial cybersecurity measures that may be implemented, please visit
https://www.siemens.com/cybersecurity-industry.

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customer's exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Cybersecurity RSS Feed under
https://new.siemens.com/cert.

Further information is provided on the Internet:

Configuration Manual Industrial Cybersecurity (https://support.industry.siemens.com/cs/ww/en/view/109975311)

**⚠ WARNING**

**Unsafe operating states resulting from software manipulation**

Software manipulations, e.g. viruses, Trojans, or worms, can cause unsafe operating states in your system that may lead to death, serious injury, and property damage.

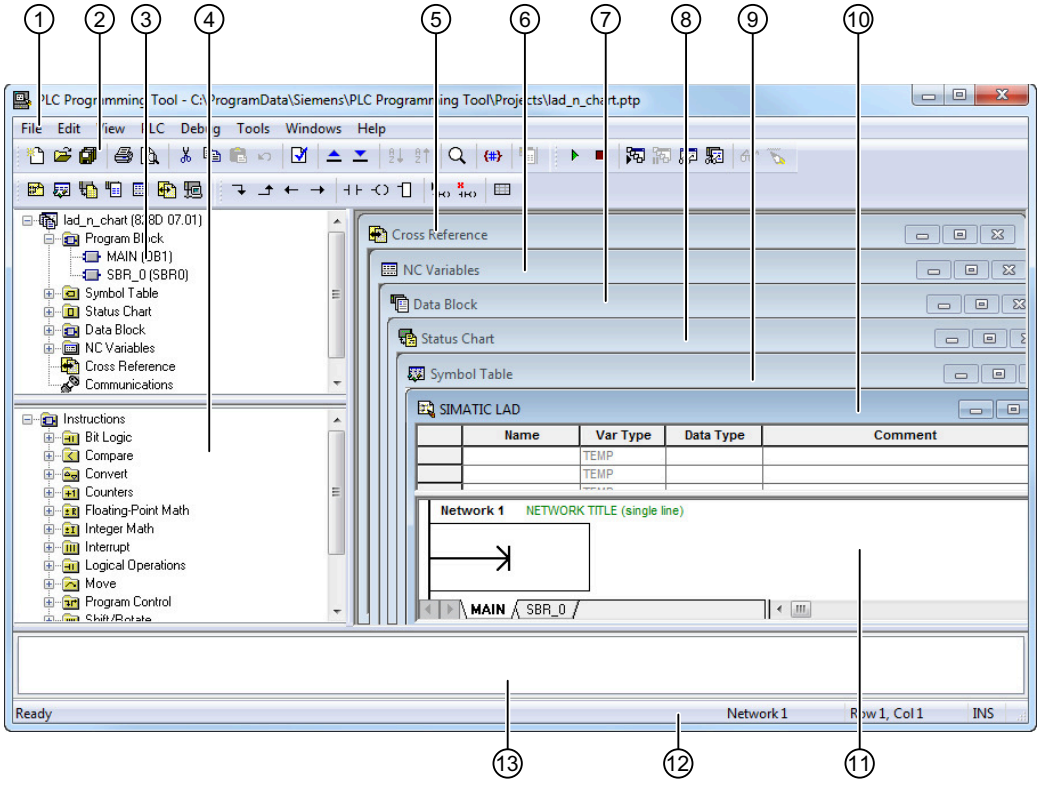
- Keep the software up to date.
- Incorporate the automation and drive components into a state-of-the-art, integrated industrial cybersecurity concept for the installation or machine.
- Make sure that you include all installed products in the integrated industrial cybersecurity concept.
- Protect files stored on exchangeable storage media from malicious software by with suitable protection measures, e.g. virus scanners.
- Carefully check all cybersecurity-related settings once commissioning has been completed.

# Getting started

# 3

## 3.1 Chapter 1 Welcome to Programming Tool

### 3.1.1 Window elements




① Menu bar (Page 378)
② Toolbar (Page 499)
③ Project tree (Page 447)
④ Instruction tree (Page 449)
⑤ Cross-references (Page 439)
⑥ NC variables (Page 434)
⑦ Data block (Page 424)
⑧ Status chart (Page 420)
⑨ Symbol table (Page 413)
⑩ Program block (Page 412)
⑪ Local variable table (Page 229)

⑫ Status bar (Page 456)
⑬ Output window (Page 451)

### Explanation of the window elements

Menu bar (Page 378)

Allows you to perform operations using either a mouse or keystrokes.

Toolbars (Page 499)

Provides easy mouse access to the most commonly used Programming Tool operations. You can customize (Page 487) the content and appearance of each of the toolbars.

Shortcut keys (Page 496)

You can use shortcut keys in the Programming Tool to make a variety of tasks more efficient.

Project tree (Page 447)

The project tree displays all of the elements of the project. You can right-click project elements to insert additional Program Organization Units (POUs). You can right-click an individual Program Organization Unit to open, rename, delete, or edit its properties.

Instruction tree (Page 449)

Provides a tree view of all project elements and all the instructions that are available for your current program editor (LAD or STL). You can right-click project elements to insert additional Program Organization Units (POUs). You can right-click an individual Program Organization Unit to open, rename, delete, or edit its properties. You can right-click a folder or individual instruction in the Instructions portion of the tree to hide the entire tree.

Once you open an instruction folder, you can drag and drop individual instructions or double-click to automatically insert the selected instruction at the cursor location in the Program Editor window.

**Note**

In STL programs, the instruction tree is for reference only: You must enter the instructions in the LAD editor. You cannot transfer them from the instruction tree.

Local variable table (Page 229)

Contains assignments that you have made to local variables (in other words, variables that are used by your subprograms and interrupt programs). Variables created in the Local Variable Table use temporary memory. Address assignment is handled for you by the system. Use of the variable is restricted to the POU where it was created.

Program Editor Window (Page 412)

Contains the Local Variable Table and the program view for the editor (LAD (Page 111) or STL (Page 258)) that you are using for this project. You can drag the split bar to expand the program view and cover up the Local Variable Table if desired. When you create subprograms (Page 223) or interrupt programs (Page 227) in addition to the main program (OB1), tabs appear at the bottom of the Program Editor window. You can click the tabs to move between the subprograms, interrupt programs, and OB1.

Output window (Page 451)

Provides information messages when you compile your program. When the output window lists program errors, you can double-click an error message and the appropriate network is displayed in the Program Editor window.

Status bar (Page 456)

Provides information about the status of the operations that you perform in Programming Tool.

Cross-references (Page 439)

In this window you can view Cross Reference and Element Usage information for your program.

Symbol Table Window (Page 413)

Allows you to assign and edit global symbols (in other words, symbolic values that can be used in any POU, not just the POU where the symbol was created). You can create multiple symbol tables. There is also a tab in the Symbol Table for system-defined symbols that you may want to use in your program.

Status Chart Window (Page 420)

Allows you to track the status of program inputs, outputs, or variables by putting them into the chart. You can create multiple status charts in order to view elements from different portions of your program. Each status chart has its own tab in the Status Chart window.

Data Block Editor Window (Page 74)

Allows you to display and edit the content of your data block.

NC Variable Editor Window (Page 434)

Allows you to display and edit the NC variables.

## See also

Print (File menu) (Page 394)

New (File menu) (Page 378)

Open (File menu) (Page 379)

Save (File menu) (Page 379)

Print Preview (File menu) (Page 393)

Cut (Edit menu) (Page 398)

Copy (Edit menu) (Page 399)

Paste (Edit menu) (Page 400)

Undo (Edit menu) (Page 398)

Compile (Page 459)

Uploading from the CPU (File menu) (Page 382)

Downloading to the CPU (File menu) (Page 382)

Sorting table and chart columns (Page 446)

Zoom (Page 452)

RUN / STOP (Page 457)
Displaying the status in the program editor (Page 60)
Displaying the status in a status chart (Page 68)
Entering instructions in LAD (Page 36)
Insert (Edit menu) (Page 401)
Delete (Edit menu) (Page 404)
Symbol information table (Page 445)
Bookmarks in the STL program (Page 258)
Communication (Page 443)
File menu (Page 378)
Edit menu (Page 398)
View menu (Page 411)
Menu PLC (Page 457)
Menu Debug (Page 471)
Menu Tools (Page 487)
Menu Window (Page 494)
Menu Help (Page 495)

## 3.1.2 Shortcut keys

You can use shortcut keys in the Programming Tool to make a variety of tasks more efficient.

**Note**

If multiple keys are listed, all keys must be pressed together in order for the shortcut to work.

These help topics are explained in the following:

### LAD shortcuts

**Navigation keys**

| Key sequence | Action |
|---|---|
| HOME | Moves cursor to the first column of the same row |
| END | Moves cursor to the last column of the same row |
| PAGE UP | Moves one screen up vertically |
| PAGE DOWN | Moves one screen down vertically |
| LEFT ARROW | Moves cursor one field to the left |
| RIGHT ARROW | Moves cursor one field to the right |
| UP ARROW | Moves cursor one field up |

| Key sequence | Action |
|---|---|
| DOWN ARROW | Moves cursor one field down |
| CTRL + HOME | Moves cursor to the first field of the first network |
| CTRL + END | Moves cursor to the last field of the last network |
| F1 | Help on the current field instruction |
| CTRL + PAGE UP | Displays the next POU<br>Moves left through POU tabs |
| CTRL + PAGE DOWN | Displays the next POU<br>Moves right through POU tabs |

**Editing**

| Key sequence | Action |
|---|---|
| CTRL + X | When series of networks is selected, cuts series. When cursor is on network title, cuts entire network. When cursor is on field, cuts content of field. |
| CTRL + C | When series of networks is selected, copies series. When cursor is on network title, copies entire network. When cursor is on field, copies content of field. |
| CTRL + V | When series of networks is selected, pastes series. When cursor is on network title, pastes network. When cursor is on field, pastes field. Pastes selection before/above current location of cursor. |
| F2 | Edits current field's mnemonic or operand |
| SPACEBAR | Edits current field's mnemonic or operand |
| ENTER | Edits current field's mnemonic or operand |
| CTRL + LEFT ARROW | Drops a horizontal line and moves the cursor one field to the left |
| CTRL + RIGHT ARROW | Drops a horizontal line and moves the cursor one field to the right |
| CTRL + UP ARROW | Drops a vertical line and moves the cursor one field up |
| CTRL + DOWN ARROW | Drops a vertical line and moves the cursor one field down |
| DELETE | Deletes current field, network, or selection of networks |
| BACKSPACE | Moves cursor one field to the left and deletes current field |
| SHIFT + LEFT ARROW | Selects current network and moves cursor over one field to left |
| SHIFT + RIGHT ARROW | Selects current network and moves cursor over one field to right |
| SHIFT + HOME | Selects current network and places cursor at first field of current row |
| SHIFT + END | Selects current network and places cursor at last field of current row |
| SHIFT + DOWN ARROW | Selects from current network downward |
| SHIFT + UP ARROW | Selects from current network upward |
| SHIFT + PAGE UP | Selects a page of adjacent networks from the current network upward |
| SHIFT + PAGE DOWN | Selects a page of adjacent networks from the current network downward |
| CTRL + SHIFT + HOME | Selects all adjacent networks from cursor to top of POU |
| CTRL + SHIFT + END | Selects all adjacent networks from cursor to bottom of POU |
| CTRL + A | Selects all networks in a POU |
| CTRL + Y | Toggles between symbolic and absolute addressing mode |
| CTRL + B | Toggles between data block and variable address view |
| CTRL + T | Displays symbol information tables for each network |

### Zooming

| Key sequence | Action |
|---|---|
| CTRL + PLUS SIGN | Zooms in (from numeric keypad only) |
| CTRL + MINUS SIGN | Zooms out (from numeric keypad only) |

### Dropping instructions

| Key sequence | Action |
|---|---|
| F4 | List box containing all contact mnemonic types |
| F6 | List box containing all coil mnemonic types |
| F9 | List box containing all box mnemonic types |

## Data block, symbol table, local variable table, status chart

| Key sequence | Action |
|---|---|
| CTRL + X | Cuts selected text |
| CTRL + C | Copies selected text |
| CTRL + V | Pastes selected text |
| DOWN ARROW | Moves cursor down (adds new rows to the end of the table) |
| UP ARROW | Moves cursor up |
| LEFT ARROW | Moves cursor left |
| RIGHT ARROW | Moves cursor right |
| DELETE | Deletes selected text |
| F1 | Provides help on current window |
| F2 | Lets you edit current field |
| CTRL + A | Selects all fields |
| TAB | Moves cursor from left to right and top to bottom |
| SHIFT + TAB | Moves cursor from right to left and bottom to top |
| ENTER | Confirms field and moves cursor to the right |
| HOME | Moves cursor to the first column |
| END | Moves cursor to the last column |
| CTRL + PAGE UP | Displays the next table/ chart. Moves left through the Symbol Table/ Status Chart tabs. |
| CTRL + PAGE DOWN | Displays the next table/ chart. Moves right through the Symbol Table/ Status Chart tabs. |
| SHIFT + ARROWS | Selects text |
| CTRL + I | Inserts new row above the current row |

## Changing between absolute and symbolic addressing mode

| Key sequence | Action |
|---|---|
| CTRL + Y | Toggles between symbolic and absolute addressing mode |
| CTRL + T | Displays symbol information tables for each network |
| CTRL + B | Toggles between data block and variable address view |

**Note**

You can choose between two representation forms for the symbolic addressing:

- Display only symbolic addresses
- Or display symbolic and absolute addresses

To do this, go to the **Tools > Options** menu and open the "LAD editor" tab.

## 3.1.3 Using the online help

**Context-sensitive help**

To call the help, select the menu item or open the dialog box for which you want help and press the "F1" key. The context-sensitive help for that topic will be displayed. In some cases, you can press "Shift" and "F1".

**Help menu**

The Help menu in the Programming Tool offers the following selections:

- **Contents and Index** allows you to navigate this help system by means of a Table of Contents browser or a searchable index.

  **Note**

  The help topics are grouped as chapters in the online help. The chapters are represented as book symbol.

- **What's This?** provides definitions of interface elements. You can also access What's This? help by pressing the shift and F1 keys simultaneously. The cursor changes to a question mark; use it to click the item for which you want help.
- **828 on the Web** provides access to Siemens Internet websites for technical support and product information.
- **About** lists product and copyright information for the Programming Tool.

**The Contents browser**

In the **Programming Tool**, you can open the Contents browser by choosing **Help>Contents and Index** from the menu bar.

From **inside a help topic**, you can bring up the Contents browser by clicking the "Help Topics" button.

**The index**

To display an alphabetical listing of topic keywords, select the "Index" tab in the help.

**Printing help**

You can print topics from the Contents browser or from inside an individual topic:

- To print **all of the topics in a book** from the "Table of contents" browser, select the book title and click the "Print" button.
- To print **an individual topic** from the Contents browser, select the topic title and click the "Print" button.
- To print from inside **an individual topic**, use the "Print" button that is located above the topic title.

**Note**

The help topics in the "First steps" section of the help are numbered successively like the sections in a manual. This makes it easier to use this part of the help as print-out.

## 3.1.4 Specifying the representation of the Programming Tool

The Programming Tool offers many ways to access and display information.

To improve clarity, you can initially hide the output window.

You can cover over or minimize windows that you need while programming (such as the local variable table and symbol table) and display them only when necessary. This gives you the maximum amount of space for the essentials: the instruction tree (for programming in LAD) and the program editor (for programming in LAD and viewing in STL).

**Note**

The output window is displayed automatically during the compilation.

Possibilities for arranging the various components of the Programming Tool working area:

- **View or hide various window elements**
  From the "View" menu, you can display or hide various objects. The objects that have checkmarks are the objects that are currently open in the Programming Tool.
- **Arrangement of windows overlapped, side-by-side or stacked vertically**
  To do this, select one of the menu commands **Window > Overlapped, Window > Side-by-side or Window > Stacked vertically.**
- **Minimize, restore, maximize or close a window**
  Use the minimize, restore, maximize and close buttons that are located in the title bar of each window.

  **Note**

  When you maximize a window, its buttons are displayed in the menu bar area beneath the buttons of the main Programming Tool window. When you maximize a window, it covers the display of any other window that you may have opened, but does not cause the other window(s) to close.

- **View various components of a window using tabs**
  Windows such as the Program Editor, the Status Chart, the Data Block or the Symbol Table have several tabs. For instance, the Program Editor window contains tabs that allow you to navigate between the main program (OB1), the subprograms, and the interrupt programs.

  **Note**

  If you have several open windows, under some circumstances not all tabs of the various windows are displayed. You can use the scroll buttons located next to the tabs to navigate from the first tab to the last. The buttons are not used to scroll through the window content. There are separate scroll buttons for this purpose.

- **Size or remove the Local Variable Table**
  Position the cursor over the split bar that divides the Program Editor and the Local Variable Table and drag to increase or decrease the size of the Local Variable Table. Drag the Program Editor all the way up to completely cover the Local Variable Table if your program does not have subprograms or interrupt programs that require you to define any local variables.

  **Note**

  You cannot remove the Local Variable Table by using the View menu because the Local Variable Table is part of the Program Editor window.

- **Undock or hide toolbars**
  By default, the "Standard", "Test", and "Instruction" toolbars appear below the menu bar in the Programming Tool. However, you can undock any toolbar and move it by placing the cursor inside the toolbar area, clicking, and dragging. If you drag the toolbar near the border of any of the windows in the Programming Tool, the toolbar will dock to the border of that window. Otherwise, the toolbar remains independent. You can show and hide toolbars by selecting the appropriate bar (Standard, Test, Instruction) in the **View >Toolbars** menu.

**See also:**

Configuring the toolbars (Page 487)

Selecting color settings (Page 489)

Setting up menus and expanding tools (Page 488)

Working with the zoom function (Page 452)

# 3.2 Chapter 2 Concepts for programming

## 3.2.1 Mode of operation of the program

When you download a program to your PLC and place the PLC in RUN mode, the central processing unit (CPU) of the PLC executes the program in the following sequence:

1. The CPU reads the status of all inputs connected to the automation system. This data is stored in the input memory area, known as the process image input.
2. The CPU uses these inputs to execute the logic of the program.

3. As the program is evaluated, the CPU stores the results of the program logic in the output memory area, known as the process image output.
4. At the end of the program, the CPU writes the data from the process image output to the physical outputs.
5. The cycle of tasks is repeated.

This cyclic execution of CPU tasks is known as the **scan cycle**. The scan cycle in a CPU includes the following tasks:

- Reading the digital inputs
- Executing the program
- Processing any communication requests
- Executing the CPU self-test diagnostics
- Writing to the digital outputs

**See also:**

RUN/STOP modes of the PLC (Page 361)

## 3.2.2 Overview of the addressing

**Understanding absolute and symbolic addresses**

You can identify the operands of the instructions in your program absolutely or symbolically. An absolute reference uses the memory area and bit or byte location to identify the address. A symbolic reference uses a combination of alphanumeric characters to identify the address.

**Display examples of addresses in the Program Editor**

| | |
|---|---|
| I0.0 | Absolute address is designated by memory area and address number. |
| #INPUT1 | # symbol precedes a local variable. |
| INPUT1 | Global symbol name |
| ??.? or ???? | Red question marks indicate an undefined address (must be defined before the program is compiled). |

**Global versus local scope**

Symbolic addresses that were assigned in the symbol table are globally valid.

Symbolic addresses assigned in a Local Variable Table have local scope.

**Global symbols**

You make global symbol assignments by using the Symbol Table (Page 413). You do not have to make symbol assignments before you use symbols in your program; you can make symbol assignments at any time.

**Local variables**

Local variables are assigned in the Local Variable Table (Page 229) of the respective program organization unit (POU). Local variables are limited in scope to the POU where they were created. Each POU has its own separate Local Variable Table.

**Example:**

You define a variable called INPUT1 in the Local Variable Table of a subprogram called SBR1. When you refer to INPUT1 from within SBR1, the Program Editor recognizes it as a local variable of SBR1.

However, if you refer to INPUT1 from elsewhere in the program (for instance, from OB1 or from a second subprogram), the Program Editor does not recognize it as a local variable (because it is outside of SBR1) and treats INPUT1 as an undefined global symbol.

**Tips**

- If you use the same name for an address at the local and at the global level, then the local use has priority. In other words, if the Program Editor finds a definition for the name in the Local Variable Table for a given program block, that definition is used. If no definition is found, the Program Editor then checks the Symbol Table.
  **Example:**
  You define PumpOn as a global symbol. You also define it as a local variable in SBR2, but not SBR1. When the program is compiled, the local definition is used for PumpOn in SBR2. The global definition is used for PumpOn in SBR1.
- Local variables use temporary PLC local memory, rather than requiring PLC program memory space. Subprograms that use only local variable parameters, or no parameters at all, are portable and can be reused in more than one program. You are not required to use local variables. It is an option for advanced programming techniques. If you want to use a parameter in several POUs, it may make more sense to define it as a global symbol in the Symbol Table than to define it as a local variable. Because you would have to make separate assignments to the Local Variable Table of each POU.
- Because local variables use temporary memory, be sure that you initialize your local variables within the POU each time the POU is called. You cannot be sure that a local variable will retain a data value from one iteration to the next.

**See also:**

Direct addressing (Page 261)

Subprogram (Page 223)

Interrupt program (Page 227)

Constants (Page 271)

Mnemonics (Page 489)

Entering addresses in LAD (GS 3.5) (Page 40)

## 3.2.3 Structuring a program

### Basic elements of a program

Your program consists of the following types of Program Organization Unit (POUs):

| | |
|---|---|
| Main program | The main body of the program (known as OB1) is where you place the instructions that control your application. The instructions in the main program are executed sequentially, once per cycle of the CPU. |
| Subprograms | A subprogram is an optional set of instructions located in a separate block that is executed outside of MAIN (OB1) when called from the main program. |
| Interrupt programs | An interrupt program is an optional set of instructions located in a separate block that is executed outside of MAIN (OB1). The interrupt program is executed with higher priority and can interrupt MAIN (OB1). |

The Programming Tool organizes your program for you by providing separate tabs in the Program Editor window for each POU. The main program, OB1, is always the first tab, followed by any subprograms and interrupt programs that you may have created.

### Terminating POUs

Because your program is compartmentalized (each POU occupies a separate tab), there is no question about where OB1 or your various subprograms and interrupt programs end. The compiler terminates each POU for you with an absolute END, RET, or RETI.

### Subprograms

Subprograms are useful in cases where you want to execute a function repeatedly. Write the logic once in a subprogram and reinvoke this subprogram as often as required during the processing of the main program. This means you do not need to write the same logic at every location in the main program where the function is to be performed. This provides several benefits:

- Your overall code size is reduced.
- Your cycle time is also reduced because you have moved the code out of the main program (where it is automatically evaluated every cycle, whether it is executed or not). Your subprogram can be called conditionally and is not evaluated during the cycles in which it is not called.
- Subprograms are easily portable. They allow you to isolate a function and copy it to other programs with hardly any or no rework.

**Note**

Use of variable memory limits the portability of your subprograms because it is possible for variable memory address assignment from one program to conflict with an assignment in another program. Subprograms that use the Local Variable Table for all address assignments, by contrast, are highly portable because there is no concern about addressing conflicts.

### Interrupt programs

You can write interrupt programs processing specific predefined interrupt events: When an interrupt event occurs, interrupt programs are called up by the operating system of the PLC instead of the main program. It cannot be predicted when the system issues an interrupt and the interrupt program enters data into the memory used at other program locations. Use the

Local Variable Table to ensure that your interrupt programs only occupy temporary memories and do not overwrite data from other program locations.

**See also:**

Subprogram (Page 223)

Interrupt program (Page 227)

Local variable table (Page 229)

## 3.2.4 Project components and their functions

A project in the Programming Tool consists of the following components:

**Project**

A project is the largest possible organizational unit for the user when working with the PLC. It can be compared with a container and can

- Accept program and data blocks, symbol tables, status charts, cross references as well as interface and debug settings.
- Be stored on data media or downloaded from these media.
- Be loaded into the CPU and read out from the CPU.

**Note**

When calling the Programming Tool, a new project with the default name "Project1" is implicitly created if you do not specify an existing project.

**Data classes**

Data classes are user-selectable project subcontainers for the input of data blocks.

Data classes have been introduced to

- Make different responsibility hierarchies of the PLC data transparent and delimitable.
- Be able to separately manage defined project subsets which can be assigned to different user groups or hierarchies.

Data classes can

- Be stored on data media or downloaded from these media (export, import).
- Be loaded into the control and downloaded from the control.

**Program (POU)**

A program (also program organization unit, "POU") is a block for a sequence of commands (including comments) that the user assembles via the LAD editor to solve the relevant task.

The users have three types of these POUs at their disposal:

- The main program (MAIN): There is only one main program which the system calls in the PLC cycle. It is permanently stored in the "Manufacturer" data class.
- Subprograms (SBR_xyz): Subprograms are used to structure and encapsulate functions. Once they have been coded, they can be called up several times. They are permanently stored in the "Manufacturer" data class.
- Interrupt programs: These are processed with higher priority at another location in the PLC cycle and are reserved for special tasks. The data class cannot be changed.

**Data block**

A data block is a block for data (initial values, actual values) and comments with the following properties:

- The data is stored in the exact sequence specified by the user. This means that the inner structure of the data block is defined and if several data blocks are created with the same inner structure (i.e. the same type), then a certain data item is always located at a specific location. This location is called offset and is the relative length in bytes from the beginning of the data block (DB) up to the actual data item.
- Data can be assigned initial values which it assumes when being loaded for the first time into the PLC.
- The actual values of the data can be read online from the control, changed, and also saved with the project.

**Symbol table**

The symbol table is used for symbolic addresses. Frequently, symbols simplify programming and increase the transparency of programs. In the compiled program being loaded into the target system, all symbols are converted into absolute addresses. The data relating to the symbol table is loaded into the target system.

**Status charts**

The status charts show how the process values change during program execution. Status charts are not loaded into the target system. They are only intended to monitor the activities of the target system.

**Cross references**

The cross reference window shows the following in table form:

- The symbolic or absolutely addressed operands and their target locations;
- The bytes used, and
- The bits used.

Cross references are not loaded into the target system.

**Tips**

If you did not download something to the target system, you cannot upload it from the target system. Hence, you cannot upload status charts or cross references from the target system.

If you do not download it to the target system, your process will not change. If you change a project in the Programming Tool, this does not have any effects on your process, unless you load the changes into the target system.

## 3.2.5 The LAD and STL program editors

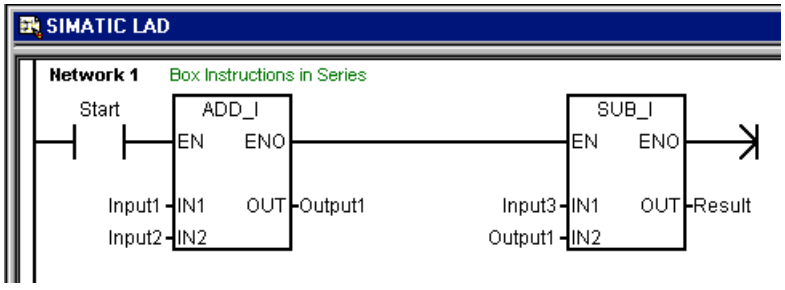
The Programming Tool provides the LAD editor with which you can write programs using the appropriate instructions. The STL editor allows you to view the program execution sequence. This means you can simply create programs in a graphic editor and then view them in a type of text editor.

These two program editors are shown below.

**LAD editor (ladder logic)**

The LAD editor allows you to build programs that resemble an electrical circuit diagram. The LAD editor is the method of choice for many PLC programmers and maintenance personnel; however, it is also well suited to less experienced programmers. Basically, the ladder programs allow the CPU to emulate the flow of electric current from a power source through a series of logical input conditions that in turn enable logical output conditions. The logic is usually separated into small, easy-to-understand pieces that are often called rungs or networks. The program executes one network at a time, from left to right and then top to bottom, as indicated by the program. Once the CPU has reached the end of the program, it starts over again at the top of the program.

The diagram below shows an example of a ladder program.




The various instructions are represented by graphic symbols and include three basic forms.

| | |
|---|---|
| Contacts | Representation of logical input conditions such as switches, buttons, internal conditions, etc. |
| Coils | Representation of logical output conditions such as lamps, motor starters, interposing relays, internal output conditions, etc. |
| Boxes | Representation of additional functions such as timers, counters, math instructions, etc. |

You can create networks from simple to highly complex in the logic ladder. You can build networks with midline outputs; you can even connect multiple box instructions in series. Boxes that can be connected in series are labeled with an Enable Output (ENO) line.

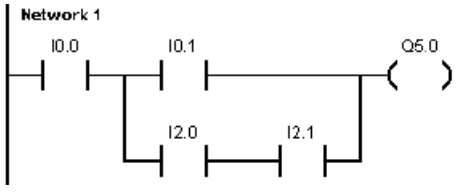
**STL editor (statement list)**

You can view programs in the STL editor that you created in the LAD editor; however, you cannot modify them. In general, the STL editor is more suitable for experienced programmers who are familiar with automation systems (PLCs) and programming techniques. This is because in STL you are viewing the "native language" of the CPU, in which information does not have to be additionally made transparent.

The diagram below shows an example of a program in the statement list.

```
NETWORK 1
LD      I0.0
LD      I0.1
LD      I2.0
A       I2.1
OLD
ALD
=       Q0.5
```

As you can see in the diagram, this type of programming in the text editor is very similar to programming with Assembler-type languages. The CPU executes each instruction in the sequence specified by the program, from top to bottom, and then starts again at the top. STL and Assembler-type languages are also similar in other regards. The control logic is realized using a logical stack. The LAD editor automatically inserts the instructions that are necessary to process the stack operation. In STL, these instructions for processing the stack are also shown. A simple program in LAD and the corresponding program in STL are shown in the diagram below.




```
Network 1
LD                              I0.0
LD                              I0.1
LD                              I2.0
A                               I2.1
OLD
ALD
=                               Q0.5
```

The diagram below shows the sequences in the stack:

| | Instructions | | | | | |
|---|---|---|---|---|---|---|
| **Stack** | **LD I0.0** | **LD I0.1** | **LD I2.0** | **A** | **OLD** | **ALD** |
| S0 | I0.0 | I0.1 | I2.0 | I2.0 & I2.1 | (I2.0 & I2.1) OR I0.1 | I0.0 AND [(I2.0 & I2.1) OR I0.1] |
| S1 | | I0.0 | I0.1 | I0.1 | I0.0 | |
| S2 | | | I0.0 | I0.0 | | |
| S3 | | | | | | |
| S4 | | | | | | |
| S5 | | | | | | |
| S6 | | | | | | |
| S7 | | | | | | |
| S8 | | | | | | |

# 3.3 Chapter 3 Entering a ladder logic program

## 3.3.1 Creating a project

### Creating a new project

To open the application, double-click on the following button **"PLC Programming Tool"**:

Or, in the Windows start menu, select the command **PLC Programming Tool > PLC Programming Tool**. A new Programming Tool project opens.

### Opening an existing project

From inside the Programming Tool, select one of the following options in the **File** menu:

- **Open** - Allows you to navigate to an existing project and open it.
- **File name** - If you have recently worked in a project, it is listed under the File menu and you can select it directly, without having to use the Open dialog box.

You can also navigate to the required directory in Windows Explorer and open the project directly from there without having to first start the Programming Tool. Your project is contained in a single file with a *.ptp extension.

**Tip**

Once you have created a project, you can begin to write your program. However, you should perform the following tasks before you begin:

- **Range-check by PLC type**
  You can select a PLC type (Page 334) before you write your program, so that the Programming Tool can check the parameter range according to your PLC. If you have specified a CPU type for your project, the Instruction Tree shows the instructions with a red x: , which cannot be used for your PLC.
- **Setting up communication**
  You can set up communication now (Page 50), or wait until you are ready to download a program.
- **Customizing your workspace**
  You can customize your workspace in a variety of ways (Page 24) if desired.

## 3.3.2 Ladder logic elements and their function

Ladder logic (LAD) is a graphical programming language that resembles electrical circuit diagrams. When you write a program in LAD, you use graphical components and arrange them to form a network of logic. The following types of elements are available for you to use in creating your program:

| | | |
|---|---|---|
| Contacts |  | Representation of a switch through which current can flow. Current flows through a normally open contact only when the contact is closed (a logical value of one). Current flows through a normally closed or negated (NOT) contact only when the contact is open (a logical value of zero). |
| Coils |  | Representation of a relay or output that is energized by signal flow. |
| Boxes |  | Representation of a function (e.g. a timer, a counter or a math instruction) which is executed when the signal flow reaches the box. |

A network is composed of these elements and represents a complete circuit. The current flows from the left-hand power rail through the closed contacts and activates coils or boxes. The power rail is represented in the LAD editor as a vertical line at the left-hand edge of the window.

## 3.3.3 Rules for constructing simple, series, and parallel networks in LAD

**Rules about placing contacts**

Each network must begin with a contact.

You cannot terminate a network with a contact.

**Rules about placing coils**

You cannot begin a network with a coil. Coils are used to terminate a network. A single network can have several coils, as long as the coils are placed on parallel branches of that particular network. You cannot place multiple coils in a series on the same rung of a network.

### Rules about placing boxes

If a box has ENO, signal flow extends beyond the box; this means that you can place further instructions after the box. You can have several boxes with ENO in a series on the same rung of a network. If the box does not have ENO, you cannot place any instructions after it.

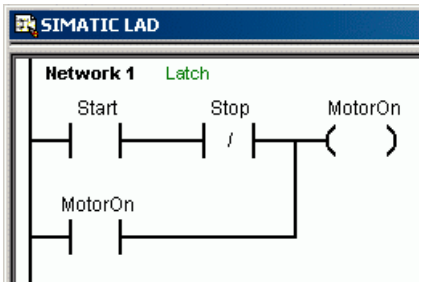
### Network size limits

You must consider the program editor as being a grid that is divided into fields. In a field, you can assign an instruction, assign a value to a parameter or drag a connection line. Inside that grid, an individual network can extend no further than 32 fields horizontally and 32 fields vertically.

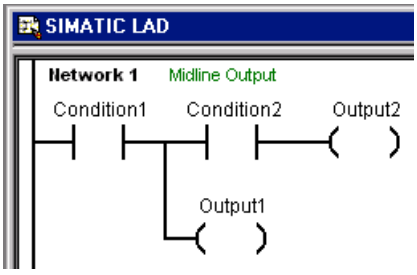
### Examples

The diagrams below illustrate some of the logic constructions possible in the Programming Tool LAD editor.

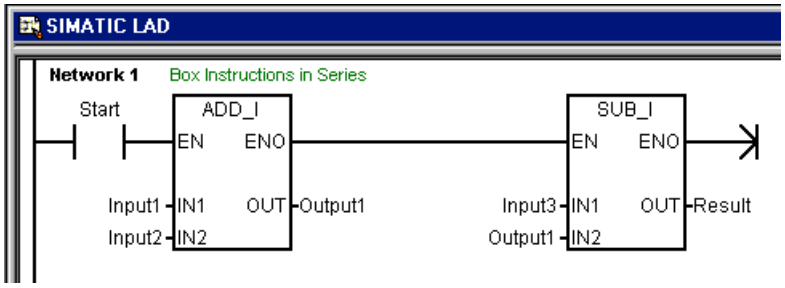
1. This network uses a normal contact ("Start") and a negated (NOT) contact ("Stop"). Once the motor is successfully activated, it latches and remains on until the Stop condition is met.



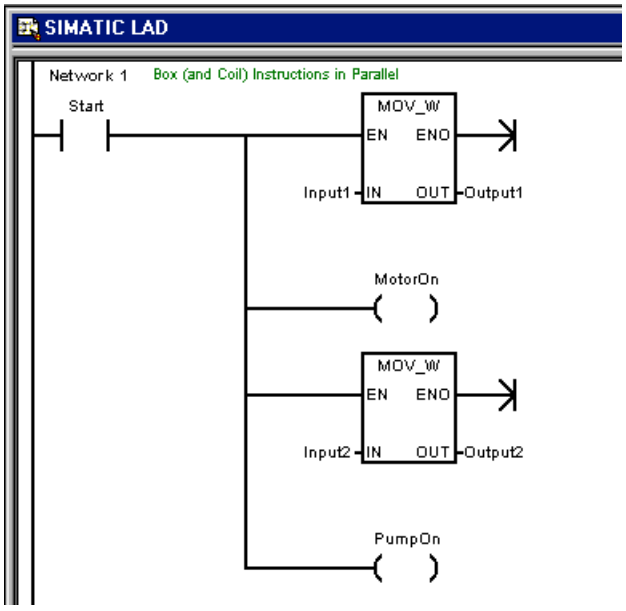

2. Notice that if the first condition is met, the preliminary output (Output 1) comes on before the second condition is evaluated. You can create multiple rungs that have midline outputs.



3. If the first box instruction evaluates successfully, the signal flows down the network to the second box instruction. You can cascade multiple ENO instructions in a series on the same rung of the network. If any instruction fails, the rest of the instructions in the series are not evaluated; signal flow stops. (Errors are not cascaded through the series.)




4. When the start condition is met, all outputs (boxes and coils) are activated. If one output fails to evaluate successfully, the signal still flows to the others; they are not affected by the instruction that failed.

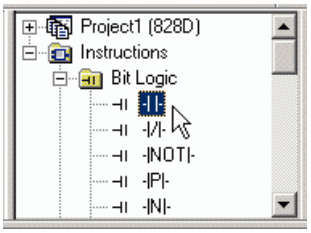



## 3.3.4 Entering instructions in LAD

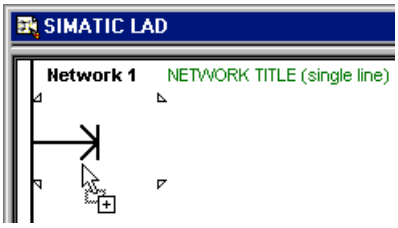
There are several ways to enter a LAD instruction:

**Instruction tree: Drag-and-drop**

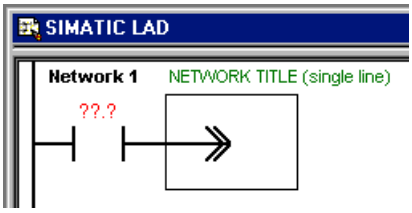
1. Select instruction.




2. Drag instruction to desired location.




3. Drop instruction in desired location by releasing the mouse button.

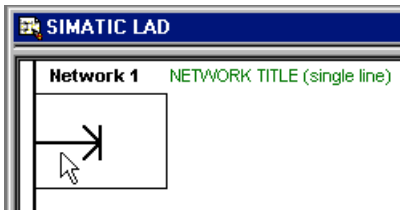



**Note**

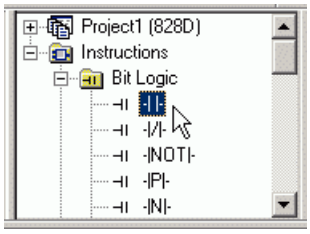
The cursor automatically prevents you from dropping an instruction at an illegal location, e.g. onto a network title or the parameter of another instruction.

**Instruction tree: Double-click**

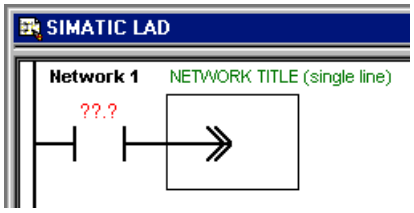
1. Place the cursor at desired location in Program Editor window. A selection box appears around the location.




2. In the Instruction Tree, navigate to the desired instruction and double-click it.

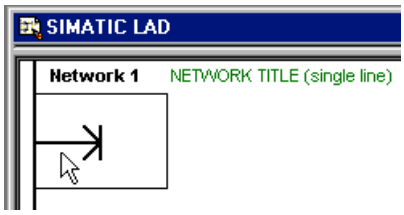



3. After you double click, the instruction appears in the Program Editor window.



**Toolbar button or function key**

1. Place the cursor at desired location in Program Editor window. A selection box appears around the location.



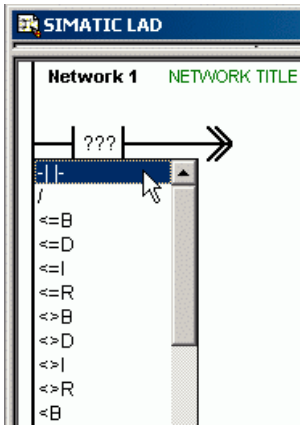

2. Either click the appropriate toolbar button...


... or use the appropriate function key (F4=contact, F6=coil, F9=box) to insert a generic instruction.

**Note**

A generic instruction cannot be addressed and will prevent your program from being compiled. You must first select a specific instruction type as shown in Step 3.

3. A drop-down list appears. Scroll or type the first few letters to navigate to the desired instruction. Double-click the desired instruction or use the ENTER key to insert it.
(If you do not choose to select the specific instruction type at this time, you can return to the network and click the mnemonic field of the generic instruction (which contains ??? instead of a mnemonic), or you can select the instruction and press the ENTER key to recall the list.)



**Insert versus overstrike mode**

The Programming Tool allows you to toggle the INSERT key on your keyboard to switch between two editing modes:

- In insert mode (selected when you press the INSERT key), if you drop an instruction onto another instruction, the Program Editor moves the existing instruction over to make room for the new one.
- In overstrike mode (the default when the INSERT key has not been pressed), if you drop an instruction onto another instruction, the Program Editor deletes the existing instruction and replaces it with the new one.

**Parameter retention in overstrike mode**

If you replace (overstrike) one instruction with another instruction that has the same profile, any assignments that you made to the old parameters are transferred to the new parameters. (That is to say, if the second instruction has the same number of signal flow inputs, input parameters, signal flow outputs, and output parameters as the first instruction, then the parameter assignments are retained when you overstrike the first instruction with the second.)

**Drawing connecting lines**

You can use the horizontal and vertical connecting line toolbar buttons, or hold down the CTRL key and press the LEFT, RIGHT, UP, or DOWN ARROW key on your keyboard, to draw lines between the elements of your network and/or the left-hand power rail.

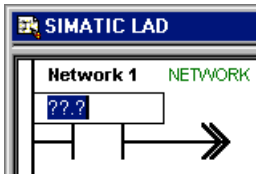

## 3.3.5 Entering addresses in LAD

When you enter an instruction in LAD, the parameters are initially represented by question marks: ??.? or ????.

The question marks indicate that the parameter is unassigned. You can assign a constant value or an absolute, symbolic, or variable address for the parameters of the element. You can enter the address immediately when you enter the instruction or you can come back later. Your program will not be compiled properly if any parameters remain unassigned.

**Assigning addresses**

To assign a constant value (e.g. **100**) or an absolute address (e.g. **I0.1**), simply type the desired value in the address field of the instruction. (Use the mouse or the ENTER key to select the address field for specifying the data type.)



To assign a symbolic address (a global symbol or local variable that uses a name such as **INPUT1**), you must perform the following simple steps:

1. Type the symbol or variable name in the address field of the instruction.
2. For **Global Symbols**, use the Symbol Table (Page 413) to assign an absolute address to the symbol name.

**Note**

You do not have to predefine symbols to use them in your program. You can define the addresses later.

For **Local Symbols**, use the Local Variable Table (Page 229) above the Program Editor window. Enter the symbol name in the "Name" column. You do not enter an address for a local variable because the compiler automatically assigns local memory addresses. You can minimize the size of the Local Variable Table by dragging the corner of the table.
The use of local variables is an advanced programming technique. Inexperienced programmers should consider assigning all symbolic values as global symbols in the Symbol Table.

**Note**

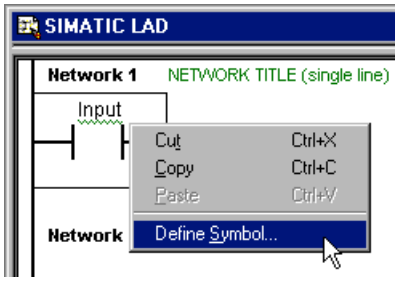
The Program Editor automatically formats address values after you enter them. You do not need to type the formatting characters; they will appear in the display after you finish typing.

### Examples of how the Program Editor displays addresses

| | |
|---|---|
| I0.0 | Absolute address is designated by memory area and address number. |
| #INPUT1 | # symbol precedes a local variable |
| INPUT1 | Global symbol name |
| ??.? or ???? | Red question marks indicate an undefined address (must be defined before the program is compiled). |

### Matching addresses and defining symbols

When you **right-click** on a parameter of an instruction, the pop-up menu is displayed. This allows you to quickly define addresses in the symbol table.



**Note**

Use the ENTER key to cycle through all the instructions of a network, one parameter at a time, in order to quickly edit any necessary addresses.

Alternatively, select individual parameters by right-clicking, and define symbols or find matches for those addresses by using the pop-up menu.

**Valid and invalid symbolic names**

Symbolic names are permitted to contain alphanumeric characters and underscores. They are also permitted to contain extended characters (ASCII 128 to ASCII 255). The first character is restricted to alpha and extended characters only.

| Valid names | Illegal names |
|---|---|
| a11 | 1loop |
| a_b_1_2 | I:kdl";ld |
| ÀÁñòÓÞßü_1284938 | |

Such illegal names begin with a number or contain characters that are not alphanumeric or in the extended character set.

**See also:**

Showing entry errors in the Program Editor in LAD (GS 3.9) (Page 48)

Addressing (GS 2.2) (Page 26)

Constants (Page 271)

Subprogram (Page 223)

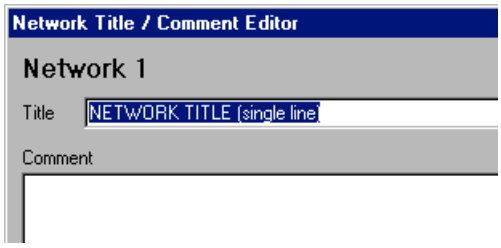
Interrupt (Page 227)

## 3.3.6 Entering comments about the program in LAD

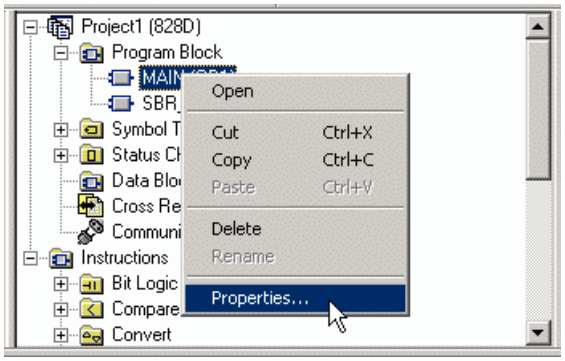
There are two levels of comment in the LAD editor.

**Network comments**

Place your cursor anywhere in the network title line and double-click or hit the ENTER key in order to bring up the Network Title / Comment Editor. You can enter a title that identifies the network and a comment about the contents of the network. The network title is visible within the Program Editor. The network comment is visible only within the Network Title / Comment Editor and when you print out program comments.
The maximum number of characters permitted in the network title is 127. The maximum number of characters permitted in the network comment is 1,023.

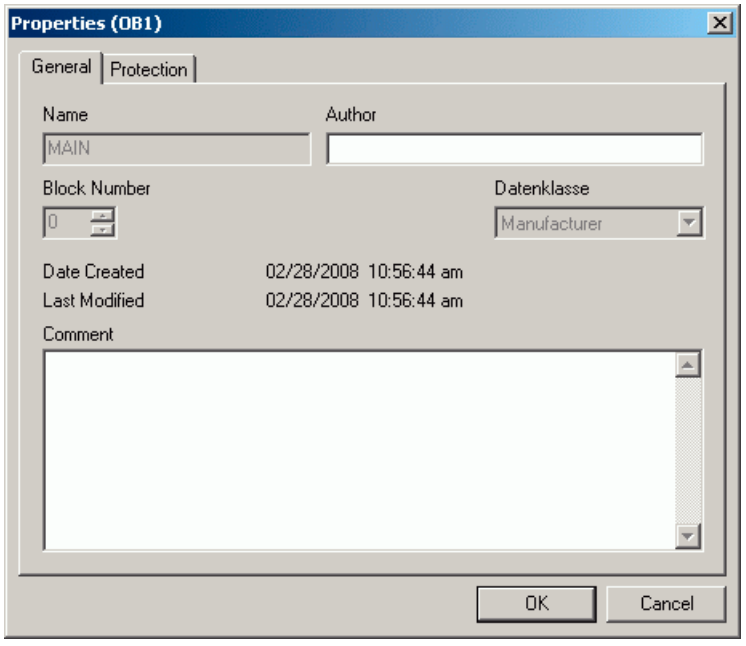



### Project component comments

| | |
|---|---|
| Navigate in the project tree to a project component and right-click it to bring up the menu with the "Properties" dialog box for that item. |  |



Each component in your project has a Properties dialog box that allows you to name the component, identify the author of the component, and record comments about the components. For instance, if you have multiple status charts, the charts are represented as separate tabs within the Status Chart window and as separate components under the Status Chart icon in the Instruction Tree window. Each status chart has its own Properties box.



**Note**

You can save the properties of "MAIN (OB1)" in the comment of a version identification.

For additional information, see the following: Version identification (Page 82)

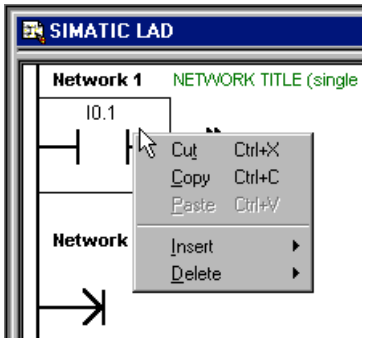
## 3.3.7 Editing program elements in LAD

### Cutting, copying, pasting, or deleting multiple networks

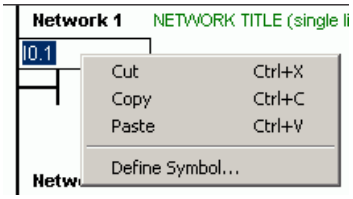
You can select multiple adjacent networks by dragging the mouse or using the SHIFT key with the UP and DOWN ARROW keys. You can cut, copy, paste, or delete your selection. Use the toolbar buttons; choose a command from the Edit menu, or right-click to bring up a pop-up menu of editing options.

### Editing fields, instructions, addresses, and networks

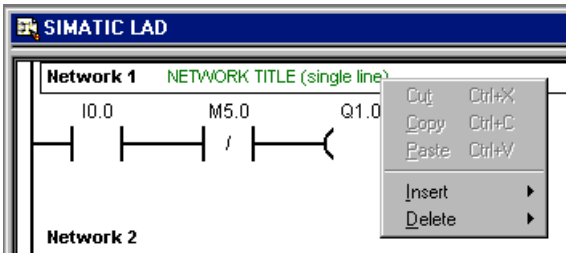
- When you click an **empty field** in the Program Editor, a box appears to indicate which field you have selected. You can use the pop-up menu to paste text from the clipboard into the empty field or insert a new row, column, vertical line, or network at that location. You can also delete the network from the location of the empty field.
- When you click an **instruction**, a box appears around the instruction to indicate which instruction you have selected. You can use the pop-up menu to cut, copy, or paste over the instruction, as well as to insert or delete a row, column, vertical line, or network at that location.



- When you click an **instruction parameter**, a box appears around the field to indicate which parameter you have selected. You can use the pop-up menu to undo your typing, to cut, copy, paste, or delete information, or to quickly select all the contents of the field. You can also use a double-click to select all.




- When you click a **network title**, a box appears around the title area. However, the pop-up menu allows you to perform edits on the entire network, not just the title. You can cut, copy, or paste over the network, insert a new network or delete the existing network.



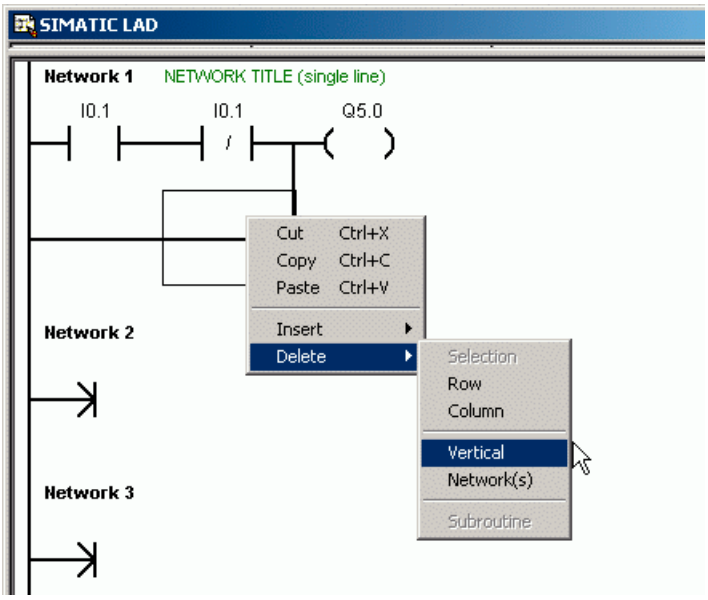

You can also use toolbar buttons, standard Windows control keys, and the Edit menu to cut, copy, or paste a selection.

### Moving elements closer together

You can cut and paste elements and lines, and delete rows or columns. However, the Program Editor requires a certain amount of space between elements. In some cases, you may not be able to place one element closer to another element (for instance, a horizontal line segment is required between box instructions and cannot be removed).

### Deleting elements

You can delete individual fields by using the DELETE or BACKSPACE key. You can delete rows, columns, vertical lines, and networks by using the Edit menu or right-clicking to bring up the pop-up menu.




**Note**

In order to correctly select a vertical line for deletion, always place the cursor on the field to the left of the vertical line.

## 3.3.8 Find, Replace and Go To functions

To use Find, Replace or Go To:

- Select the **Edit > Find, Edit > Replace** or **Edit > Go To** menu command.
- Press CTRL+F for Find, CTRL+H for Replace, or CTRL+G for Go To

**Working with the function:**

For details, see the "Search function" and "Paste function" sections.

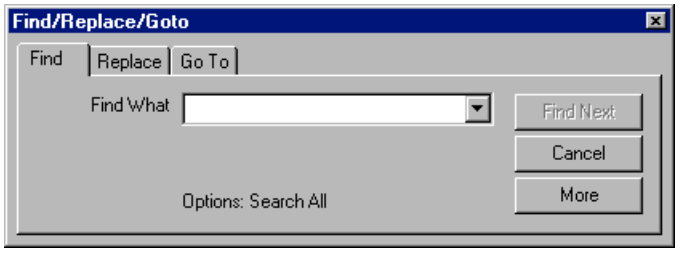
**Where they can be used**

Find, Replace, and Go To can be used in the Program Editor window, and in Local Variable Tables, Symbol Tables, Status Charts, the Cross Reference table, and the Data Block.

**How they work**

- The Find function allows you to locate a specified string, such as an operand, network title, or instruction mnemonic. (Find does not search network comments, only network titles. Find does not search the network symbol information tables that are available in LAD.)
- The Replace function allows you to replace the specified string. (Replace does not operate for instruction mnemonics.)
- The Go To function allows you to move quickly to another location by specifying the number of the network or row to which you wish to navigate.

### The Find function




1. To search, type a string in the "Find What" field.
2. To move to the next occurrence of the search string, click the "Find Next" button.

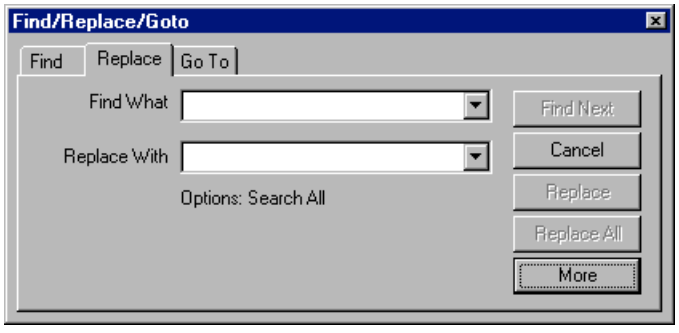
**Note**

Under some circumstances, the Find Next command may appear to search program code in an irregular order; actually, this order simply reflects the way that operands are stored in the code.

To define your search further, click the "More" button, which allows you to customize the search as
described below:

- You can choose a direction for the search by using the "Search" list box.
- You can search only for strings that have the same uppercase/lowercase value as the string that you entered. To do this, select the "Match Case" checkbox.
- You can eliminate strings that contain your search phrase as part of a larger word by selecting the "Whole Word" checkbox.
- You can search through all POUs (OB1 and all subprograms and interrupt programs) or all instances of a Local Variable Table, Symbol Table, or Status Chart, by selecting the appropriate checkbox.
- You can specify a range of lines to search. If you have selected a range of networks in the Program Editor, they appear as the default range in the Find dialog box. You can also type in the network or row numbers for the beginning and end of the search.
- You can specify whether or not to search network titles and/or program code by selecting the appropriate checkboxes.

### The Replace function



1. Type the string that you are searching for in the "Find What" field.
2. Type the string that you want to use as the replacement for the search string in the "Replace With" field.
3. To find an occurrence of the search string, click the "Find Next" button.

   **Note**

   Under some circumstances, the Find Next command may appear to search program code in an irregular order; actually, this order simply reflects the way that operands are stored in the code.

4. If you want to replace the string, click "Replace". If you have defined your search carefully, and there is no risk that strings could be modified erroneously, you can click "Replace All" to replace all instances of the string without examining them individually.

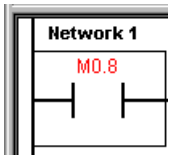
To define your search further, click the "More" button, which allows you to customize the search as described below:

- You can choose a direction for the search by using the "Search" list box.
- You can search only for strings that have the same uppercase/lowercase value as the string that you entered. To do this, select the "Match Case" checkbox.
- You can eliminate strings that contain your search phrase as part of a larger word by selecting the "Whole Word" checkbox.
- You can search through all POUs (OB1 and all subprograms and interrupt programs) or all instances of a Local Variable Table, Symbol Table, or Status Chart, by selecting the appropriate checkbox.
- You can specify a range of lines to search. If you have selected a range of networks in the Program Editor, they appear as the default range in the Find dialog box. You can also type in the network or row numbers for the beginning and end of the search.
- You can specify whether or not to search network titles and/or program code by selecting the appropriate checkboxes.

## 3.3.9 Showing input errors in the Program Editor in LAD

**Red text**

indicates illegal syntax.

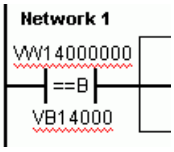



**Note**

When you replace the illegal address value or symbol with a valid value, the font automatically changes to the default font color (black, unless you have customized the window).

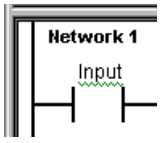
**A red wavy line underneath a value**

indicates that the value is either out of range or incorrect for this type of instruction.




**A green wavy line underneath a value**

indicates that the variable or symbol being used has not been defined. The Programming Tool permits you to write your program before defining variables and symbols. At any time, you can add the value to your Local Variable Table or Symbol Table.




**Tips:**

- Use the "Define Symbol" dialog box to resolve undefined symbols.

## 3.3.10 Compilation in LAD

**Compiling**

You can compile the program by using the toolbar buttons or the target system menu commands.

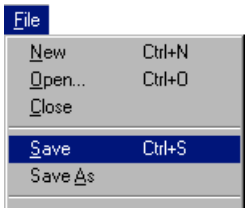

- "Compile"  compiles your project.

**Using the Output Window to resolve errors**

When you compile, the Output Window lists any errors that occur. The errors are identified by location (network, row, and column) as well as type of error. You can double-click an error line to bring up the network of code that contains the error in the Program Editor.

### 3.3.11 Saving your work

You can save your work by using the Save button on the toolbar or the **Save** and **Save as** options from the File menu.




- "Save" allows you to quickly save any changes to your work. (The first time you save a project, however, you are prompted to confirm or modify the default choices of name and directory for your current project.)
- "Save as" allows you to modify the name and/or directory location of your current project.

By default, the Programming Tool supplies the name "Project1.ptp" for your project when you first create it. You can accept or modify this name. If you accept it, the default name for the next project is automatically incremented to "Project2.ptp."

The default directory location for the Programming Tool projects is a folder called "Projects". This is located in the directory in which you have installed the Programming Tool. You do not have to accept the default location.

**Note**

**Naming projects**

You can add a version identification to project names.

The maximum length of the file name without file extension is 46 characters.

For additional information, see the following: Version identification (Page 82)

## 3.4 Chapter 4 Setting up communication and downloading a program to the target system

### 3.4.1 Communication overview

How you establish communications between the personal computer on which you are running the Programming Tool and your PLC depends on your hardware installation. If you are using nothing more than a PLC802(PPI)-interface/RS232 cable connection between your personal computer and the PLC, all you have to do is connect the cable and accept the default parameters that are assigned in the Programming Tool for the personal computer and PLC when you install the Programming Tool software.

You can establish communications or edit communication settings at any time.

Here are the tasks that are typically required for establishing communications:

- Connect the PLC and the PC on which the Programming Tool is installed using a cable or establish the connection via Ethernet.
- Optional: Make sure the PLC specified (Page 334) in the Programming Tool matches the actual type of your PLC.
- If you are using a simple PC/PPI connection, you can accept the default communications protocol (PLC802(PPI)) that is offered in the "Setting the PG/PC Interface" dialog box when you install the Programming Tool. Otherwise, select a different communications protocol (Page 443) and verify the parameters (station address, baud rate, etc.) for your personal computer from the "Setting the PG/PC Interface" dialog box.

**See also:**

Setting and changing parameters (Page 97)

Installing and removing communication interfaces (Page 96)

Setting up the PLC802(PPI)/RS232 interface (Page 88)

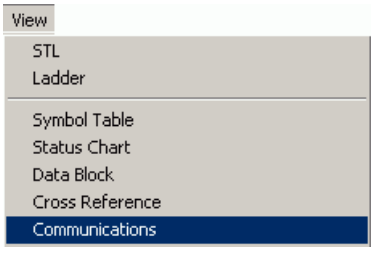
Setting up an Ethernet connection (Page 90)

## 3.4.2 Testing the communications network

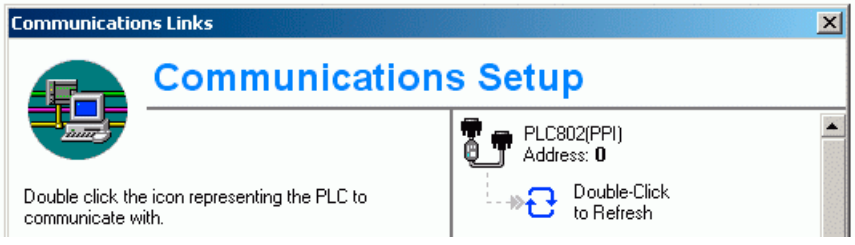
1. In the Programming Tool, click on the communication icon in the navigation bar:

   Or select the menu command **View > Communications**.

2. From the right-hand pane of the Communications dialog box, click the blue text that says "Double-Click to Refresh".

   If you have successfully established communications between your personal computer and the devices on your network, a list of the devices (with their model types and station addresses) is displayed.

The Programming Tool only communicates with one PLC at any time. A green box appears around the button that is currently in communication with the Programming Tool. You can double-click a different PLC to change communications to that PLC.

## 3.4.3 Downloading a program into the CPU

If you have successfully established communications between the personal computer on which you are running the Programming Tool and a PLC, you can download your program to that PLC. Follow the procedure outlined below.

**Note**

When you download a program block or data block from your personal computer to the PLC, the contents of the block that you are downloading from the personal computer overwrite the contents of the block that is currently in the PLC (if there already is one in the PLC). Be sure that you want to overwrite the block that is in the PLC before you initiate a download.

### Downloading in the RUN mode

Use a CPU which supports loading in the RUN mode. In this case, you need not set the CPU into the STOP mode before starting the load procedure.

| | |
|---|---|
| 828D | 808-PPU14x |
| 828D Step 2 | 808-PPU16x |

### Procedure

1. Click the Download button  in the toolbar or select the **File > Download to CPU** menu command. This displays the "Download" dialog box.
2. Click "OK" to confirm the settings. Then confirm that loading should be performed in the RUN mode or whether the CPU should be set to the STOP mode. If the download is successful, the following message is displayed: "Download successful". If loading is performed in the RUN mode, the user program is loaded during operation. If loading is performed in the STOP mode, you have to reset the PLC to the RUN mode.

**Note**

When loading in the RUN mode, you may not change the number and structure of data blocks. This is monitored during loading and rejected in case of change when loading in the RUN mode.

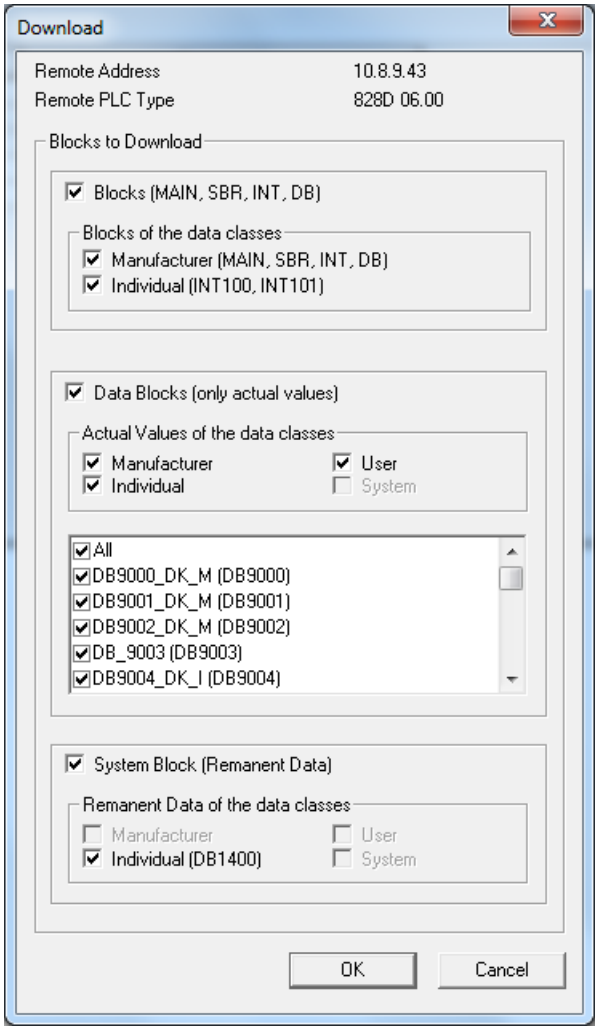
**⚠ CAUTION**

**Downloading in the RUN mode**

Downloading the program during the RUN mode should only be performed by authorized personnel who understand the limitations of downloading a program in RUN mode with respect to system operation. Downloading the program during the RUN mode can result in unexpected system operation causing serious injury, death, and/or damage to equipment.

On all CPUs, loading can be performed in the STOP mode.

1. Before you can start downloading, you must switch the CPU to the STOP mode. Check the mode display on the CPU. If the PLC is not in the STOP mode, click the STOP button in the toolbar or select the **PLC > STOP** menu command.
2. Click the Download button in the toolbar or select the **File > Download to CPU** menu command. The "Download" dialog box appears.
3. Click "OK" to confirm the settings. If the download is successful, the following message is displayed: "Download Successful". Proceed to Step 7.
4. If the value in the Programming Tool for your CPU type does not match your actual PLC, a warning box appears with this message:
"The PLC type selected for the project does not match the remote PLC type. Continue download?"
5. To correct the PLC type selection, select "Set Type to Remote PLC".
6. The CPU is read out and automatically set to the correct value. A window displays the detected CPU. Confirm with "OK" to close the dialog box. The download process is reinitiated.
7. Once the download is successfully completed, you must switch the CPU from the STOP mode back to the RUN mode before you can run the program in the PLC. Click the RUN button in the toolbar or select the **PLC > RUN** menu command to switch the PLC back to the RUN mode.




**Note**

**Saving retentive data**

The user-specific retentive data must be selected to be able to save them in the user project.

You can find information on this in the Chapter "Saving retentive data" (Page 345).

**See also:**

Communications overview (GS 4.1) (Page 50)

Testing the communications network (GS 4.2) (Page 51)

Downloading to the PLC (File > Download to CPU) (Page 382)

Correcting compilation and loading errors (GS 4.4) (Page 55)

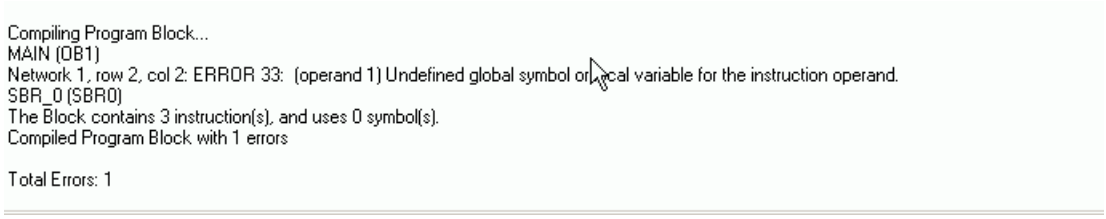
Uploading from a PLC (GS 6.3) (Page 78)

Uploading from the PLC (File > Upload from CPU) (Page 382)

### 3.4.4 Correcting compilation and loading errors

The output window displays messages automatically when compiling or loading the program.

The messages specify the network, the column, and the row, if an error has occurred, as well as the error number and a description of the error.




Double-click the error message to show the faulty network in the program editor.

If you have closed the output window, you can use the menu command **View > Output Window** to show it again.

## 3.5 Chapter 5 Monitoring and testing your program

### 3.5.1 Overview of test and monitoring functions

#### Diagnostics functions

Once you have established communication between your programming device, on which the PLC Programming Tool is installed, and a PLC and have loaded a program into the PLC, you can work with the diagnostic functions of the PLC Programming Tool and test new programs as well as monitor programs that are already being processed.

These help topics are explained in the following:

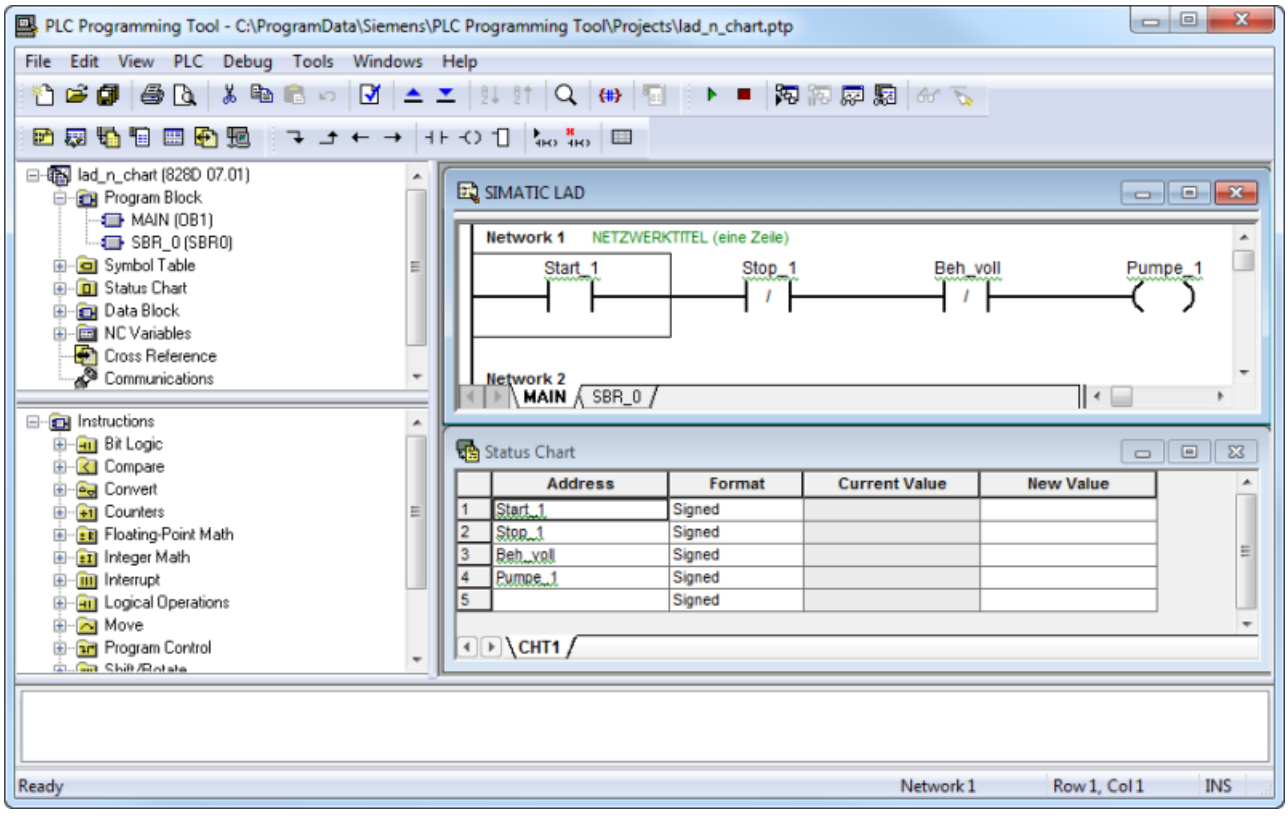
#### What is "status"?

"Status" refers to the display of the actual values of operands while executing the program in the PLC. You can display status information in a status chart, with the data block status or by enabling the program status in the program editor.

In the status chart in each case the values are recorded at the end of a scan cycle (end-of-scan status). It is not possible to display the status of the local data memory and the accumulators.
With the data block status, the actual values are also recorded at the end of a scan cycle in each case.

If the execution status is activated for the program status on the "LAD Status" tab (Tools > Options) (Page 492), the status values are recorded at the time of execution of the instructions. The states of the local data memory and the accumulators are displayed. The status values are updated and displayed only if the PLC is in the RUN mode.

An example for status information in the status chart and in the program editor of the PLC Programming Tool is shown in the following diagram:




**Note**

Please note that unnecessary project components (e.g. the project tree, the instruction tree, and the output window) have been omitted, in order to provide more space to display the required components (LAD program editor, status chart, function bar to test and symbol for the status chart in the navigation bar). Using the menu "View", you can set-up the environment in the PLC Programming Tool corresponding to your particular tasks, i.e. you can display just the project components that you require.

## Supporting CPUs

The execution status is not available for all CPUs. The following table shows which CPUs are available:

| | |
|---|---|
| 828D (as of PLC 05.04) | 808D-PPU14x |
| 828D Step 2 | 808D-PPU15x |
| | 808D-PPU16x |

### Execution status for unsupported CPUs

If the execution status is activated for a CPU that does not support this status, the end-of-scan status is automatically executed.

If the execution status is not activated, the end-of-scan status is executed. For this, the values are recorded at the end of a scan cycle in each case. It is not possible to display the status of the local data memory and the accumulators.

### Preconditions of the status update

Before you can update the status to monitor and test your program, you must execute the following tasks:

- Your program must be able to be compiled error-free.
- You must have set up communication between the PLC Programming Tool and the PLC.
- Your program must have been loaded error-free into the PLC.

  **Note**

  The time stamps of your project in the Programming Tool and in the PLC must match, so that you can enable the status. The comparison of the time stamps (carried out automatically if you attempt to enable the status) ensures that the representation of the project status in the Programming Tool exactly reflects the status of the program in the PLC. If the time stamps do not match, you can compare the programs via the dialog box "Time Stamps Do Not Match". Look at the drop-down list box in order to find out which networks are different.

- After you have loaded the program into the PLC, then you should again bring this into the RUN mode. Otherwise, the address status is displayed, however, the PLC cannot execute the program, so you are not shown the logic operations that you expect.

### Operating state of the PLC

The type of monitoring and test functions that you can execute depends on the mode of your PLC.

Even if your program is not executed in the STOP mode, the operating system of the PLC still monitors the PLC (status of RAM and I/O) and transfers the data status to the Programming Tool. If the PLC is in the STOP mode, then you can execute the following functions:

- You can display the actual values of the operands as the end-of-scan status in the chart status, the data block status or the program status. (This is the same as the function "Single Read", as the program is not executed.)
- In the chart status and the data block status, you can write values.
- You can execute a certain number of scan cycles and display the effect in a status chart, in the data block status and/or in the program status.

If the PLC is in the RUN mode, you cannot execute the functions "First Scan" or "Multiple Scans". You can write values in a status chart, you can also execute the following functions (not in the STOP mode):

- In the chart status, you can carry out continuous updates. (If you wish to only execute one update, you must disable the chart status so that you can execute the command "Single Read".)
- In the data block status, you can carry out continuous updates.
- You can execute continuous updates in the program status.

## Communication and scan cycle

In a continuous scan cycle, the PLC reads the inputs, it executes the program, writes to the outputs, and executes system functions as well as communication. This scan cycle runs with an extremely high speed of many times per second. Even if the PLC Programming Tool issues status requests in a fast sequence, you cannot monitor each individual event that takes place in the PLC. When using the program status or the chart status, if you read data values from a PLC program, the data is scanned by taking samples (spot check). The update rate of the status values read from the PLC depends on the communication baud rate. Communicate with maximum baud rate, if you want to achieve the highest update rate.

## Different ways to update the status

### Chart status

Click the "Chart Status" button or select the menu command **Debug > Chart Status** to view continuous updates of the status when the PLC is in the RUN mode. Disable the chart status and use the function "Single Read" if you need to update the status just once and you do not want to switch the PLC into the STOP mode. If you switch the PLC into the STOP mode and enable the chart status, you receive a status update too. Furthermore, you can use the functions "Multiple Scans" and "First Scan" while you are viewing a status chart. In the chart status, the end-of-scan status is always used.

### Data Block Status

Click the "Data Block Status" button or select the menu command **Debug > Data block status** to view continuous updates of the current values of a data block when the PLC is in the RUN mode. If you switch the PLC into the STOP mode and enable the data block status, you receive a current value update too. Furthermore, you can use the functions "Multiple Scans" and "First Scan" while you are viewing a data block status. With the data block status, the actual values are always recorded at the end of a scan cycle.

**Program status**

Click the "Program Status" button or select the menu command **Debug > Program Status** to display the status of the data in the PLC in the program editor. The status data is recorded in the mode previously selected on the "LAD Status" tab (Tools > Options) (Page 492):

- Execution status (execution status activated)
  In the execution status in LAD, the value of operands and the signal flow for execution of the individual instructions during the execution of the program in the scan cycle are displayed. The execution status shows the values of the operands at the time of execution of the instruction; this may be overwritten again by subsequent program instructions. All of the displayed data values of the PLC are recorded in a single program scan cycle. The states of the local data memory and the accumulators are displayed. The status values are only updated and displayed if the PLC is in the RUN mode.

  **Note**

  The execution status may only be run as the program status once for each CPU at any one time. If you attempt to run the execution status in a different application, such as in a second Programming Tool, which is connected to the same PLC, an error will be displayed.

- End-of-scan status (execution status deactivated or CPU does not support execution status)
  The end-of-scan status shows the status results that are read at the end of the program scan cycle. These results may not indicate all changes to the values of the addresses of data in the PLC, because subsequent program instructions write a value and overwrite it again before the end of the program scan cycle is reached. The end-of-scan status shows the data values that are scanned at the end of multiple scan cycles. This results from the different speeds between the fast scan cycle of the PLC and the relatively slow transmission of the PC status data. The states of the local data memory and the accumulators are not displayed. If you switch the PLC into the STOP mode, you receive a status update too.

You can receive the status values of the program status continuously or as a snapshot.

- **Continuous:**
  Open the program editor window and enable the program status in order to view continuous updates of the program execution status when the PLC is in the RUN mode.

  **Note**

  "Continuous" should not be understood as "real time"; it means that the programming device scans the PLC for status information continuously and displays it on your screen, updating the display as quickly as your communication setup permits. Some rapidly fluctuating values may not be recorded and displayed on your screen. It is also possible that the values change too quickly for you to read them.

- **Snapshot:**
  If the PLC is in the STOP mode, you can use the menu command **Debug > Multiple Scans...** to display the status values of one or more scan cycles, or, with the menu command **Debug > First Scan**, to display the status values of the first scan cycle.

With the button "Pause Program Status" or the menu command Debug > Pause Program Status you can pause the status display of the program status. This can be necessary if values change too quickly for you to read them. In the execution status this can also make a single execution of a program section, e.g. a subprogram, visible. To do this, set the networks which are not being processed in the program editor and pause the status display with the

execution status activated. The next time the networks are executed in the program editor, the status display is updated. The status update then stops.

### Simulating process conditions

You can simulate process conditions by writing new values to operands: To do so, use the status chart.

### Checking cross references and the elements used

If you test your program, you may wish to supplement, delete or change the parameters.

In the window "Cross Reference", you can determine how the parameters are presently assigned in your program. This helps you to avoid assigning values twice.

### Loading changes in your program into the CPU

If you test a program with the program status and thus determine that you want to change the program, you must disable the program status before you can change the program. Once you have loaded the changed program into the CPU, you can enable the program status to check whether the changes are correct.

### See also

Displaying the status in the editors (GS 5.2) (Page 60)

Displaying the status in the status chart (GS 5.3) (Page 68)

Executing a certain number of scan cycles (GS 5.4) (Page 73)

Displaying the data block status (GS 5.5) (Page 74)

Downloading a program to the CPU (GS 4.3) (Page 52)

Cross-references and elements used (Page 311) (Ensure that program changes do not produce duplicate assignment.)

RUN/STOP modes of the PLC (Page 101)

## 3.5.2 Displaying the status in the program editor

### Program status in LAD

These help topics are explained in the following:

### Enabling the program status

After you have successfully established communication between the Programming Tool and a PLC and have loaded a program into the PLC, you can operate and test the networks in your program with the function "Program Status".

Set up the program editor in such a way that the section and the networks of the program that you want to test are displayed.

**Note**

If you load a program into the PLC and are using a CPU that does not support loading in RUN mode, you are prompted to bring the PLC into STOP mode. If you want to be able to view continuous updates of the program status, you must make sure that you switch the PLC back into the RUN mode.

## CPUs that support download in the RUN mode

828D
828D Step 2
808D-PPU14x
808D-PPU15x
808D-PPU16x

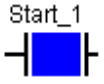
## Procedure

To enable the program status, proceed in one of the following ways:

- Select the menu command "Debug > Program Status".
- To test, click the "Program status"  button in the function bar.

The status data is recorded in the mode previously selected on the "LAD Status" tab (Tools > Options) (Page 492). See description below.

The program status is displayed in the program editor.




**Note**

If you enable the program status, many other functions in the PLC Programming Tool are deactivated. For instance, you cannot change your program unless you disable the program status again. Other functions, e.g. switching the display from one program editor to another, mean that the program status is automatically disabled. If you wish to display the status again, you must reselect the command "Program Status".

### Troubleshooting

If you have problems when enabling the program status, please consider the following prerequisites:

- You must have set up communication (so that you can load your program into the PLC).
- You must have selected the correct CPU version so that you can load your program into the PLC.

- Your program must be able to be compiled error-free.
- Your program must be able to be loaded error-free into the PLC.
- Your PLC must be in the RUN mode in order to be able to display continuous updates of the status. Otherwise, only changes at the inputs and outputs (if they are available) are displayed. As the program is not executed in the PLC, changes at the inputs and outputs do not have the same effects as you would expect on the program logic in the displayed program status.
- If you display another program area, which is not executed (e.g. an interrupt program or subprogram or an area which was skipped due to a jump instruction), the status is not displayed as the code is not scanned.

**Pause Program Status**

- Select the menu command **Debug > Pause Program Status**.
- To test, click the "Pause program status"  button in the function bar.

This can be necessary if values change too quickly for you to read them. In the execution status this can also make a single execution of a program section, e.g. a subprogram, visible. To do this, set the networks which are not being processed in the program editor and pause the status display with the execution status activated. The next time the networks are executed in the program editor, the status display is updated. The status update then stops.

## Displaying the program status

The program status can be executed for CPUs in the "execution status" mode and for all CPUs in the "end-of-scan status" mode. The selection is made on the "LAD Status" tab (Tools > Options) (Page 492). If the execution status is activated for a CPU that does not support this status, the end-of-scan status is automatically executed.

**Note**

If you display the program status in LAD, the Boolean operations (contacts, coils) are displayed as colored blocks if the value of the address is 1 (i.e. the bit is enabled). The actual data value from other addresses is displayed next to the address, or instead of the address. The display is updated when changes are read from the PLC. The value of non-Boolean addresses is displayed and updated as quickly as the communication permits it.

## CPUs that support the execution status

828D (as of PLC 05.04)
828D Step 2

808D-PPU14x
808D-PPU15x
808D-PPU16x

## Execution status

### Execution status (execution status activated)

In the execution status in LAD, the value of operands and the signal flow for execution of the individual instructions during the execution of the program in the scan cycle are displayed. The execution status shows the values of the operands at the time of execution of the instruction; this may be overwritten again by subsequent program instructions. All of the displayed data values of the PLC are recorded in a single program scan cycle. The statuses of the local data memory and the accumulators are displayed. The status values are only updated and displayed if the PLC is in the RUN mode.

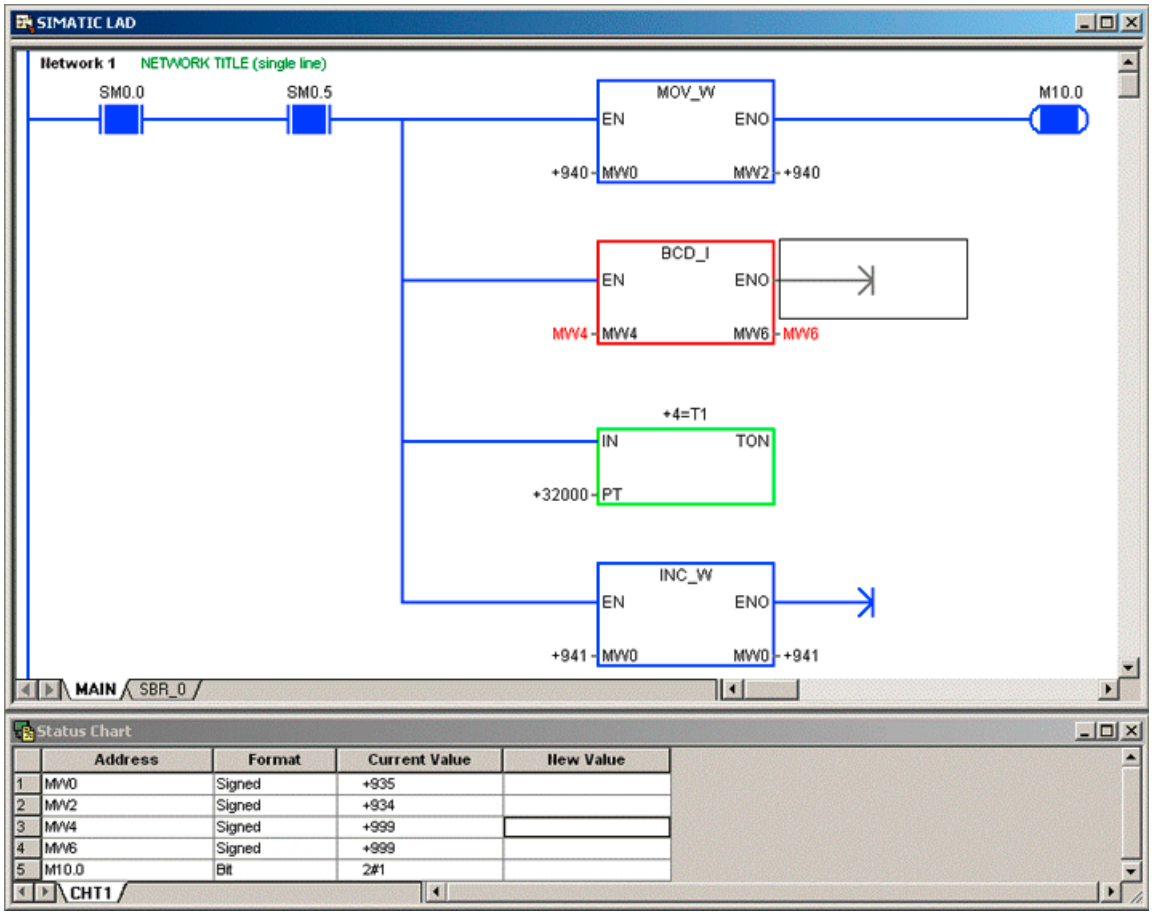
### Colors for execution status:

- The busbar is marked with a color if the program is being scanned, default setting: blue.
- The signal flow in the diagrams is marked with a color, default setting: blue.
- Contacts: The instruction is marked with a color if the contact is switched on, default setting: blue.
- Coils: The instruction is marked with a color if the output is switched on, default setting: blue.
- Boxes and subprograms: Boxes and subprograms are marked in color if the instruction is enabled and is being executed without error, default setting: blue.
- Timers and counters: Timers and counters are marked in color if they have valid data, default setting: green.
- For jump and jump mark instructions the signal flow is marked in color if it is active. If it is not active, this is color-coded differently, default setting: gray.
- Red (red is the default setting) indicates that an instruction was not executed error-free.
- Gray (gray is the default setting) indicates that no signal flow is available, that the instruction is not being scanned (it is skipped or not called) or that the PLC is in the STOP mode.

You can also set your own colors on the "LAD Status" tab (Tools > Options) (Page 492).

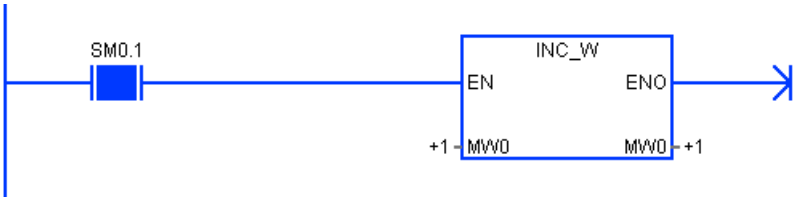
### Example for the execution status

The following example uses the execution status in the program editor in LAD. The status charts only use the end-of-scan status data acquisition. Some differences in the displayed values of rapidly changing data can be seen if the LAD diagram and the corresponding status chart are compared. These differences are created by the various types of acquisition of the status data.




**Note**

If the same operand is used as the input and output parameter in an instruction, the value of this operand can only be determined by the execution status at one time. In this case, it is the value after execution of the instruction. The value after execution of the instruction is also displayed for the input parameter and not the actual input value.




**End-of-scan status (execution status deactivated or CPU does not support execution status)**

The end-of-scan status shows the status results that are read at the end of the program scan cycle. These results may not indicate all changes to the values of the addresses of data in the PLC, because subsequent program instructions write a value and overwrite it again before the end of the program scan cycle is reached. The end-of-scan status shows the data values that are scanned at the end of multiple scan cycles. This results from the different speeds

between the fast scan cycle of the PLC and the relatively slow transmission of the PC status data. The statuses of the local data memory and the accumulators are not displayed. If you switch the PLC into the STOP mode, you receive a status update too.

**Colors for end-of-scan status:**

- Contacts and coils: Contacts and coils are marked in color if they are live or logically true, default setting: blue (as signal flow).

You can also set your own colors on the "LAD Status" tab (Tools > Options) (Page 492).

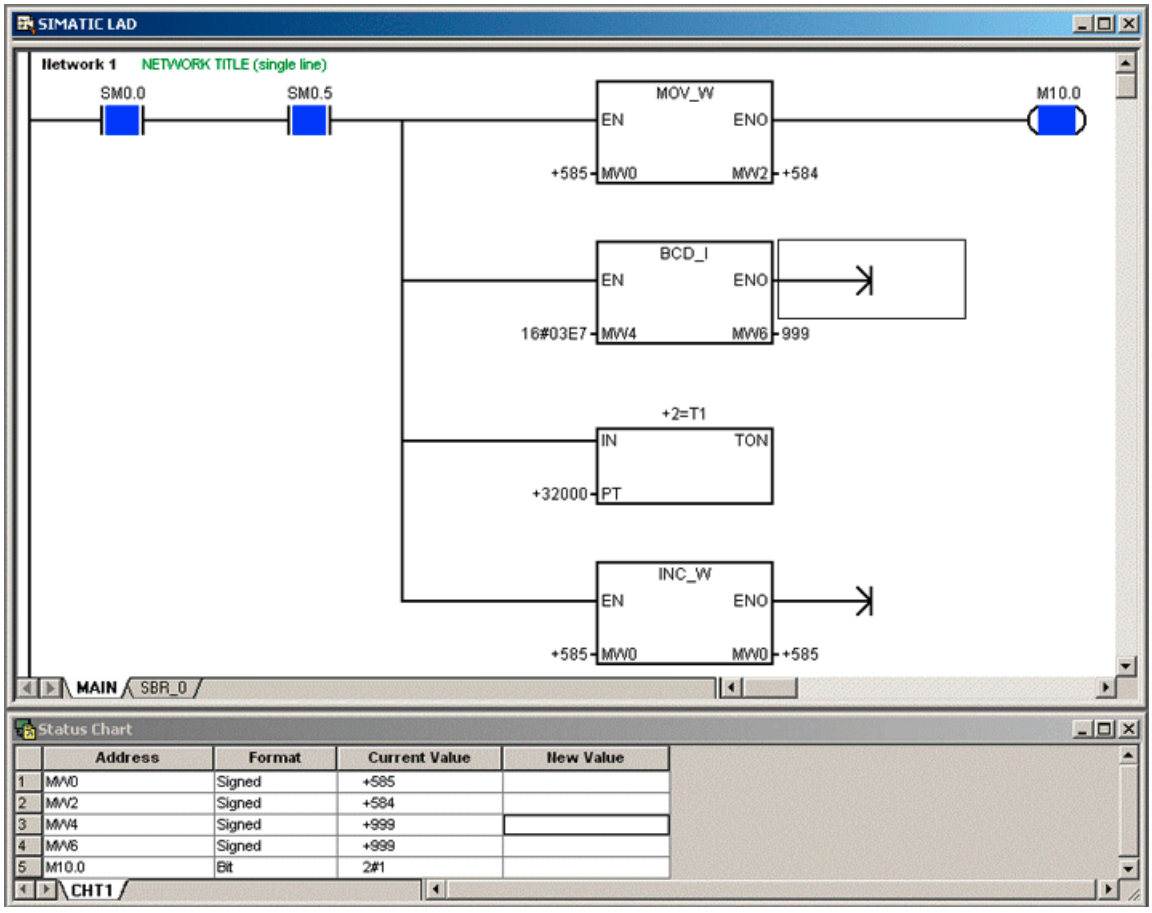
**Note**

The color marking for "signal flow" does not mean that there is always a signal flow. Since only the values from the end of the scan cycle are displayed, it can sometimes be difficult to evaluate just what the "signal flow" representation really means. Boolean contacts and coils are marked in color in the program status in LAD corresponding to the value of the bit operand. If the bit value is 1 (bit is enabled), then the instruction is marked in color. However, this does not necessarily mean that the instruction was actually executed. There are multiple conditions that can result in an ambiguous signal flow display:

- If, when evaluating the status, the PLC is in the STOP mode, contacts can be activated, however, coils and boxes are not switched on because the program is not being executed.
- If the program contains a jump instruction, then it is possible that the networks that you are investigating do not display the expected results because the PLC skipped these instructions while executing the program.
- It is a similar situation if you consider a subprogram. The Boolean operands can be activated, however, the subprogram logic can only be executed if the subprogram is activated. If the subprogram was not called from the main program, then no logic of the networks was executed regardless of what the bit values of the instructions display.

**Example for the end-of-scan status**

The following example uses the end-of-scan status in the program editor in LAD. Because both windows (Program Status and Chart Status) use the same data of the end-of-scan status, the two windows show the same data values of the PLC. This can be misleading because a program can assign the same address many values before the status is recorded at the end of the scan cycle. Temporary values from intermediate calculations are not displayed. In this example, the box BCD_I was not executed error-free, because the input does not contain a valid BCD value. In contrast to the example for the execution status, in which the box is displayed in red, you must read the status values in order to determine that the BCD value is invalid.




## Restrictions regarding the program status

The advantage of displaying the status in the program editor is due to the fact that you obtain a graphic display of the events in your program. However, not all of the same tools are available as in the status chart (e.g. the functions "Single Read", "Write All").

It is important that you know the restrictions of the program status when you are testing and monitoring your program.

## Adjusting the display of the program status

Select the menu command **Tools > Options** and open the "LAD Status" tab in order to edit the following settings:

| | |
|---|---|
| Zoom factor | To edit the scaling<br>**Shortcut key:**<br>You can use the shortcut key to quickly set the zoom factor in the program status. Press the Ctrl key and the plus key in the numerical keypad of the keyboard to increase the display size. Press the Ctrl key and the minus key in the numerical keypad of the keyboard to reduce the display size. |
| Field width and height | To edit the grid settings<br>You can increase the field width so that information can be displayed which otherwise would be cut off.<br>You can reduce the field height so that there is space to show more networks on the screen. |
| Colors | You can change the colors for the various status categories. For the end-of-scan status, only the color for the signal flow is relevant. |
| Operand display | You can display the operands either within or outside the instructions. You can also display the status value without changing the name or the address of the operand. |
| Execution status | You can activate the execution status. |

**Note**

If the execution status is activated for a CPU that does not support this status, the end-of-scan status is automatically executed.

In addition, you can rearrange other windows in the Programming Tool in order to create more space for the window of the program editor (or to display it simultaneously with another window such as the status chart, the symbol table or the cross references). With the commands from the menus "View" and "Window", you can enlarge or reduce the windows with your mouse and move them to different positions until all windows are arranged according to your needs.

## See also

Overview of test and monitoring functions (GS 5.1) (Page 55)

Displaying the status in the status chart (GS 5.3) (Page 68)

Executing a certain number of scan cycles (GS 5.4) (Page 73)

Displaying the data block status (GS 5.5) (Page 74)

Downloading a program to the CPU (GS 4.3) (Page 52)

Cross-references and elements used (Page 311) (Ensure that program changes do not produce duplicate assignment.)

RUN/STOP modes of the PLC (Page 101)

## 3.5.3 Displaying the status in a status chart

### Precondition

After you have loaded your program into the PLC, you can generate one or multiple status charts to monitor and test program execution.

The program is continuously executed if the PLC is in the RUN mode. You can enable the chart status so that the status values in the chart are continuously updated (not interrupted). As an alternative, using the function "Single Read", you can generate a "snapshot" of the status values in the chart without having to enable the status chart.

While you are viewing a status chart, you can also switch the PLC into the STOP mode and only execute the first or a certain number of scan cycles, in which you monitor program execution.

**Note**

Please note that you cannot change your chart if the chart status is enabled.

Disable the chart status if you wish to edit the chart.

These help topics are explained in the following:

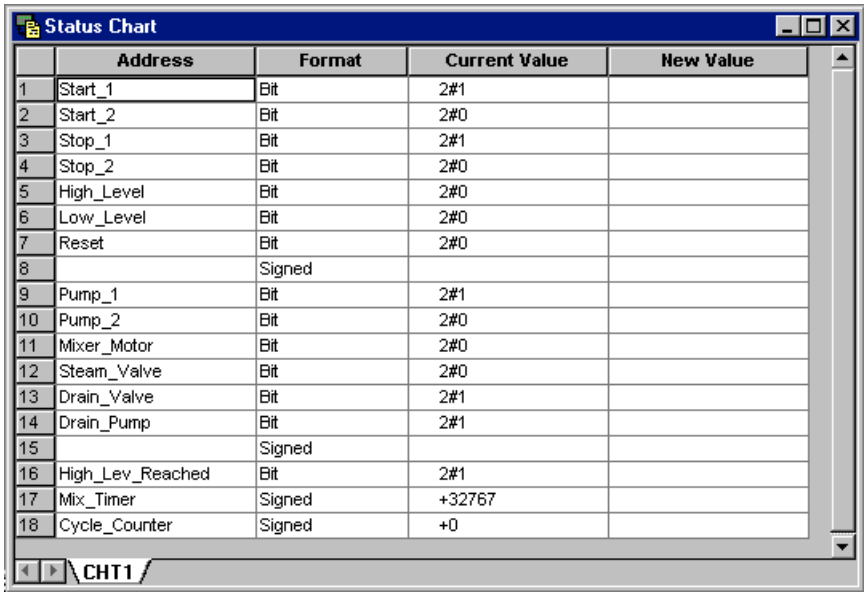
### Example for the status chart

Opening a status chart is not the same as enabling a status chart. You can open and evaluate or change a status chart: However, if you do not execute the command "Single Read" (in the menu "Debug" or in the toolbar) or enable the chart status (in the menu "Debug" or in the toolbar), no status information will be displayed in the column "Actual Value".

If you use the function "Single Read" (this is only available when the chart status is disabled) to evaluate a status chart, the actual values of the PLC are accepted and displayed in the column "Actual Value". However, the values are not updated while the PLC is executing the program.

If you enable the chart status (in the menu "Debug" or in the toolbar), the actual values of the PLC are regularly updated. If changes are received from the PLC, then the column "Actual Value" is updated.

You can assign (write) certain values in the PLC using the column "New Value".




## Opening or enabling a status chart

Open a status chart to evaluate or change the contents of the chart. Enable the status chart so that the status information can be updated.

To open a status chart, proceed in one of the following ways:

- Click on the button of the status chart in the "Navigation" toolbar (Page 500)
- Select the menu command **View > Status Chart**.
- Open the directory of the status chart in the project tree (Page 447) and double-click the symbol of a table.
- If your project includes more than one status chart, using the tab for the CHT1 CHT2 CHT3 status charts at the lower edge of the window, you can switch over between the individual charts.

To enable a status chart, proceed in one of the following ways:

- If you wish to continually update the status information in the status chart, enable the chart status: Select the menu command **Debug > Chart Status** or click the appropriate button in the toolbar.
- If you only require a "snapshot" of the values, execute the function "Single Read": Select the menu command **Debug > Single Read** or click the appropriate button in the toolbar (if the chart status is enabled, then the function "Single Read" is deactivated).

**Tips:**

If you open a status chart, then the status is still not displayed. You must enable the status chart so that the status information is updated.

Enabling the status chart will have no effect if the chart is still empty: You must first create your status chart by entering program values (operands) in the column "Address" and entering a data type in the column "Format" for each address.

## Working with multiple status charts

You can create additional status charts in several ways:

- In the instruction tree, right-click the "Status Chart" folder and, in the shortcut menu, select the command **Insert Status Chart**.
- Open the window "Status Chart" and call the menu "Edit" or right-click and select the command **Insert Contents > Chart**.

**Note**

- After you have inserted a new status chart, a new tab 

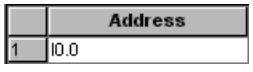
is displayed at the lower edge of the "Status Chart" window. If you wish to switch between the status charts, simply click the tab of the required status chart.
- Sometimes, a tab is hidden by the buttons on the right-hand side that are used to scroll. If a tab is not visible, drag the demarcation line between the tab area and the scroll buttons to display additional tabs.

## Creating a status chart

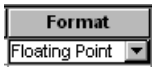
You can enter addresses in a status chart in order to monitor and control values from your program. Values of timers and counters can be displayed as bits or words. If you wish to display the value of a timer or a counter as a bit, then the status of the output is displayed (on or off). If you wish to display the value of a timer or a counter as a word, then the actual value is used.

To create a status chart, proceed as follows:

1. In the address field you enter the addresses of the desired values.
   Most of the memory types listed in the memory areas of the automation system (Page 262) are valid, with the exception of data constants and accumulators.
   To edit an address field, select the desired field using the cursor keys or the mouse. If you enter data, existing data are deleted and the new characters are entered. The field is selected if you double click with the mouse or press key "F2". You can then move the cursor using the cursor keys to the position that you wish to edit.


2. If the element involves a bit (e.g. I, Q or M), then the bit format is displayed in the second column. If the element involves a byte, word or double word, select the field in the column "Format" and double-click or press the space bar or ENTER in order to scroll through the valid formats until the correct format is displayed.


To insert a new row, open the menu "Edit" or right-click a field in the status chart so that the shortcut menu is displayed and select the command **Insert Contents > Row**. A new row is then inserted in the status chart above the cursor position. You can also move the cursor to a field in the last row and press the down arrow key in order to insert a row at the bottom of the status chart.

**Tips:**

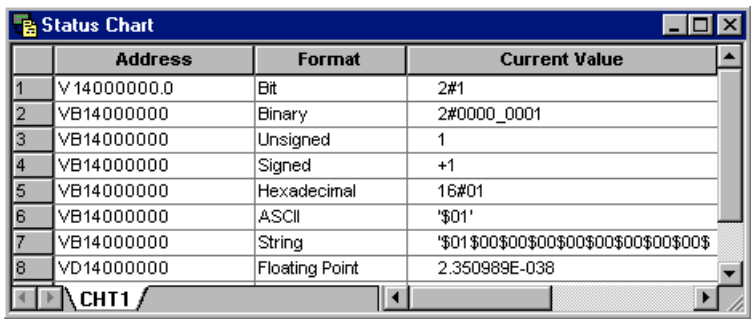
You can select addresses in the symbol table and copy these into the status chart in order to generate your chart faster.
You can display the status a multiple number of times. You can classify the elements in logical groups to display each group in an individual chart. In this way, you avoid having to scroll through extremely long lists.

## Shortcut key combinations for editing

- To insert a new row with the following address and the same data format, select an address field and press the Enter key.
- To scroll through the possible data formats for a specific address, select the field "Data Format" and press the Enter key several times. To display all available data formats, open the drop-down list box.
- Use the TAB key to jump into the next field of the chart.
- To set the width of a column, position the mouse cursor on the edge of a column until the appearance of the cursor changes and then drag to increase or decrease the width of the column.
- To select a complete row (to cut or copy), click once the number of the row.
- To insert a new row, select a field or a row and right-click. In the shortcut menu select the command **Insert Contents > Row**. The row is inserted at the cursor position. The subsequent rows are shifted down by one row.
- To insert a row at the bottom of the status chart, move the cursor to a field in the last row and press the down arrow key.
- To delete a field or a row, select the field or the row and right-click. Select the menu command **Delete > Selection**. If you delete a row, the subsequent rows are shifted up by one row.
- To select the entire status chart, click once the upper left corner above the row numbers.

## Data formats

The data format that you assign to a value defines how the value is represented in the status chart.
In the example below, VD14000000 (and therefore VB14000000 and V14000000.0) contains the value 1.



**Tips:**

Bit and binary values are both introduced by the number 2 and the # symbol. Hexadecimal values are introduced by the number 16 and the # symbol. Bit values have one digit. Binary values have eight digits.
Signed and unsigned values use the basis 10 (decimal).

## Single read or continuous chart status

- If you only require a "snapshot" of the values, execute the function "Single Read": Select the menu command Debug > Single Read or click the appropriate button in the toolbar  (if the chart status is enabled, then the function "Single Read" is deactivated).
- If you wish to continually update the status information in the status chart, enable the chart status: Select the menu command Debug > Chart Status or click the appropriate button in the toolbar  .

## Working with test functions in the status chart

You access the test functions (Single Read, Write All) using the menu Debug or using the toolbar with the test functions.

| | |
|---|---|
|  | **Single Read**<br>Use Single Read if you require a "snapshot", i.e. a single update of the program status of all values. By default, the chart status continually scans the PLC for status updates. If you click a status chart and the chart status is disabled, then the button for Single Read is activated. |
|  | **Write All**<br>After you have entered the values in the column "New Value" in the status chart, write the required changes to the PLC using the command "Write All". |

## See also

Overview of test and monitoring functions (GS 5.1) (Page 55)

Displaying the status in the editors (GS 5.2) (Page 60)

Executing a certain number of scan cycles (GS 5.4) (Page 73)

Displaying the data block status (GS 5.5) (Page 74)

Downloading a program to the CPU (GS 4.3) (Page 52)

Cross-references and elements used (Page 311) (Ensure that program changes do not produce duplicate assignment.)

RUN/STOP modes of the PLC (Page 101)

## 3.5.4 Execute a certain number of scan cycles

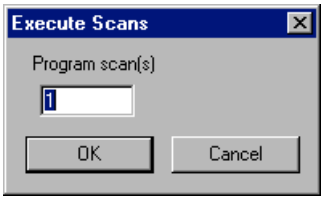
You can specify that the PLC should process a certain number of scan cycles of your program (from one scan cycle up to 65,535 scan cycles). If you specify that the PLC should execute a certain number of scan cycles, you can monitor the processing of the process variables.
In the first scan cycle, the value of SM0.1 = 1 (ON).

**Executing a single scan cycle:**

1. The PLC must be in the STOP mode. If it is not already in STOP, switch the PLC into the STOP mode (Page 457).
2. Select the menu command **Debug > First Scan**.

**Executing multiple scan cycles:**

1. The PLC must be in the STOP mode. If it is not already in STOP, switch the PLC into the STOP mode (Page 457).
2. To execute multiple scan cycles, select the menu command **Debug > Multiple Scans...**. This opens the dialog box "Execute Scans".




3. Specify how many scan cycles should be executed and confirm with "OK".

**Note**

Make sure that you switch the PLC back into the RUN mode (this requires that you click the RUN button in the toolbar or select the menu command **PLC > RUN**) if you wish to return to normal program processing.

**See also:**

Overview of test and monitoring functions (GS 5.1) (Page 55)

Displaying the status in the editors (GS 5.2) (Page 60)

Displaying the status in the status chart (GS 5.3) (Page 68)

Displaying the data block status (GS 5.5) (Page 74)

Downloading a program to the CPU (GS 4.3) (Page 52)

Cross-references and elements used (Page 311) (Ensure that program changes do not produce duplicate assignment.)

RUN/STOP modes of the PLC (Page 101)

## 3.5.5 Display the data block status

### Precondition

If the CPU that you are using supports data blocks, you can, after you have loaded your program into the PLC, display the current values of the data blocks in the PLC.

### CPUs that support data blocks


828D
828D Step 2

808D-PPU14x (only special data blocks)
808D-PPU15x
808D-PPU16x

### Procedure

The program is continuously executed if the PLC is in the RUN mode. You can enable the data block status so that the actual values in the data block are continuously updated (not interrupted).

While you are viewing a data block chart, you can also switch the PLC into the STOP mode and only execute the first or a certain number of scan cycles, in which you monitor program execution. The actual values of the data block are then continuously updated.

**Note**

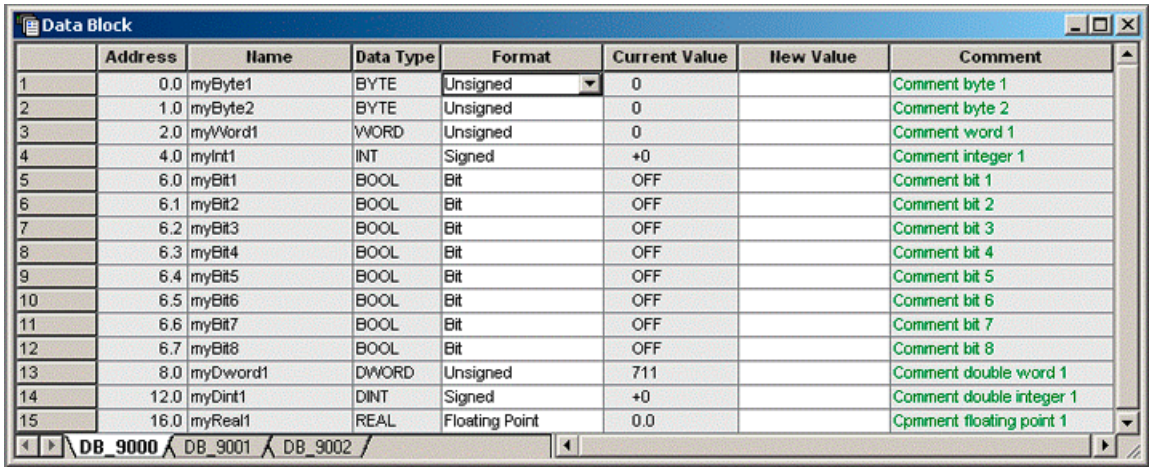
Please note that you cannot change your data block if the status is enabled.

Disable the status if you wish to edit the data block.

### Example for the data block status

If you enable the data block status (in the menu "Debug" or in the toolbar), the values in the column "Actual Value" are regularly updated with the actual values of the PLC.

You can assign (write) certain values in the PLC using the column "New Value".


Data Block

| | Address | Name | Data Type | Format | Current Value | New Value | Comment |
|---|---|---|---|---|---|---|---|
| 1 | 0.0 | myByte1 | BYTE | Unsigned | 0 | | Comment byte 1 |
| 2 | 1.0 | myByte2 | BYTE | Unsigned | 0 | | Comment byte 2 |
| 3 | 2.0 | myWord1 | WORD | Unsigned | 0 | | Comment word 1 |
| 4 | 4.0 | myInt1 | INT | Signed | +0 | | Comment integer 1 |
| 5 | 6.0 | myBit1 | BOOL | Bit | OFF | | Comment bit 1 |
| 6 | 6.1 | myBit2 | BOOL | Bit | OFF | | Comment bit 2 |
| 7 | 6.2 | myBit3 | BOOL | Bit | OFF | | Comment bit 3 |
| 8 | 6.3 | myBit4 | BOOL | Bit | OFF | | Comment bit 4 |
| 9 | 6.4 | myBit5 | BOOL | Bit | OFF | | Comment bit 5 |
| 10 | 6.5 | myBit6 | BOOL | Bit | OFF | | Comment bit 6 |
| 11 | 6.6 | myBit7 | BOOL | Bit | OFF | | Comment bit 7 |
| 12 | 6.7 | myBit8 | BOOL | Bit | OFF | | Comment bit 8 |
| 13 | 8.0 | myDword1 | DWORD | Unsigned | 711 | | Comment double word 1 |
| 14 | 12.0 | myDint1 | DINT | Signed | +0 | | Comment double integer 1 |
| 15 | 16.0 | myReal1 | REAL | Floating Point | 0.0 | | Comment floating point 1 |

DB_9000 DB_9001 DB_9002

## Enabling the data block status

Open a data block and enable the data block status so that the actual values from the PLC are displayed. If you enable the data block status without having opened a data block, the first data block from the data block list is opened automatically.

To enable the data block status, proceed in one of the following ways:

- Click the "Data block status" button in the toolbar.
- Select the menu command **Debug > Data Block Status**.

If your project includes more than one data block, if the data block status is active then you can switch between the individual data blocks. To do this, proceed in one of the following ways:

- Switch between the individual data blocks using the tab for the data blocks DB_9000 DB_9001 DB_9002 at the lower edge of the window.
- In the project tree, double-click the desired data block.

**Note**

- When the data block status is activated, the structure of the variables (name, data type, initial value, and comment) is write-protected. The format of the variables can be changed.
- The structure of the data block in the project must match the structure of the data block in the PLC. If you have modified the structure of the data block, then the complete project has to be loaded again.

## Changing the actual values in the data block status

If the data block is not write-protected, you can change the actual values in the PLC with the data block status active. To do so, enter the desired values in the column "New Value" and write the changes to the PLC with the command "Write All".

**See also**

Overview of test and monitoring functions (GS 5.1) (Page 55)

Displaying the status in the editors (GS 5.2) (Page 60)

Displaying the status in the status chart (GS 5.3) (Page 68)

Executing a certain number of scan cycles (GS 5.4) (Page 73)

Saving data in the variable memory with the help of a data block (Page 424)

Downloading a program to the CPU (GS 4.3) (Page 52)

Cross-references and elements used (Page 311) (Ensure that program changes do not produce duplicate assignment.)

RUN/STOP modes of the PLC (Page 101)

# 3.6 Chapter 6 Managing projects

## 3.6.1 Print

**Preview**

You can view your project before printing it. To do this, select the menu command **File > Print Preview**.

In the dialog box "Print Preview" you have the following options:

- You can have the program displayed page by page, or two pages at the same time.
- You can make particular program organization unit settings in order to display a different part of the program.
- You can move to the next page and you can also return to the previous page.
- You can zoom the program display in or out for the print preview (zoom functions).

**Essential settings for printing**

Select the menu command **File > Print** to open the dialog box "Print".

In the dialog box "Print" you have the following options:

- You can select the printer.
- You can print in color or monochrome.
- You can print a network area or rows from a project component.
- You can print several project components.

**Print options**

In the dialog box "Print", click the "Options" 

button to open the dialog box "Print Options".

In the dialog box "Print Options" you have the following options:

- You can print the properties for your project.
- You can print the local variable table.
- You can print your data blocks.
- You can set the number of columns (width of the printout) for an LAD project.
- You can print the row numbers in an STL project.

**Page setup settings**

Select the menu command **File > Page Setup** or click the "Page Setup" 

button in the dialog box "Print" to open the dialog box "Page Setup".

In the dialog box "Page Setup" you have the following options:

- You can set the left, right, top, and bottom edges of the printout.
- You can set the paper orientation (portrait or landscape).
- You can set the format of the paper.
- You can set the paper feed.

**Header and footer settings**

In the dialog box "Page Setup", click the button 

to open the dialog box "Header/Footer".

In the dialog box "Header/Footer" you can arrange the following elements in the header or footer:

- Name of the project
- Name of the object (e.g. POU, symbol table, status chart or data block)
- Date
- Time
- No. of pages
- User-specific text

You can also specify whether the elements in the header and footer are to be arranged left- or right-justified or centered. You can create a comment box for project footers.

**See also:**

Print preview (Page 393)

Print (Page 394)

## 3.6.2 Moving, copying, renaming and sending

The projects are saved in the Programming Tool in a single file with the extension ".ptp". This enables you to quite easily move, copy or rename the projects in Windows Explorer, and also send them by email.

You can also rename a project or the project folder in the dialog box "Save as (Page 257)".

**See also:**

Copying program segments from one program to another (Page 80)

Importing files (Page 380)

Exporting files (Page 381)

## 3.6.3 Download from a target system

With the help of buttons in the toolbar or the menu **File > Upload**, you can load your program from the PLC onto the computer on which the Programming Tool is installed.

**Loading the blocks**

You can load program blocks (OB1, interrupt programs and subprograms) and data blocks from the PLC. The PLC contains no information for the symbol table or the status chart. You can therefore load no symbol table or status chart from the PLC. (See also: Project components and their functions (Page 29).)

**Loading into a new, empty project**

This is a risk-free way of recording the information for the program block or the data blocks. Since the project is empty, data cannot be destroyed accidentally. If you also want to include information for the status chart or the symbol table, which has been created for this project, you can open the Programming Tool for a second time and copy the corresponding information from the other project file. (See also: Copying program segments from one program to another (Page 80).)

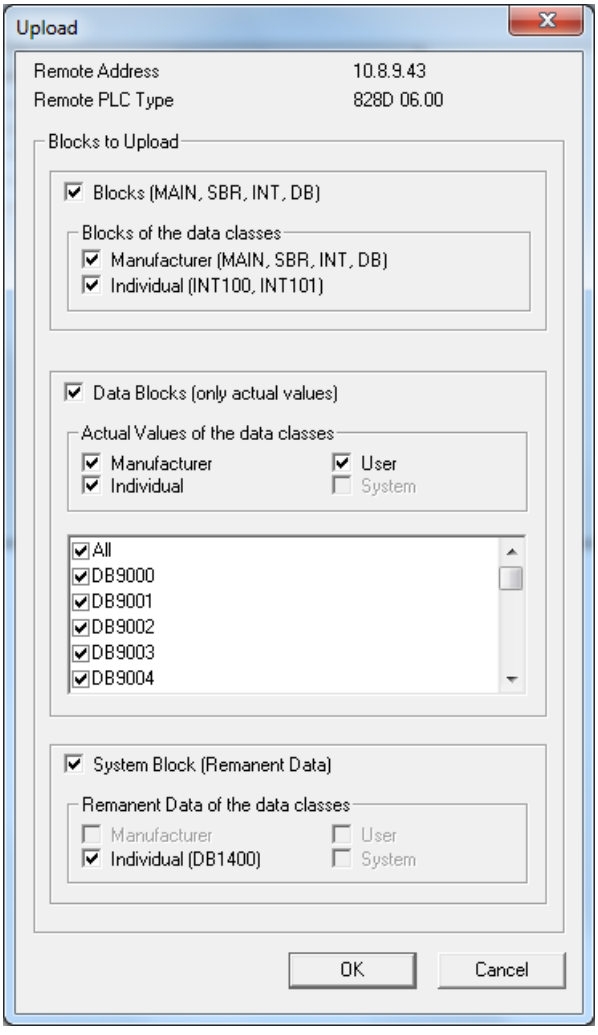
**Loading into an existing project**

This method is suitable if you want to overwrite all changes that have been made to the program since the last load procedure into the PLC.

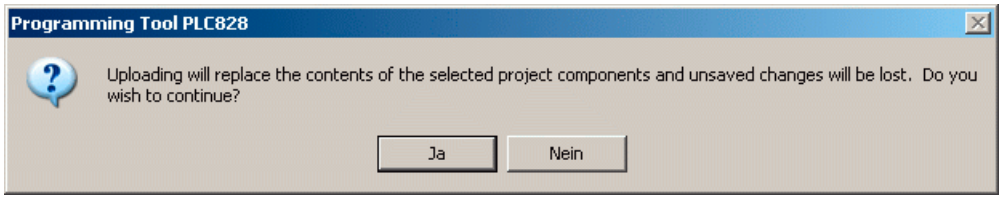
The method is not suitable if you want to save some changes that you have made to the program block and/or data block after loading into the PLC, because loading from the PLC will overwrite the blocks.

**Procedure:**

1. Open the project in the Programming Tool which contains the blocks that you want to load from the PLC.
   - If you want to load into an empty project, select the menu command **File > New** or click the button in the toolbar .
   - If you want to load into an existing project, select the menu command **File > Open** or click the button in the toolbar .
2. Start the loading procedure with the menu command **File > Upload from CPU** or click the button in the toolbar . The dialog box "Upload" contains checkboxes for program blocks and data blocks (Page 424) and their data classes (Page 243). Activate the checkboxes of the blocks you want to load from the PLC and confirm with "OK".




3. The Programming Tool displays the following warning:




Click the "Yes" button.

The Programming Tool displays a message if the blocks are successfully loaded from the PLC onto your PC.

**Note**

**Saving retentive data**

The user-specific retentive data must be selected to be able to save them in the user project.

You can find information on this in the Chapter "Saving retentive data" (Page 345).

**See also:**

Uploading from the PLC (File > Upload from CPU) (Page 382)

Communications overview (GS 4.1) (Page 50)

Testing the communications network (GS 4.2) (Page 51)

Downloading a program to the CPU (GS 4.3) (Page 52)

Downloading to the PLC (File > Download to CPU) (Page 382)

## 3.6.4 Copying project segments

### Getting started

Use the commands from the menu "Edit" and the mouse or the standard Microsoft keyboard shortcuts, such as `as Ctrl + A` to select all, `Ctrl + C` to copy, `Ctrl + X` to cut and `Ctrl + V` to paste.

**Note**

If you paste elements, these are inserted into the row or the network above the current position of the cursor.

These help topics are explained in the following:

### Copying a range of networks or a program organization unit

- You can copy a range of networks or a whole program organization unit in the LAD program editor.
- You can paste the selected elements into a different program organization unit of your program, simply by opening the appropriate tab at the bottom edge of the program editor.
- You can also paste the selected elements into a different project. (See also below: "Opening the Programming Tool twice in order to copy elements from one project to another one")

**Note**

The network area that you select for cutting/copying/pasting must consist of linked networks: The networks must all be next to each other. If you want to copy several networks or network areas from different parts of a program organization unit, you must repeat the commands to cut/copy and paste for each network area.

### Copying data from the data block

- You can copy data from a data block and paste it into a different data block. If the data violates the syntax rules, however, the data block editor will mark the appropriate points.
- You can copy all the columns with data from a data block and paste them into a Microsoft Excel spreadsheet. You can also copy individual columns with data from Excel and paste them into a data block. If the data violates the syntax rules, however, the data block editor will mark this.

### Local variable table

Due to the write-protected assignments in the local data memory, which must be unique in every table, no linked data area may be copied from one local variable table to another.

### Opening the Programming Tool twice in order to copy elements from one project to another

This is done as follows:

1. Open the Programming Tool twice. To do this, double-click on the Programming Tool icon:

   Or, in the Windows start menu, select the command **PLC Programming Tool > PLC Programming Tool**.
2. Repeat this step.
3. In the first Programming Tool, select the menu command **File > Open** and open the project from which you want to copy.
4. Select what you want to copy and press Ctrl + C or select the menu command **Edit > Copy**.

5. In the second Programming Tool, select the menu command **File > Open** and open the project into which you want to paste.
6. Position the cursor at the desired position and press Ctrl + V or select the menu command **Edit > Paste**.

**Note**

Save your changes in this project before you close it again.

### Opening the Programming Tool once in order to copy elements from one project to another

This is done as follows:

1. Use the menu command **File > Open** to open the first project from which you want to copy.
2. Select what you want to copy and press Ctrl + C or select the menu command **Edit > Copy**.
3. Close the first project with the menu command **File > Close**.
4. Use the menu command **File > Open** to open the second project into which you want to paste.
5. Position the cursor at the desired position and press Ctrl + V or select the menu command **Edit > Paste**.

**Note**

Save your changes in this project before you close it again.

## 3.6.5 Working with projects from earlier versions

If you want to work with a project, which was created with an earlier version of the Programming Tool, simply click the button for opening  or select the menu command **File > Open** and select the desired project.

## 3.6.6 Version identification

You can save a version identification in the PLC user project. The version identification is displayed at the control in the user interface under "Diagnostics" > "Version" for PLC user programs or supplementary PLC programs. The version identification can be entered at two different locations:

### 1. Information in the project file name (*.ptp file)

You can enter the project version at any position in the file name. The version itself is not evaluated. Only the character string that was entered is accepted. The following formats are identified:

- xx.yy
- xx.yy.zz
- ww.xx.yy.zz
- x.y
- x.y.z
- w.x.y.z

**Examples:**

- "Testproject_Version_01.02.03.ptp"
- "1.2_Testproject.ptp"

### 2. Information in the comment for "MAIN (OB1)"

A project version can be entered in the comment for the program block "MAIN (OB1)". The characters up to the first line break are evaluated; however, a maximum of the first 68 characters of the comment. The project version can be inserted at any position within this area. The version itself is not evaluated. Only the character string that was entered is accepted. The following formats are identified:

- xx.yy
- xx.yy.zz
- ww.xx.yy.zz
- x.y
- x.y.z
- w.x.y.z

**Examples:**

- "Comment in the OB1 test project version 01.02.03"
- "1.2 comment in the OB1 for test project"
- "Version 01.02 test project comment OB1"

In addition to the version identification, a date can also be entered into the comment, which is also displayed at the control. The date itself is not evaluated. Only the character string that was entered is accepted. A combination with the version is possible. The following formats are identified:

- xx/xx/xxxx
- xxxx/xx/xx
- xx/xx/xx

**Examples:**

- "Comment in the OB1 test project version 01.02.03 01/01/2015"
- "1.2 2015/01/01 comment in the OB1 for test project"
- "01/01/15 version 01.02 test project comment OB1"

**Note**

The date that the PLC user project was last changed is displayed at the control, when no date is entered in the comment.

**Note**

When version identifications are saved in the file name and also in the comment, then the contents of the file name are used for the display at the control.

# Setting up communication

# 4

## 4.1 Options for communication

In order to establish a connection between the control and PG/PC, depending on the installed hardware, you have the following possibilities:

- PLC802 (PPI) interface / RS232 cable (V24) (Page 88)

| | | |
|---|---|---|
| 808D-PPU14x | 802Dsl TM value | SINUMERIK 802D |
| 808D-PPU15x | 802Dsl TM plus | SINUMERIK 802SC |
| | 802Dsl TM pro | |
| | 802Dsl CU pro | |
| | 802Dsl GN plus | |
| | 802Dsl GN pro | |

CPUs that support PLC802(PPI) interfaces
For a simple cable connection between the PC and CPU, you can use the default parameters of the Programming Tool. No additional adaptation is required.

- Network (Ethernet) (Page 90)

| | | |
|---|---|---|
| 828D | 808D-PPU15x | 802Dsl TM pro |
| 828D Step 2 | 808D-PPU16x | 802Dsl CU pro |
| | | 802Dsl GN pro |

CPUs that support a network
You must adapt the communication settings of your control and in the Programming Tool when establishing a communication connection through a network.

- Establishing a connection using a modem connection (Page 93)

| | |
|---|---|
| 802Dsl TM value | SINUMERIK 802D |
| 802Dsl TM plus | |
| 802Dsl TM pro | |
| 802Dsl CU pro | |
| 802Dsl GN plus | |
| 802Dsl GN pro | |

CPUs that support a modem
In order to communicate with the CPU via a modem connection, you must change the settings in the Programming Tool.

You can set up the communication or you can edit the communication settings at any time (as long as you are in the LAD editor).

## Setting up the communication configuration

The table shows the possible hardware configurations and the baud rates, which the Programming Tool supports.

Table 4-1 Hardware supported by the Programming Tool - Configurations

| Hardware supported | Type | Baud rate supported | Comments |
|---|---|---|---|
| RS232 cable (V24) | Cable connection to communication interface of the PC | 9600 baud<br>19,200 baud<br>38,400 baud<br>57,600 baud<br>115,200 baud | Supports PPI protocol |
| Ethernet | | | |
| Modem | Modem port on communication interface of the PC | 9600 baud<br>19,200 baud<br>38,400 baud<br>57,600 baud<br>115,200 baud | Supports PPI protocol |

### Communication via the PLC802(PPI) interface/RS232 cable (V24)

The Programming Tool supports establishing a connection to the CPU via a PLC802(PPI) interface. The connection is activated on the operator panel of the CPU in the System operating area via the softkeys. The active or inactive state is kept even after Power On (except on run-up with the default data).
You set up the interface card and the protocol in the Programming Tool in the dialog box "Set PG/PC Interface".

Setting up the PLC802(PPI)/RS232 interface (Page 88)

### Communication via an Ethernet connection

The Programming Tool supports establishing a connection to the CPU via an Ethernet connection. For Ethernet communications, the Programming Tool requires port 102.

You set up the interface card and the protocol in the Programming Tool in the dialog box "Set PG/PC Interface".

Setting up an Ethernet connection (Page 90)

### Setting up communication

You can set up communication at the following points:

- In the last step when installing the Programming Tool.
- In the Programming Tool.

### Setting up communication in the Programming Tool

The Programming Tool displays the dialog box "Set Up Communication", in which you can set up your communication. Proceed as follows to open this dialog box:

- Click the Communication button in the "Navigation" toolbar (Page 500).
- Select the menu command View > Communications.
- Click the Communication button in the project tree (Page 447).

In the dialog box "Set Up Communication", double-click the uppermost symbol on the right-hand side. The dialog box "Set PG/PC Interface" is then opened.

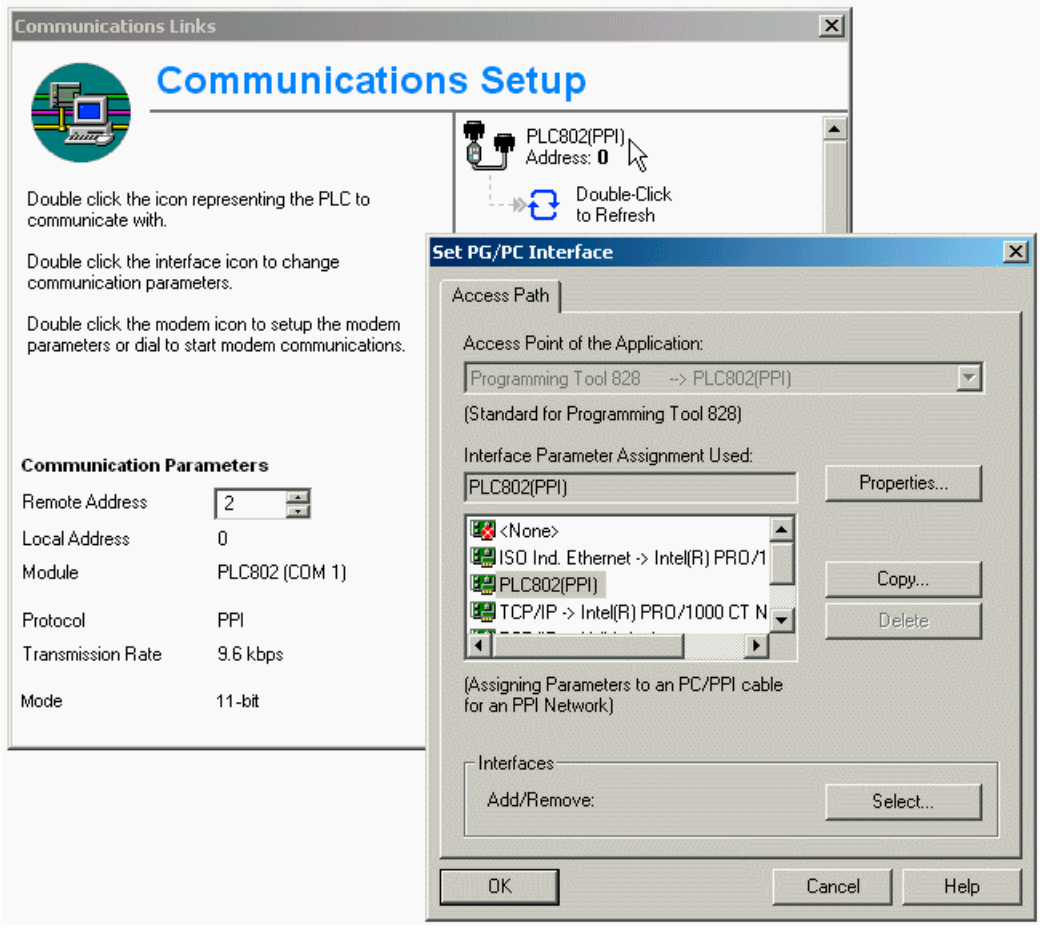



Figure 4-1 Dialog box "Set PG/PC Interface"

## See also:

Setting and changing parameters (Page 97)

Installing and removing communication interfaces (Page 96)

Setting up the PLC802(PPI)/RS232 interface (Page 88)

Setting up an Ethernet connection (Page 90)

## 4.2 Setting up communication via a PLC802(PPI) interface/RS232 cable (V24)

Establishing communication via a PLC802(PPI) interface is not supported by all controls.

| | | |
|---|---|---|
| 808D-PPU14x | 802Dsl TM value | SINUMERIK 802D |
| 808D-PPU15x | 802Dsl TM plus | SINUMERIK 802SC |
| | 802Dsl TM pro | |
| | 802Dsl CU pro | |
| | 802Dsl GN plus | |
| | 802Dsl GN pro | |

CPUs that support PLC802(PPI) interfaces

To establish a connection between the CPU and PG/PC, ensure that both are switched on. Activate the connection on the control. This is done directly at the CPU using the softkeys.

If you have not made any changes to the communication parameters, then no adaptation is required in the Programming Tool. If changes were made, then you can specify your parameters in the following section. You can now proceed as follows to establish your communication connection.

1. Make sure that you are in the LAD editor (**View > LAD**) and open the dialog box "Communication Connections" in one of the following ways:
   - Click the Communication button  in the "Navigation" toolbar (Page 500).
   - Select the menu command **View > Communications**.
   - Click the Communication button  in the project tree (Page 447).
2. In the dialog box "Communication Connections", double-click the refresh button to start a scan that identifies the CPU in the network.
3. If your CPU was found, you can cancel the search request. The communication connection has now been established.

**Setting up the PLC802(PPI) interface/RS232 cable (V24) parameters**

This section explains how to set up the parameters for the PLC802(PPI) interface if the default settings were changed.

Connect your PC to the CPU using the RS232 cable and open the dialog box "Communication Connections" in one of the following ways:

- Click the Communication button  in the toolbar of the navigation (Page 500).
- Select the menu command **View > Communications**.
- Click the Communication button  in the project tree (Page 447).

Double-click the PLC symbol. The dialog box "Set PG/PC Interface" opens. In the field "Interface Parameter Assignment Used", select "PLC802(PPI)" and click the "Properties" button. The properties for the PC/PPI cable (PPI) are then displayed.

The PLC802(PPI) protocol is set as default in the Programming Tool for communication with the CPUs.

Proceed as follows to set up the parameters:

1. Enter an address in the field "Station Parameters" on the tab "PPI". This is the Programming Tool address in the automation system network. Station address 0 is the default address for the computer on which the programming tool is installed. The default station address for the first PLC in your network is address 2. Every device (PC, automation system etc.) in your network requires a unique station address. Never assign an address a multiple number of times.
2. Enter a value for the timeout. This value specifies how long the communication driver should attempt to establish communication for. The default value should be sufficient.
3. For the data transmission rate, enter the baud rate with which the Programming Tool should communicate in the network. The PLC802(PPI) interface supports 9600 baud, 19,200 baud, 38,400 baud, 57,600 baud, and 115,200 baud.
4. Enter the highest node address. This is the address beyond which the Programming Tool should not continue to search for other masters in the network.

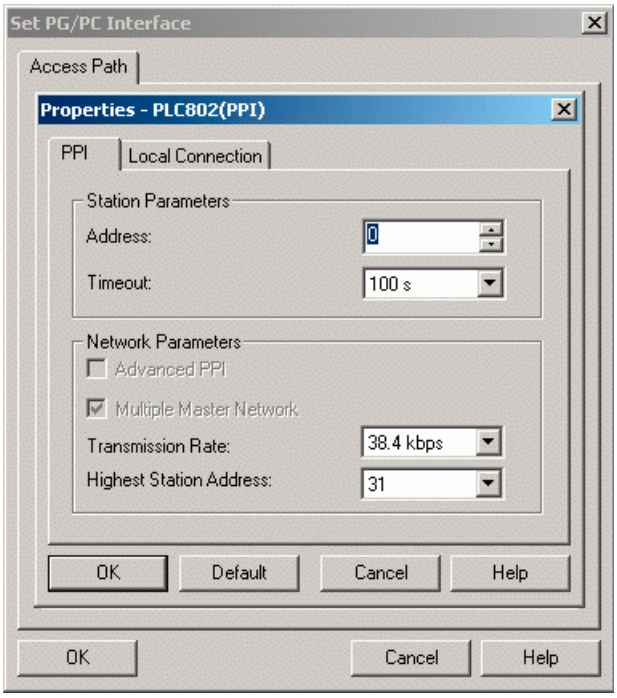



Figure 4-2 Dialog box "Properties -PLC802(PPI)", tab "PPI"

5. Open the tab "Local Connection".

6. On the tab "Local Connection", specify the COM port to which the RS232 cable is connected. If the checkbox "Modem connection" is activated, then deactivate it. Click "OK" to exit the dialog box "Set PG/PC Interface" and to accept the changes.
7. On the right of the dialog box "Communication Connections", click the blue text "Double-click to refresh".

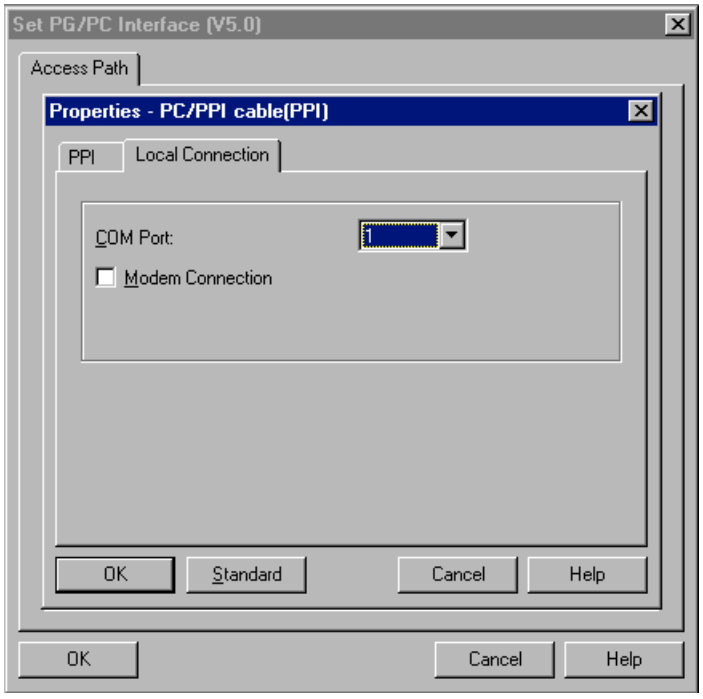



Figure 4-3 Dialog box "Properties -PLC802(PPI)", tab "Local Connection"

**Note**

The connection must be activated on the control. For more information, refer to the documentation for the control.

**See also:**

Options for communication (Page 85)

Communication interfaces (Page 96)

## 4.3 Setting up communication via an Ethernet connection

It is not possible to set up a network for all controls.

| | | |
|---|---|---|
| 828D | 808D-PPU15x | 802Dsl TM pro |
| 828D Step 2 | 808D-PPU16x | 802Dsl CU pro |
| | | 802Dsl GN pro |

CPUs that support a network

The CPU is network-capable as a result of the integrated network card. The following connections are possible:

- Peer-to-peer: A direct connection between the CPU and the PG/PC using a crossover Ethernet cable.
- Twisted pair: Integration of the CPU into an existing, local network using an Ethernet cable (patch cable).

To establish a connection between the CPU and PG/PC, ensure that both are switched on. Activate the connection on the control. This is done directly at the CPU using the softkeys.

**Setting up the parameters in the Programming Tool for the Ethernet connection**

This section explains how you set up the parameters for the CPUs that have an Ethernet connection.

Make sure that you are in the LAD editor (**View > LAD**) and open the dialog box "Communication Connections" in one of the following ways:

- Click the Communication button  in the "Navigation" toolbar (Page 500).
- Select the menu command **View > Communications**.
- Click the Communication button  in the project tree (Page 447).

Proceed as follows to set up communication with a TCP/IP Ethernet network:

1. Double-click the PLC symbol. The dialog box "Set PG/PC Interface" opens. In the field "Interface Parameter Assignment Used", select "TCP/IP" with the arrow to the Ethernet card for your computer and click the "OK" button. The name of your Ethernet card can be found under the network connections (Start > Settings > Network Connections). The corresponding device name is displayed there (TCP/IP -> device name of the Ethernet card). If you cannot find your interface, then you must first install (Page 96) it before you can proceed with establishing communication.

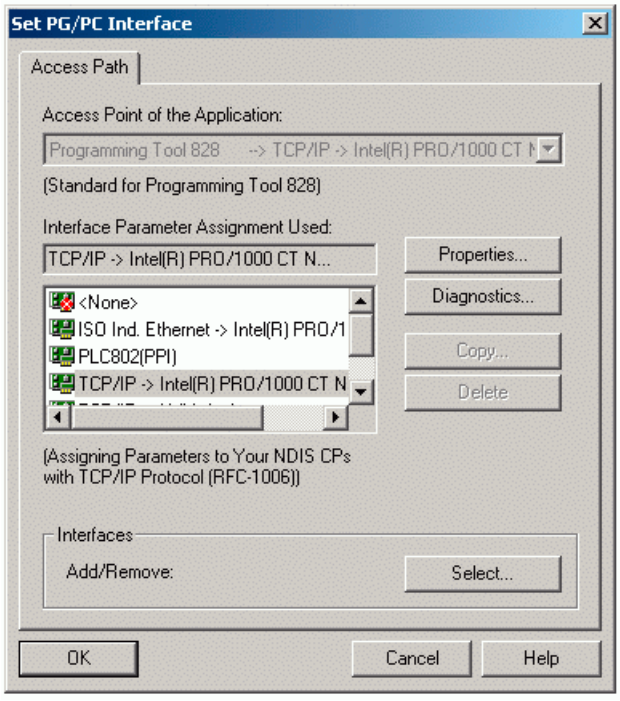



Figure 4-4 Dialog box "Set PG/PC Interface"

2. In the dialog box "Communication Connections", enter the IP address for the corresponding control under "Communication Parameters".
3. Double-click the Refresh symbol to establish a connection to the specified IP address.
   - If the connection has been established and the type of the PLC can be successfully determined, then the appropriate PLC symbol is displayed in the dialog box "Communication".
   - If the connection attempt fails, then the IP address is displayed as "Not available" in the dialog box "Communication".
   - If a connection is established, but the Programming Tool cannot determine the PLC type, then the IP address is displayed as "Unknown".

**Note**

The Programming Tool requires Port 102 to establish Ethernet communications. For information on how to activate Ethernet communication on the control, refer to the documentation for the control.

**See also:**

Options for communication (Page 85)

Communication interfaces (Page 96)

## 4.4 Setting up communication via a modem connection

It is not possible to communicate via a modem connection for all controls.

| | |
|---|---|
| 802Dsl TM value | SINUMERIK 802D |
| 802Dsl TM plus | |
| 802Dsl TM pro | |
| 802Dsl CU pro | |
| 802Dsl GN plus | |
| 802Dsl GN pro | |

CPUs that support a modem

To establish a connection between the CPU and PG/PC, ensure that both are switched on. Activate the connection on the control. This is done directly at the CPU using the softkeys.

**Setting up the parameters in the Programming Tool for communication via a modem connection**

This section shows you how to set up the parameters for the CPUs that have a modem connection.

Make sure that you are in the LAD editor (**View > LAD**) and open the dialog box "Communication Connections" in one of the following ways:

- Click the Communication button in the "Navigation" toolbar (Page 500).

- Select the menu command **View > Communications**.
- Click the Communication button in the project tree (Page 447).

Proceed as follows to set up communication via a modem:

1. Double-click the PLC symbol. The dialog box "Set PG/PC Interface" opens.
2. In the field "Interface Parameter Assignment Used", select "PLC802(PPI)" and click the "Properties" button.
3. Enter an address in the field "Station Parameters" on the tab "PPI". This is the Programming Tool address in the automation system network. Station address 0 is the default address for the computer on which the programming tool is installed. The default station address for the first PLC in your network is address 2. Every device (PC, automation system etc.) in your network requires a unique station address. Never assign an address a multiple number of times.

4. Enter a value for the timeout. This value specifies how long the communication driver should attempt to establish communication for. The default value should be sufficient.
5. For the data transmission rate, enter the baud rate with which the Programming Tool should communicate in the network. The PLC802(PPI) interface supports 9600 baud, 19,200 baud, 38,400 baud, 57,600 baud, and 115,200 baud.
6. Enter the highest node address. This is the address beyond which the Programming Tool should not continue to search for other masters in the network.

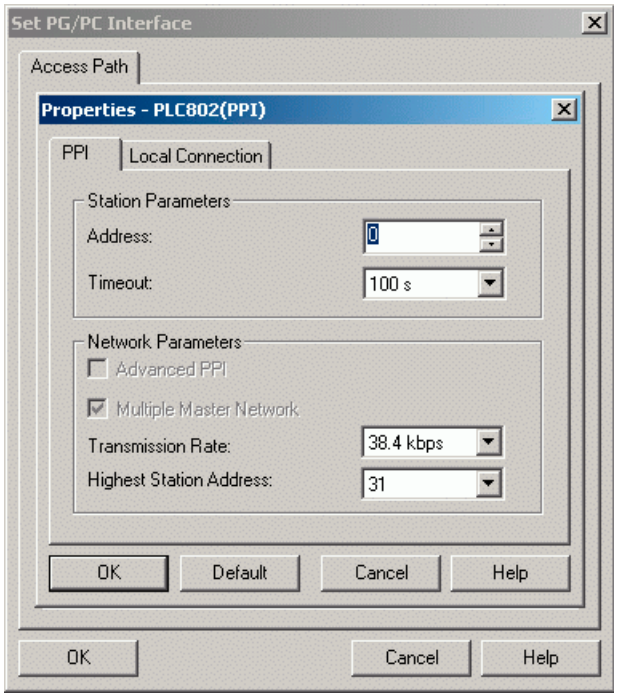



Figure 4-5 Dialog box "Properties -PLC802(PPI)", tab "PPI"

7. On the tab "Local Connection", enter the COM port of your modem and activate the check box "Modem connection".

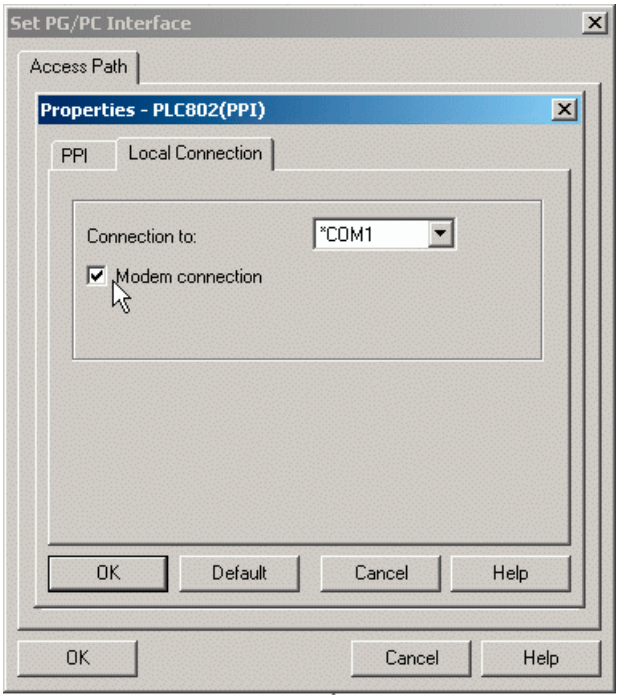



Figure 4-6 Dialog box "Properties -PLC802(PPI)", tab "Local Connection"

8. Click "OK" to exit the dialog box "Set PG/PC Interface" and to accept the changes.
9. On the right-hand side of the dialog box "Communication Connections", click the modems to set the sender and receiver.
10. Click "Connect modem" to enter the telephone number of the remote modem. Click"OK" to establish a connection.

**Note**

The connection must be activated on the control. For more information, refer to the documentation for the control.

**See also:**

Options for communication (Page 85)

Communication interfaces (Page 96)

## 4.5 Installing and removing communication interfaces

You can install or remove communications hardware by using the dialog box "Install/Remove Interfaces". On the left-hand side of the dialog box is a list of hardware that you have still not installed. On the right-hand side of the dialog box is a list of currently installed hardware.

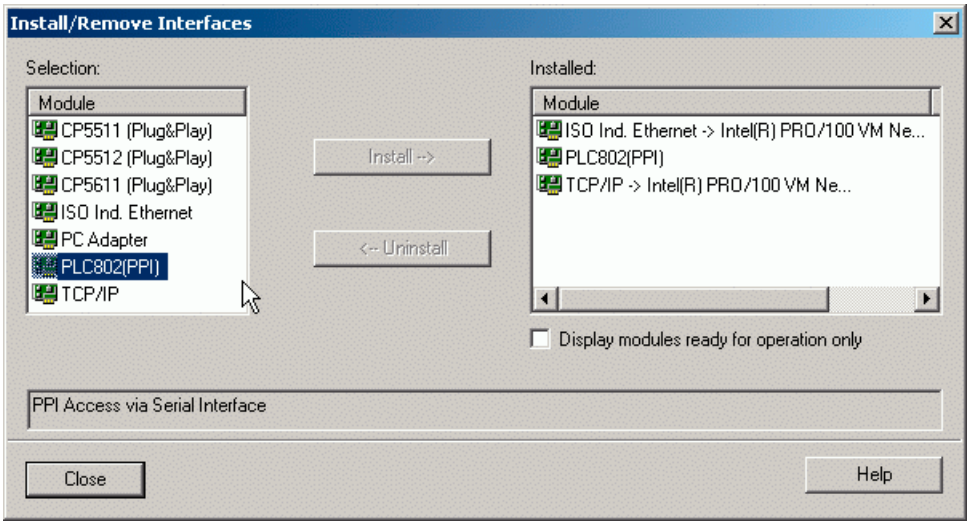



Figure 4-7 Dialog box to install and remove interfaces

**Installing the hardware:**

Proceed as follows to install the hardware:

1. In the dialog box "Set PG/PC Interface", click the "Install" button to open the dialog box "Install/ Remove Interfaces".
2. From the text box "Selection", select the hardware that you have and that you wish to install. A description of your selection is shown in the lower part of the dialog box.
3. Click the "Install" button.
4. When you have finished installing the required hardware, click the "Close" button. The dialog box "Set PG/PC Interface" dialog box is opened and your selection is displayed in the field "Interface Parameter Assignment Used".

**Removing the hardware:**

Proceed as follows to remove hardware:

1. Select the hardware that has already been installed in the text field on the right.
2. Click the "Remove" button.
3. When you have finished removing the required hardware, click the "Close" button. The dialog box "Set PG/PC Interface" dialog box is opened and your selection is displayed in the field "Interface Parameter Assignment Used".

**See also:**

Options for communication (Page 85)

Setting up the PLC802(PPI)/RS232 interface (Page 88)

Setting up an Ethernet connection (Page 90)

## 4.6 Setting and changing parameters

**Selecting the correct interface parameter assignment and setting it up**

If you are already in the dialog box "Set PG/PC Interface", you can set the properties for communication with your hardware. First enter the protocol that you wish to use for your network. You should use the PLC802(PPI) protocol for all CPUs.

After you have selected the correct interface parameter assignment, you must set up the individual parameters for the current configuration. To do this, click the "Properties..." button in the dialog box "Set PG/PC Interface". This action takes you to one of several possible dialog boxes depending on the interface parameter assignment that you selected. The various dialog boxes are described in detail in the following sections.

If you wish to delete an interface parameter assignment, then proceed as follows:

1. Ensure that your hardware has already been installed.
2. Enter the protocol with which you wish to work. You should use the PLC802(PPI) protocol for all CPUs.
3. In the dialog box "Set PG/PC Interface", select the appropriate settings in the text field "Interface Parameter Assignment Used".
4. To do this, click the "Properties..." button in the dialog box "Set PG/PC Interface".

From this point onwards, you should make all of the additional settings corresponding to the interface parameter assignment that you selected.

**Setting up the PLC802(PPI) interface/RS232 cable (V24) parameters**

This section explains how to set up the RS232 cable parameters if default settings have been changed.

Use an RS232 cable and, in the dialog box "Set PG/PC Interface", click the "Properties" button; the properties for the PC/PPI cable (PPI) are then displayed.

The PLC802(PPI) protocol is set as default in the Programming Tool for communication with the CPUs.

Proceed as follows to set up the parameters:

1. Enter an address in the field "Station Parameters" on the tab "PPI". This is the Programming Tool address in the automation system network. For the computer on which the Programming Tool is installed, station address 0 is the default address. The default station address for the first PLC in your network is address 2. Each device (PC, automaton system, etc.) in your network requires a unique station address. Never assign an address a multiple number of times.
2. Enter a value for the timeout. This value specifies how long the communication driver should attempt to establish communication for. The default value should be sufficient.
3. For the data transmission rate, enter the baud rate with which the Programming Tool should communicate in the network. The PLC802(PPI) interface supports 9600 baud, 19,200 baud, 38,400 baud, 57,600 baud, and 115,200 baud.

4. Enter the highest node address. This is the address beyond which the Programming Tool should not continue to search for other masters in the network.

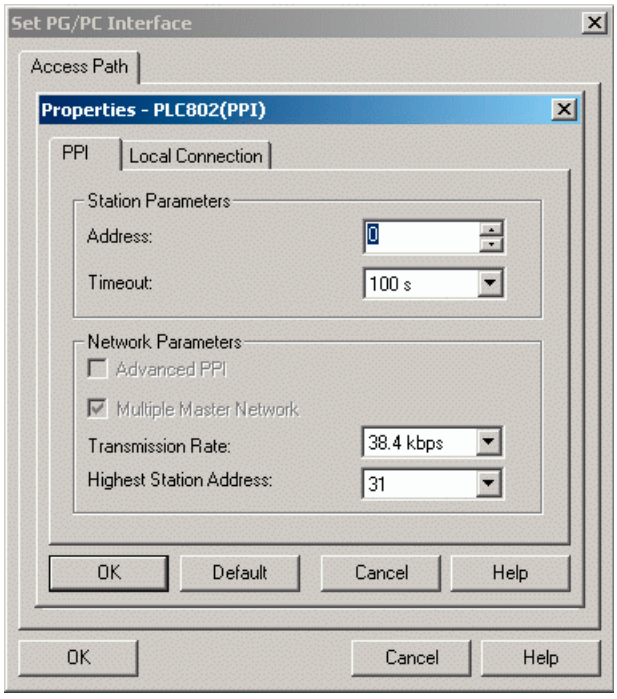



Figure 4-8 Dialog box "Properties - RS232 cable (PPI)", tab "Local Connection"

5. Open the tab "Local Connection".
6. On the tab "Local Connection", specify the COM port where the PLC802(PPI) interface is located. If you are working with a modem, specify the COM port to which the modem is connected and activate the check box "Modem connection".

7. Click"OK" to exit the dialog box "Set PG/PC Interface".
8. On the right of the dialog box "Communication", click the blue text "Double-click to refresh".

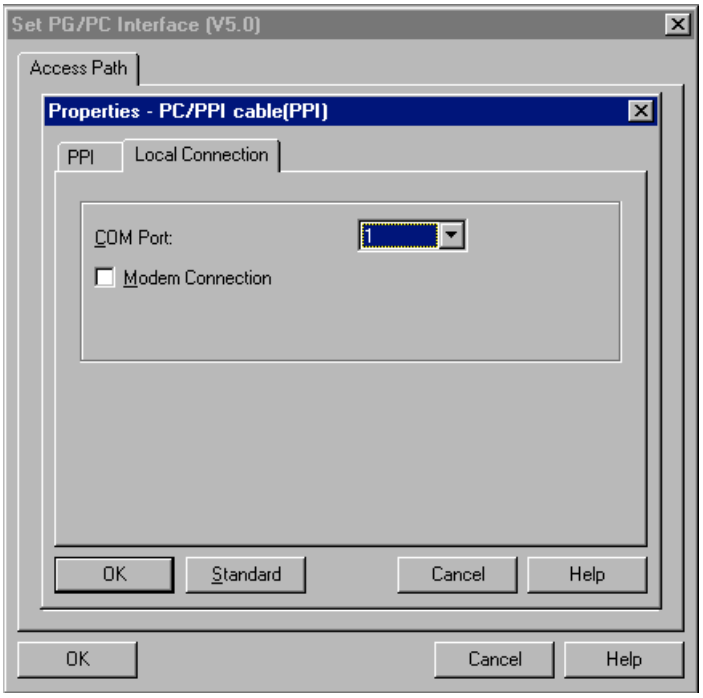



Figure 4-9 Dialog box "Properties - RS232 cable (PPI)", tab "Local Connection"

**Note**

The connection must be activated on the control. For more information, refer to the documentation for the control.

### Setting up the parameters for the Ethernet connection

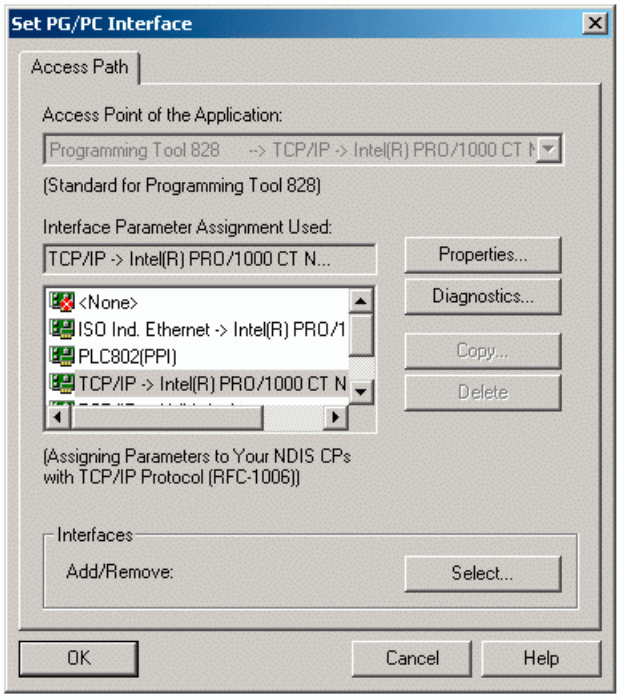
This section explains how you set up the parameters for the CPUs that have an Ethernet connection.

If you are using an Ethernet connection, in the dialog box "Set PG/PC Interface", select the Ethernet card for your computer. Click "OK".

**Note**

Please observe the following information when setting the maximum number of connections:

1. The number of connections specified here must be at least the same as the configured number of connections.
2. The maximum number of connections depends on the license used and the parameter entry here. The lower limit of the two will be used.




Proceed as follows to set up the parameters:

Proceed as follows to set up communication with a TCP/IP Ethernet network:

1. Enter the IP address for the corresponding control in the dialog box "Communication".
2. Double-click the Refresh symbol to establish a connection to the specified IP address.
   - If the connection has been established and the type of the PLC can be successfully determined, then the appropriate PLC symbol is displayed in the dialog box "Communication".
   - If the connection attempt fails, then the IP address is displayed as "Not available" in the dialog box "Communication".
   - If a connection is established, but the Programming Tool cannot determine the PLC type, then the IP address is displayed as "Unknown".

**Note**

The Programming Tool requires port 102 to establish Ethernet communication. For information on how to activate Ethernet communication on the control, refer to the documentation for the control.

**See also:**

Communication options (Page 85)

# Mode of operation of the automation system 5

## 5.1 RUN/STOP modes of the PLC

**Mode of operation of the program:**

If you load a program into your PLC and switch the PLC into the RUN mode, the PLC (CPU) of the PLC executes the program in a repeated scan cycle. This scan cycle is called in the interval of MD10075 PLC_CYCLE_TIME:

1. **Reading the inputs**
   The CPU reads the status of all inputs connected to the automation system. This data is stored in the input memory area, known as the process image input.
2. **Executing the optional interrupt program INT100**
3. **Executing the program**
   The CPU calls MAIN and executes the logic of the program. In this respect, it has read and write access to the process image, variables, data blocks, etc.
4. **Executing the optional interrupt program INT101**
5. **Writing to the outputs**
   At the end of the program, the CPU writes the data from the process image output to the physical outputs.
6. **Processing communication requests (NCK, HMI)**
7. **CPU self-diagnosis**

**Changing the mode of the PLC**

- Click the RUN button to place the PLC into the RUN mode. Click the STOP button to place the PLC into the STOP mode.
- Select the PLC > RUN menu command to place the PLC into the RUN mode. Select the PLC > STOP menu command to place the PLC into the STOP mode.

---

**Note**

In order that you can set the RUN/STOP mode using the Programming Tool, communication (Page 85) between the Programming Tool and the PLC must be established.

---

**PLC operating mode details:**

The PLC has two operating modes: STOP and RUN. In the STOP mode, you can create and edit your program. However, your program is not executed in the STOP mode. The program is executed in the RUN mode. Further, in the RUN mode, you can observe the mode of operation and data of the program. Test functions are available to better observe the mode of operation of the program and to identify programming errors (bugs).

The test functions to execute the first or several scan cycles can be used in the STOP mode. The RUN mode is then only assumed for the specified number of scan cycles.

Fatal errors are saved in the CPU operating system and force a mode change from the RUN to the STOP mode. If the PLC has identified a fatal error, then it can no longer be brought from the STOP mode into the RUN mode as long as the error is present. Although the operating system of the CPU stores minor errors that can be evaluated later, they do not require a transition from RUN to STOP.

In the **STOP** mode

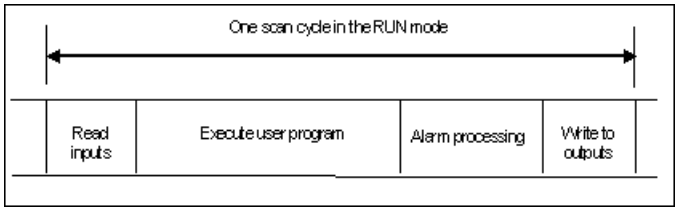
- The PLC is, in effect, in a quasi-idle state (the execution of the user program is interrupted)
- You can load the user program into and from the PLC.

If one or several devices are communicating with the PLC via the communication interface, then the PLC responds to all of the requests one after the other. The PLC does not prevent several communicating devices mutually interfering with one another. Your system design must provide the necessary safety precautions to prevent this mutual interference between communicating partners.

In the **RUN** mode, the PLC

- Reads the inputs
- Executes the user program
- Writes to the outputs
- Responds to communications requests
- Performs internal administrative tasks
- Responds to interrupt conditions

The PLC does not support fixed scan cycle times to execute the scan cycle in the RUN mode. These tasks (with the exception of the interrupts) are executed according to their priority, in the sequence that they occur. This continuous execution of the CPU tasks is called the scan cycle and is shown in the following diagram.




Each scan cycle begins by reading the actual values of the inputs and writing these values into the process image input. Input bits that have no corresponding physical input but which are in the same byte as bits with physical inputs are set to zero in the process image in each scan cycle.

After the inputs have been read, your program executes the instructions from the first through to the last one.

During the phase in which the alarms are processed in the scan cycle, all alarms are processed for which changes occurred during the scan cycle.

Finally, the values are written from the process image into the output modules. This completes one scan cycle.

**See also:**

Information on using the software (Page 375)

First steps: Contents (Page 17)

## 5.2 Elements of a program

**LAD programs**

In LAD programs, the main control elements are represented as contacts, coils, and boxes. Connected elements that form a closed circuit are referred to as a network.

A hardwired input is represented by the symbol for a contact (Page 111). When closed, an NO contact enables the signal flow. A contact can also be normally closed. An NC contact becomes current-conducting when it is open.

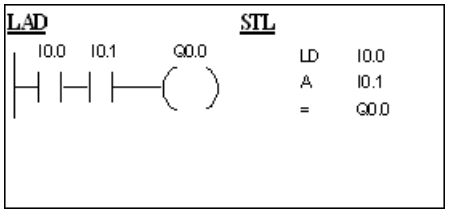
A hardwired output is represented by the symbol for a coil (Page 117). If there is a signal flow to the coil, the output is switched on.

A box represents a complex instruction which is executed by the CPU. A box simplifies the programming of complex instructions. For example, timers (Page 173) and counters (Page 169) are shown as boxes.

**STL programs**

The elements of an STL program are represented by instructions that execute the desired instructions. In contrast to LAD programs, which are represented graphically, STL programs are shown in text format.

The following example shows the various representations.




## 5.3 Overview of test and monitoring functions

### Diagnostics functions

Once you have established communication between your programming device, on which the PLC Programming Tool is installed, and a PLC and have loaded a program into the PLC, you can work with the diagnostic functions of the PLC Programming Tool and test new programs as well as monitor programs that are already being processed.

These help topics are explained in the following:

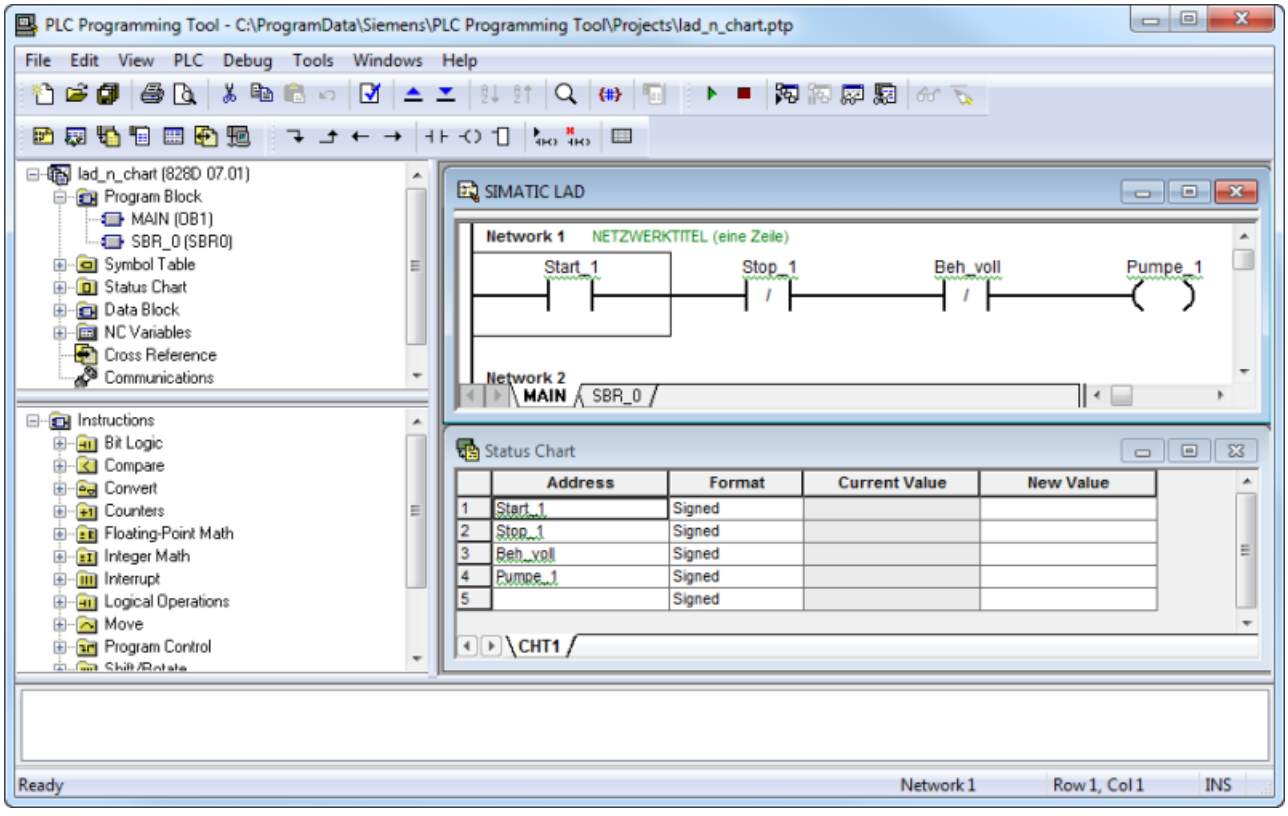
## What is "status"?

"Status" refers to the display of the actual values of operands while executing the program in the PLC. You can display status information in a status chart, with the data block status or by enabling the program status in the program editor.

In the status chart in each case the values are recorded at the end of a scan cycle (end-of-scan status). It is not possible to display the status of the local data memory and the accumulators.
With the data block status, the actual values are also recorded at the end of a scan cycle in each case.

If the execution status is activated for the program status on the "LAD Status" tab (Tools > Options) (Page 492), the status values are recorded at the time of execution of the instructions. The states of the local data memory and the accumulators are displayed. The status values are updated and displayed only if the PLC is in the RUN mode.

An example for status information in the status chart and in the program editor of the PLC Programming Tool is shown in the following diagram:



**Note**

Please note that unnecessary project components (e.g. the project tree, the instruction tree, and the output window) have been omitted, in order to provide more space to display the required components (LAD program editor, status chart, function bar to test and symbol for the status chart in the navigation bar). Using the menu "View", you can set-up the environment in the PLC Programming Tool corresponding to your particular tasks, i.e. you can display just the project components that you require.

## Supporting CPUs

The execution status is not available for all CPUs. The following table shows which CPUs are available:

| | |
|---|---|
| 828D (as of PLC 05.04) | 808D-PPU14x |
| 828D Step 2 | 808D-PPU15x |
| | 808D-PPU16x |

## Execution status for unsupported CPUs

If the execution status is activated for a CPU that does not support this status, the end-of-scan status is automatically executed.

If the execution status is not activated, the end-of-scan status is executed. For this, the values are recorded at the end of a scan cycle in each case. It is not possible to display the status of the local data memory and the accumulators.

## Preconditions of the status update

Before you can update the status to monitor and test your program, you must execute the following tasks:

- Your program must be able to be compiled error-free.
- You must have set up communication between the PLC Programming Tool and the PLC.

- Your program must have been loaded error-free into the PLC.

  **Note**

  The time stamps of your project in the Programming Tool and in the PLC must match, so that you can enable the status. The comparison of the time stamps (carried out automatically if you attempt to enable the status) ensures that the representation of the project status in the Programming Tool exactly reflects the status of the program in the PLC. If the time stamps do not match, you can compare the programs via the dialog box "Time Stamps Do Not Match". Look at the drop-down list box in order to find out which networks are different.

- After you have loaded the program into the PLC, then you should again bring this into the RUN mode. Otherwise, the address status is displayed, however, the PLC cannot execute the program, so you are not shown the logic operations that you expect.

### Operating state of the PLC

The type of monitoring and test functions that you can execute depends on the mode of your PLC.

Even if your program is not executed in the STOP mode, the operating system of the PLC still monitors the PLC (status of RAM and I/O) and transfers the data status to the Programming Tool. If the PLC is in the STOP mode, then you can execute the following functions:

- You can display the actual values of the operands as the end-of-scan status in the chart status, the data block status or the program status. (This is the same as the function "Single Read", as the program is not executed.)
- In the chart status and the data block status, you can write values.
- You can execute a certain number of scan cycles and display the effect in a status chart, in the data block status and/or in the program status.

If the PLC is in the RUN mode, you cannot execute the functions "First Scan" or "Multiple Scans". You can write values in a status chart, you can also execute the following functions (not in the STOP mode):

- In the chart status, you can carry out continuous updates. (If you wish to only execute one update, you must disable the chart status so that you can execute the command "Single Read".)
- In the data block status, you can carry out continuous updates.
- You can execute continuous updates in the program status.

### Communication and scan cycle

In a continuous scan cycle, the PLC reads the inputs, it executes the program, writes to the outputs, and executes system functions as well as communication. This scan cycle runs with an extremely high speed of many times per second. Even if the PLC Programming Tool issues status requests in a fast sequence, you cannot monitor each individual event that takes place in the PLC. When using the program status or the chart status, if you read data values from a PLC program, the data is scanned by taking samples (spot check). The update rate of the status values read from the PLC depends on the communication baud rate. Communicate with maximum baud rate, if you want to achieve the highest update rate.

## Different ways to update the status

### Chart status

Click the "Chart Status" button or select the menu command **Debug > Chart Status** to view continuous updates of the status when the PLC is in the RUN mode. Disable the chart status and use the function "Single Read" if you need to update the status just once and you do not want to switch the PLC into the STOP mode. If you switch the PLC into the STOP mode and enable the chart status, you receive a status update too. Furthermore, you can use the functions "Multiple Scans" and "First Scan" while you are viewing a status chart. In the chart status, the end-of-scan status is always used.

### Data Block Status

Click the "Data Block Status" button or select the menu command **Debug > Data block status** to view continuous updates of the current values of a data block when the PLC is in the RUN mode. If you switch the PLC into the STOP mode and enable the data block status, you receive a current value update too. Furthermore, you can use the functions "Multiple Scans" and "First Scan" while you are viewing a data block status. With the data block status, the actual values are always recorded at the end of a scan cycle.

### Program status

Click the "Program Status" button or select the menu command **Debug > Program Status** to display the status of the data in the PLC in the program editor. The status data is recorded in the mode previously selected on the "LAD Status" tab (Tools > Options) (Page 492):

- Execution status (execution status activated)
  In the execution status in LAD, the value of operands and the signal flow for execution of the individual instructions during the execution of the program in the scan cycle are displayed. The execution status shows the values of the operands at the time of execution of the instruction; this may be overwritten again by subsequent program instructions. All of the displayed data values of the PLC are recorded in a single program scan cycle. The states of the local data memory and the accumulators are displayed. The status values are only updated and displayed if the PLC is in the RUN mode.

  **Note**

  The execution status may only be run as the program status once for each CPU at any one time. If you attempt to run the execution status in a different application, such as in a second Programming Tool, which is connected to the same PLC, an error will be displayed.

- End-of-scan status (execution status deactivated or CPU does not support execution status)
  The end-of-scan status shows the status results that are read at the end of the program scan cycle. These results may not indicate all changes to the values of the addresses of data in the PLC, because subsequent program instructions write a value and overwrite it again before the end of the program scan cycle is reached. The end-of-scan status shows the data values that are scanned at the end of multiple scan cycles. This results from the different speeds between the fast scan cycle of the PLC and the relatively slow transmission of the PC status data. The states of the local data memory and the accumulators are not displayed. If you switch the PLC into the STOP mode, you receive a status update too.

You can receive the status values of the program status continuously or as a snapshot.

- **Continuous:**
  Open the program editor window and enable the program status in order to view continuous updates of the program execution status when the PLC is in the RUN mode.

  **Note**

  "Continuous" should not be understood as "real time"; it means that the programming device scans the PLC for status information continuously and displays it on your screen, updating the display as quickly as your communication setup permits. Some rapidly fluctuating values may not be recorded and displayed on your screen. It is also possible that the values change too quickly for you to read them.

- **Snapshot:**
  If the PLC is in the STOP mode, you can use the menu command **Debug > Multiple Scans...** to display the status values of one or more scan cycles, or, with the menu command **Debug > First Scan**, to display the status values of the first scan cycle.

With the button "Pause Program Status" or the menu command Debug > Pause Program Status you can pause the status display of the program status. This can be necessary if values change too quickly for you to read them. In the execution status this can also make a single execution of a program section, e.g. a subprogram, visible. To do this, set the networks which are not being processed in the program editor and pause the status display with the execution status activated. The next time the networks are executed in the program editor, the status display is updated. The status update then stops.

### Simulating process conditions

You can simulate process conditions by writing new values to operands: To do so, use the status chart.

### Checking cross references and the elements used

If you test your program, you may wish to supplement, delete or change the parameters.

In the window "Cross Reference", you can determine how the parameters are presently assigned in your program. This helps you to avoid assigning values twice.

### Loading changes in your program into the CPU

If you test a program with the program status and thus determine that you want to change the program, you must disable the program status before you can change the program. Once you have loaded the changed program into the CPU, you can enable the program status to check whether the changes are correct.

### See also

Displaying the status in the editors (GS 5.2) (Page 60)

Displaying the status in the status chart (GS 5.3) (Page 68)

Executing a certain number of scan cycles (GS 5.4) (Page 73)

Displaying the data block status (GS 5.5) (Page 74)

Downloading a program to the CPU (GS 4.3) (Page 52)

Cross-references and elements used (Page 311) (Ensure that program changes do not produce duplicate assignment.)

RUN/STOP modes of the PLC (Page 101)

# LAD and STL editors

# 6

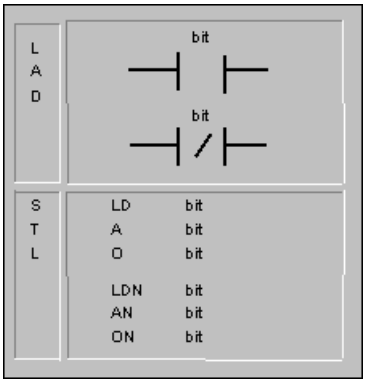
## 6.1 Programming in the ladder logic

### 6.1.1 LAD instructions

#### 6.1.1.1 Bit logic operations

**NO contact, NC contact**

| Inputs/outputs | | Addresses | Data types |
|---|---|---|---|
| Bit (LAD, STL) | | V, I, Q, M, SM, T, C, L | BOOL |
| Memory areas (Page 262) | ENO (Page 259) | Supported instructions (Page 259) | Mnemonics (Page 260) |




These instructions take the value specified for the data types I and Q from the memory or from the process image. You can use a maximum of seven inputs for the AND and OR boxes in each case.

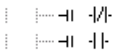
The **NO contact** is closed (on) if the bit at address n is equal to 1.

The **NC contact** is closed (on) if the bit is equal to 0.

In LAD, normally closed and normally open instructions are represented by contacts.

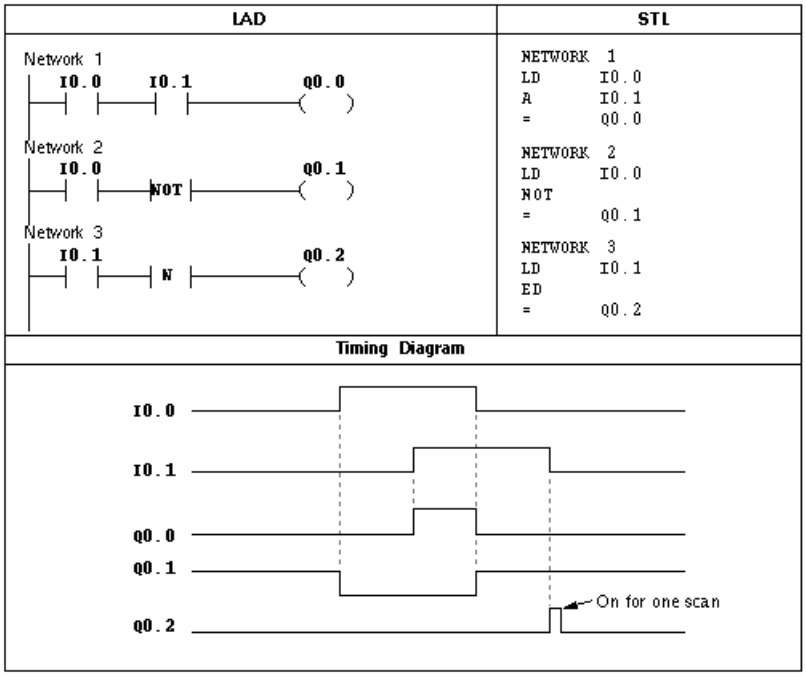
In STL, the NO contact is represented by the instructions **load bit value**, **combine bit value for logic AND**, and **combine bit value for logic OR**. These instructions load the bit value of the address bit as the topmost stack value or combine the bit value with the topmost stack value for logic AND or OR. In STL, the NO contact is represented by the instructions **load inverted bit value**, **combine inverted bit value for logic AND**, and **combine inverted bit value for logic OR**. These instructions load the inverted bit value of the address bit as the topmost stack value or combine the inverted bit value with the topmost stack value for logic AND or OR.

## Supporting CPUs



| | | | |
|---|---|---|---|
| 828D | 808D-PPU14x | 802Dsl TM value | SINUMERIK 802D |
| 828D Step 2 | 808D-PPU15x | 802Dsl TM plus | SINUMERIK 802SC |
| | 808D-PPU16x | 802Dsl TM pro | |
| | | 802Dsl CU pro | |
| | | 802Dsl GN plus | |
| | | 802Dsl GN pro | |

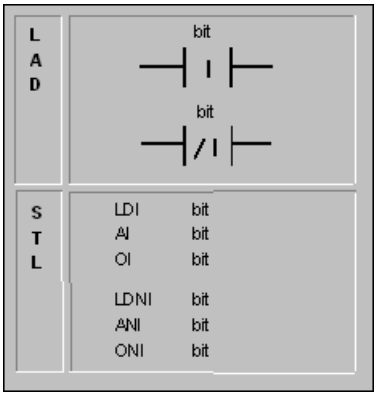
## Example



## Direct NO contact, NC contact

## Direct NO contact, NC contact operation

| Inputs/outputs | | Operands | Data types |
|---|---|---|---|
| Bit (LAD, STL) | | I | BOOL |
| Memory areas (Page 262) | ENO (Page 259) | Supported operations (Page 259) | Mnemonics (Page 260) |




The direct operation reads the value from the physical input when the operation is executed. The process image is not updated.

The **direct NO contact** is closed (on) if the physical input (bit) is equal to 1.

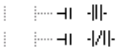
The **direct NC contact** is closed (on) if the physical input (bit) is equal to 0.

In LAD, the direct NC contact and direct NO contact operations are represented by contacts.

In STL, the direct NO contact is represented by the operations **load bit value directly**, **combine bit value directly for logic AND**, and **combine bit value directly for logic OR**. These operations load the value of the physical input directly as the topmost stack value or combine the value directly with the topmost stack value for logic AND or OR.

In STL, the direct NC contact is represented by the operations **load inverted bit value directly**, **combine inverted bit value directly for logic AND**, and **combine inverted bit value directly for logic OR**. These operations load the inverted value of the physical input directly as the topmost stack value or combine the inverted value directly with the topmost stack value for logic AND or OR.

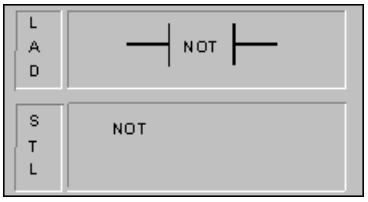
## Supporting CPUs



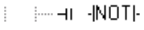
828D
828D Step 2

## NOT

| Inputs/outputs | | Addresses | Data types |
|---|---|---|---|
| None | | None | None |
| Memory areas (Page 262) | ENO (Page 259) | Supported instructions (Page 259) | Mnemonics (Page 260) |

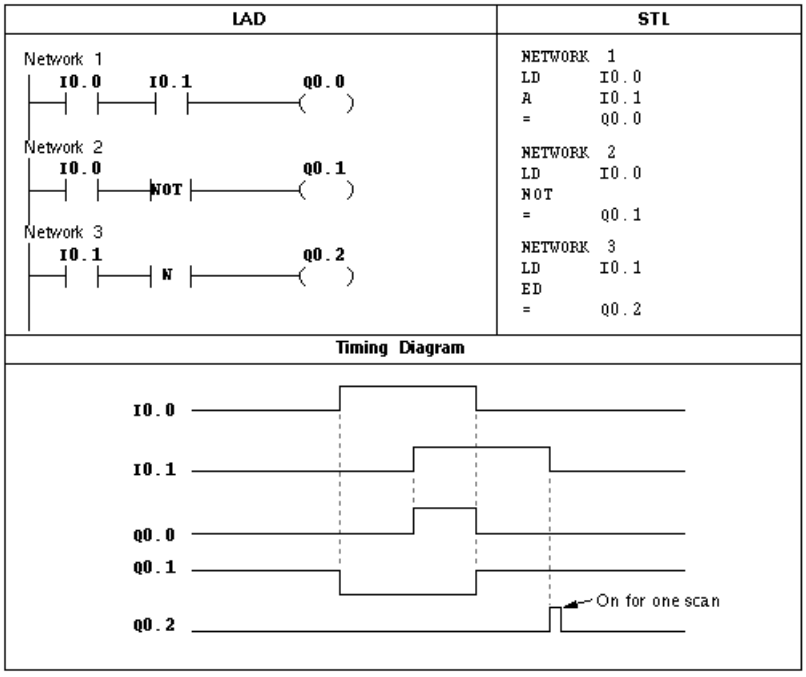
| | |
|---|---|
|  | The **NOT** contact changes the state of the signal flow input. If the signal flow reaches the NOT contact, it is stopped at the contact. If the signal flow does not reach the NOT contact, it is generated at the contact.<br><br>In LAD, the **NOT** instruction is represented as a contact.<br><br>In STL, the **NOT** instruction sets the topmost stack value from 0 to 1 or from 1 to 0. |

## Supporting CPUs



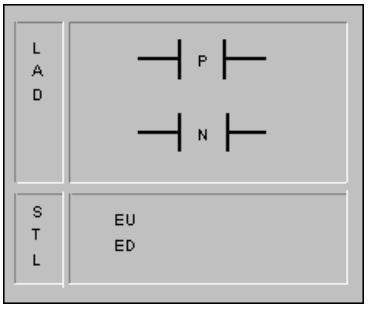
| | | | |
|---|---|---|---|
| 828D | 808D-PPU14x | 802Dsl TM value | SINUMERIK 802D |
| 828D Step 2 | 808D-PPU15x | 802Dsl TM plus | SINUMERIK 802SC |
| | 808D-PPU16x | 802Dsl TM pro | |
| | | 802Dsl CU pro | |
| | | 802Dsl GN plus | |
| | | 802Dsl GN pro | |

## Example

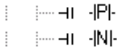



## Positive, negative edge

| | | | |
|---|---|---|---|
| Memory areas (Page 262) | ENO (Page 259) | Supported instructions (Page 259) | Mnemonics (Page 260) |

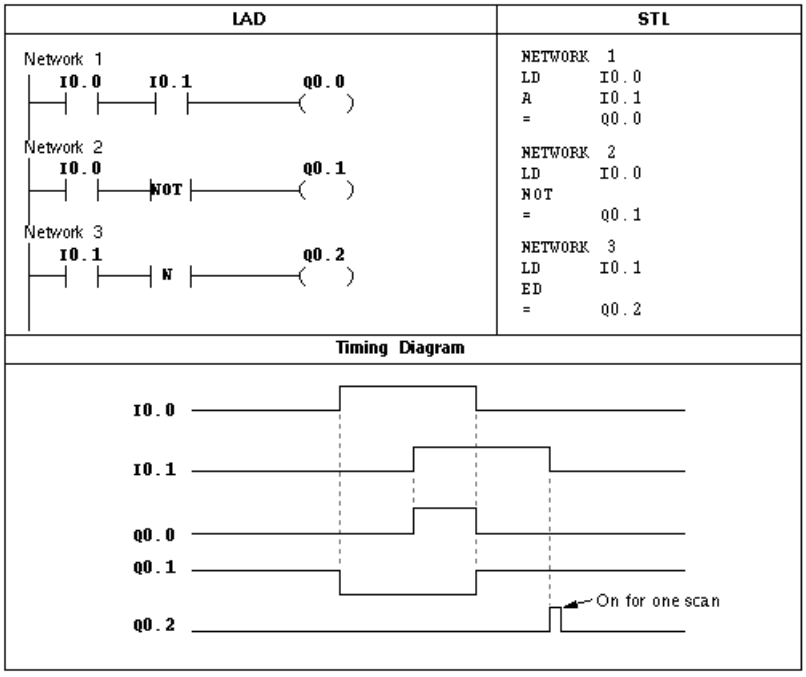
| | |
|---|---|
|  | The **positive edge** contact permits the signal flow for one scan cycle,<br>on every positive edge.<br>The **negative edge** contact permits the signal flow for one scan cycle,<br>on every negative edge.<br><br>In LAD, the positive and negative edge instructions are represented<br>by contacts.<br><br>In STL, the positive edge contact is represented by the **detection of positive edge** instruction. If<br>a positive edge (change from 0 to 1) is detected in the topmost stack value,<br>the topmost stack value is set to 1. If no positive edge is detected, the value is set to 0.<br>In STL, the negative edge contact is represented by the **detection of negative edge** instruction. If<br>a negative edge (change from 1 to 0) is detected in the topmost stack value,<br>the topmost stack value is set to 1. If no negative edge is detected, the value is set to 0. |

## Supporting CPUs



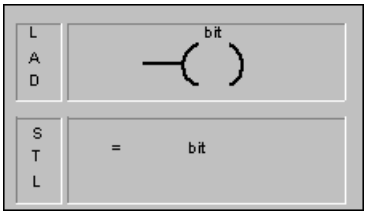

| | | | |
|---|---|---|---|
| 828D | 808D-PPU14x | 802Dsl TM value | SINUMERIK 802D |
| 828D Step 2 | 808D-PPU15x | 802Dsl TM plus | SINUMERIK 802SC |
| | 808D-PPU16x | 802Dsl TM pro | |
| | | 802Dsl CU pro | |
| | | 802Dsl GN plus | |
| | | 802Dsl GN pro | |

## Example




## Assign bit value

| Inputs/outputs | | Addresses | Data types |
|---|---|---|---|
| Bit | | V, I, Q, M, SM, T, C, L | BOOL |
| Memory areas (Page 262) | ENO (Page 259) | Supported instructions (Page 259) | Mnemonics (Page 260) |

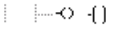



If the **assign** instruction is executed, the output bit is enabled in the process image.

In LAD, when the assign instruction is executed, the specified bit is set according to the state of the signal flow.

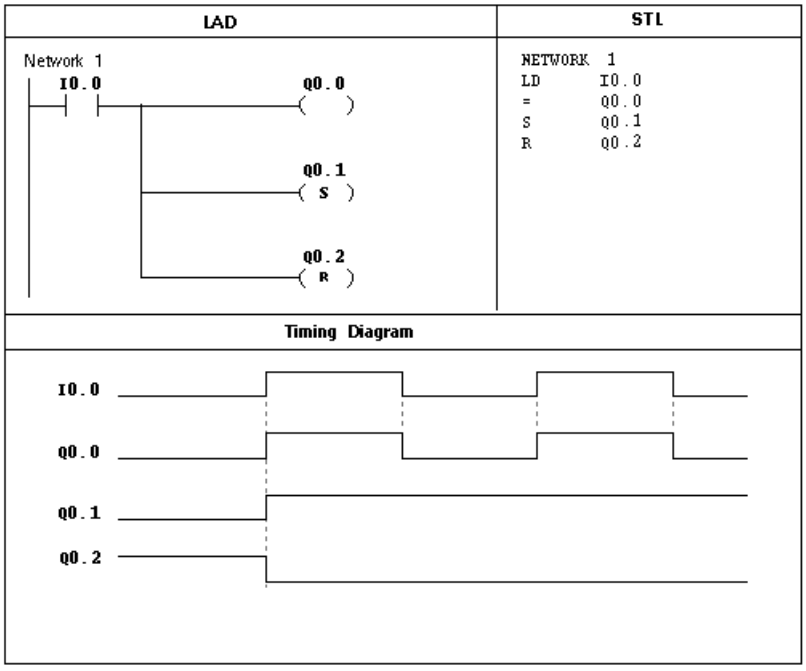
In STL, the assign instruction copies the topmost stack value into the specified bit.

## Supporting CPUs



| | | | |
|---|---|---|---|
| 828D | 808D-PPU14x | 802Dsl TM value | SINUMERIK 802D |
| 828D Step 2 | 808D-PPU15x | 802Dsl TM plus | SINUMERIK 802SC |
| | 808D-PPU16x | 802Dsl TM pro | |
| | | 802Dsl CU pro | |
| | | 802Dsl GN plus | |
| | | 802Dsl GN pro | |

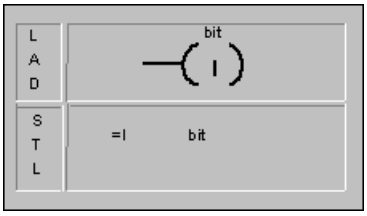
### Example



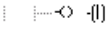

## Assign bit value directly

## Instruction to directly assign a bit value

| Inputs/outputs | | Addresses | Data types |
|---|---|---|---|
| Bit (LAD, STL) | | Q | BOOL |
| Memory areas (Page 262) | ENO (Page 259) | Supported instructions (Page 259) | Mnemonics (Page 260) |

| | |
|---|---|
|  | If the **assign bit value directly** instruction is executed, then<br>the physical output (bit or OUT) is set according to the state of the signal flow.<br>The "I" shows direct access. The new value is written to the<br>process image, as well as directly to the physical output, when the instruction is executed. This is where a direct instruction differs from other instructions where the value for<br>the input or output being addressed is only written to the process image.<br>The direct digital onboard<br>outputs are a special case, as they have no process image.<br><br>In STL, the assign bit value directly instruction copies the topmost<br>stack value directly to the specified physical output (bit). |

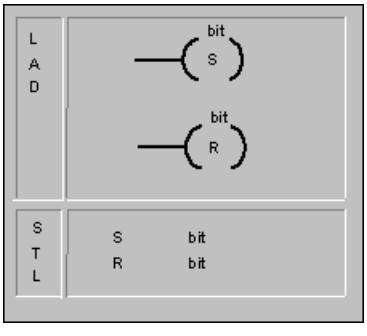
## Supporting CPUs



828D
828D Step 2

## Set, reset

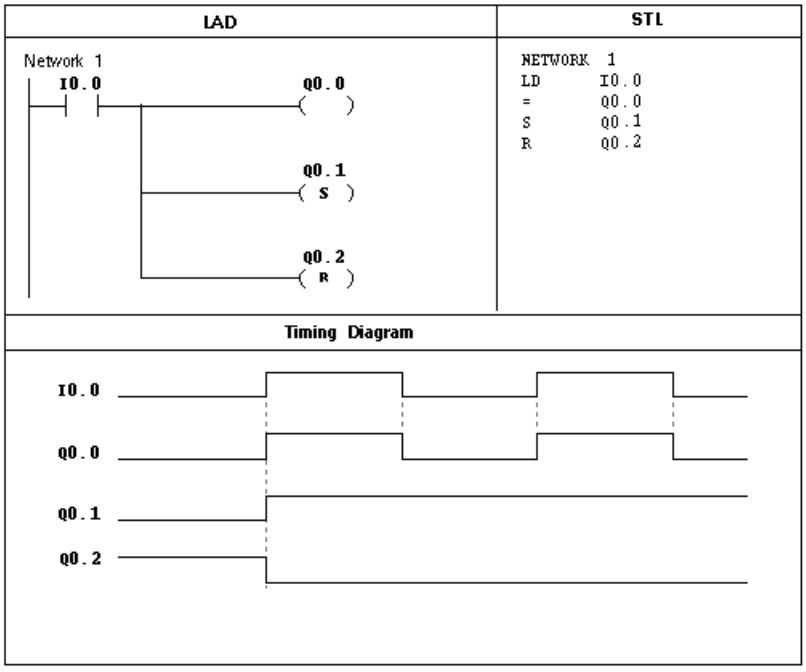
| Inputs/outputs | | Addresses | Data types |
|---|---|---|---|
| Bit | | V, I, Q, M, T, C, L | BOOL |
| Memory areas (Page 262) | ENO (Page 259) | Supported instructions (Page 259) | Mnemonics (Page 260) |

| | |
|---|---|
|  | If the **set** (S) and **reset** (R) instructions<br>are executed, then the output specified by the bit parameter<br>is set (switched on) or reset<br>(switched off).<br>If a T or C bit is specified for the bit for the reset instruction, then the timer or counter bit is reset and<br>the actual value of the timer or counter is deleted. |

## Supporting CPUs

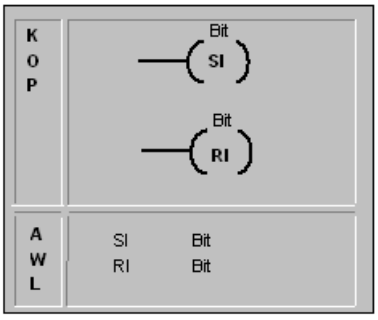

| | | | |
|---|---|---|---|
| 828D | 808D-PPU14x | 802Dsl TM value | SINUMERIK 802D |
| 828D Step 2 | 808D-PPU15x | 802Dsl TM plus | SINUMERIK 802SC |
| | 808D-PPU16x | 802Dsl TM pro | |
| | | 802Dsl CU pro | |
| | | 802Dsl GN plus | |
| | | 802Dsl GN pro | |

## Example




## Set, reset bit value directly

| Inputs/outputs | | Addresses | Data types |
|---|---|---|---|
| Bit | | Q | BOOL |
| Memory areas (Page 262) | ENO (Page 259) | Supported instructions (Page 259) | Mnemonics (Page 260) |




If the instructions **set bit value directly** (SI) and **reset bit value directly** (RI) are executed, then the output specified by the parameter bit is set directly (switched on) or reset directly (switched off).

The "I" shows direct access. The new value is written to the process image, as well as directly to the physical output, when the instruction is executed. This is where a direct instruction differs from other instructions in which the value for the input or output being addressed is only written to the process image.

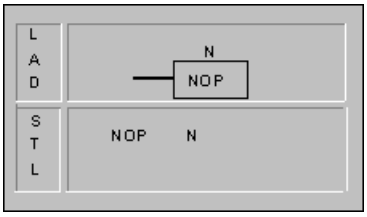
The direct digital onboard outputs are a special case, as they do not have a process image.

## Supporting CPUs


828D
828D Step 2

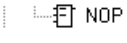
## Null instruction

| Inputs/outputs | | Addresses | Data types |
|---|---|---|---|
| | | N: constant (0 to 255) | BYTE |
| Memory areas (Page 262) | ENO (Page 259) | Supported instructions (Page 259) | Mnemonics (Page 260) |




The **null instruction** (NOP) has no influence on the processing of the user program. The operand N is a number between 0 and 255.

## Supporting CPUs

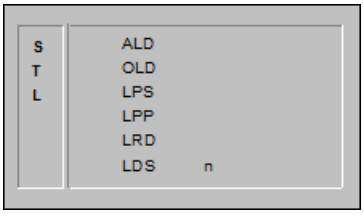



| | | | |
|---|---|---|---|
| 828D | 808D-PPU14x | 802Dsl TM value | SINUMERIK 802D |
| 828D Step 2 | 808D-PPU15x | 802Dsl TM plus | SINUMERIK 802SC |
| | 808D-PPU16x | 802Dsl TM pro | |

802Dsl CU pro
802Dsl GN plus
802Dsl GN pro

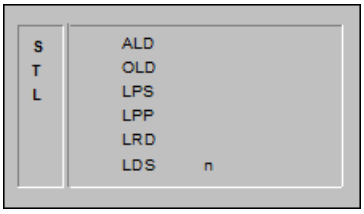
## Logic stack (bit logic instructions)

| Inputs/outputs | | Addresses | Data types none |
|---|---|---|---|
| | | **Addresses for the instruction LDS** | n (1-8) |
| Memory areas (Page 262) | ENO (Page 259) | Supported instructions (Page 259) | Mnemonics (Page 260) |

| | |
|---|---|
|  | The instruction **Link the first and second stack level with AND** (ALD) links the values of the first and second level of the stack with AND. The result is loaded to the top of the stack. After the ULD instruction is performed, the stack contains one bit less.<br>The instruction **Link the first and second stack level with OR** (OLD) links the values of the first and second level of the stack with OR. The result is loaded to the top of the stack. After the OLD instruction is performed, the stack contains one bit less.<br>The instruction **Duplicate topmost stack value** (LPS) duplicates the topmost stack value and inserts it into the stack. The bottom stack value is moved out of the stack and is lost.<br>The instruction **Move topmost stack value out of stack** (LPP) removes the topmost value from the stack. The second stack value is moved to the top of the stack.<br>The instruction **Copy second stack value** (LRD) copies the second stack value to the top of the stack. No value is loaded to the stack and no value is moved out of the stack. The previous topmost stack value is overwritten with the new value.<br>The instruction **Load stack** (LDS) duplicates the stack bit n in stack and stores it in the topmost stack value. The bottom stack value is moved out of the stack and is lost. |

## ENO (bit logic instructions)

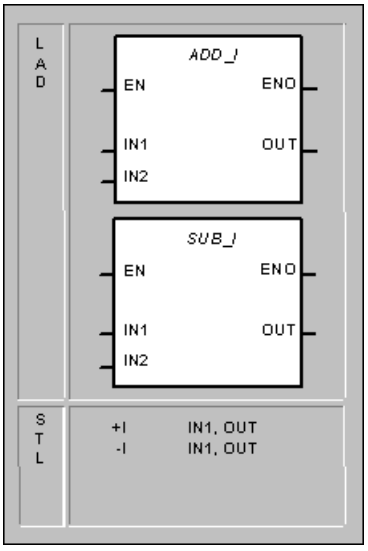
| Inputs/outputs | | Addresses | Data types |
|---|---|---|---|
| | | None | None |
| Memory areas (Page 262) | ENO (Page 259) | Supported instructions (Page 259) | Mnemonics (Page 260) |

| | |
|---|---|
|  | **ENO** is a Boolean output from LAD boxes. If a signal flow is present at a box at input EN and the box is being executed without any error, the ENO output transfers the signal flow to the next element. ENO can be used as a release bit that indicates the successful performance of an instruction.<br>The ENO bit is used together with the topmost stack value to influence the signal flow for the performance of subsequent instructions.<F0><br>STL instructions do not have an EN input. The topmost stack value must logically be 1 for the instruction to be performed.<br>In STL there is no ENO output, but the STL instructions corresponding to the LAD instructions with ENO outputs also set a special ENO bit. The instruction **AND ENO (AENO)** can access this bit. The AENO instruction has the same effect as the ENO bit of a box. The AENO instruction is only available in STL.<br>AENO links the ENO bit with the topmost stack value using AND. The result of the AND instruction is the new value at the top of the stack. |

### 6.1.1.2 Integer maths

## Add, subtract integers (16 bits)

| Inputs/outputs | | Addresses | Data types |
|---|---|---|---|
| IN1, IN2 | | VW, T, C, IW, QW, MW, AC, constant, LW | INT |
| OUT | | VW, T, C, IW, QW, MW, AC, LW | INT |
| Memory areas (Page 262) | ENO (Page 259) | Supported instructions (Page 259) | Mnemonics (Page 260) |




The instructions **add integers (16 bits)** and **subtract integers (16 bits)** add or subtract two integers (16 bits) and return a result (16 bits) to OUT.

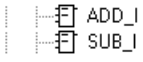
In LAD:

- IN1 + IN2 = OUT
- IN1 - IN2 = OUT

In STL:

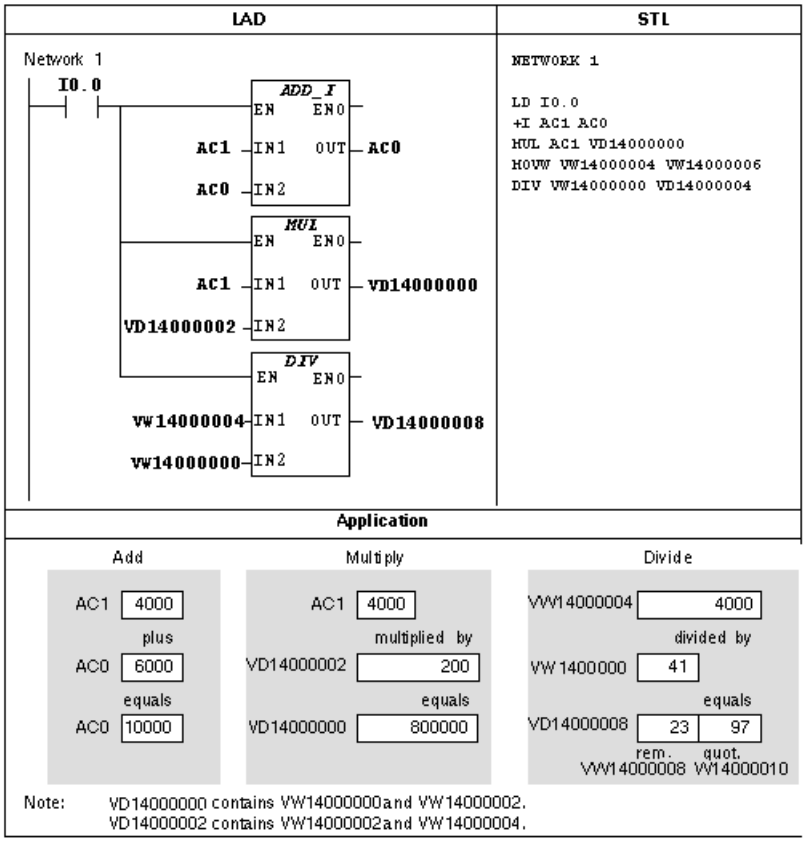
- IN1 + OUT = OUT
- OUT - IN1 = OUT

## Supporting CPUs




| | | | |
|---|---|---|---|
| 828D | 808D-PPU14x | 802Dsl TM value | SINUMERIK 802D |
| 828D Step 2 | 808D-PPU15x | 802Dsl TM plus | SINUMERIK 802SC |
| | 808D-PPU16x | 802Dsl TM pro | |
| | | 802Dsl CU pro | |
| | | 802Dsl GN plus | |
| | | 802Dsl GN pro | |

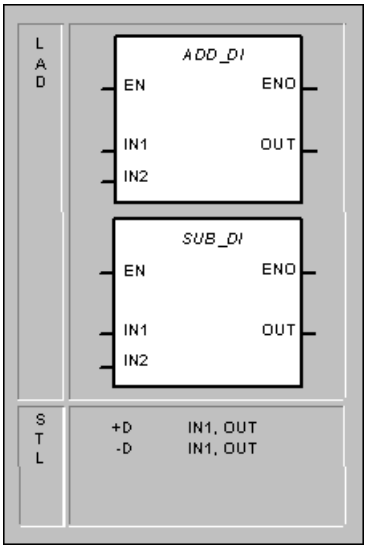
## Example




## Add, subtract integers (32 bits)

## Instructions for adding and subtracting integers (32 bits)

| Inputs/outputs | | Addresses | Data types |
|---|---|---|---|
| IN1, IN2 | | VD, ID, QD, MD, AC, constant, LD | DINT |
| OUT | | VD, ID, QD, MD, AC, LD | DINT |
| Memory areas (Page 262) | ENO (Page 259) | Supported instructions (Page 259) | Mnemonics (Page 260) |




The instructions **add integers (32 bits)** and **subtract integers (32 bits)** add or subtract two integers (32 bits) and return a result (32 bits) to OUT.

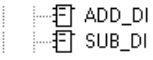
In LAD:

- IN1 + IN2 = OUT
- IN1 - IN2 = OUT

In STL:

- IN1 + OUT = OUT
- OUT - IN1 = OUT

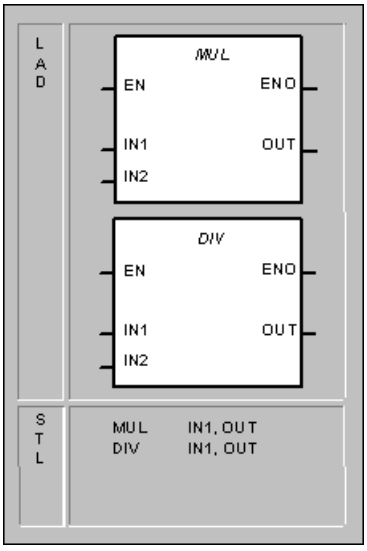
## Supporting CPUs




| | | | |
|---|---|---|---|
| 828D | 808D-PPU14x | 802Dsl TM value | SINUMERIK 802D |
| 828D Step 2 | 808D-PPU15x | 802Dsl TM plus | SINUMERIK 802SC |
| | 808D-PPU16x | 802Dsl TM pro | |
| | | 802Dsl CU pro | |
| | | 802Dsl GN plus | |
| | | 802Dsl GN pro | |

## Multiply, divide integers (16 bits) into integers (32 bits)

| Inputs/outputs | | Addresses | Data types |
|---|---|---|---|
| IN1, IN2 | | VW, T, C, IW, QW, MW, AC, constant, LW | INT |
| OUT | | AC (only for MUL), VD, ID, QD, MD, LD | DINT |
| Memory areas (Page 262) | ENO (Page 259) | Supported instructions (Page 259) | Mnemonics (Page 260) |




The instruction **multiply integers (16 bits) into integers (32 bits)**
**multiplies** two integers (16 bits) and delivers a 32-bit result.

The instruction **divide integers (16 bits) into integers (32 bits)**
**divides** two integers (16 bits) and delivers a 32-bit result, comprised of a 16-bit remainder (most significant) )and a 16-bit quotient (least significant).

When multiplying in STL, the least significant word (16 bits)
of OUT (32 bits) is used as one of the factors.
When dividing in STL, the least significant word (16 bits) of OUT (32 bits) is used as the dividend.

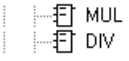
In LAD:

- IN1 * IN2 = OUT
- IN1 / IN2 = OUT

In STL:

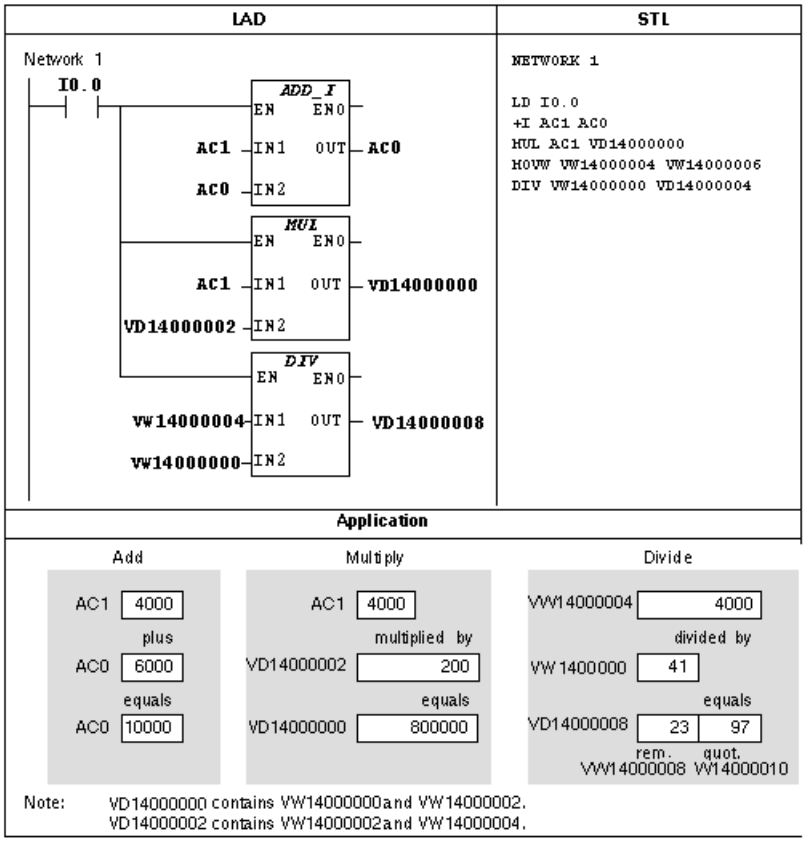
- IN1 * OUT = OUT
- OUT / IN1 = OUT

## Supporting CPUs




| | | | |
|---|---|---|---|
| 828D | 808D-PPU14x | 802Dsl TM value | SINUMERIK 802D |
| 828D Step 2 | 808D-PPU15x | 802Dsl TM plus | SINUMERIK 802SC |
| | 808D-PPU16x | 802Dsl TM pro | |
| | | 802Dsl CU pro | |
| | | 802Dsl GN plus | |
| | | 802Dsl GN pro | |

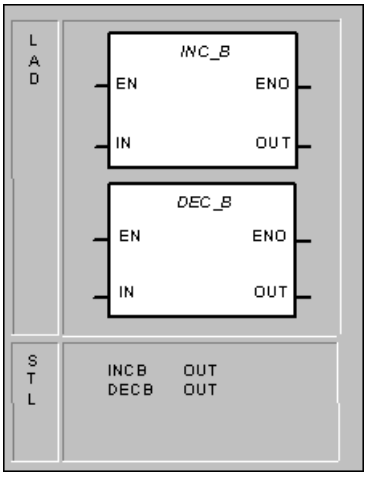
## Example



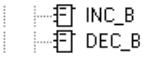

## Increase, decrease byte by 1

## Instructions for increasing and decreasing a byte by 1

| Inputs/outputs | | Addresses | Data types |
|---|---|---|---|
| IN | | VB, IB, QB, MB, AC, constant, LB | BYTE |
| OUT | | VB, IB, QB, MB, AC, LB | BYTE |
| Memory areas (Page 262) | ENO (Page 259) | Supported instructions (Page 259) | Mnemonics (Page 260) |

| | |
|---|---|
|  | The instructions **increase byte by 1** and **decrease byte by 1** add or subtract the value 1 to or from the input byte (IN) and store the result in the variable specified by OUT.<br>The instructions increase byte by 1 and decrease byte by 1 are unsigned.<br><br>In LAD:<br>• IN + 1 = OUT<br>• IN - 1 = OUT<br>In STL:<br>• OUT + 1 = OUT<br>• OUT - 1 = OUT |

## Supporting CPUs

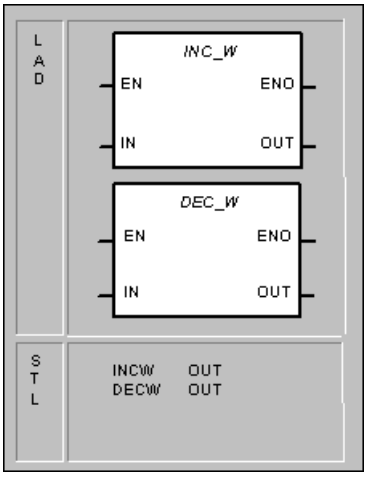



| | | | |
|---|---|---|---|
| 828D | 808D-PPU14x | 802Dsl TM value | SINUMERIK 802D |
| 828D Step 2 | 808D-PPU15x | 802Dsl TM plus | SINUMERIK 802SC |
| | 808D-PPU16x | 802Dsl TM pro | |
| | | 802Dsl CU pro | |
| | | 802Dsl GN plus | |
| | | 802Dsl GN pro | |

## Increase, decrease word by 1

## Instructions for increasing and decreasing a word by 1

| Inputs/outputs | | Addresses | Data types |
|---|---|---|---|
| IN | | VW, T, C, IW, QW, MW, AC, constant, LW | INT |
| OUT | | VW, T, C, IW, QW, MW, AC, LW | INT |
| Memory areas (Page 262) | ENO (Page 259) | Supported instructions (Page 259) | Mnemonics (Page 260) |




The instructions**increase word by 1** and **decrease word by 1**
add or subtract the value 1 to or from the input word (IN) and store the result in OUT.

The instructions increase word by 1 and decrease word by 1
are signed (116#7FFF > 16#8000).

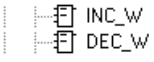
In LAD:

- IN + 1 = OUT
- IN - 1 = OUT

In STL:

- OUT + 1 = OUT
- OUT - 1 = OUT

## Supporting CPUs

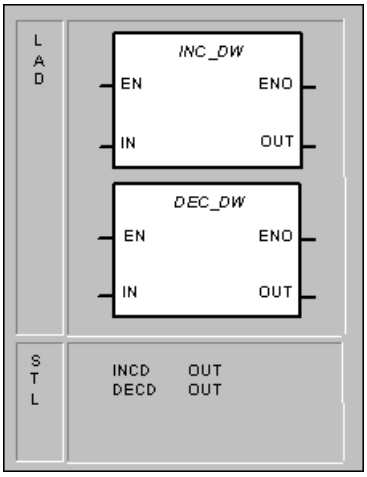



| | | | |
|---|---|---|---|
| 828D | 808D-PPU14x | 802Dsl TM value | SINUMERIK 802D |
| 828D Step 2 | 808D-PPU15x | 802Dsl TM plus | SINUMERIK 802SC |
| | 808D-PPU16x | 802Dsl TM pro | |
| | | 802Dsl CU pro | |
| | | 802Dsl GN plus | |
| | | 802Dsl GN pro | |

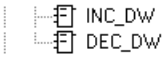
## Increase, decrease double word by 1

## Instructions for increasing and decreasing a double-word by 1

| Inputs/outputs | | Addresses | Data types |
|---|---|---|---|
| IN | | VD, ID, QD, MD, AC, constant, LD | DINT |
| OUT | | VD, ID, QD, MD, AC, LD | DINT |
| Memory areas (Page 262) | ENO (Page 259) | Supported instructions (Page 259) | Mnemonics (Page 260) |

| | |
|---|---|
|  | The instructions**increase double word by 1** and **decrease double word by**<br>**1** add or subtract the value 1 to or from the input double word<br>(IN) and store the result in OUT.<br><br>In LAD:<br>• IN + 1 = OUT<br>• IN - 1 = OUT<br>The instructions increase double word by 1 and decrease double word<br>by 1 are signed (16#7FFFFFFF > 16#80000000).<br><br>In STL:<br>• OUT + 1 = OUT<br>• OUT - 1 = OUT |

### Supporting CPUs



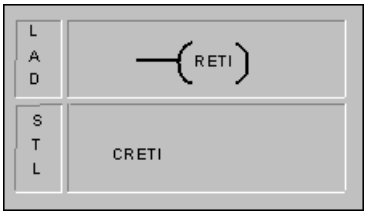

| | | | |
|---|---|---|---|
| 828D | 808D-PPU14x | 802Dsl TM value | SINUMERIK 802D |
| 828D Step 2 | 808D-PPU15x | 802Dsl TM plus | SINUMERIK 802SC |
| | 808D-PPU16x | 802Dsl TM pro | |
| | | 802Dsl CU pro | |
| | | 802Dsl GN plus | |
| | | 802Dsl GN pro | |

## 6.1.1.3 Interrupt instructions

### Conditionally end interrupt program (INT)

### Instruction for conditionally ending an interrupt program

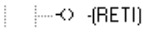
| | | Addresses | Data types |
|---|---|---|---|
| | | None | None |
| Memory areas (Page 262) | ENO (Page 259) | Supported instructions (Page 259) | Mnemonics (Page 260) |

| | |
|---|---|
|  | The conditionally end interrupt program instruction ends a network in your project with the commands of your interrupt program. To insert an interrupt, select "Interrupts" in the instruction tree and double-click "RETI".<br>**Guidelines for the use of interrupt programs**<br>With the interrupt processing routine you can quickly respond to particular internal or external events. You should set up your interrupt program such that it performs a specific task and then transfers control back to the main program. Program interrupt programs that are as short as possible with precise information, so that the programs can be processed quickly and other processes are not interrupted for long. If you disregard this guideline, it can lead to unpredictable states, which can interfere with the operation of devices controlled by the main program. For interrupt programs, the motto "the shorter, the better" applies.<br>Interrupt (Page 227) |

**Note**

At the end of an interrupt program, the Programming Tool automatically enters the instruction absolutely end interrupt program.

### Supporting CPUs

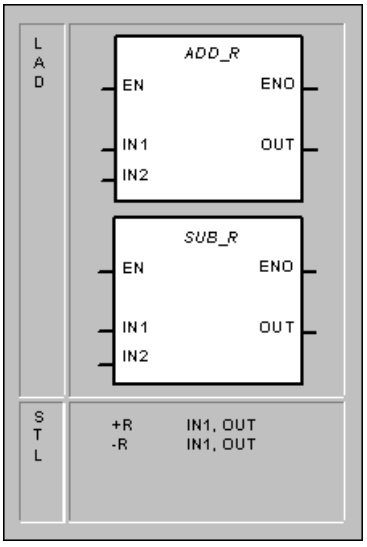



828D
828D Step 2

## 6.1.1.4 Floating-point maths

### Add, subtract floating-point numbers

| Inputs/outputs | | Operands | Data types |
|---|---|---|---|
| IN1, IN2 | | VD, ID, QD, MD, AC, constant, LD | REAL |
| OUT | | VD, ID, QD, MD, AC, LD | REAL |
| Memory areas (Page 262) | ENO (Page 259) | Supported operations (Page 259) | Mnemonics (Page 260) |




The operations **add real numbers** and **subtract real numbers** add or subtract two real numbers (32 bits) and deliver one real number (32 bits) as a result (OUT).

In LAD:

- IN1 + IN2 = OUT
- IN1 - IN2 = OUT

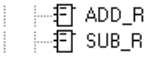
In STL:

- IN1 + OUT = OUT
- OUT - IN1 = OUT

**Note**

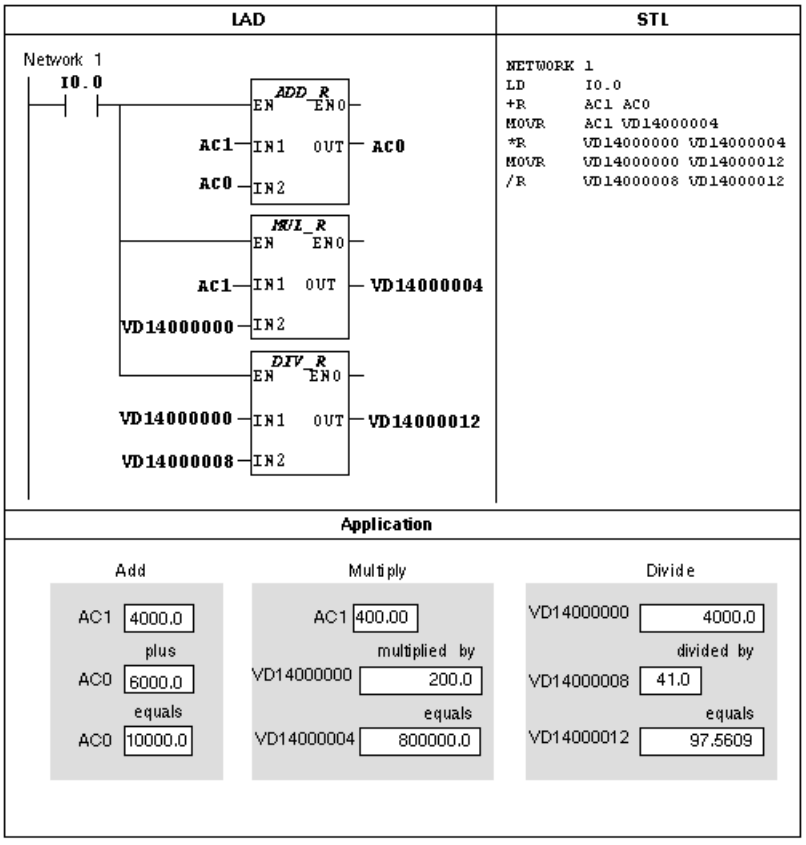
Floating-point numbers or real numbers are displayed in the format described in the directive ANSI/IEEE 754-1985 (single-precision). For detailed information on these numbers, refer to the guidelines.

## Supporting CPUs



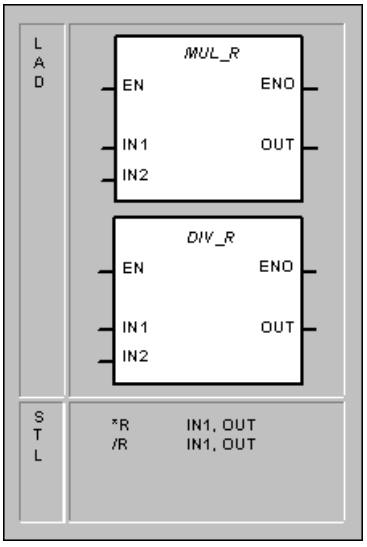

| | | | |
|---|---|---|---|
| 828D | 808D-PPU14x | 802Dsl TM value | SINUMERIK 802D |
| 828D Step 2 | 808D-PPU15x | 802Dsl TM plus | SINUMERIK 802SC |
| | 808D-PPU16x | 802Dsl TM pro | |
| | | 802Dsl CU pro | |
| | | 802Dsl GN plus | |
| | | 802Dsl GN pro | |

## Example




## Multiply, divide floating-point numbers

| Inputs/outputs | | Operands | Data types |
|---|---|---|---|
| IN1, IN2 | | VD, ID, QD, MD, AC, constant, LD | REAL |
| OUT | | VD, ID, QD, MD, AC, LD | REAL |
| Memory areas (Page 262) | ENO (Page 259) | Supported operations (Page 259) | Mnemonics (Page 260) |




The operation **multiply real numbers** multiplies two real numbers (32 bits) and delivers one real number (32 bits) as a result (OUT).

The operation **divide real numbers** divides two real numbers (32 bits) and delivers one real number (32 bits) as the quotient.

In LAD:

- IN1 * IN2 = OUT
- IN1 / IN2 = OUT

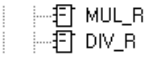
In STL:

- IN1 * OUT = OUT
- OUT / IN1 = OUT

**Note**

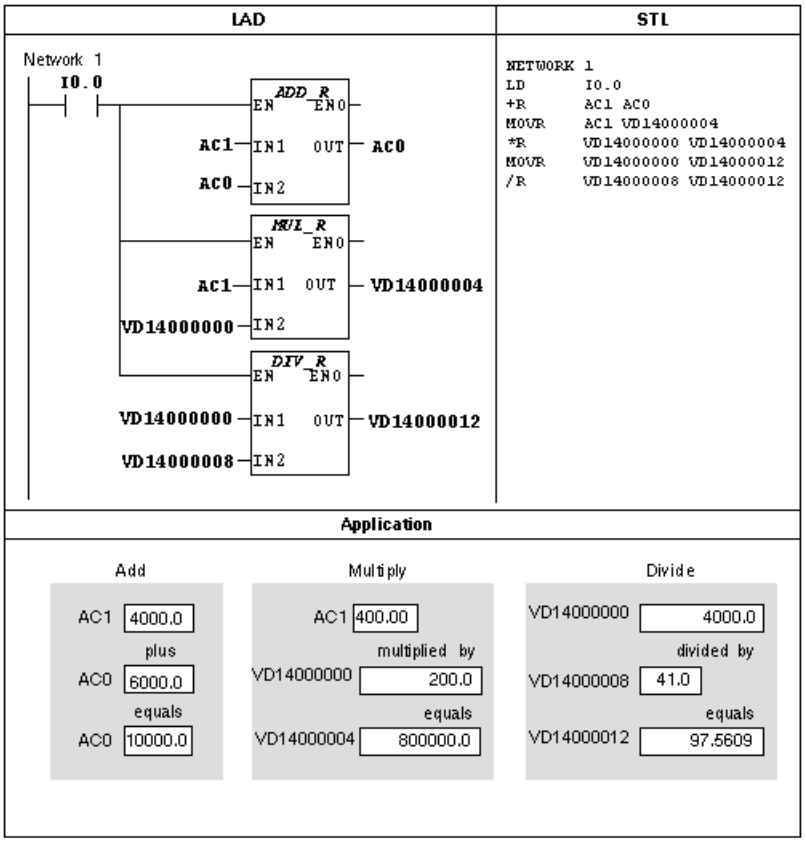
Floating-point numbers or real numbers are displayed in the format described in the directive ANSI/IEEE 754-1985 (single-precision). For detailed information on these numbers, refer to the guidelines.

## Supporting CPUs




| | | | |
|---|---|---|---|
| 828D | 808D-PPU14x | 802Dsl TM value | SINUMERIK 802D |
| 828D Step 2 | 808D-PPU15x | 802Dsl TM plus | SINUMERIK 802SC |
| | 808D-PPU16x | 802Dsl TM pro | |
| | | 802Dsl CU pro | |
| | | 802Dsl GN plus | |
| | | 802Dsl GN pro | |

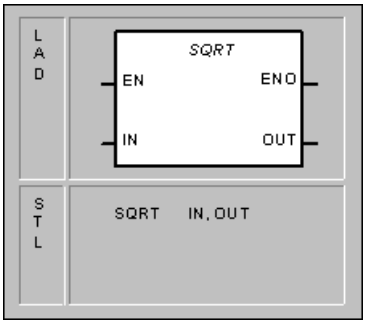
## Example



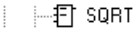

## Extract square root of a floating-point number

## Operation for extracting the square root of a real number

| Inputs/outputs | | Operands | Data types |
|---|---|---|---|
| IN | | VD, ID, QD, MD, AC, constant, LD | REAL |
| OUT | | VD, ID, QD, MD, AC, LD | REAL |
| Memory areas (Page 262) | ENO (Page 259) | Supported operations (Page 259) | Mnemonics (Page 260) |

| | |
|---|---|
|  | The operation **extract square root of a real number** extracts the<br>square root of a real number (32 bits) (IN) and delivers a real number (32 bits) as the result (OUT). This is shown by the following equation:<br><br>√IN = OUT |

### Supporting CPUs



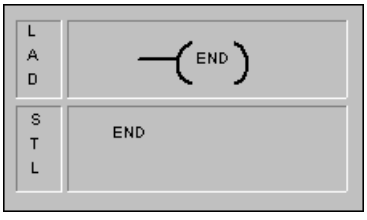

| | | | |
|---|---|---|---|
| 828D | 808D-PPU14x | 802Dsl TM value | SINUMERIK 802D |
| 828D Step 2 | 808D-PPU15x | 802Dsl TM plus | SINUMERIK 802SC |
| | 808D-PPU16x | 802Dsl TM pro | |
| | | 802Dsl CU pro | |
| | | 802Dsl GN plus | |
| | | 802Dsl GN pro | |

## 6.1.1.5 Program control instructions

### Conditionally end processing (END)

### Instruction for conditionally ending the processing

| Inputs/outputs | | Addresses | Data types |
|---|---|---|---|
| | | None | None |
| | | | |
| Memory areas (Page 262) | ENO (Page 259) | Supported instructions (Page 259) | Mnemonics (Page 260) |

| | |
|---|---|
|  | The instruction **conditionally end processing** (END) ends<br>the main program depending on the state of the preceding logic operation. |

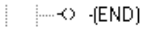
**Note**

You can use the instruction **conditionally end processing** in the main program, but not in subprograms or interrupt programs.

**Note**

The Programming Tool automatically inserts an absolute end at the end of the main program.

## Supporting CPUs




| | | | |
|---|---|---|---|
| 828D | 808D-PPU14x | 802Dsl TM value | SINUMERIK 802D |
| 828D Step 2 | 808D-PPU15x | 802Dsl TM plus | SINUMERIK 802SC |
| | 808D-PPU16x | 802Dsl TM pro | |
| | | 802Dsl CU pro | |
| | | 802Dsl GN plus | |
| | | 802Dsl GN pro | |

## Going into STOP (program control instructions)

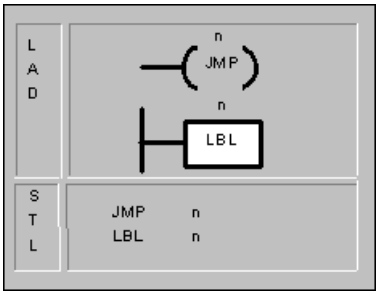
| Inputs/outputs | | Addresses | Data types |
|---|---|---|---|
| | | None | None |
| | | | |
| Memory areas (Page 262) | ENO (Page 259) | Supported instructions (Page 259) | Mnemonics (Page 260) |

| | |
|---|---|
|  | The operation **Go to STOP** immediately ends the processing of the user program by transferring the CPU from RUN mode to STOP mode. |

## Jump to, define jump mark

## Instruction for jumping to a jump mark

| Inputs/outputs | | Addresses | Data types |
|---|---|---|---|
| | | n: constant (0 to 127) | WORD |
| | | | |
| Memory areas (Page 262) | ENO (Page 259) | Supported instructions (Page 259) | Mnemonics (Page 260) |

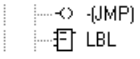



The instruction **jump to jump mark** branches within the program to the specified jump mark n. If a jump is executed, the topmost stack value is always 1.

The instruction **define jump mark** (LBL) specifies the destination (n) to be jumped to.

The jump instruction and the corresponding jump mark must be both either in the main program or in a subprogram or an interrupt program. You cannot jump from the main program to a jump mark located in a subprogram or an interrupt program. Nor can you jump from a subprogram or an interrupt program to a jump mark located outside of the respective subprogram or interrupt program.

## Supporting CPUs

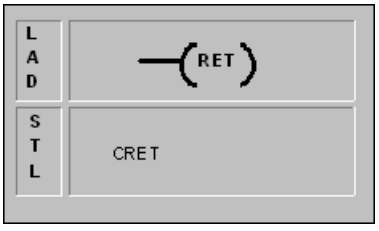



| | | | |
|---|---|---|---|
| 828D | 808D-PPU14x | 802Dsl TM value | SINUMERIK 802D |
| 828D Step 2 | 808D-PPU15x | 802Dsl TM plus | SINUMERIK 802SC |
| | 808D-PPU16x | 802Dsl TM pro | |
| | | 802Dsl CU pro | |
| | | 802Dsl GN plus | |
| | | 802Dsl GN pro | |

## Conditionally end subprogram (RET)

## Instruction for conditionally ending a subprogram

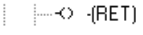
| Inputs/outputs | | Addresses | Data types |
|---|---|---|---|
| | | None | None |
| | | | |
| Memory areas (Page 262) | ENO (Page 259) | Supported instructions (Page 259) | Mnemonics (Page 260) |

| | |
|---|---|
|  | The instruction **conditionally end subprogram** ends a subprogram depending on the state of the preceding logic operation.<br>Subprogram (Page 223) |

**Note**

The Programming Tool automatically adds the instruction absolutely end subprogram at the end of a subprogram.

## Supporting CPUs



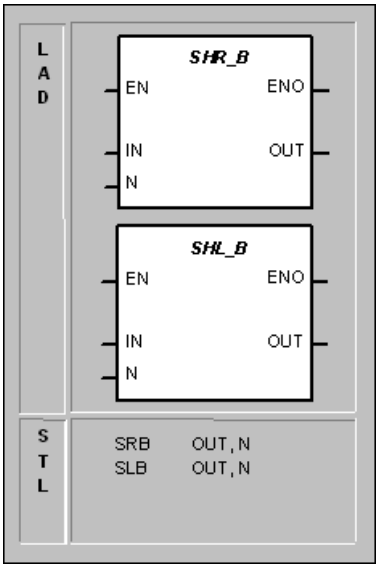

| | | | |
|---|---|---|---|
| 828D | 808D-PPU14x | 802Dsl TM value | SINUMERIK 802D |
| 828D Step 2 | 808D-PPU15x | 802Dsl TM plus | SINUMERIK 802SC |
| | 808D-PPU16x | 802Dsl TM pro | |
| | | 802Dsl CU pro | |
| | | 802Dsl GN plus | |
| | | 802Dsl GN pro | |

### 6.1.1.6 Shift/rotate instructions

## Shift byte left/right

## Instructions for shifting a byte right or left

| Inputs/outputs | | Addresses | Data types |
|---|---|---|---|
| IN | | VB, IB, QB, MB, LB, AC, constant | BYTE |
| N | | VB, IB, QB, MB, AC, LB, constant | BYTE |
| OUT | | VB, IB, QB, MB, AC, LB | BYTE |
| Memory areas (Page 262) | ENO (Page 259) | Supported instructions (Page 259) | Mnemonics (Page 260) |

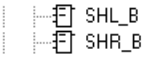



The instructions **shift byte right and shift byte left** shift the value of the input byte (IN) to the right or left by the shift number (N) and load the result into the output byte (OUT).

The shift instructions assign zeros to the locations from where the bits have been shifted. If the shift value (N) is greater than or equal to 8, then the value is shifted a maximum of 8 times.

The instructions shift byte right and shift byte left are unsigned.

## Supporting CPUs

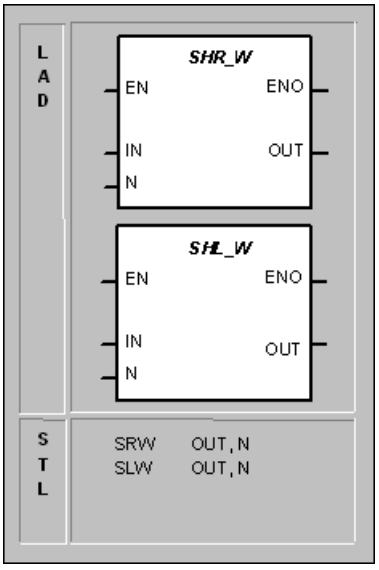



| | | | |
|---|---|---|---|
| 828D | 808D-PPU14x | 802Dsl TM value | SINUMERIK 802D |
| 828D Step 2 | 808D-PPU15x | 802Dsl TM plus | SINUMERIK 802SC |
| | 808D-PPU16x | 802Dsl TM pro | |
| | | 802Dsl CU pro | |

802Dsl GN plus
802Dsl GN pro

## Shift word left/right

| Inputs/outputs | | Addresses | Data types |
|---|---|---|---|
| IN | | VW, T, C, IW, QW, MW, AC, constant, LW | WORD |
| N | | VB, IB, QB, MB, AC, constant, LB | BYTE |
| OUT | | VW, T, C, IW, QW, MW, AC, LW | WORD |
| Memory areas (Page 262) | ENO (Page 259) | Supported instructions (Page 259) | Mnemonics (Page 260) |




The instructions **rotate word to the right** and **rotate word to the left**
rotate the value of the input word (IN) to the right or left by the shift number (N)
and load the result into the output word (OUT).

The shift instructions assign zeros to the locations from where the bits have been shifted. If the shift value (N) is greater
than or equal to 16, then the value is shifted a maximum of 16 times.
The instructions shift word right and shift word left are unsigned.

## Supporting CPUs

SHL_B
SHR_B

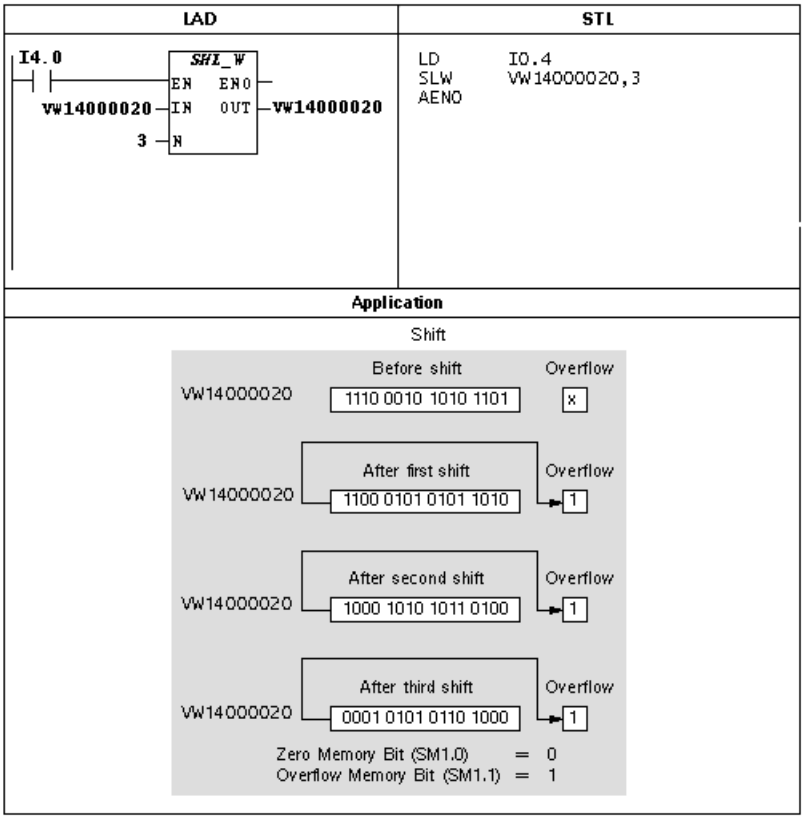
828D
828D Step 2

808D-PPU14x
808D-PPU15x
808D-PPU16x

802Dsl TM value
802Dsl TM plus
802Dsl TM pro
802Dsl CU pro
802Dsl GN plus
802Dsl GN pro

SINUMERIK 802D
SINUMERIK 802SC

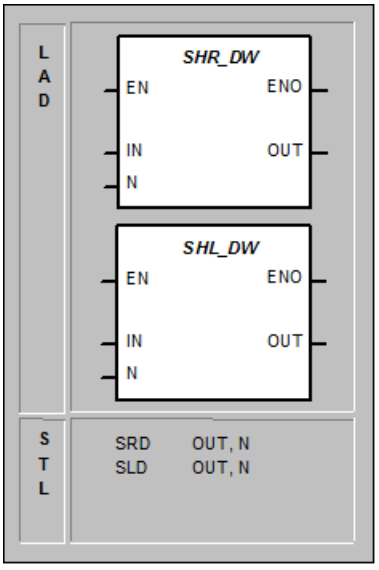
## Example




## Shift double word left/right

## Instructions for shifting a double-word right or left

| Inputs/outputs | | Addresses | Data types |
|---|---|---|---|
| IN (LAD) | | VD, ID, QD, MD, AC, constant, LD | DWORD |
| N | | VB, IB; QB, MB, AC, constant, LB | BYTE |
| OUT | | VD, ID, QD, MD, AC, LD | DWORD |
| Memory areas (Page 262) | ENO (Page 259) | Supported instructions (Page 259) | Mnemonics (Page 260) |

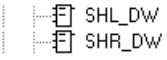



The instructions **shift double word right** and **shift double word left** shift the value of the input double word (IN) to the right or left by the shift number (N) and load the result into the output double word (OUT).

The shift instructions assign zeros to the locations from where the bits have been shifted. If the shift value (N) is greater than or equal to 32, then the value is shifted a maximum of 32 times.

The instructions shift double word right and shift double word left are unsigned.

### Supporting CPUs



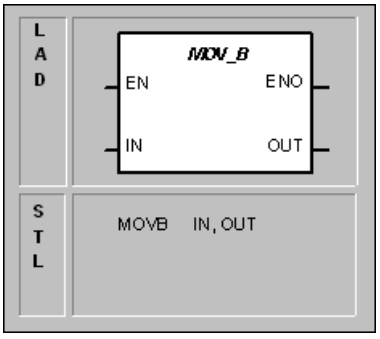

| | | | |
|---|---|---|---|
| 828D | 808D-PPU14x | 802Dsl TM value | SINUMERIK 802D |
| 828D Step 2 | 808D-PPU15x | 802Dsl TM plus | SINUMERIK 802SC |
| | 808D-PPU16x | 802Dsl TM pro | |
| | | 802Dsl CU pro | |
| | | 802Dsl GN plus | |
| | | 802Dsl GN pro | |

## 6.1.1.7 Transfer instructions

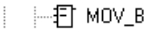
### Transfer byte

### Operation for transferring a byte

| Inputs/outputs | | Operands | Data types |
|---|---|---|---|
| IN | | VB, IB, QB, MB, AC, constant, LB | BYTE |
| OUT | | VB, IB, QB, MB, AC, LB | BYTE |
| Memory areas (Page 262) | ENO (Page 259) | Supported operations (Page 259) | Mnemonics (Page 260) |

| | |
|---|---|
|  | The operation **transfer byte** (MOVB) transfers the input byte (IN) to the output byte (OUT). This does not change<br>the input byte. |

## Supporting CPUs

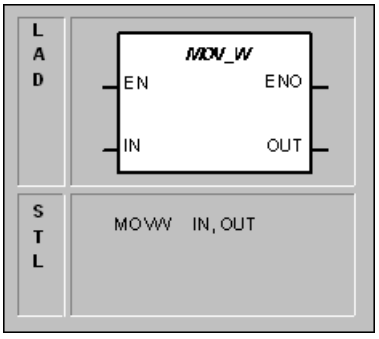



| | | | |
|---|---|---|---|
| 828D | 808D-PPU14x | 802Dsl TM value | SINUMERIK 802D |
| 828D Step 2 | 808D-PPU15x | 802Dsl TM plus | SINUMERIK 802SC |
| | 808D-PPU16x | 802Dsl TM pro | |
| | | 802Dsl CU pro | |
| | | 802Dsl GN plus | |
| | | 802Dsl GN pro | |

## Transfer word

## Transfer word operation

| Inputs/outputs | | Operands | Data types |
|---|---|---|---|
| IN | | VW, T, C, IW, QW, MW, AC, constant, LW | WORD, INT |
| OUT | | VW, T, C, IW, QW, MW, AC, LW | WORD, INT |
| Memory areas (Page 262) | ENO (Page 259) | Supported operations (Page 259) | Mnemonics (Page 260) |

| | |
|---|---|
|  | The operation **transfer word** (MOVW) transfers the input word (IN) to the output word (OUT). This does not change<br>the input word. |

## Supporting CPUs

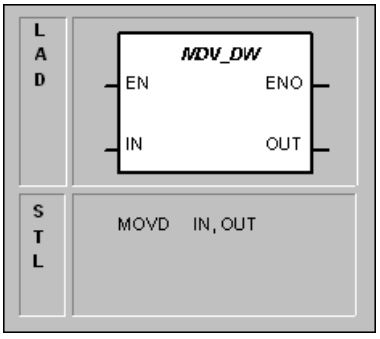
MOV_W

| | | | |
|---|---|---|---|
| 828D | 808D-PPU14x | 802Dsl TM value | SINUMERIK 802D |
| 828D Step 2 | 808D-PPU15x | 802Dsl TM plus | SINUMERIK 802SC |
| | 808D-PPU16x | 802Dsl TM pro | |
| | | 802Dsl CU pro | |
| | | 802Dsl GN plus | |
| | | 802Dsl GN pro | |

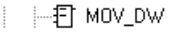
## Transfer double word

## Transfer double word operation

| Inputs/outputs | | Operands | Data types |
|---|---|---|---|
| IN | | VD, ID, QD, MD, AC, constant, LD | DWORD, DINT |
| OUT | | VD, ID, QD, MD, AC, LD | DWORD, DINT |
| Memory areas (Page 262) | ENO (Page 259) | Supported operations (Page 259) | Mnemonics (Page 260) |

| | |
|---|---|
|  | The operation **transfer double word** (MOV_DW) transfers the<br>input double word (IN) to the output double word (OUT). This does not change<br>the input double word. |

## Supporting CPUs

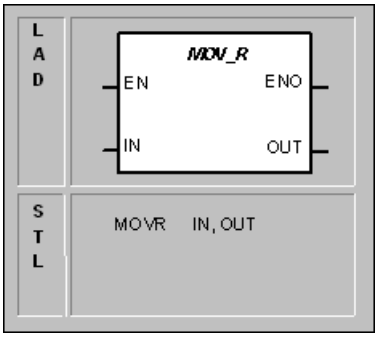



| | | | |
|---|---|---|---|
| 828D | 808D-PPU14x | 802Dsl TM value | SINUMERIK 802D |
| 828D Step 2 | 808D-PPU15x | 802Dsl TM plus | SINUMERIK 802SC |
| | 808D-PPU16x | 802Dsl TM pro | |
| | | 802Dsl CU pro | |
| | | 802Dsl GN plus | |
| | | 802Dsl GN pro | |

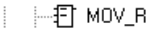
## Transfer floating-point number

## Transfer real number operation

| Inputs/outputs | | Operands | Data types |
|---|---|---|---|
| IN | | VD, ID, QD, MD, AC, constant, LD | REAL |
| OUT | | VD, ID, QD, MD, AC, LD | REAL |
| Memory areas (Page 262) | ENO (Page 259) | Supported operations (Page 259) | Mnemonics (Page 260) |

| | |
|---|---|
|  | The operation **transfer real number** (MOVR) transfers the<br>input real number (double word, 32 bits) (IN) to the output double word (OUT). This does not change the input real number. |

## Supporting CPUs

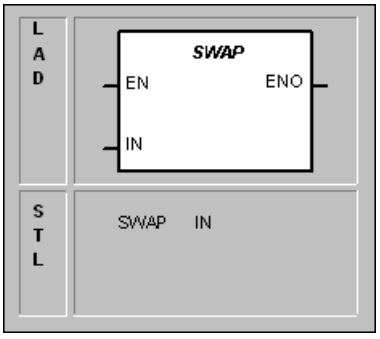



| | | | |
|---|---|---|---|
| 828D | 808D-PPU14x | 802Dsl TM value | SINUMERIK 802D |
| 828D Step 2 | 808D-PPU15x | 802Dsl TM plus | SINUMERIK 802SC |
| | 808D-PPU16x | 802Dsl TM pro | |
| | | 802Dsl CU pro | |
| | | 802Dsl GN plus | |
| | | 802Dsl GN pro | |

## Replace bytes in the word

## Replace bytes in the word operation

| Inputs/outputs | | Operands | Data types |
|---|---|---|---|
| IN | | VW, IW, QW, MW, T, C, AC, LW | WORD |
| | | | |
| Memory areas (Page 262) | ENO (Page 259) | Supported operations (Page 259) | Mnemonics (Page 260) |

| | |
|---|---|
|  | The operation **swap bytes in the word** (SWAP) swaps the most significant byte with the least significant byte of the word (IN). |

## Supporting CPUs

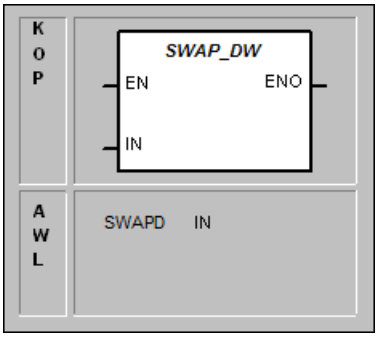
SWAP

| | | | |
|---|---|---|---|
| 828D | 808D-PPU14x | 802Dsl TM value | SINUMERIK 802D |
| 828D Step 2 | 808D-PPU15x | 802Dsl TM plus | SINUMERIK 802SC |
| | 808D-PPU16x | 802Dsl TM pro | |
| | | 802Dsl CU pro | |
| | | 802Dsl GN plus | |
| | | 802Dsl GN pro | |

## Replace bytes in the double word

## Swap bytes in the double word operation

| Inputs/outputs | | Operands | Data types |
|---|---|---|---|
| IN | | VD, ID, QD, MD, AC, LD | DWORD |
| | | | |
| Memory areas (Page 262) | ENO (Page 259) | Supported operations (Page 259) | Mnemonics (Page 260) |

| | |
|---|---|
|  | The operation **swap bytes in the double word** (SWAP_DW) swaps the most significant word with the least significant word of the double word (IN) and within the words the most significant byte with the least significant byte of the word. |

## Supporting CPUs

SWAP_DW

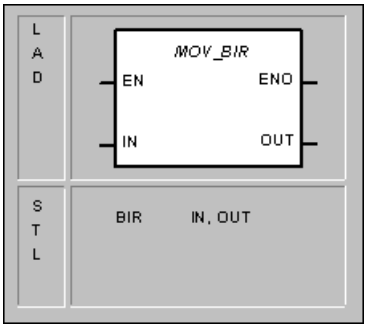
828D Step 2 (as of PLC 07.00)
828D (as of PLC 07.00)

808D-PPU14x (as of PLC 07.00)
808D-PPU15x
808D-PPU16x (as of PLC 7.00)

## Read and transfer byte directly

## Read and transfer byte directly operation

| Inputs/outputs | | Operands | Data types |
|---|---|---|---|
| IN | | IB | BYTE |
| OUT | | VB, IB, QB, MB, AC, LB | BYTE |
| Memory areas (Page 262) | ENO (Page 259) | Supported operations (Page 259) | Mnemonics (Page 260) |

| | |
|---|---|
|  | The operation **read and transfer byte directly** reads the physical input IN (as bytes) and writes the result to OUT. |

## Supporting CPUs

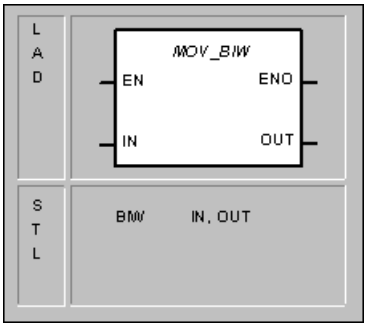
MOV_BIR

828D
828D Step 2

## Write and transfer byte directly

## Write and transfer byte directly operation

| Inputs/outputs | | Operands | Data types |
|---|---|---|---|
| IN | | VB, IB, QB, MB, AC, constant, LB | BYTE |
| OUT | | QB | BYTE |
| Memory areas (Page 262) | ENO (Page 259) | Supported operations (Page 259) | Mnemonics (Page 260) |




The operation **write and transfer byte directly** reads from the
address IN and writes (in bytes) to the physical output OUT.

## Supporting CPUs

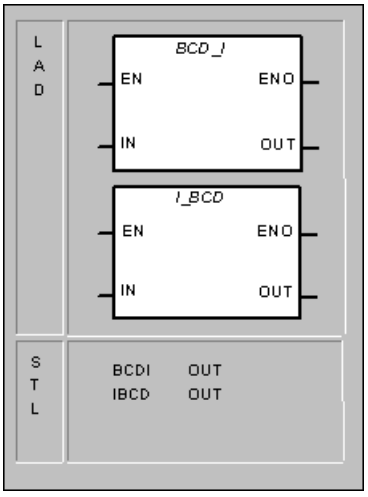
MOV_BIW

828D
828D Step 2

### 6.1.1.8 Conversion instructions

## Convert BCD into integer and vice versa

| Inputs/outputs | | Addresses | Data types |
|---|---|---|---|
| IN (LAD) | | VW, T, C, IW, QW, MW, AC, constant, LW | WORD |
| OUT | | VW, T, C, IW, QW, MW, AC, LW | WORD |
| Memory areas (Page 262) | ENO (Page 259) | Supported instructions (Page 259) | Mnemonics (Page 260) |



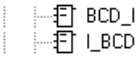

The instruction **convert BCD into integer** converts a binary-coded
decimal value (IN) into an integer value and loads the result into the variable specified by OUT.

The instruction **convert integer into BCD** converts an integer value (IN) into a binary-coded decimal value and loads the result into the variable specified by OUT.

**Error conditions that set ENO = 0:**

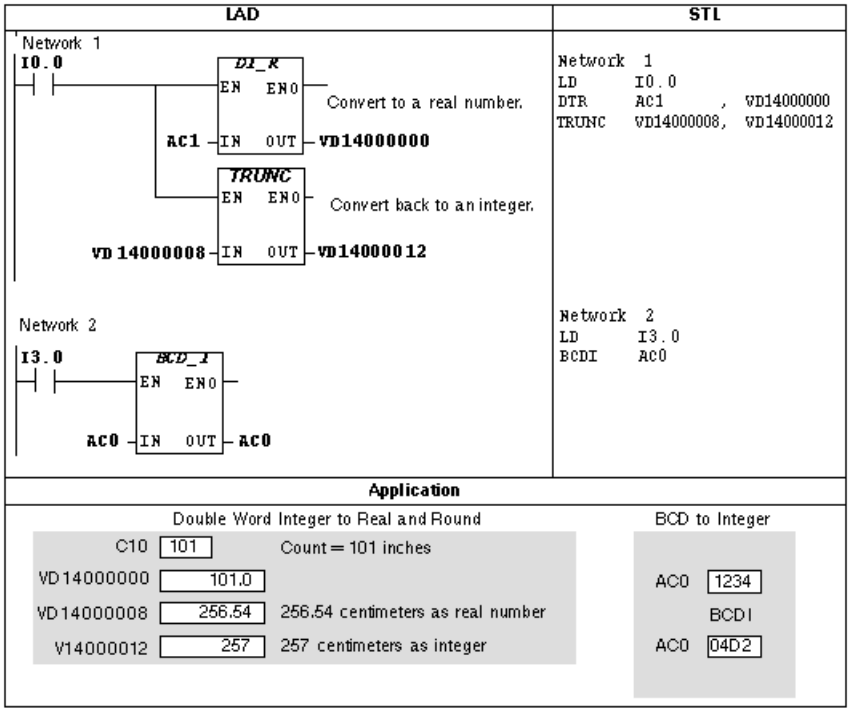
BCD range invalid (valid ranges for the BCD format (Page 216))

## Supporting CPUs



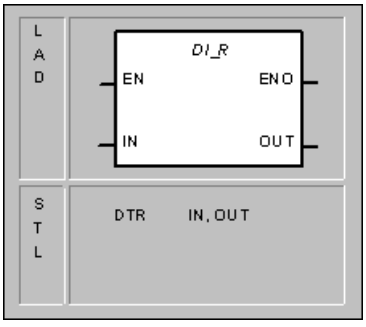

| | | | |
|---|---|---|---|
| 828D | 808D-PPU14x | 802Dsl TM value | SINUMERIK 802D |
| 828D Step 2 | 808D-PPU15x | 802Dsl TM plus | SINUMERIK 802SC |
| | 808D-PPU16x | 802Dsl TM pro | |
| | | 802Dsl CU pro | |
| | | 802Dsl GN plus | |
| | | 802Dsl GN pro | |

## Example




## Convert integer (32 bits) into floating-point number

| Inputs/outputs | | Addresses | Data types |
|---|---|---|---|
| IN | | VD, ID, QD, MD, AC, constant, LD | DINT |
| OUT | | VD, ID, QD, MD, AC, LD | REAL |
| Memory areas (Page 262) | ENO (Page 259) | Supported instructions (Page 259) | Mnemonics (Page 260) |



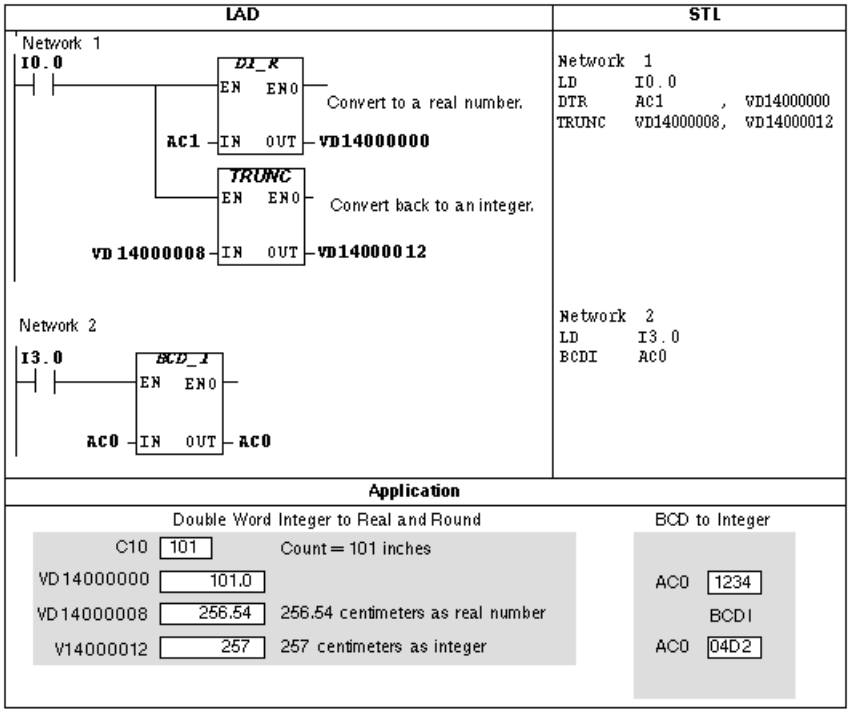

The instruction **convert integer (32 bits) into floating-point number** converts
an integer (32 bits) with sign (IN) into a floating-point number (32
bits) and loads the result into the variable specified by OUT.

## Supporting CPUs

DI_R

| | | | |
|---|---|---|---|
| 828D | 808D-PPU14x | 802Dsl TM value | SINUMERIK 802D |
| 828D Step 2 | 808D-PPU15x | 802Dsl TM plus | SINUMERIK 802SC |
| | 808D-PPU16x | 802Dsl TM pro | |
| | | 802Dsl CU pro | |
| | | 802Dsl GN plus | |
| | | 802Dsl GN pro | |

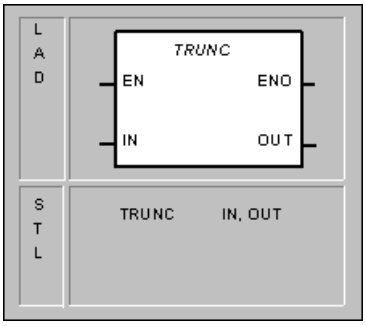
## Example



## Convert floating-point number into integer (32 bits)

## Convert floating-point number into integer (32 bits) instruction

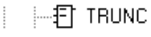
| Inputs/outputs | | Addresses | Data types |
|---|---|---|---|
| IN | | VD, ID, QD, MD, AC, constant, LD | REAL |
| OUT | | VD, ID, QD, MD, AC, LD | DINT |
| Memory areas (Page 262) | ENO (Page 259) | Supported instructions (Page 259) | Mnemonics (Page 260) |




The instruction **convert floating-point number into integer (32 bits)** (TRUNC)
converts a floating-point number (32 bits) (IN) into an integer value (32
bits) with sign and loads the result into the variable specified by OUT. Only the integer part of the floating-point number
is converted (and the rest is discarded).

If the value that you want to convert is not a valid floating-point number
or is too large to be displayed in the output, the overflow bit is set and the output is not changed.

## Supporting CPUs



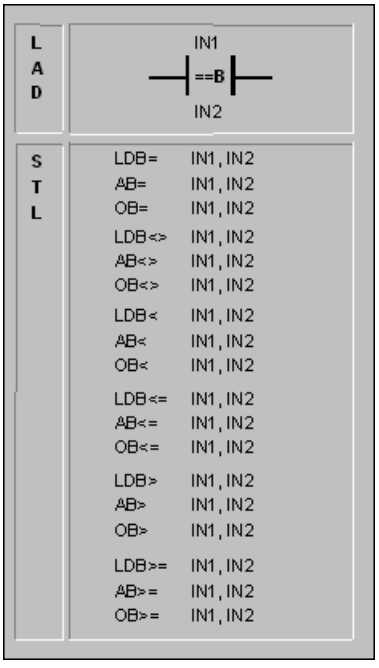

| | | | |
|---|---|---|---|
| 828D | 808D-PPU14x | 802Dsl TM value | SINUMERIK 802D |
| 828D Step 2 | 808D-PPU15x | 802Dsl TM plus | SINUMERIK 802SC |
| | 808D-PPU16x | 802Dsl TM pro | |
| | | 802Dsl CU pro | |
| | | 802Dsl GN plus | |
| | | 802Dsl GN pro | |

### 6.1.1.9 Comparison instructions

## Byte comparison: equal to, not equal to, greater than or equal to, less than or equal to, greater than, less than

## Byte comparison instruction

| Inputs/outputs | | Addresses | Data types |
|---|---|---|---|
| Inputs | | VB, IB, QB, MB, SMB, AC, constant, LB | BYTE |
| Memory areas (Page 262) | ENO (Page 259) | Supported instructions (Page 259) | Mnemonics (Page 260) |



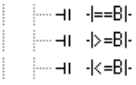

The instruction **byte comparison** compares two values with each other:
IN1 and IN2. The comparison can be as follows: IN1 = IN2, IN1 >= IN2, IN1
<= IN2, IN1 > IN2, IN1 < IN2 or IN1 <> IN2.

Byte comparisons are unsigned.

In LAD, the contact is activated if the comparison is true.

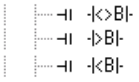
In STL, the instructions load the value 1 as the topmost stack value or combine the value 1 with the topmost stack value
for logic AND or OR if the comparison is true.

## Supporting CPUs




| | | | |
|---|---|---|---|
| 828D | 808D-PPU14x | 802Dsl TM value | SINUMERIK 802D |
| 828D Step 2 | 808D-PPU15x | 802Dsl TM plus | SINUMERIK 802SC |
| | 808D-PPU16x | 802Dsl TM pro | |
| | | 802Dsl CU pro | |

802Dsl GN plus
802Dsl GN pro



828D
828D Step 2

808D-PPU14x
808D-PPU15x
808D-PPU16x

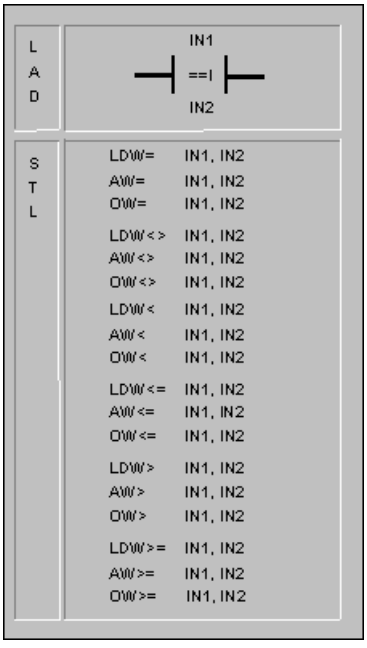
802Dsl TM value
802Dsl TM plus
802Dsl TM pro
802Dsl CU pro
802Dsl GN plus
802Dsl GN pro

SINUMERIK 802D
SINUMERIK 802SC

## Integer comparison: equal to, not equal to, greater than or equal to, less than or equal to, greater than, less than

### Integer comparison instruction

| Inputs/outputs | | Addresses | Data types |
|---|---|---|---|
| Inputs | | VW, T, C, IW, QW, MW, AC, constant, LW | IINT |
| Memory areas (Page 262) | ENO (Page 259) | Supported instructions (Page 259) | Mnemonics (Page 260) |



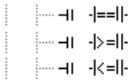

The instruction **integer comparison** compares two values with each other: IN1 and IN2. The comparison can be as follows: IN1 = IN2, IN1 >= IN2, IN1 <= IN2, IN1 > IN2, IN1 < IN2 or IN1 <> IN2.

Integer comparisons are signed (16#7FFF > 16#8000).

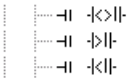
In LAD, the contact is activated if the comparison is true.

In STL, the instructions load the value 1 as the topmost stack value or combine the value 1 with the topmost stack value
for logic AND or OR if the comparison is true.

### Supporting CPUs



| | | | |
|---|---|---|---|
| 828D | 808D-PPU14x | 802Dsl TM value | SINUMERIK 802D |
| 828D Step 2 | 808D-PPU15x | 802Dsl TM plus | SINUMERIK 802SC |
| | 808D-PPU16x | 802Dsl TM pro | |
| | | 802Dsl CU pro | |
| | | 802Dsl GN plus | |
| | | 802Dsl GN pro | |

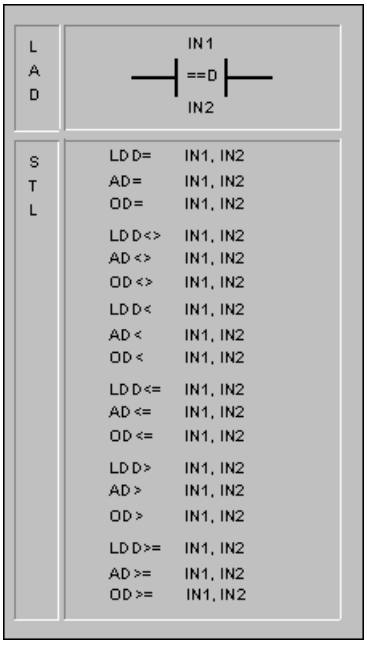


| | | | |
|---|---|---|---|
| 828D | 808D-PPU14x | 802Dsl TM value | SINUMERIK 802D |
| 828D Step 2 | 808D-PPU15x | 802Dsl TM plus | SINUMERIK 802SC |
| | 808D-PPU16x | 802Dsl TM pro | |
| | | 802Dsl CU pro | |
| | | 802Dsl GN plus | |
| | | 802Dsl GN pro | |

## Integer comparison (32 bits): equal to, not equal to, greater than or equal to, less than or equal to, greater than, less than

## Integer comparison (32 bits) instruction

| Inputs/outputs | | Addresses | Data types |
|---|---|---|---|
| Inputs | | VD, ID, QD, MD, AC, constant, LD | DINT |
| Memory areas (Page 262) | ENO (Page 259) | Supported instructions (Page 259) | Mnemonics (Page 260) |



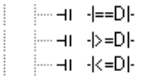

The instruction **integer comparison (32 bits)** compares two
values with each other: IN1 and IN2. The comparison can be as follows: IN1 = IN2, IN1
>= IN2, IN1 <= IN2, IN1 > IN2, IN1 < IN2 or IN1 <> IN2.

Double word comparisons are signed (16#7FFFFFFF > 16#80000000).

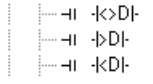
In LAD, the contact is activated if the comparison is true.

In STL, the instructions load the value 1 as the topmost stack value or combine the value 1 with the topmost stack value
for logic AND or OR if the comparison is true.

## Supporting CPUs




| | | | |
|---|---|---|---|
| 828D | 808D-PPU14x | 802Dsl TM value | SINUMERIK 802D |
| 828D Step 2 | 808D-PPU15x | 802Dsl TM plus | SINUMERIK 802SC |
| | 808D-PPU16x | 802Dsl TM pro | |
| | | 802Dsl CU pro | |
| | | 802Dsl GN plus | |
| | | 802Dsl GN pro | |

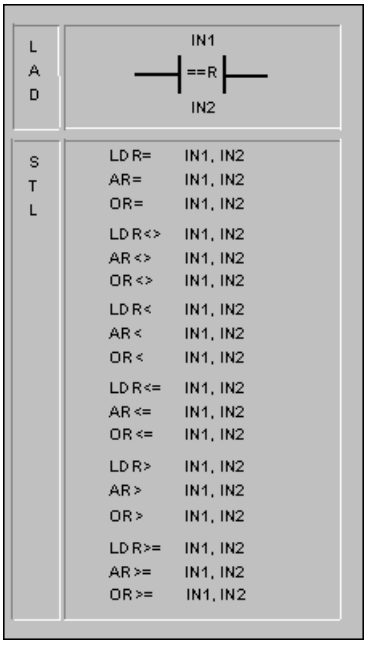



| | | | |
|---|---|---|---|
| 828D | 808D-PPU14x | 802Dsl TM value | SINUMERIK 802D |
| 828D Step 2 | 808D-PPU15x | 802Dsl TM plus | SINUMERIK 802SC |
| | 808D-PPU16x | 802Dsl TM pro | |
| | | 802Dsl CU pro | |
| | | 802Dsl GN plus | |
| | | 802Dsl GN pro | |

## Floating-point number comparison: equal to, not equal to, greater than or equal to, less than or equal to, greater than, less than

### Floating-point number comparison instruction

| Inputs/outputs | | Addresses | Data types |
|---|---|---|---|
| Inputs | | VD, ID, QD, MD, AC, constant, LD | REAL |
| Memory areas (Page 262) | ENO (Page 259) | Supported instructions (Page 259) | Mnemonics (Page 260) |



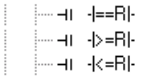

The instruction **floating-point number comparison** compares two values with each other: IN1 and IN2. The comparison can be as follows: IN1 = IN2, IN1 >= IN2, IN1 <= IN2, IN1 > IN2, IN1 < IN2 or IN1 <> IN2.

Floating-point number comparisons are signed.

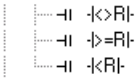
In LAD, the contact is activated if the comparison is true.

In STL, the instructions load the value 1 as the topmost stack value or combine the value 1 with the topmost stack value for logic AND or OR if the comparison is true.

### Supporting CPUs




| | | | |
|---|---|---|---|
| 828D | 808D-PPU14x | 802Dsl TM value | SINUMERIK 802D |
| 828D Step 2 | 808D-PPU15x | 802Dsl TM plus | SINUMERIK 802SC |
| | 808D-PPU16x | 802Dsl TM pro | |
| | | 802Dsl CU pro | |
| | | 802Dsl GN plus | |
| | | 802Dsl GN pro | |



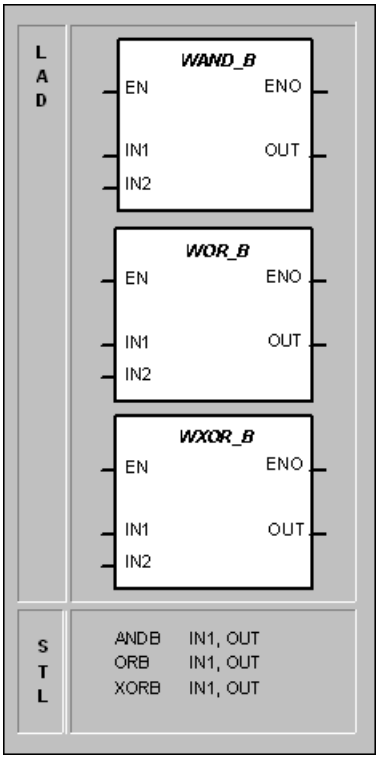
| | | | |
|---|---|---|---|
| 828D | 808D-PPU14x | 802Dsl TM value | SINUMERIK 802D |
| 828D Step 2 | 808D-PPU15x | 802Dsl TM plus | SINUMERIK 802SC |
| | 808D-PPU16x | 802Dsl TM pro | |
| | | 802Dsl CU pro | |
| | | 802Dsl GN plus | |
| | | 802Dsl GN pro | |

### 6.1.1.10 Logic operations

#### Combine bytes for logic AND/OR/EXCLUSIVE OR

#### Combine bytes instruction

| Inputs/outputs | | Addresses | Data types |
|---|---|---|---|
| IN1, IN2 | | VB, IB, QB, MB, AC, constant, LB | BYTE |
| OUT | | VB, IB, QB, MB, AC, LB | BYTE |
| Memory areas (Page 262) | ENO (Page 259) | Supported instructions (Page 259) | Mnemonics (Page 260) |

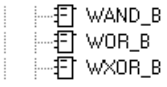



The instruction **combine bytes for logic AND** combines the
corresponding bits of two input bytes for logic AND and loads
the result (OUT) into a byte.

The instruction **combine bytes for logic OR** combines the
corresponding bits of two input bytes for logic OR and loads
the result (OUT) into a byte.

The instruction **combine bytes for logic EXCLUSIVE OR** combines the
corresponding bits of two input bytes for logic EXCLUSIVE OR and loads
the result (OUT) into a byte.

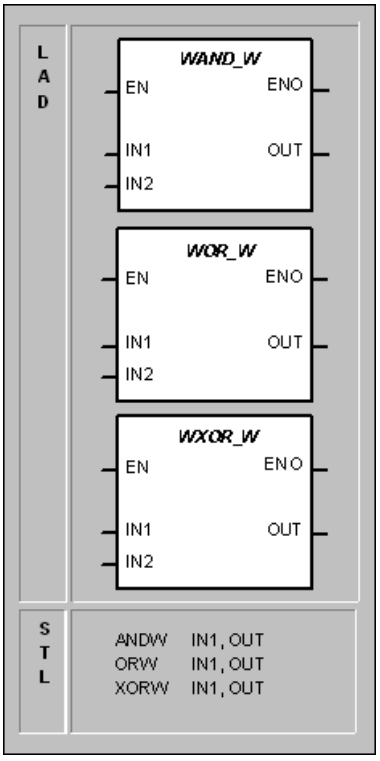
## Supporting CPUs




| | | | |
|---|---|---|---|
| 828D | 808D-PPU14x | 802Dsl TM value | SINUMERIK 802D |
| 828D Step 2 | 808D-PPU15x | 802Dsl TM plus | SINUMERIK 802SC |
| | 808D-PPU16x | 802Dsl TM pro | |
| | | 802Dsl CU pro | |
| | | 802Dsl GN plus | |
| | | 802Dsl GN pro | |

## Combine words for logic AND/OR/EXCLUSIVE OR

| Inputs/outputs | | Addresses | Data types |
|---|---|---|---|
| IN1, IN2 | | VW, T, C, IW, QW, MW, AC, constant, LW | WORD |
| OUT | | VW, T, C, IW, QW, MW, AC, LW | WORD |
| Memory areas (Page 262) | ENO (Page 259) | Supported instructions (Page 259) | Mnemonics (Page 260) |

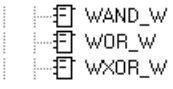



The instruction **combine words for logic AND** combines the corresponding bits of two input words for logic AND and loads the result (OUT) into a word.

The instruction **combine words for logic OR** combines the corresponding bits of two input words for logic OR and loads the result (OUT) into a word.

The instruction **combine words for logic EXCLUSIVE OR** combines the corresponding bits of two input words for logic EXCLUSIVE OR and loads the result (OUT) into a word.

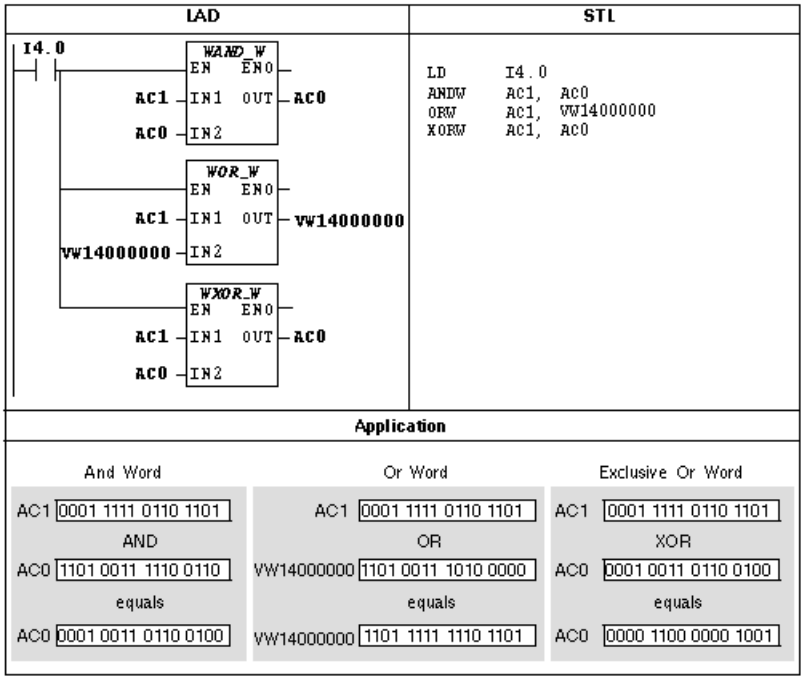
## Supporting CPUs




| | | | |
|---|---|---|---|
| 828D | 808D-PPU14x | 802Dsl TM value | SINUMERIK 802D |
| 828D Step 2 | 808D-PPU15x | 802Dsl TM plus | SINUMERIK 802SC |
| | 808D-PPU16x | 802Dsl TM pro | |
| | | 802Dsl CU pro | |

802Dsl GN plus
802Dsl GN pro

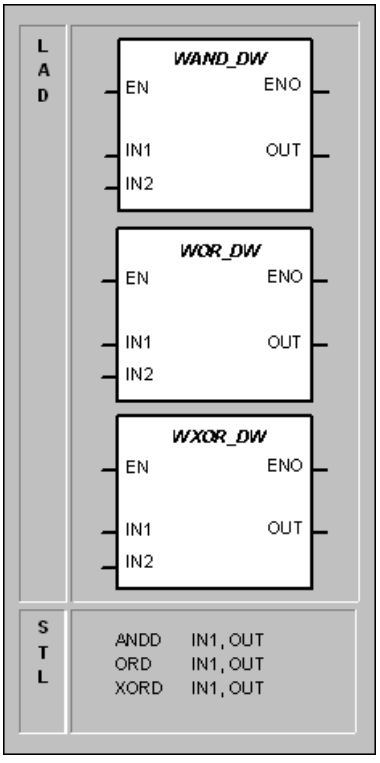
## Example




## Combine double words for logic AND/OR/EXCLUSIVE OR

## Combine double-words instruction

| Inputs/outputs | | Addresses | Data types |
|---|---|---|---|
| IN1, IN2 | | VD, ID, QD, MD, AC, constant, LD | DWORD |
| OUT | | VD, ID, QD, MD, AC, LD | DWORD |
| Memory areas (Page 262) | ENO (Page 259) | Supported instructions (Page 259) | Mnemonics (Page 260) |

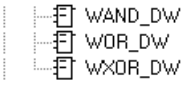



The instruction **combine double words for logic AND** combines the
corresponding bits of two double word inputs for logic AND and
loads the result (OUT) to a double word.

The instruction **combine double words for logic OR** combines the
corresponding bits of two double word inputs for logic OR and
loads the result (OUT) to a double word.

The instruction **combine double words for logic EX-CLUSIVE OR**
combines the
corresponding bits of two double word inputs for logic EXCLUSIVE OR and
loads the result (OUT) to a double word.

## Supporting CPUs

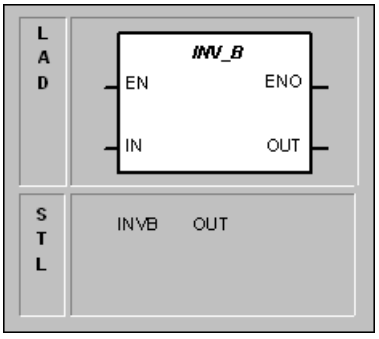



| | | | |
|---|---|---|---|
| 828D | 808D-PPU14x | 802Dsl TM value | SINUMERIK 802D |
| 828D Step 2 | 808D-PPU15x | 802Dsl TM plus | SINUMERIK 802SC |
| | 808D-PPU16x | 802Dsl TM pro | |
| | | 802Dsl CU pro | |
| | | 802Dsl GN plus | |
| | | 802Dsl GN pro | |

## Create one's complement of byte

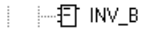
## Create one's complement of a byte instruction

| Inputs/outputs | | Addresses | Data types |
|---|---|---|---|
| IN | | VB, IB, QB, MB, AC, constant, LB | BYTE |
| OUT | | VB, IB, QB, MB, AC, LB | BYTE |
| Memory areas (Page 262) | ENO (Page 259) | Supported instructions (Page 259) | Mnemonics (Page 260) |




The instruction **create one's complement of byte** forms the
one's complement of the input byte IN and loads the result
to the byte value OUT.

## Supporting CPUs

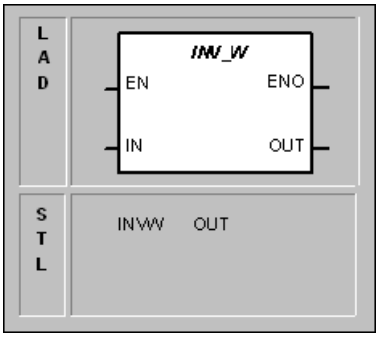



| | | | |
|---|---|---|---|
| 828D | 808D-PPU14x | 802Dsl TM value | SINUMERIK 802D |
| 828D Step 2 | 808D-PPU15x | 802Dsl TM plus | SINUMERIK 802SC |
| | 808D-PPU16x | 802Dsl TM pro | |
| | | 802Dsl CU pro | |
| | | 802Dsl GN plus | |
| | | 802Dsl GN pro | |

## Create one's complement of integer (16 bits)

## Create one's complement of an integer (16 bits) instruction

| Inputs/outputs | | Addresses | Data types |
|---|---|---|---|
| IN1, IN2 | | VW, T, C, IW, QW, MW, AC, constant, LW | WORD |
| OUT | | VW, T, C, IW, QW, MW, AC, LW | WORD |
| Memory areas (Page 262) | ENO (Page 259) | Supported instructions (Page 259) | Mnemonics (Page 260) |




The instruction **create one's complement of integer (16 bits)** forms the one's complement of the input word IN and loads the result to the word value OUT.

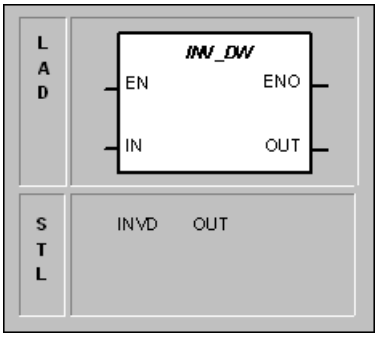
## Supporting CPUs


| | | | |
|---|---|---|---|
| 828D | 808D-PPU14x | 802Dsl TM value | SINUMERIK 802D |
| 828D Step 2 | 808D-PPU15x | 802Dsl TM plus | SINUMERIK 802SC |
| | 808D-PPU16x | 802Dsl TM pro | |
| | | 802Dsl CU pro | |
| | | 802Dsl GN plus | |
| | | 802Dsl GN pro | |

## Create one's complement of integer (32 bits)

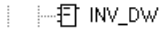
## Create one's complement of an integer (32 bits) instruction

| Inputs/outputs | | Addresses | Data types |
|---|---|---|---|
| IN | | VD, ID, QD, MD, AC, constant, LD | DWORD |
| OUT | | VD, ID, QD, MD, AC, LD | DWORD |
| Memory areas (Page 262) | ENO (Page 259) | Supported instructions (Page 259) | Mnemonics (Page 260) |




The instruction **create one's complement of integer (32 bits)** forms the one's complement of the input double word IN
and loads the result to the double word value OUT.

## Supporting CPUs



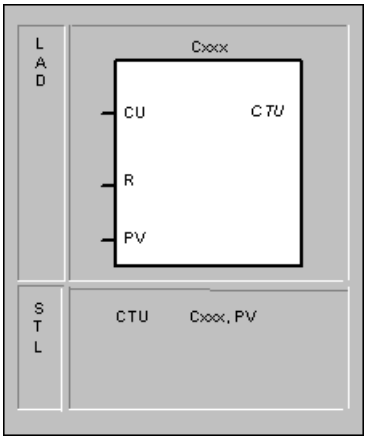

| | | | |
|---|---|---|---|
| 828D | 808D-PPU14x | 802Dsl TM value | SINUMERIK 802D |
| 828D Step 2 | 808D-PPU15x | 802Dsl TM plus | SINUMERIK 802SC |
| | 808D-PPU16x | 802Dsl TM pro | |
| | | 802Dsl CU pro | |
| | | 802Dsl GN plus | |
| | | 802Dsl GN pro | |

### 6.1.1.11 Counter

## Count up

## Count up operation

| Inputs/outputs | | Operands | Data types |
|---|---|---|---|
| Cxxx | | Constant | WORD |
| CU (LAD) | | Signal flow | BOOL |
| R (LAD) | | Signal flow | BOOL |
| PV | | VW, T, C, IW, QW, MW, AC, constant, LW | INT |
| Memory areas (Page 262) | ENO (Page 259) | Supported operations (Page 259) | Mnemonics (Page 260) |




The counter operation (Page 221) **count up** (CTU) counts up to the maximum value on a positive edge at the count up input (CU). If the actual value (Cxxx) is greater than or equal to the preset value (PV), then the counter bit (Cxxx) is activated. The counter is reset when the reset input is activated. The counter stops counting when the maximum value (32767) is reached.

In STL, the reset input is the topmost stack value and the count up input is the value in the second stack level.

**Note**

Since each counter has its own actual value, you may not assign the same number to several counters (up-counters, up-down counters, and down-counters with the same number have access to the same actual value).

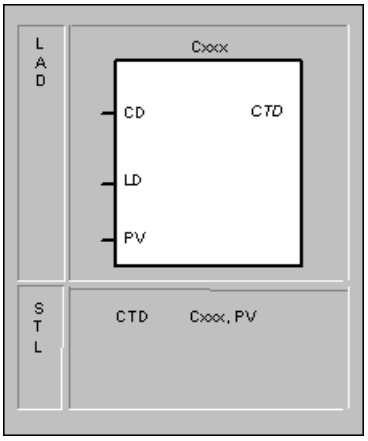
## Supporting CPUs


| | | | |
|---|---|---|---|
| 828D | 808D-PPU14x | 802Dsl TM value | SINUMERIK 802D |
| 828D Step 2 | 808D-PPU15x | 802Dsl TM plus | SINUMERIK 802SC |
| | 808D-PPU16x | 802Dsl TM pro | |

802Dsl CU pro
802Dsl GN plus
802Dsl GN pro

## Count down

| Inputs/outputs | | Operands | Data types |
|---|---|---|---|
| Cxxx | | Constant | WORD |
| CD (LAD) | | Signal flow | BOOL |
| LD (LAD) | | Signal flow | BOOL |
| PV | | VW, T, C, IW, QW, MW, AC, constant, LW | INT |
| Memory areas (Page 262) | ENO (Page 259) | Supported operations (Page 259) | Mnemonics (Page 260) |




The count operation (Page 221) **count down** (CTD) counts down from the preset value on a positive edge at the count down input (CD). If the actual value is equal to zero, then the counter bit (Cxxx) is activated. The counter resets the counter bit (Cxxx) and loads the actual value to the preset value (PV), when the load input (LD) is activated. The down-counter stops counting when it reaches zero.

In STL, the CTD load input is the topmost stack value and the count down input is the value in the second stack level.

**Note**

Since each counter has its own actual value, you may not assign the same number to several counters (up-counters, up-down counters, and down-counters with the same number have access to the same actual value).

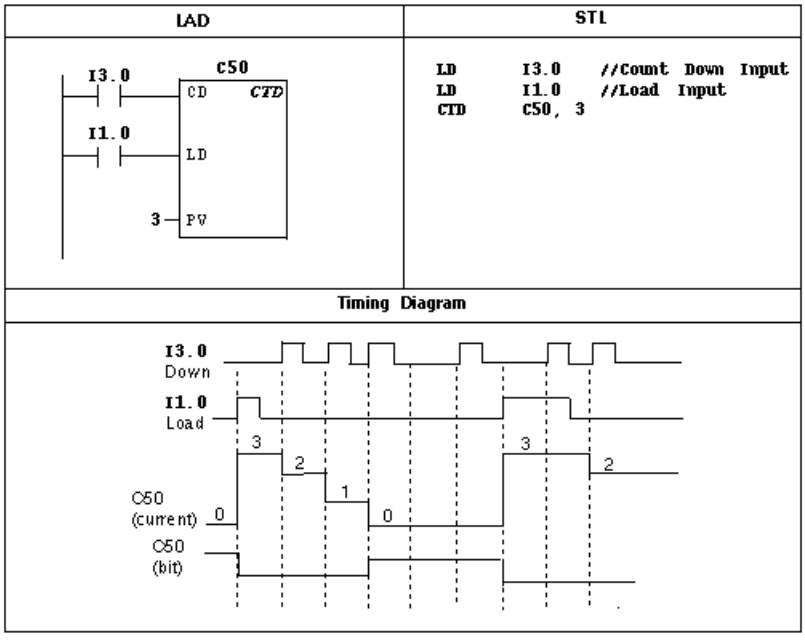
## Supporting CPUs


| | | | |
|---|---|---|---|
| 828D | 808D-PPU14x | 802Dsl TM value | SINUMERIK 802D |
| 828D Step 2 | 808D-PPU15x | 802Dsl TM plus | SINUMERIK 802SC |
| | 808D-PPU16x | 802Dsl TM pro | |

802Dsl CU pro
802Dsl GN plus
802Dsl GN pro

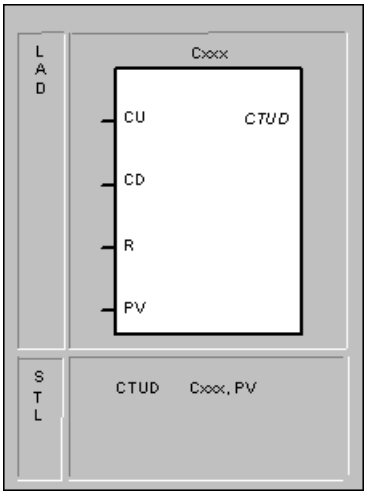
## Example

Down-counter with pulse diagram:




## Count up-down

| Inputs/outputs | | Operands | Data types |
|---|---|---|---|
| Cxxx | | Constant | WORD |
| CU, CD (LAD) | | Signal flow | BOOL |
| R (LAD) | | Signal flow | BOOL |
| PV | | VW, T, C, IW, QW, MW, AC, constant, LW | INT |
| Memory areas (Page 262) | ENO (Page 259) | Supported operations (Page 259) | Mnemonics (Page 260) |

| | |
|---|---|
|  | The count operation (Page 221) **count up-down** (CTUD) counts forward on a positive edge at the count up input (CU). On a positive edge at the count down input (CD), the operation counts backward. If the actual value (Cxxx) is greater than or equal to the preset value (PV), then the counter bit (Cxxx) is activated. The counter is reset when the reset input is activated.<br>In STL, the reset input is the topmost stack value. The count up input is the value in the second stack level and the count down input is the value in the third stack level. |

**Note**

Since each counter has its own actual value, you may not assign the same number to several counters (up-counters, up-down counters, and down-counters with the same number have access to the same actual value).

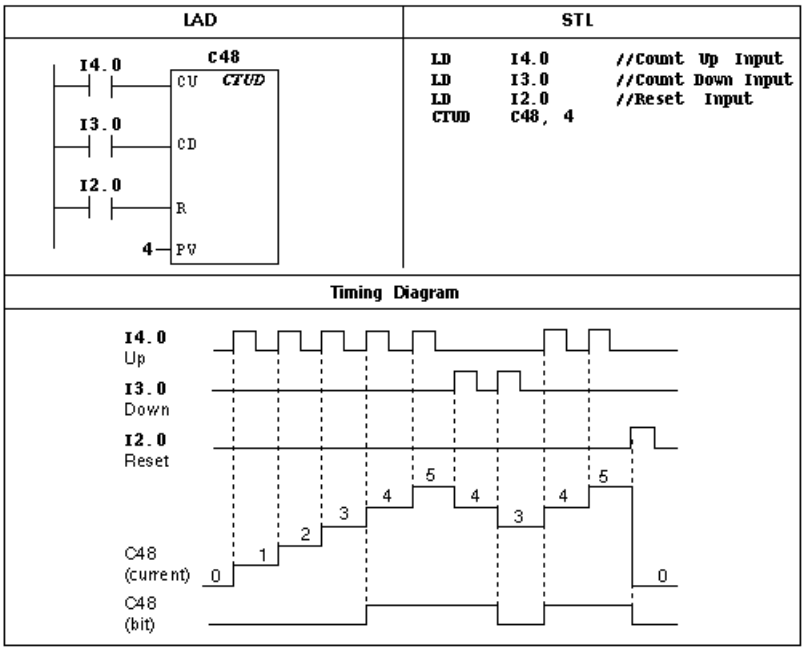
## Supporting CPUs

CTUD

| | | | |
|---|---|---|---|
| 828D | 808D-PPU14x | 802Dsl TM value | SINUMERIK 802D |
| 828D Step 2 | 808D-PPU15x | 802Dsl TM plus | SINUMERIK 802SC |
| | 808D-PPU16x | 802Dsl TM pro | |
| | | 802Dsl CU pro | |
| | | 802Dsl GN plus | |
| | | 802Dsl GN pro | |

### Example

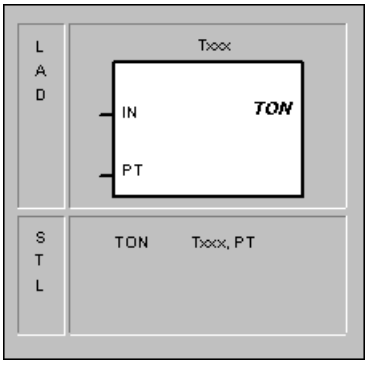
Up-down counter with pulse diagram:




## 6.1.1.12 Timer

### Start ON-delay timer

| Inputs/outputs | | Operands | Data types |
|---|---|---|---|
| Txxx | | Constant (T0-T15 [100 ms], after T16 [10 ms]) | WORD |
| IN (LAD) | | Signal flow | BOOL |
| PT | | VW, T, C, IW, QW, MW, AC, constant, LW | INT |
| Memory areas (Page 262) | ENO (Page 259) | Supported operations (Page 259) | Mnemonics (Page 260) |




The operation **start ON-delay timer** (TON) counts the time when the enable input is switched on. If the actual value (Txxx) is greater than or equal to the preset time value (PT), then the counter bit is switched on.
The actual value of the time is deleted when the enable input is switched off.

This timer continues to count after the default setting is reached, and stops at the maximum value of 32,767.

The timer operations (Page 218) TON, TONR, and TOF are available with two different resolutions. The resolution depends on the number of the timer (see following table). Each increase of 1 to the actual value represents a multiple of the time base. For a timer with a resolution of 10 ms, for example, a counter value of 50 corresponds to an actual value of 500 ms.

| Resolution | Maximum value | Number of the timer |
|---|---|---|
| 100 ms | 3276.7 s | T0-T15 |
| 10 ms | 327.67 s | After T16 |

**Note**

You cannot assign the same number to the timers TOF and TON. For example, you may not assign a timer of TON T32 and one of TOF T32 at the same time.
You can use the reset operation (R) (Page 119) to reset the timers. The reset operation performs the following:

Timer bit = OFF and actual value of the timer = 0

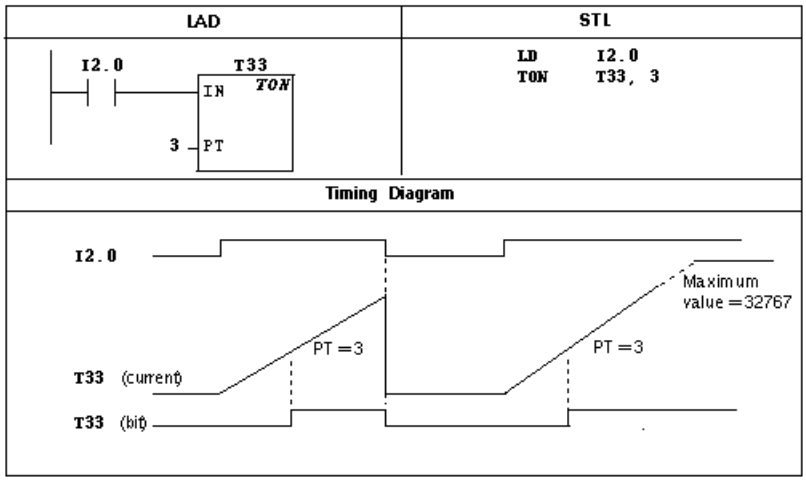
## Supporting CPUs

TON

| | | | |
|---|---|---|---|
| 828D | 808D-PPU14x | 802Dsl TM value | SINUMERIK 802D |
| 828D Step 2 | 808D-PPU15x | 802Dsl TM plus | SINUMERIK 802SC |
| | 808D-PPU16x | 802Dsl TM pro | |
| | | 802Dsl CU pro | |
| | | 802Dsl GN plus | |
| | | 802Dsl GN pro | |

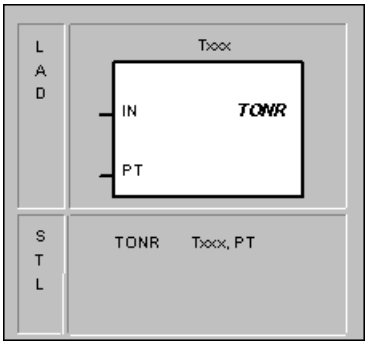
## Example

ON delay with pulse diagram:




## Start latching ON-delay timer

| Inputs/outputs | | Operands | Data types |
|---|---|---|---|
| Txxx | | Constant (T0-T15 [100 ms], after T16 [10 ms]) | WORD |
| IN (LAD) | | Signal flow | BOOL |
| PT | | VW, T, C, IW, QW, MW, AC, constant, LW | INT |
| Memory areas (Page 262) | ENO (Page 259) | Supported operations (Page 259) | Mnemonics (Page 260) |




The operation **start latching ON-delay timer** (TONR) counts the time when the enable input is switched on. If the actual value (Txxx) is greater than or equal to the preset time value (PT), then the counter bit is switched on.

The actual value of the latching ON delay is saved once the enable input is switched off. With the latching ON delay, you can accumulate the time value over several starting cycles of the enable input. With the reset operation (R) (Page 119), you delete the actual value of the latching ON delay.

This timer continues to count after the default setting is reached, and stops at the maximum value of 32,767.

The timer operations (Page 218) TON, TONR, and TOF are available with two different resolutions. The resolution depends on the number of the timer (see following table). Each increase of 1 to the actual value represents a multiple of the time base. For a timer with a resolution of 10 ms, for example, a counter value of 50 corresponds to an actual value of 500 ms.

| Resolution | Maximum value | Number of the timer |
|---|---|---|
| 100 ms | 3276.7 s | T0-T15 |
| 10 ms | 327.67 s | After T16 |

**Note**

With the reset operation (R), you can reset the timers. The reset operation performs the following:

Timer bit = OFF and actual value of the timer = 0

The timer TONR can only be reset with the reset operation.

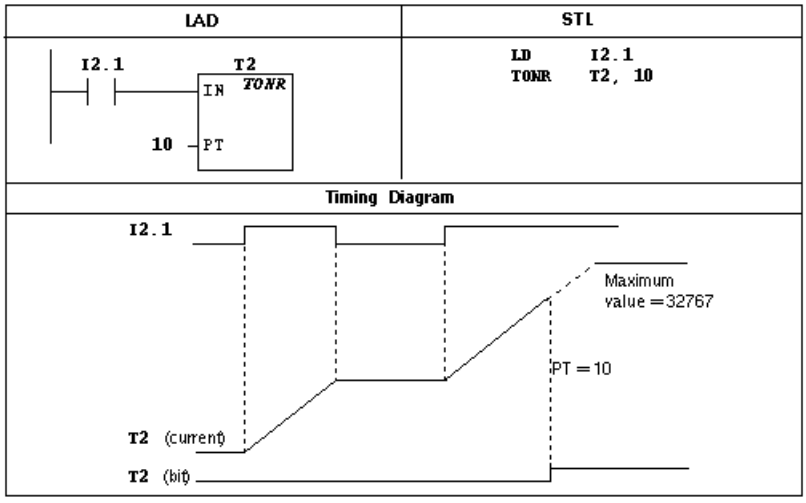
## Supporting CPUs

TONR

| | | | |
|---|---|---|---|
| 828D | 808D-PPU14x | 802Dsl TM value | SINUMERIK 802D |
| 828D Step 2 | 808D-PPU15x | 802Dsl TM plus | SINUMERIK 802SC |
| | 808D-PPU16x | 802Dsl TM pro | |
| | | 802Dsl CU pro | |
| | | 802Dsl GN plus | |
| | | 802Dsl GN pro | |

## Example

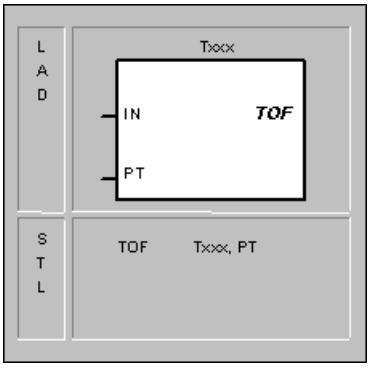
Latching ON delay with pulse diagram:




## Start OFF-delay timer

| Inputs/outputs | | Operands | Data types |
|---|---|---|---|
| Txxx | | Constant (T0-T15 [100 ms], after T16 [10 ms]) | WORD |
| IN (LAD) | | Signal flow | BOOL |
| PT | | VW, IW, QW, MW, LW, T, C, AC, constant | INT |
| Memory areas (Page 262) | ENO (Page 259) | Supported operations (Page 259) | Mnemonics (Page 260) |

## Programming




The operation **start OFF-delay timer** (TOF) is used to delay the switch-off of an output for a particular period of time after the input has been switched off. If the enable input is switched on, the timer bit is enabled immediately and the actual value is set to 0. If the enable output is switched off, the timer continues to count until the time elapsed reaches the preset time value. If the default setting is reached, the timer bit is disabled and the actual value stops counting. If the input is switched off for a period of time that is shorter than the preset value, the timer bit remains enabled. The TOF operation requires a negative edge so that it starts to count.

The timer operations (Page 218) TON, TONR, and TOF are available with two different resolutions. The resolution depends on the number of the timer (see following table). Each increase of 1 to the actual value represents a multiple of the time base. For a timer with a resolution of 10 ms, for example, a counter value of 50 corresponds to an actual value of 500 ms.

| Resolution | Maximum value | Number of the timer |
|---|---|---|
| 100 ms | 3276.7 s | T0-T15 |
| 10 ms | 327.67 s | After T16 |

**Note**

You cannot assign the same number to the timers TOF and TON. For example, you may not assign a timer of TON T32 and one of TOF T32 at the same time. With the reset operation (R), you can reset the timers. The reset operation (R) (Page 119) performs the following:

Timer bit = OFF and actual value of the timer = 0

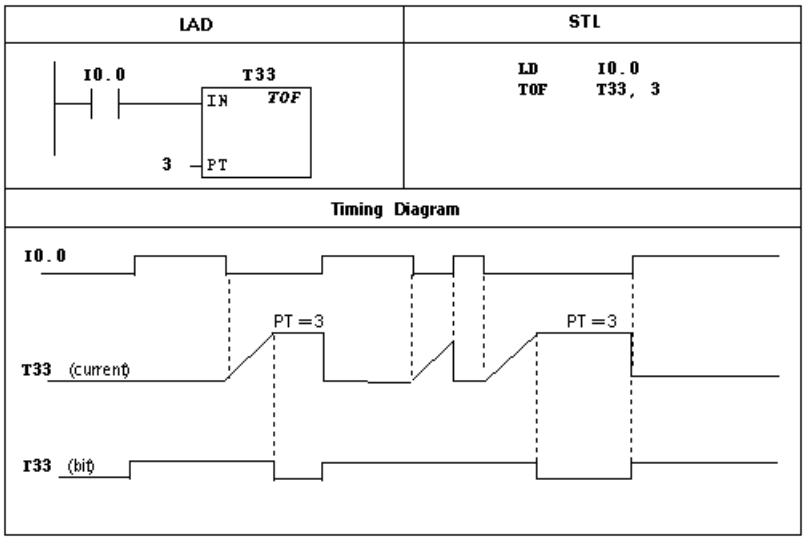
After a TOF timer is reset, the enable input must switch from ON to OFF so that the timer can be restarted.

## Supporting CPUs


| | | | |
|---|---|---|---|
| 828D | 808D-PPU14x | 802Dsl TM value | SINUMERIK 802D |
| 828D Step 2 | 808D-PPU15x | 802Dsl TM plus | SINUMERIK 802SC |
| | 808D-PPU16x | 802Dsl TM pro | |
| | | 802Dsl CU pro | |
| | | 802Dsl GN plus | |
| | | 802Dsl GN pro | |

### Example

OFF delay with pulse diagram:




## 6.1.1.13 Libraries

### Interrupt Programs

| | |
|---|---|
| Programming of interrupt instructions (Page 131) | |

### Supporting CPUs

INT

828D
828D Step 2

### Using interrupt programs

Create an interrupt program:

- You will find these in the instruction tree under **Libraries > Interrupt programs**. You can make use of them by double-clicking or via copy and paste (in the project tree).

The actual program organization unit (POU) is then no longer displayed in the program editor, but a new interrupt program is instead. A new tab is inserted at the bottom of the program editor, which contains the new interrupt program.

- INT_0: servo-synchronous version
  For extremely fast responses to events for which such responses are absolutely essential.

  **Note**

  This interrupt program may contain a maximum of 500 instructions.

- INT_100: executed before MAIN, after reading the inputs
  Especially for marshaling data that is accessed (ready only) in the subsequent scan cycle.
- INT_101: executed after MAIN, before writing the outputs
  Especially for marshaling data that was previously written in the scan cycle, but which is to be output to other or additional outputs.

Now you can program the new interrupt program or return to the POU which you were working on previously:

- Note that each POU in your program has its own local variable table (Page 229). You must define the local variables for this interrupt program in the local variable table, which will be displayed if you have opened the tab for the interrupt program. Make sure that you always work on the correct tab when you are editing the local variable table.
- If you want to write the logic for the interrupt program, as long as the tab is open you just need to start working in the program editor.
- If you want to work with another program organization unit, open the tab for the desired POU so that it is displayed in the program editor.

The CPU processes interrupts outside the main(OB1). Only one program is ever active to process interrupts. If an interrupt program is presently being processed, then this program is terminated. It cannot be interrupted.

**Note**

Restrictions when using interrupt programs:

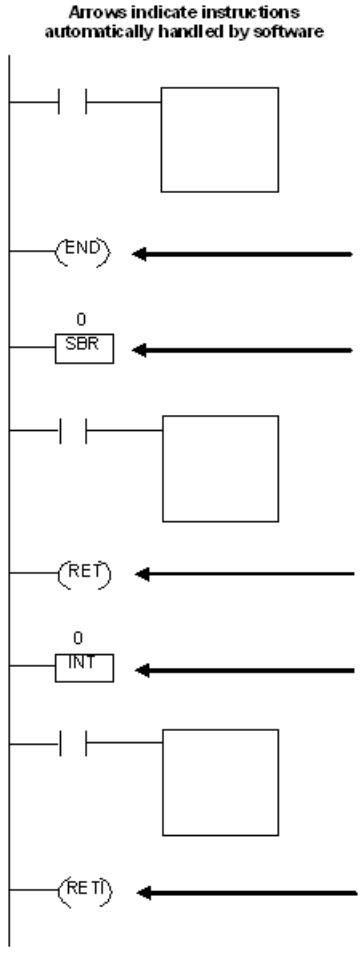
- You may not insert the END instruction into an interrupt program.

There are commands with direct access for fast responses to I/O signals in the interrupt program:

| | |
|---|---|
| NCI_CONTACT | \|I\| |
| NOI_CONTACT | \|/I\| |
| OUTI_COIL | (I) |
| SETI_COIL | (SI) |
| RESETI_COIL | (RI) |
| RBI_BOX | MOV_BIR |
| WBI_BOX | MOV_BIW |

**Note**

The editor automatically inserts the instructions to absolutely terminate program organization units (END for OB1, RET for SBR, and RETI for INT). An example is shown below.




## Special data blocks

If you wish to work with data blocks, then select a CPU (Page 469) that supports data blocks.

## CPUs that support data blocks

DB

828D
828D Step 2
808D-PPU14x (only special data blocks)
808D-PPU15x (as of PLC 07.03)
808D-PPU16x (as of PLC 07.03)

### Procedure

Use one of the following methods to access the data block:

- Click the button of the data block  in the "Navigation" toolbar (Page 500).
- Select the menu command **View > Data Block**.
- Click the button of the data block  in the project tree (Page 447).

These help topics are explained in the following:

### Data block properties

The data block structures are, just like the POUs, part of the project. They are compiled, saved, imported or exported with the project. They are loaded into/from the PLC with the project. The data blocks themselves only contain actual values. Actual values may be loaded into/from the PLC independently of the project. It is therefore essential that the structure of the data block to be loaded in the PLC is exactly the same as that of the project opened in the Programming Tool. Data blocks are listed like POUs in their own symbol table.

When changing the CPU, existing data blocks are not lost. However, if you select a CPU in which data blocks are not available, then please note that the variables from the data block are now displayed with their absolute address (e.g. DB9000.DBB0 or VB90000000). Whether this V range exists is first checked when the PLC is brought into the RUN state. In this case, the program cannot be executed.

To rename your data blocks and change their properties (Page 452), right-click the corresponding data block in the operation tree. Select "Properties". The "Data Block Properties" dialog box opens. Here, you can change the name, block number, and data class (Page 243), as well as add an author and comments. You can also activate the property "Non-retain" for data blocks. This ensures that every time the PLC is brought into the RUN state, the data block in the PLC is preset with the initial values. Data blocks can be write-protected, that is, the actual values of the data block cannot be changed with the Programming Tool. This property cannot be changed.

**Note**

If you just want to rename your data block, right-click the object you want to rename in the operation tree and select "Rename".

You can also easily edit your data block in another program (e.g. Microsoft Excel) (see Cut, Copy and Paste section in the data block editor).

### Assigning addresses and initial values in the data block

When you create a variable, you assign its name and data type. The default initial value is set to zero/OFF, however, this can also be changed. You can optionally insert comments. The Programming Tool automatically assigns the address. Each address is aligned according to the size of its data type, which means that gaps can occur. The actual values are edited in the data view. However, you may assign the initial values to all actual values.

The maximum size of a data block is limited (512 bytes), if it is exceeded then the Programming Tool marks the excess variable addresses using a red wavy line. The Programming Tool returns an error when compiling the project.

### Assigning actual values

Actual values are saved, exported or imported in your project file as part of your data block. You can only assign individual actual values in the data view. The structure of the variable (name, data type, initial value, and comment) is write-protected in the data view. You can only change the actual value and therefore specifically set initial values, for example. After loading the data block (only actual values), these values become effective in the PLC.

The structure of the data block in the project must match the structure of the data block in the PLC. If you have modified the structure of the data block, then the complete project has to be loaded again.

You can access the data view in the following ways:

- Click the Display data block values button in the toolbar.
- Select the menu command **View > View Data Block Values**.

### Accepting initial values

If you accept your initial values, then all of the actual values of your data block are overwritten. If you have not selected an initial value for a variable, then the Programming Tool sets it to zero/ OFF. Other data blocks will not be changed.

In order to reset all actual values in the actual data block to the initial values,

- Right-click in the table > Accept Initial Values.
- Select the menu command **Edit > Accept Initial Values**.

### Displaying and changing actual values

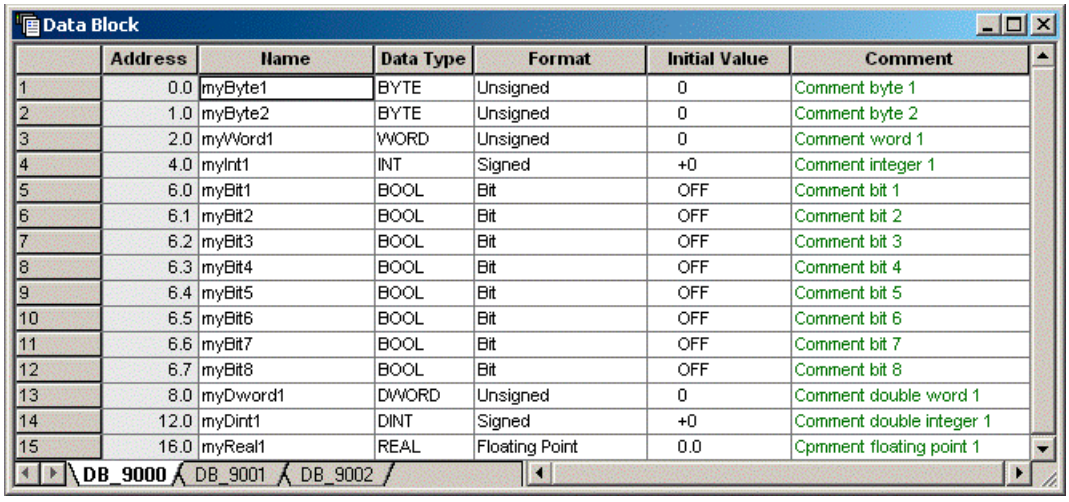
You can view the actual values of a data block in the PLC with the data block status (Page 369) and, if the data block is not write-protected, change them too. The structure of the data block in the project must match the structure of the data block in the PLC. If you have modified the structure of the data block, then the complete project has to be loaded again. When the data block status is activated, the structure of the variable (name, data type, initial value, and comment) is write-protected. If you switch on the data block status, the values in the column "Actual Value" are regularly updated with the actual values of the PLC.

After you have entered values in the column "New Value", write the required changes to the PLC using the command "Write All" .

You can activate the data block status in the following ways:

- Click the "Data block status" button in the toolbar.
- Select the menu command **Debug > Data Block Status**.

## Example for a data block



Data Block

| | Address | Name | Data Type | Format | Initial Value | Comment |
|---|---|---|---|---|---|---|
| 1 | 0.0 | myByte1 | BYTE | Unsigned | 0 | Comment byte 1 |
| 2 | 1.0 | myByte2 | BYTE | Unsigned | 0 | Comment byte 2 |
| 3 | 2.0 | myWord1 | WORD | Unsigned | 0 | Comment word 1 |
| 4 | 4.0 | myInt1 | INT | Signed | +0 | Comment integer 1 |
| 5 | 6.0 | myBit1 | BOOL | Bit | OFF | Comment bit 1 |
| 6 | 6.1 | myBit2 | BOOL | Bit | OFF | Comment bit 2 |
| 7 | 6.2 | myBit3 | BOOL | Bit | OFF | Comment bit 3 |
| 8 | 6.3 | myBit4 | BOOL | Bit | OFF | Comment bit 4 |
| 9 | 6.4 | myBit5 | BOOL | Bit | OFF | Comment bit 5 |
| 10 | 6.5 | myBit6 | BOOL | Bit | OFF | Comment bit 6 |
| 11 | 6.6 | myBit7 | BOOL | Bit | OFF | Comment bit 7 |
| 12 | 6.7 | myBit8 | BOOL | Bit | OFF | Comment bit 8 |
| 13 | 8.0 | myDword1 | DWORD | Unsigned | 0 | Comment double word 1 |
| 14 | 12.0 | myDint1 | DINT | Signed | +0 | Comment double integer 1 |
| 15 | 16.0 | myReal1 | REAL | Floating Point | 0.0 | Cpmment floating point 1 |

DB_9000 DB_9001 DB_9002

## Addressing

### Direct addressing

- **Absolute address input**
  Input in absolute format, the address of a variable in your data block comprises the data block number (e.g. DB9000), a period, and the address of the variable. However, you can also directly access the variable via a V address (e.g. V9000000.0). The display of the absolute address depends on how the address display is set and this can be selected in the menu **View > DB Address View (Ctrl + B)**.

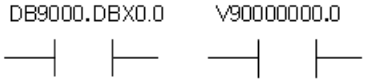



Figure 6-1 Absolute addressing using the data block number

- **Symbolic address input**
  In the symbol table, you can assign an additional name to the variables from your data block. If you wish to use your variable, then you no longer have to specify the complete address - a name in the symbol table is sufficient.
  If you have not listed your variable from the data block in your symbol address, then the symbolic address comprises the name of the corresponding data block (e.g. DB_9000) and the name of the variable in the data block (e.g. Ax1).

| Address | Name of the variables in the data block | Entry in the symbol table | Absolute representation | Symbolic representation |
|---|---|---|---|---|
| VB90000000 | VarName | FeedCorrAx1 | VB90000000 | FeedCorrAx1 |
| or | | | | |
| DB9000.DBB0 | VarName | FeedCorrAx1 | DB9000.DBB0 | FeedCorrAx1 |
| VB90000000 | VarName | | VB90000000 | NameDB.VarName |
| or | | | | |
| DB9000.DBB0 | VarName | | DB9000.DBB0 | NameDB.VarName |

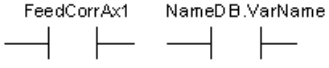



Figure 6-2 Symbolic address representation

The switch between the absolute and symbolic representation is carried out using the menu command **View > Symbolic Addressing (Ctrl + Y)**.

### Indirect addressing

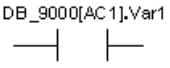
You may be able to simplify the programming if data blocks with the same structure are used in your program. You can indirectly address your data blocks using the accumulators (AC0 to AC3). The accumulators are used to inform the program which data block is to be handled. The value in the AC is then treated as an index.

For example, for axis DBs, the program text can be somewhat reduced by not having to write a dedicated program for each axis, but instead accessing the appropriate axis via the various data blocks and the index (AC). As this value is treated as an index, it starts at 0 for the first axis. Indirect addressing is only possible using ACs. The address cannot be broken down, as

the contents of the accumulator are not known offline. This also means that the absolute address cannot be determined. V addresses cannot be used for indirect addressing.

**Note**

Accumulators are used by both subprograms and the calling program. Calling a subprogram does not cause the accumulators to be saved or restored.




| Absolute input | Symbolic input |
|---|---|
| DB3800[AC1].DBX2.1 | ToAxis[AC1].ControlEnable |
| DB9000[AC0].DBW0 | Prototyp1[AC0].MyWord1 |

**Note**

**Not permitted:**

DB3800[1].DBX2.1 ToAxis[5].ControlEnable

**Indirect addressing using V addresses is not permitted:**

V3800[5]0002.1

V380[5]0002.1

V3800[AC0]0002.1

V38[AC0]0002.1

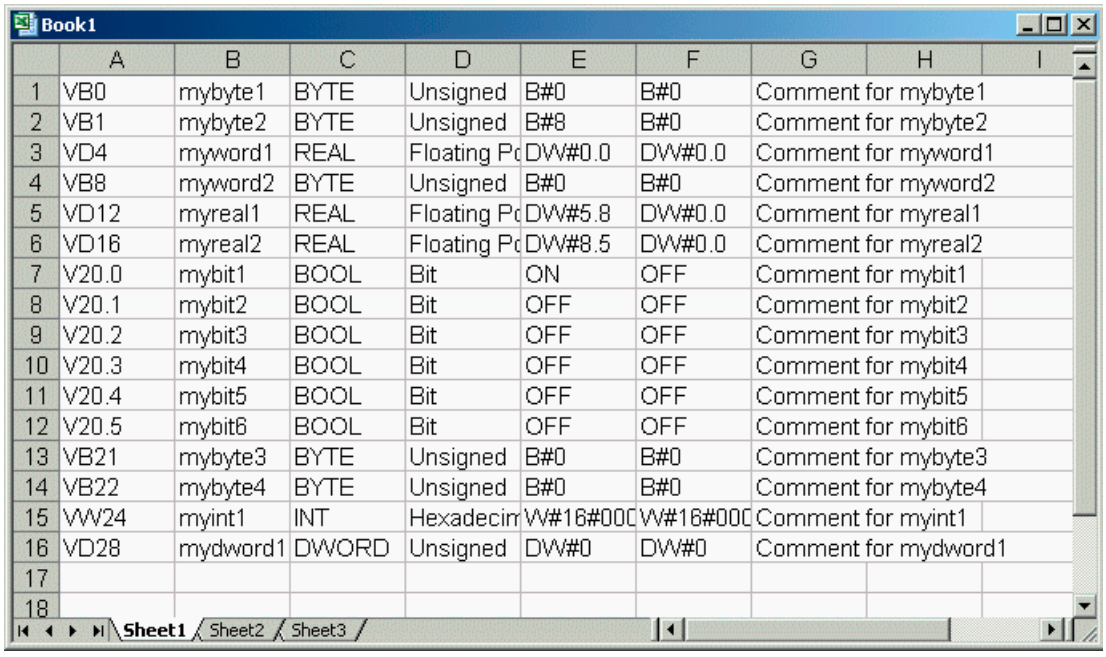
## Using Cut, Copy and Paste in the data block editor

You can save data (rows, columns, and cells) of your data block to the Microsoft clipboard in order to edit it in a different program. You can do this in one of the following ways:

- By right-clicking Cut/Copy/Paste
- By pressing the shortcut keys Ctrl+X/Ctrl+C/Ctrl+V
- By using the menu commands **Edit > Cut**, **Edit > Copy, Edit > Paste**

| | |
|---|---|
| Cut | Selected data is copied to the clipboard and is deleted |
| Copy | Selected data is copied to the clipboard and is not deleted |
| Paste | If data is in the clipboard, the data is pasted |
| Delete | Selected data block/section is deleted, it is not saved in the clipboard |

You can easily edit your data block in this way, e.g. in Microsoft Excel.



Book1

| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 1 | VB0 | mybyte1 | BYTE | Unsigned | B#0 | B#0 | Comment for mybyte1 | | |
| 2 | VB1 | mybyte2 | BYTE | Unsigned | B#8 | B#0 | Comment for mybyte2 | | |
| 3 | VD4 | myword1 | REAL | Floating Pc | DW#0.0 | DW#0.0 | Comment for myword1 | | |
| 4 | VB8 | myword2 | BYTE | Unsigned | B#0 | B#0 | Comment for myword2 | | |
| 5 | VD12 | myreal1 | REAL | Floating Pc | DW#5.8 | DW#0.0 | Comment for myreal1 | | |
| 6 | VD16 | myreal2 | REAL | Floating Pc | DW#8.5 | DW#0.0 | Comment for myreal2 | | |
| 7 | V20.0 | mybit1 | BOOL | Bit | ON | OFF | Comment for mybit1 | | |
| 8 | V20.1 | mybit2 | BOOL | Bit | OFF | OFF | Comment for mybit2 | | |
| 9 | V20.2 | mybit3 | BOOL | Bit | OFF | OFF | Comment for mybit3 | | |
| 10 | V20.3 | mybit4 | BOOL | Bit | OFF | OFF | Comment for mybit4 | | |
| 11 | V20.4 | mybit5 | BOOL | Bit | OFF | OFF | Comment for mybit5 | | |
| 12 | V20.5 | mybit6 | BOOL | Bit | OFF | OFF | Comment for mybit6 | | |
| 13 | VB21 | mybyte3 | BYTE | Unsigned | B#0 | B#0 | Comment for mybyte3 | | |
| 14 | VB22 | mybyte4 | BYTE | Unsigned | B#0 | B#0 | Comment for mybyte4 | | |
| 15 | VW24 | myint1 | INT | Hexadecim | W#16#000 | W#16#000 | Comment for myint1 | | |
| 16 | VD28 | mydword1 | DWORD | Unsigned | DW#0 | DW#0 | Comment for mydword1 | | |
| 17 | | | | | | | | | |
| 18 | | | | | | | | | |

Sheet1 Sheet2 Sheet3

## Using special data blocks

The Programming Tool offers you the possibility to use pre-configured data blocks for special applications. These special data blocks can be found in the operation tree under **Libraries > Special Data Blocks**. These data blocks have a fixed structure. You can make use of them by double-clicking or via copy and paste (in the operation tree).

## Resolving errors

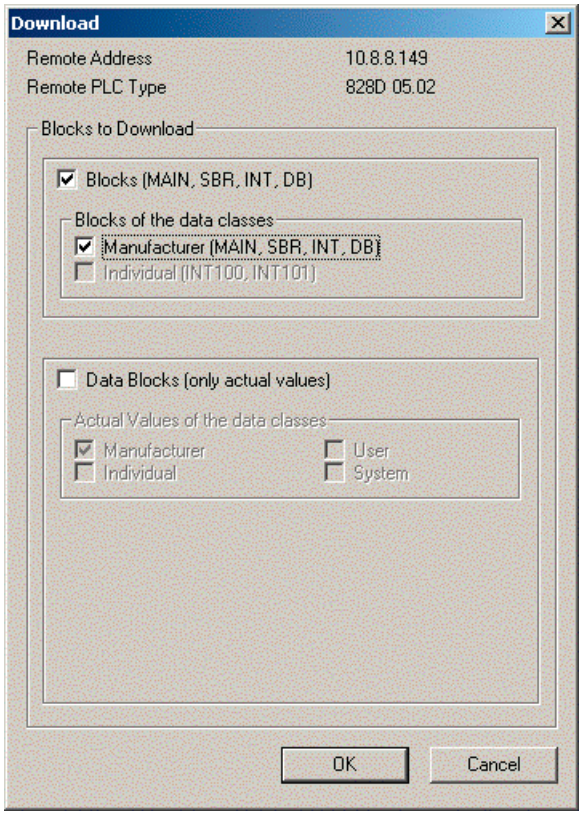
The Programming Tool marks errors made during data input as is the case in the LAD editor or the symbol table. For example, the use of invalid syntax or invalid values can cause an error. You must correct all of the errors that occur before you can compile (Page 459) without any errors and load the program into the PLC.

If any errors occur during compilation of the data block, then they are displayed in the output window. Position the cursor on an error message in the output window and double-click it so that you see the row in the data block with the error.

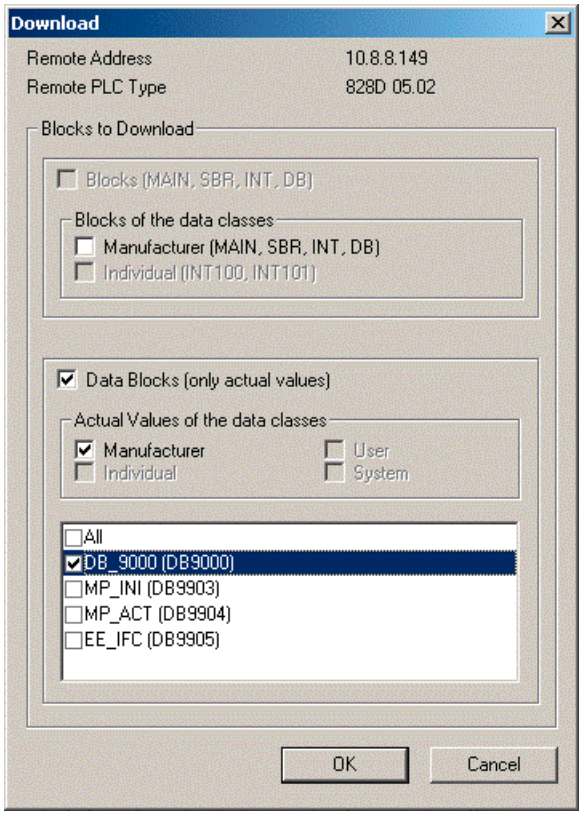
## Loading the data block

### Loading the data block into the PLC

If you edit the structure of a data block, you need to subsequently load the program block (project) into the PLC (Page 382). Your changes will only become effective after the modified project has been loaded into the PLC. Actual values and comments are not loaded into the PLC with the project. If the PLC identifies that a data block is new or has been changed, then it sets the initial values for this data block as the first actual values.

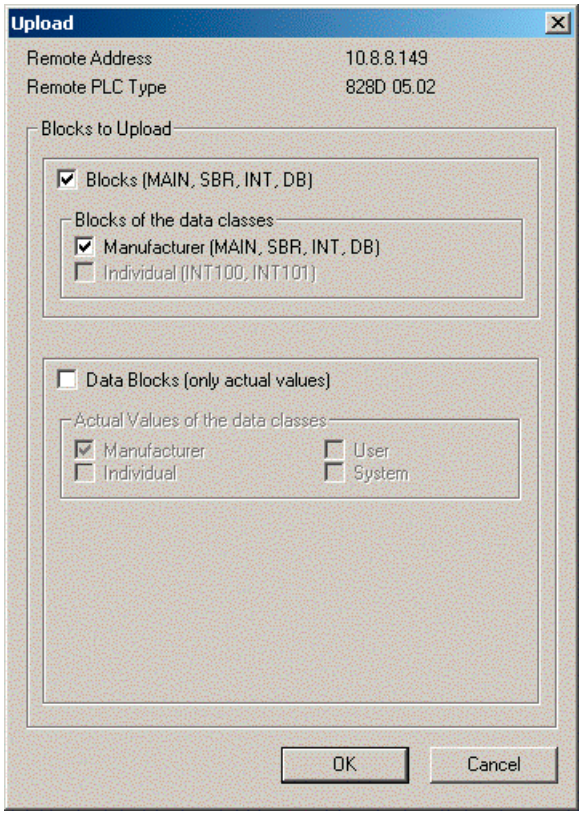



If you edit the actual values of a data block, you need to subsequently load the data block (actual values only) into the PLC (Page 382) so that these actual values become valid. The structure of the data block in the PLC must match the structure of the data block in the project.

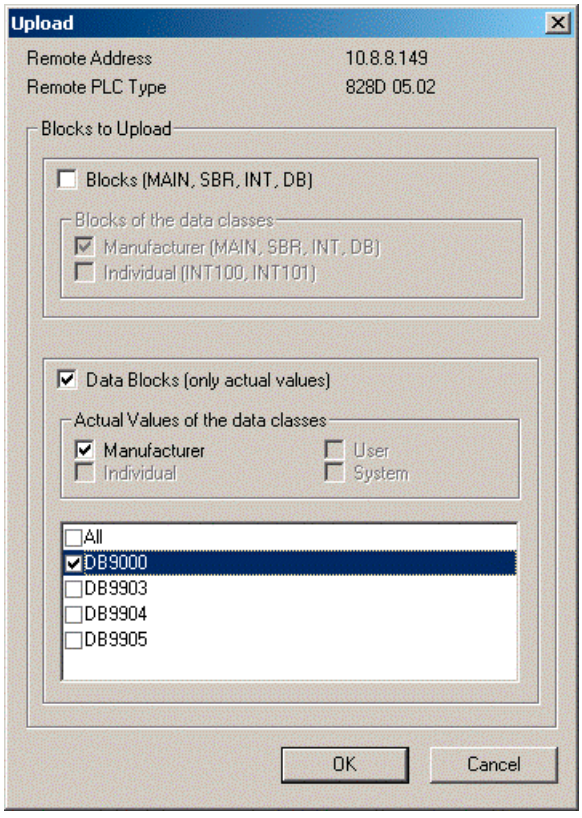



**Loading a data block from the PLC**

You must first open a project in the Programming Tool before you can load the program block (project) and therefore the structure of the data blocks contained in it from the PLC (Page 382).



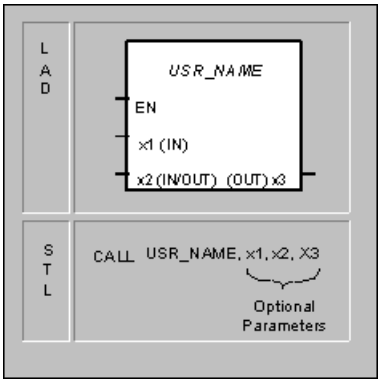

If you only want to load the actual values of a data block from the PLC, then first open the corresponding project in the Programming Tool or load it from the PLC. Now you can load the data block (only actual values) from the PLC (Page 382). You cannot load the data block if the structure of the data block in the PLC does not match that of the data block of the project that has been opened (or if the project that has been opened does not have a data block).




### 6.1.1.14 Subprograms

## Calling a subprogram

| Inputs/outputs | | Operands | Data types |
|---|---|---|---|
| | | n: Constant | Constant |
| Memory areas (Page 262) | ENO (Page 259) | Supported operations (Page 259) | Mnemonics (Page 260) |

| | |
|---|---|
|  | The operation call subprogram (Page 223) or CALL in STL calls a subprogram (n). You can use this operation with or without parameters. |

## Supporting CPUs

| | | | |
|---|---|---|---|
| 828D | 808D-PPU14x | 802Dsl TM value | SINUMERIK 802D |
| 828D Step 2 | 808D-PPU15x | 802Dsl TM plus | SINUMERIK 802SC |
| | 808D-PPU16x | 802Dsl TM pro | |
| | | 802Dsl CU pro | |
| | | 802Dsl GN plus | |
| | | 802Dsl GN pro | |

## Calling a subprogram

**Note**

If you have assigned a symbolic name to your subprogram, e.g. ANW_NAME, that name is displayed in the directory of the subprograms in the operation tree.

The parameter values are assigned to the local memory of the subprograms as follows:

1. The parameter values are assigned to the local memory in the order specified with parameters by the CALL operation, starting with L.0.
2. Between one and eight consecutive bit parameter values are assigned to a byte starting with Lx.0 to Lx.7.
3. Byte, word, and double word values are assigned to the local data area at byte boundaries (LBx, LWx or LDx).

If you perform a transfer with the operation call subprogram or CALL parameters, the parameters must exactly correspond to the variables defined in the local variable table of the subprogram. The parameters must be arranged in the following order:

- First input parameter
- Then in-out parameter
- Finally, output parameter

**Note**

If the first input parameters are defined as bit parameters, a result of logic operation is expected for these input parameters. Bit input parameters which follow a non-bit input parameter expect a bit value.

**Note**

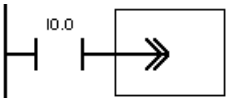
The tooltip for the subprogram in the operation tree shows the names of the individual parameters.

**Error conditions that set ENO for the CALL operation with parameters = 0:**
maximum nesting depth for subprograms exceeded

## 6.1.2 Signal flow displays in LAD

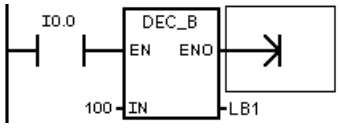
The signal flow in a valid network flows from the left busbar to the outputs, via the elements entered by the user. In order to make it easier for you to connect elements, in LAD there are two different **signal flow displays**. These displays represent the "flow" in a network graphically. Note that the signal flow displays are shown by the editor automatically and also deleted.

**Signal flow display with open circuit**




Via the position at the output of the coil, this element shows that the circuit in this network is open. You **must** close the circuit so that the network can be compiled without error.

**Optional signal flow display**

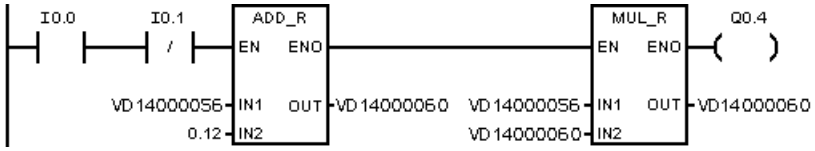



This shows that other logic can be added, although this is not required, because the network can already be compiled without error in the existing state. This element is at the ENO output of a box and displayed as the starting point of empty networks.

## 6.1.3 Connecting instructions in LAD

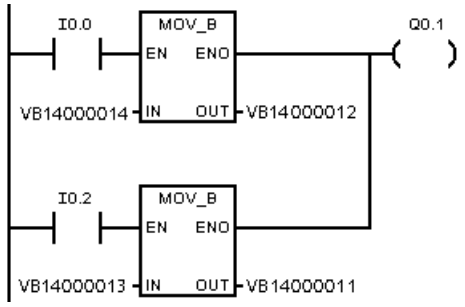
LAD offers horizontal and vertical lines to help connect instruction references in series and in parallel arrangements. By connecting the signal flow output of one instruction to the signal flow input of another, you create the logic in your program. LAD provides some horizontal connecting lines automatically and can even expand boxes vertically to align signal flows.

**Series connection**



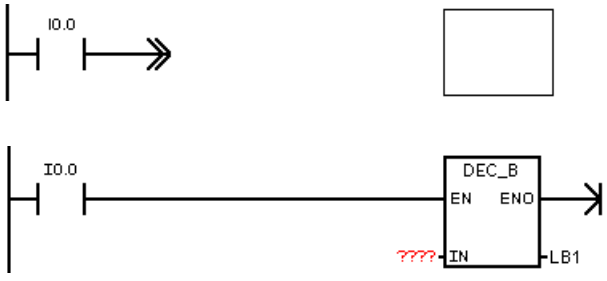

Contacts and boxes can be connected in series. Coils can be used to terminate such series. Horizontal connecting lines are sometimes needed to connect the signal flows of a series of instructions. Horizontal connecting lines are used here to connect the "add floating-point numbers" and "multiply floating-point numbers" boxes.

**Parallel connection**



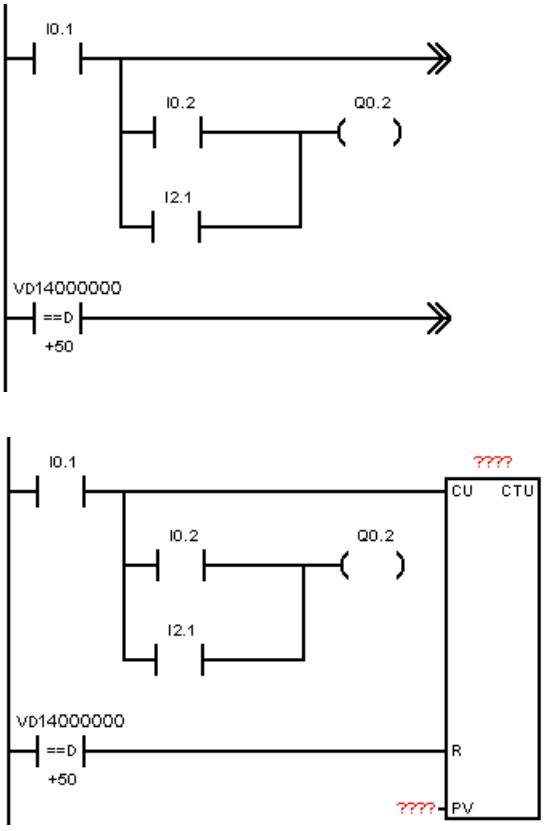

When a parallel connection is required, vertical connecting lines are needed. A vertical connecting line can extend up or down from the field of origin. In the configuration above, the "move byte" instructions are in parallel.

**Automatic signal flow connection**




When an instruction is placed in a network, the LAD editor automatically connects the signal flow input of the instruction to a signal flow display to the left of the instruction. In the network above, the LAD editor connects the signal flow to the "decrease byte by 1" instruction.

**Box alignment**



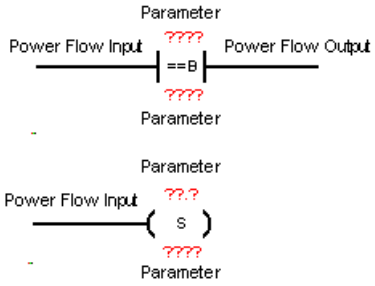

In this example, the interim instructions occupy many rows of the network. In order for the second signal flow input of the "count up" box to be connected to the output of the "compare double word" contact, the box is automatically expanded. Signal flow was extended to connect the aligned signal flow inputs of the new box with the signal flow display in the network.

When a box is placed in a network and the box contains two or more signal flow inputs, the LAD editor will automatically align the signal flow inputs with the signal flow displays of the network.

## 6.1.4 Instruction parameters in LAD

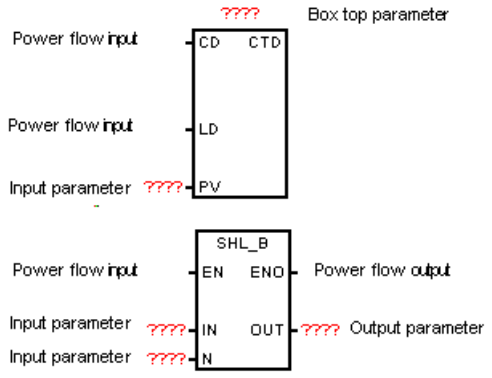
LAD has three different instruction types: contacts, coils, and boxes.

### Contacts and coils




The mnemonic of the instruction is shown in the middle of the contact or coil. Signal flow is shown as leading in from the left and out toward the right. Contacts have signal flow in, signal flow out, and can have up to two parameters. Coils have signal flow in and up to two parameters.

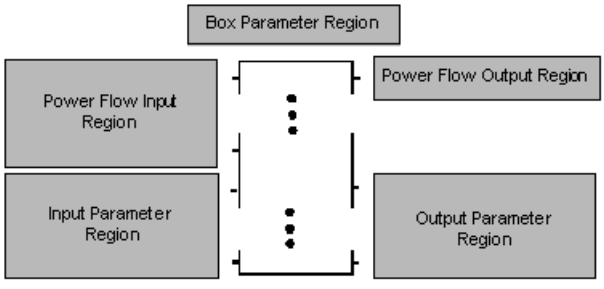
### Boxes




Boxes are represented by one of the two forms shown above. The mnemonic of the instruction is located at the top of the box.

Short lines are shown jutting outward from the box where a signal flow may be connected or a parameter may be assigned. The text of a parameter resides to the immediate left and right of the box or immediately above the box for box header parameters.

### Box parameters




The diagram above depicts the five regions of a box instruction reference that contain signal flow inputs, signal flow outputs, box header parameters, input parameters, and output

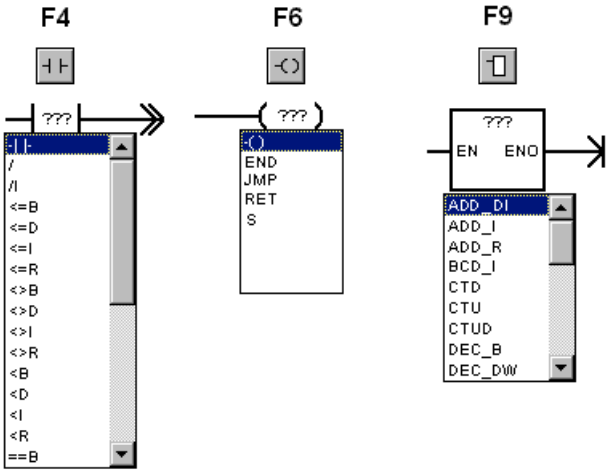
parameters. While there can only be one box header parameter and only one signal flow output, the other three regions can contain many signal flows or parameters.

## 6.1.5 Generic instructions in LAD

Generic instructions provide a quick, keyboard-controlled method for picking and placing instructions in LAD. When the F4, F6 or F9 keys (or the respective toolbar buttons) are pressed, a generic instruction will be placed at the cursor location and a list box will appear below it. The list box will contain a sorted list of instructions of the same type. The F4 key or toolbar button places a generic contact and displays a list of contacts. The F6 key or toolbar button places a generic coil and displays a list of coils. The F9 key or toolbar button places a generic box and displays a list of boxes.

The instruction that is selected from the list box is placed in the network, replacing the generic instruction.

You can undo your selection by hitting the <Escape> key. When you click the label of a generic instruction or select the generic instruction and press <Enter>, the list box will reappear.




All rules for placing instructions also apply for placing generic instructions, as well as for placing the instructions selected from the list box.

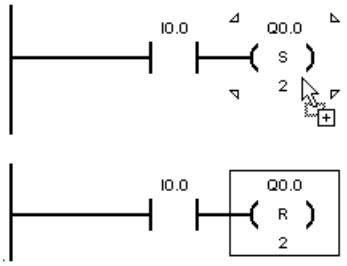
## 6.1.6 Retaining parameters in LAD

In LAD, instructions can retain the parameter assignments of the instruction that is overwritten. The following conditions must be met:

- The editor must be in Overstrike mode and the instruction must be placed directly over an existing instruction.
- The instructions must be of the same type.

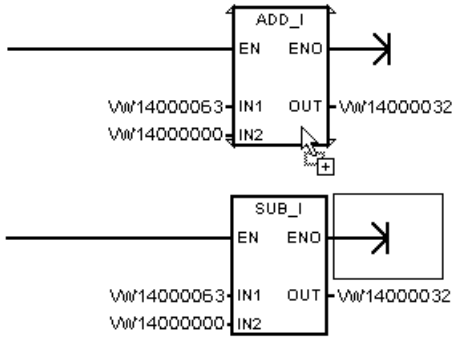
- For contacts and coils, the instructions must have the same number of parameters.
- For boxes, the instructions must have the same number of signal flow inputs and outputs and the same number of parameters in each parameter region.

**Retain coil parameters**



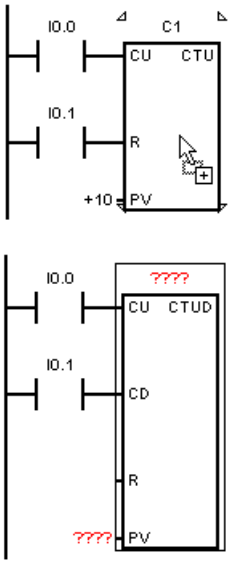

In the example above, the "reset" coil retains the parameter assignments of the "set" coil, because both instructions are coils and both contain two parameters.

**Retain box parameters**




In this example, the "subtract integers (16 bits)" box retains the parameter assignments of the "add integers (16 bits)" box. These box instructions each contain one signal flow input, one signal flow output, no box header parameter, two input parameters, and one output parameter.

**Box parameters not retained**




In this final example, the "count up-down" box does not retain the parameters of the "count up" box. The instructions do not have the same number of signal flow inputs.

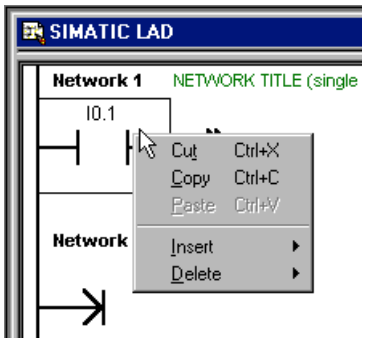
## 6.1.7 Editing instruction parameters

**Cutting, copying, pasting, or deleting multiple networks**

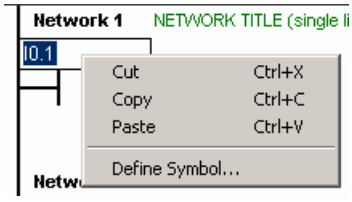
You can select multiple adjacent networks by dragging the mouse or using the SHIFT key with the UP and DOWN ARROW keys. You can cut, copy, paste, or delete your selection. Use the toolbar buttons; choose a command from the Edit menu, or right-click to bring up a pop-up menu of editing options.

### Editing fields, instructions, addresses, and networks

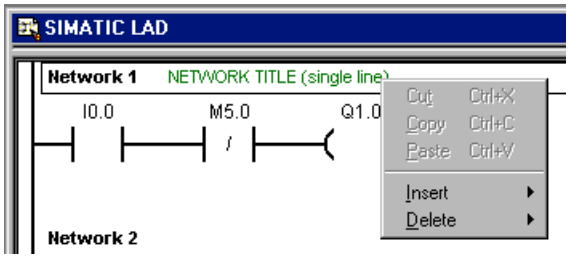
- When you click an **empty field** in the Program Editor, a box appears to indicate which field you have selected. You can use the pop-up menu to paste text from the clipboard into the empty field or insert a new row, column, vertical line, or network at that location. You can also delete the network from the location of the empty field.
- When you click an **instruction**, a box appears around the instruction to indicate which instruction you have selected. You can use the pop-up menu to cut, copy, or paste over the instruction, as well as to insert or delete a row, column, vertical line, or network at that location.




- When you click an **instruction parameter**, a box appears around the field to indicate which parameter you have selected. You can use the pop-up menu to undo your typing, to cut, copy, paste, or delete information, or to quickly select all the contents of the field. You can also use a double-click to select all.




- When you click a **network title**, a box appears around the title area. However, the pop-up menu allows you to perform edits on the entire network, not just the title. You can cut, copy, or paste over the network, insert a new network or delete the existing network.




You can also use toolbar buttons, standard Windows control keys, and the Edit menu to cut, copy, or paste a selection.

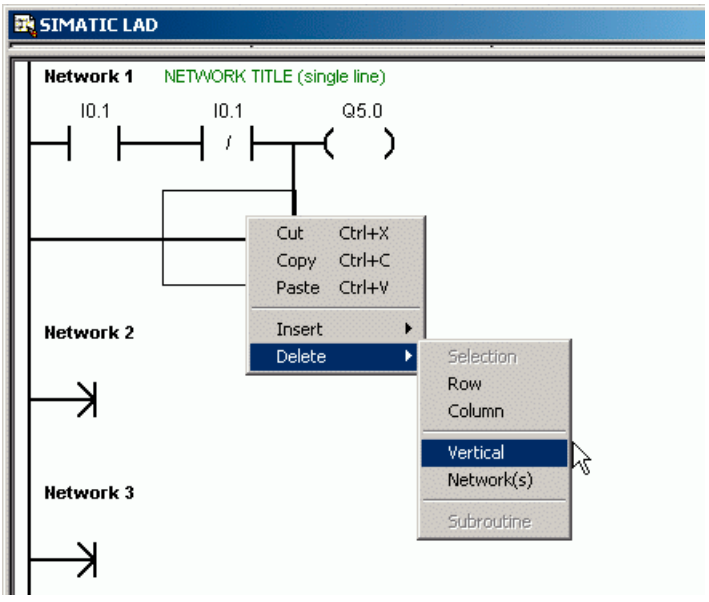
### Moving elements closer together

You can cut and paste elements and lines, and delete rows or columns. However, the Program Editor requires a certain amount of space between elements. In some cases, you

may not be able to place one element closer to another element (for instance, a horizontal line segment is required between box instructions and cannot be removed).

### Deleting elements

You can delete individual fields by using the DELETE or BACKSPACE key. You can delete rows, columns, vertical lines, and networks by using the Edit menu or right-clicking to bring up the pop-up menu.




**Note**

In order to correctly select a vertical line for deletion, always place the cursor on the field to the left of the vertical line.

## 6.1.8 Structuring a program with the help of the symbol table

Open a symbol table by using one of the following methods:

- Click the button of the symbol table in the "Navigation" toolbar (Page 500).
- Select the menu command **"View > Symbol Table"**.
- Open the directory of the symbol table in the project tree (Page 447) and double-click the button.

## Editing in a table

### Making symbol assignments in the symbol table

To assign a symbolic name to an address, proceed as follows:

1. Open the symbol table with one of the described entry types.
2. Enter the symbolic name (e.g. Input1) in the "Name" column. The maximum symbol length is 23 characters. Use the Tab, Enter or arrow keys to confirm your work and move to the next field.

   **Note**

   - Until you assign an address to the symbol, it is shown as an undefined symbol (green wavy line). After you complete the Address column assignment, the green wavy line is removed.
   - If you have selected simultaneous symbolic and absolute view of addresses for your project, lengthy symbolic names are truncated with a tilde (~) in the LAD Program Editor window. You can place your mouse pointer over the truncated name to see the entire name displayed in a tooltip.

3. Enter the address (e.g. I0.0) in the "Address" column.
4. Enter a comment (optional: 79 characters maximum).

**Note**

In the Programming Tool, you can create multiple symbol tables. You cannot, however, use any character string more than once as symbolic name - neither within the same table nor distributed over multiple tables.

By contrast, it is permissible to use the same name in various local variable tables.

### Use of quotation marks is not supported

Global symbol names must not be set in double quotation marks. The symbol table interprets them as illegal syntax (the symbol is displayed in red until the quotation marks are removed).

### Inserting additional rows

Proceed as follows to insert additional rows into a symbol table:

- Select the menu command **"Edit > Insert > Row"**. A new row is inserted above the current location of the cursor in the symbol table.
- Right-click a field of the symbol table. In the shortcut menu select the command **"Insert > Row"**. A new row is inserted above the current location of the cursor.
- You can move the cursor to a field in the last row and press the down arrow key in order to insert a row at the bottom of the symbol table.

### Defining symbols

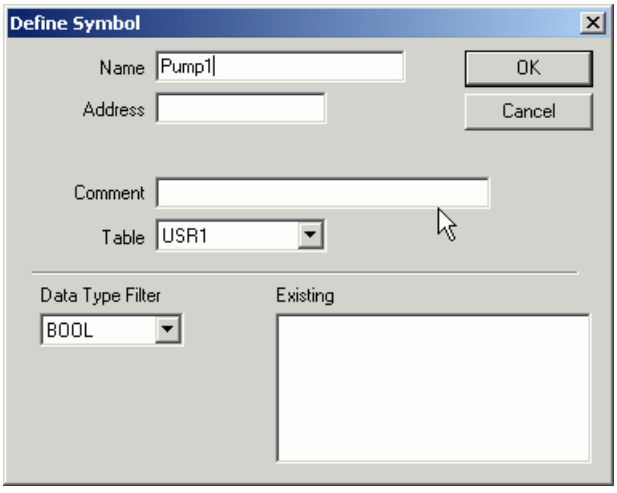
The "Define Symbol" command allows you to select an existing symbol from a list, and it also allows you to create new symbol assignments while you work in the Program Editor window or in a status chart, without having to open up the symbol table.

**Note**

You cannot use the "Define Symbol" command unless you are working in the Symbolic Addressing view. (You can check this in the "View" menu: A check mark next to the "Symbolic Addressing" option means that this view is turned on.)

Using the "Define Symbol" command:

1. Enter at least one character of the desired symbolic name in the Program Editor window or a status chart address field and press the Enter key.
2. Right-click the incomplete (or incorrect) symbolic name, then select "Define Symbol" from the shortcut menu.
3. Choose an existing symbol from the list or define a new symbol in the dialog box "Define Symbol".




4. Click "OK" to confirm your work and close the dialog box or click "Cancel" to cancel.

**Note**

If you have selected simultaneous symbolic and absolute view of addresses for your project, lengthy symbolic names are truncated with a tilde (~) in the LAD Program Editor window. You can place your mouse pointer over the truncated name to see the entire name displayed in a tooltip.

**Sorting by column**

You can sort the table by the "Name" or "Address" column. You can sort a column in ascending or descending (alphabetical) order.

To sort a column in a particular order, select the column by clicking in the body. Avoid clicking on the column header ("Name" or "Address").

- To sort the column in ascending order, from A to Z, click the Sort ascending button or choose the menu command **"View > Sort Ascending"**.
- To sort the column in descending order, from Z to A, click the Sort descending button or choose the menu command **"View > Sort Descending"**.

To reverse the sort, click the column header ("Name" or "Address"). When you click the column header, you toggle between ascending and descending sorts.

## Working with multiple tables

### Creating additional symbol tables

By default, the Symbol Table window displays one tab for user-defined symbolic names (USR1). If you have renamed any POUs, the Symbol Table window shows a tab called "POU Symbols".

There are several ways to create additional symbol tables for user-defined symbolic names:

- In the instruction tree, right-click the "Symbol Table" directory and, in the shortcut menu, select the command **Insert Symbol Table**.
- Open the window "Symbol Table" and call the menu "Edit" or right-click and select the command **Insert Contents > Table**.

**Note**

After you have inserted a new symbol table, a new tab 

is displayed at the lower edge of the "Symbol Table" window.

### Moving between symbol tables

If you have created more than one symbol table, you can access the tables by any of the following methods:

- If the Symbol Table window is open, you just need to click the 

tab of the desired symbol table at the bottom of the Symbol Table window to move between tables.
- From the instruction tree, expand the Symbol Table directory icon and double-click the desired symbol table to view the tab for that table within the Symbol Table window.

### "POU Symbols" tab

If you have assigned symbolic names for any of the POUs or other components (such as a status chart, data block or symbol table) in your project, the symbolic name assignments are listed on the "POU Symbols" tab of the Symbol Table window.

This tab is read-only; you cannot edit the assignments from here.

If you want to change an assignment, you must edit the Properties dialog box for the component in question. From the Project branch of the instruction tree, right-click the component to bring up the dialog box "Properties".

All symbolic assignments must use valid syntax. If you violate the guidelines for making a symbolic name assignment, the Programming Tool reports an error when you try to compile your program.

## Symbolic and absolute addressing

### Toggling between symbolic and absolute addressing mode

After you make symbol and absolute address associations in a symbol table, you can toggle the display between symbolic and absolute addressing. To do this, proceed in one of the following ways:

- Select the menu command **"View > Symbolic Addressing"** to toggle symbolic addressing on and off
- Use the **<Ctrl+Y>** shortcut key to toggle symbolic addressing on and off

A check mark in front of the command "Symbolic Addressing" means that symbolic addressing is on. By default, symbolic addressing is on when you create the first project.

### Viewing symbolic and absolute addresses simultaneously

To view symbolic addresses and absolute addresses simultaneously in an LAD program, use the menu command **"Tools > Options"** and open the tab "LAD Editing". Select the option button "Display symbol and address".

**Note**

Symbolic addresses are only displayed in your project when the Symbolic Addresses view is on. Otherwise, even if you select "Display symbol and address", only the absolute address is shown.

If you have selected simultaneous symbolic and absolute view of addresses for your project, lengthy symbolic names are truncated with a tilde (~) in the LAD Program Editor window. You can place your mouse pointer over the truncated name to see the entire name displayed in a tooltip.

## Symbols

### Understanding the symbol scope

In the symbol table you can assign symbolic names to the addresses of inputs and outputs and to addresses in the automation system memory. When you define a symbol in the symbol table, the symbol has a global scope. This means you can use the name of the symbol in any POU as a reference to the data at the address of that symbol. By contrast, if you assign a symbolic name using a local variable table, the local variable is restricted in scope to the POU where it was defined.

### Memory types

You can create symbolic names for the following memory areas: I, Q, M, SM, V, S, C, T.

### Entry errors

- Illegal syntax - red text
  Example:

| | Name | Address |
|---|---|---|
| 1 | 2name | I0.0 |
| 2 | | VBB0 |

The first character cannot be a digit.
Invalid address, VB14000000 would be correct
Reserved keywords cannot be used as symbols

- Illegal use - red wavy line
  Example:

| | Name | Address |
|---|---|---|
| 1 | Pump1 | I0.0 |
| 2 | Pump1 | I0.0 |

Duplicate name
Duplicate address

- Undefined symbol - green wavy line
  Example:

| | Name | Adresse |
|---|---|---|
| 1 | Pumpe1 | |
| 2 | Pumpe2 | VB14000000 |

Name with invalid address

**Note**

- Symbolic names are permitted to contain alphanumeric characters and underscores. They are also permitted to contain extended characters (ASCII 128 to ASCII 255). The first character is restricted to alpha and extended characters only.
- When naming global symbols it is not permissible to use double quotation marks. The symbol table interprets them as illegal syntax (the symbol is displayed in red until the quotation marks are removed).
- It is illegal to use keywords (Page 281) as symbolic names, or to use names that begin with a number or contain characters that are not alphanumeric or in the extended character set.
- The maximum permissible length for a symbolic name is 23 characters.
- If you attempt to create a symbol assignment to correct the "Undefined Symbol" error in your program, but do not press the TAB key, ENTER key or an ARROW key after typing the Address value and before you return to your program, the error is still displayed. To resolve this problem, return to the Symbol Table window, place the cursor in the Address field, and press the TAB, ENTER or an ARROW key to complete the address assignment.

### Tips for creating symbols

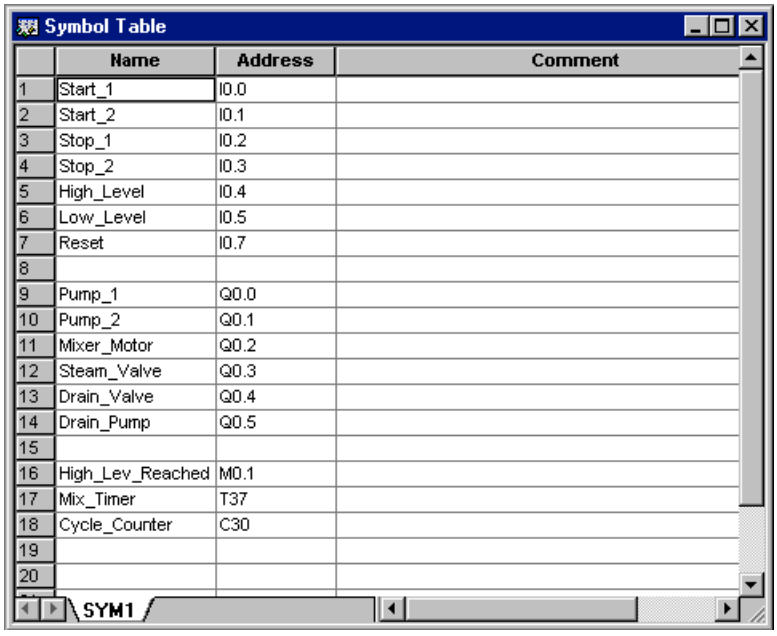
- You can make symbol assignments at any time: before, during or after the creation of program logic.
- You can document your symbols by making remarks in the "Comments" column.
- You can use the command "Define Symbol" to select from a list or create new symbolic names as you program.

- You can sort the values in the symbol table by clicking on the various table column headers.
- You can resize columns by dragging on their edges.
- You can use the menu command File > Print to print the symbol table.
- Once you have created symbolic assignments, you can display your address assignments and symbolic comments for each network in the LAD program editor by enabling the function "Symbol Information Table".

**Examples of tables**

Instruction operands are defined to accept one specific data type. For information about data types, see Data Types (Page 269).

This is an example of a symbol table:




## See also:

Local variable table (Page 229)

Symbolic addressing (Page 444)

Symbol information tables (LAD) (Page 445)

Overview of the addressing (Page 26)

Direct addressing (Page 261)

DB address representation (Page 446)

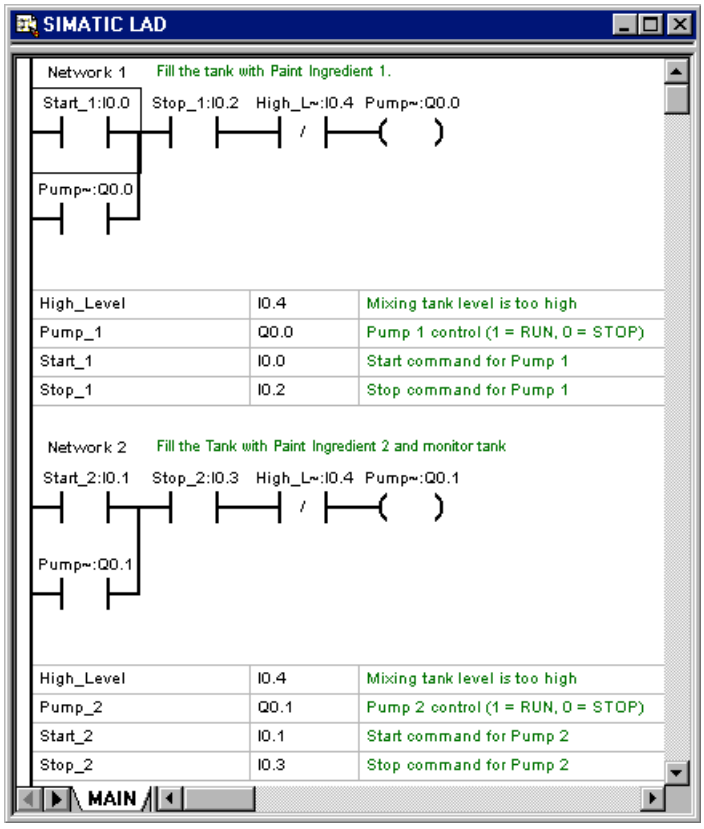
Keywords (Page 281)

## 6.1.9 Displaying symbol information tables for each network

Use one of the following methods to view or hide symbol information tables in LAD:

- Click the  button in the toolbar.
- Select the menu command **"View > Symbol Information Table"**.
- Press the **<Ctrl+T>** shortcut key.

When you view symbol information tables, the symbolic names, absolute addresses, and comments are displayed below each network in your LAD program. The table lists information for any symbols in that network; networks that do not contain global symbols do not display a symbol information table.




**Tips:**

- Use the arrow keys to scroll through the symbol information table.
- You can resize the table by dragging the splitter cursor from the column edges.
- Use the button "Options" in the dialog box "Print" to enable printing of symbol information tables when you print your program.
- To conserve vertical space on the screen and view more networks of your program, you can toggle the symbol information tables on and off by using the menu command **"View > Symbol Information Table"**.

**Troubleshooting**

If you do not see symbol information tables beneath the networks in your program, consider these points:

1. Are symbol information tables enabled? Check this in the menu "View": The menu command "Symbol Information Table" should have a check mark beside it.
2. Are symbols (or the absolute addresses that correspond to the symbols) used in the network? Networks that do not contain symbolic names do not display a symbol information table.
3. Have you entered the symbol in the symbol table? No address or comment information is available for a symbol until it is fully defined.

**See also:**

Symbol table (Page 413)

"LAD Editing" tab (Tools > Options) (Page 490) (Here you can specify how the symbol information table is displayed when a program is processed.)

"LAD Program Status" tab (Tools > Options) (Page 492) (Here you can specify how the symbol information table is displayed when you display the program status.)

Information on using the software (Page 375)

First steps: Contents (Page 17)

## 6.1.10 Data types

There is one circumstance under which you must use data types when programming in the Programming Tool:

If you assign values in the local variable table, then you must specify a data type for every local variable.

By explicitly specifying a data type for a value, you give the Programming Tool clear instructions on how much memory must be assigned for the value (e.g. the value 100 can be stored as BYTE, WORD or DWORD) and how the value is to be represented (e.g. should 0 be interpreted as BOOL or as a numeric value?).

The instructions and parameterized subprograms are recognized using a precise definition. This definition is also called the signature. For all standardized instructions, the data types permissible for the operands of the instruction are specified in the signature. For parameterized subprograms, the signature of the subprogram is generated by the user in the local variable table.

The Programming tool software implements simple data type checking. If a data type is specified for a local or global variable, the software checks that the data type of the operand corresponds to the signature of the instruction.

| Elementary data types | Description | Memory area |
|---|---|---|
| BOOL (bit) | Boolean | 0 to 1 |
| BYTE | Byte, Unsigned | 0 to 255 |
| WORD | Integer number (16 bits) without sign | 0 to 65535 |
| INT (integer) | Integer number (16 bits) with sign | -32768 to +32767 |

| Elementary data types | Description | Memory area |
|---|---|---|
| DWORD (Double Word) | Integer number (32 bits) without sign | 0 to 4294967295 |
| DINT (Double Integer) | Integer number (32 bits) with sign | -2147483648 to +2147483647 |
| REAL | 32-bit floating point | +/- 1.175495E-38<br>to +/- 3.402823E+38$^{+38}$ |

| Complex data types | Description | Memory areas | |
|---|---|---|---|
| TON | Switch-on delay | 100 ms | T0-T15 |
| | | 10 ms | From T16 |
| TOF | Switch-off delay | 100 ms | T0-T15 |
| | | 10 ms | From T16 |
| CTU | Up counter | Depending on CPU | |
| CTD | Down counter | Depending on CPU | |
| CTUD | Up-down counter | Depending on CPU | |
| SR | Bistable function block:<br>priority set | N/A | |
| RS | Bistable function block:<br>priority reset | N/A | |

**The Programming Tool has two data type check stages**

1. **Elementary data type check**
   For the elementary data type check, if a symbol or a variable is assigned a data type, then all data types which correspond to the bit size of the user-defined data type are assigned automatically. If, for instance, you specify DINT as the data type, then the local variable is also automatically assigned the DWORD data type, because both data types are 32-bit types. The REAL data type is not automatically assigned, although it is also a 32-bit data type. The REAL data type is defined so that it does not have any equivalent data types: it is always unique. The elementary data type check is performed when using local variables and variables of user data blocks.

   | User-defined data types | Equivalent data type |
   |---|---|
   | BOOL | BOOL |
   | BYTE | BYTE |
   | WORD | WORD, INT |
   | INT | WORD, INT |
   | DWORD | DWORD, INT |
   | DINT | DWORD, INT |
   | REAL | REAL |

2. **No data type check**
   This mode is only available for global variables where no data types can be specified. If a data type check is not active, all data types of the same size are automatically assigned to the symbol. Example: A symbol, which is assigned to address VD14000000, is assigned the following data types automatically by the programming software: DWORD, DINT, and REAL.

**Data types with defined size for global symbols**

| User-defined address | Assigned equivalent data type |
|---|---|
| V14000000.0 | BOOL |
| VB14000000 | BYTE |
| VW14000000 | WORD, INT |
| VD14000000 | DWORD, DINT, REAL |

**Advantages of the data type check**

The benefit of data type checking is to assist the user in avoiding common programming mistakes. If an instruction supports numbers with signs, the Programming Tool flags the use of unsigned numbers in instruction operands. Example: The comparison < I is an operation with sign. -1 is less than 0 for operands with sign. However, if the instruction < I supports unsigned data types, then the programming itself must ensure that the following does not occur:  While a program is being executed, an unsigned value of 40,000 is actually smaller than 0 for the instruction < I. If it cannot be guaranteed that the unsigned numbers for signed instructions do not exceed the positive and negative limiting values, then unpredictable events can occur in your program or in the mode of operation of the control.

**Working with instructions to convert the data type**

Conversion instructions convert one data type into another. The Programming Tool supports the following conversion instructions to transfer values between the elementary data types.

| Conversion instructions | Complete data type check, permissible addresses | | Data type check, permissible addresses | |
|---|---|---|---|---|
| BYTE in INT | IN:<br>OUT: | BYTE<br>INT | IN:<br>OUT: | BYTE<br>WORD, INT |
| INT in BYTE | IN:<br>OUT: | INT<br>BYTE | IN:<br>OUT: | WORD, INT<br>BYTE |
| INT in DINT | IN:<br>OUT: | INT<br>DINT | IN:<br>OUT: | WORD, INT<br>DWORD, DINT |
| DINT in INT | IN:<br>OUT: | DINT<br>INT | IN:<br>OUT: | DWORD, DINT<br>WORD, INT |
| DINT in REAL | IN:<br>OUT: | DINT<br>REAL | IN:<br>OUT: | DWORD, DINT<br>REAL |
| REAL in DINT (ROUND) | IN:<br>OUT: | REAL<br>DINT | IN:<br>OUT: | REAL<br>DWORD, DINT |

## See also

Open (File menu) (Page 379)

## 6.1.11 Enter corrections with the Find and Replace function

To use Find, Replace or Go To:

- Select the **Edit > Find, Edit > Replace** or **Edit > Go To** menu command.
- Press CTRL+F for Find, CTRL+H for Replace, or CTRL+G for Go To

**Working with the function:**

For details, see the "Search function" and "Paste function" sections.

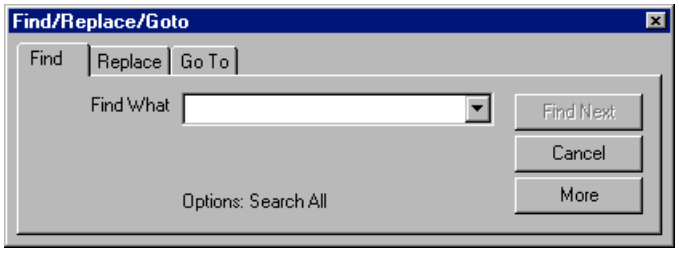
**Where they can be used**

Find, Replace, and Go To can be used in the Program Editor window, and in Local Variable Tables, Symbol Tables, Status Charts, the Cross Reference table, and the Data Block.

**How they work**

- The Find function allows you to locate a specified string, such as an operand, network title, or instruction mnemonic. (Find does not search network comments, only network titles. Find does not search the network symbol information tables that are available in LAD.)
- The Replace function allows you to replace the specified string. (Replace does not operate for instruction mnemonics.)
- The Go To function allows you to move quickly to another location by specifying the number of the network or row to which you wish to navigate.

**The Find function**




1. To search, type a string in the "Find What" field.
2. To move to the next occurrence of the search string, click the "Find Next" button.

**Note**

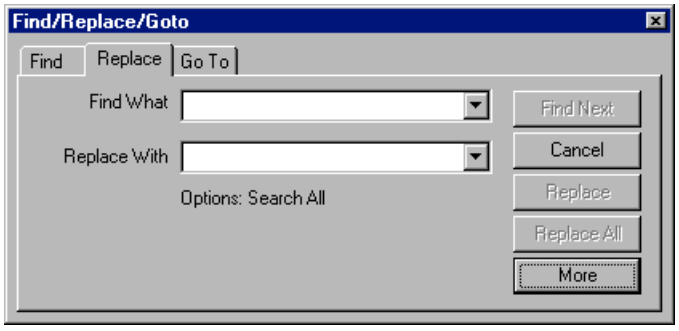
Under some circumstances, the Find Next command may appear to search program code in an irregular order; actually, this order simply reflects the way that operands are stored in the code.

To define your search further, click the "More" button, which allows you to customize the search as
described below:

- You can choose a direction for the search by using the "Search" list box.
- You can search only for strings that have the same uppercase/lowercase value as the string that you entered. To do this, select the "Match Case" checkbox.

- You can eliminate strings that contain your search phrase as part of a larger word by selecting the "Whole Word" checkbox.
- You can search through all POUs (OB1 and all subprograms and interrupt programs) or all instances of a Local Variable Table, Symbol Table, or Status Chart, by selecting the appropriate checkbox.
- You can specify a range of lines to search. If you have selected a range of networks in the Program Editor, they appear as the default range in the Find dialog box. You can also type in the network or row numbers for the beginning and end of the search.
- You can specify whether or not to search network titles and/or program code by selecting the appropriate checkboxes.

**The Replace function**




1. Type the string that you are searching for in the "Find What" field.
2. Type the string that you want to use as the replacement for the search string in the "Replace With" field.
3. To find an occurrence of the search string, click the "Find Next" button.

   **Note**

   Under some circumstances, the Find Next command may appear to search program code in an irregular order; actually, this order simply reflects the way that operands are stored in the code.

4. If you want to replace the string, click "Replace". If you have defined your search carefully, and there is no risk that strings could be modified erroneously, you can click "Replace All" to replace all instances of the string without examining them individually.

To define your search further, click the "More" button, which allows you to customize the search as described below:

- You can choose a direction for the search by using the "Search" list box.
- You can search only for strings that have the same uppercase/lowercase value as the string that you entered. To do this, select the "Match Case" checkbox.
- You can eliminate strings that contain your search phrase as part of a larger word by selecting the "Whole Word" checkbox.
- You can search through all POUs (OB1 and all subprograms and interrupt programs) or all instances of a Local Variable Table, Symbol Table, or Status Chart, by selecting the appropriate checkbox.

- You can specify a range of lines to search. If you have selected a range of networks in the Program Editor, they appear as the default range in the Find dialog box. You can also type in the network or row numbers for the beginning and end of the search.
- You can specify whether or not to search network titles and/or program code by selecting the appropriate checkboxes.

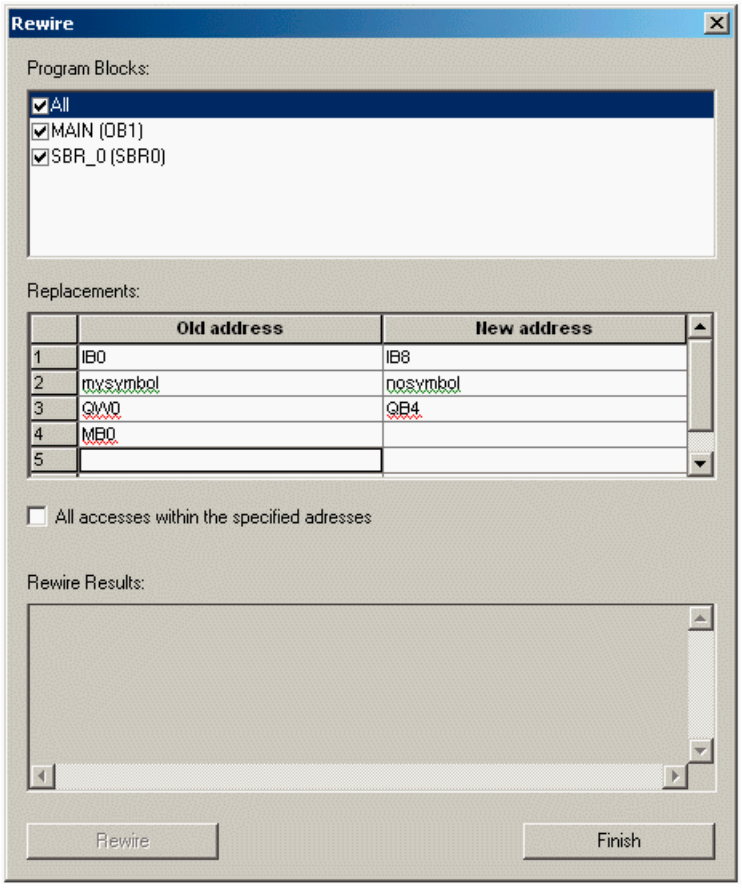
## 6.1.12 Rewire

With the "Rewire" function, PLC user program operands can be centrally modified, e.g. IW0 to IW8. This means that user programs can be quickly adapted to a modified I/O configuration.

### Example

You are writing a PLC user program for a series of machines. As the I/O configuration of some machines differs due to the differing machine configurations, user program operands must currently be modified individually. Under certain circumstances, this can involve several hundred operands. A list of operands to be changed, e.g. inputs and outputs, can be entered for these machines via the "Rewire" dialog. This list is processed with the "Rewire" function and the operands are modified in the user program.

Use the "Rewire" dialog box to rewire operands:



## Procedure

1. **"Program blocks" list**
   Displays a list of all POUs available in the project. Select the POUs for rewiring.
2. **"Replacements" list**
   In this list, enter the old and the new operands for rewiring.
   Permissible operands include:
   - Inputs
   - Outputs
   - Bit memories
   - Special bit memories
   - Variable memories / data blocks
   - Timers
   - Counters
3. Editing the list of operands:
   The following functions are supported using the shortcut menu (right mouse button):
   - Cut (Ctrl+X)
   - Copy (Ctrl+C)
   - Paste (Ctrl+V)
   - Select All (Ctrl+A)
   - Insert Row (Ctrl+I)
   - Delete Selection

   This means that it is possible to copy the list or parts of the list from other or into other applications, e.g. Microsoft Excel.
4. Old operand
   In this column, enter the name or the address of the operand which you wish to rewire.
5. New operand
   In this column, enter the new name or the new address of the operand. Please observe that the type of the new operand corresponds to the type of the old operand, e.g. old operand IW0 and new operand IW4, not IB4, or old operand DB9000.DBB0 and new operand MB0, not MW0.
6. Check the validity of the operand
   If the name of the operand (symbol) does not exist in the open project, then this is marked with a green wavy line. If the type of the old operand does not match that of the new operand, or if only one operand was entered (old or new operand), then this operand is marked with a red wavy line.
7. **Selection box "All accesses within the specified operands"**
   If this option is activated, operand ranges (BYTE, WORD, DWORD) are rewired.
   Example: You specify IW0 and IW4 as operand ranges. In this case, the operands I0.0 – I1.7 are rewired to operands I4.0 – I5.7. Operands from the unwired range (e.g. I0.1) can then no longer be entered individually into the table.

8. **"Rewiring results" list**
   After rewiring, the results are displayed in this list. This list comprises the list of operands, "Old operand" and "New operand". These list the individual blocks and the number of wiring operations that were carried out in each block. Using the shortcut menu (right mouse button), the results can be copied into other applications, e.g. Microsoft Word.
9. **"Rewire" button**
   Press this button to start rewiring.
10. **"Exit" button**
    Press this button to exit rewiring.
11. **Activate the Rewire function:**
    The "Rewire" dialog is opened in the LAD editor in the screen shortcut menu or in the "Edit > Rewire..." menu.

**Note**

Please observe the following when rewiring:

- The name or number of a POE cannot be modified with the "Rewire" function. For this, use the functions "Rename" or "Properties..." in the POU shortcut menu in the project tree (right-click, e.g. MAIN).
- Timers can only be rewired to timers, e.g. old operand T0, new operand T16, and counters only to counters, e.g. old operand C0, new operand C1.

**See also:**

Information on using the software (Page 375)

First steps: Contents (Page 17)

## 6.1.13 Valid BCD format ranges

| Size | Decimal | BCD | Hexadecimal |
|---|---|---|---|
| | **Minimum values:** | | |
| Word | 0 | 0000 0000 0000 0000 | 0000 |
| | **Maximum values:** | | |
| Word | 9999 | 1001 1001 1001 1001 | 9999 |

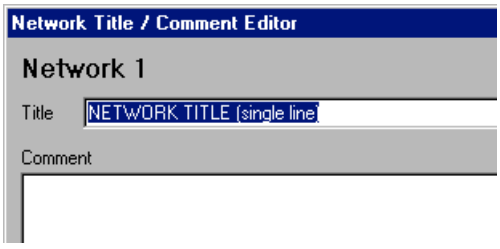
## 6.1.14 Entering comments about the program

There are two levels of comment in the LAD editor.

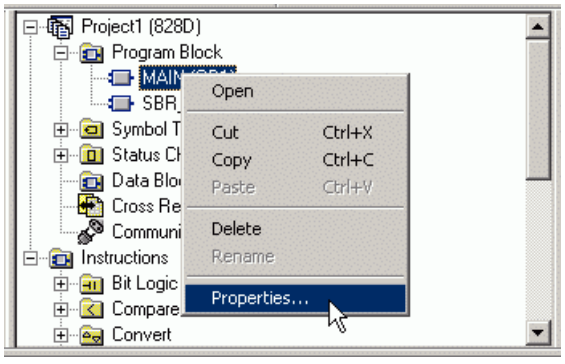
**Network comments**

Place your cursor anywhere in the network title line and double-click or hit the ENTER key in order to bring up the Network Title / Comment Editor. You can enter a title that identifies the network and a comment about the contents of the network. The network title is visible

within the Program Editor. The network comment is visible only within the Network Title / Comment Editor and when you print out program comments.
The maximum number of characters permitted in the network title is 127. The maximum number of characters permitted in the network comment is 1,023.

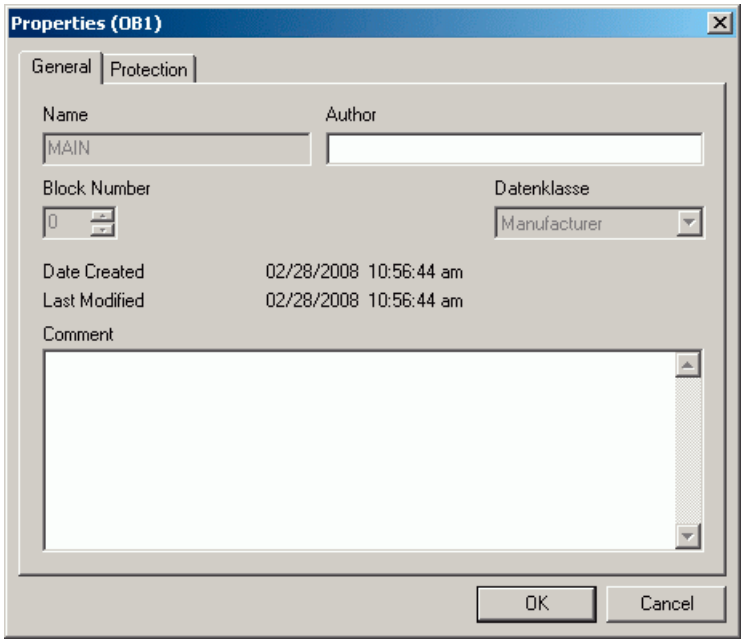



**Project component comments**

| | |
|---|---|
| Navigate in the project tree to a project component and right-click it to bring up the menu with the "Properties" dialog box for that item. |  |



Each component in your project has a Properties dialog box that allows you to name the component, identify the author of the component, and record comments about the components. For instance, if you have multiple status charts, the charts are represented as separate tabs within the Status Chart window and as separate components under the Status Chart icon in the Instruction Tree window. Each status chart has its own Properties box.




**Note**

You can save the properties of "MAIN (OB1)" in the comment of a version identification.

For additional information, see the following: Version identification (Page 82)

## 6.1.15 Description of the timer instructions

You can use timers to implement time-based counting functions. The Programming Tool instruction set provides three different types of timer.

- ON-delay timer (TON): for timing a single interval
- Retentive ON-delay timer (TONR): for accumulating a number of timed intervals
- OFF-delay timer (TOF): for extending time past a false condition (such as cooling a motor after it is turned off)

**Timer mode of operation**

| Timer | Actual value >= default | Enable input ON | Enable input OFF | Power cycle/First scan |
|---|---|---|---|---|
| **TON** | Timer bit ON, actual value continues counting to 32,767 | Actual value counts time | Timer bit OFF<br>Actual value = 0 | Timer bit OFF<br>Actual value = 0 |
| **TONR** | Timer bit ON, actual value continues counting to 32,767 | Actual value counts time | Timer bit and actual value maintain last state | Timer bit OFF, actual value may be saved(1) |
| **TOF** | Timer bit OFF, actual value = default, stops counting | Timer bit ON<br>Actual value = 0 | Timer counts after negative edge (ON-OFF) | Timer bit OFF<br>Actual value = 0 |

**Note**

With the reset instruction (R) (Page 119), you can reset the timers. The reset instruction performs the following:

Timer bit = OFF and actual value of the timer = 0

The timer TONR can only be reset with the reset instruction.

After a TOF timer is reset, the enable input must switch from ON to OFF so that the timer can be restarted.

**10-millisecond resolution**

The 10-ms timers count the number of 10-ms timer intervals that have elapsed since the active 10-ms timer was enabled. The execution of the timer instruction starts the timing; However, the 10-ms timers are updated at the beginning of each scan cycle (in other words, the actual value and timer bit remain constant throughout the scan), by adding the accumulated number of 10-ms intervals (since the beginning of the previous scan) to the actual value for the active timer.

Since the timer can be started anywhere within a 10-ms interval, the preset must be set to one time interval greater than the minimum desired timer interval. For example, to guarantee a timed interval of at least 140 ms using a 10-ms timer, the preset time value should be set to 15.

**100-millisecond resolution**

The 100-ms timers count the number of 100-ms timer intervals that have elapsed since the active 100-ms timer was enabled. These timers are updated by adding the accumulated number of 100-ms intervals (since the start of the previous scan cycle) to the timer's actual value each time the timer instruction is executed.

The actual value of a 100-ms timer is updated only if the timer instruction is executed. Consequently, if the current value of a timer is not updated, the timer is enabled but the instruction is not executed each scan cycle. and it loses time.

Likewise, if the same 100-ms timer instruction is executed multiple times in a single scan cycle, the numbers of 100-ms intervals are added to the timer's actual value multiple times and it gains time. Therefore, 100-ms timers should only be used where the timer instruction is executed exactly once per scan cycle.

Since the timer can be started anywhere within a 100-ms interval, the preset must be set to one time interval greater than the minimum desired timer interval. For example, to guarantee a timed interval of at least 2100 ms using a 100-ms timer, the preset time value should be set to 22.

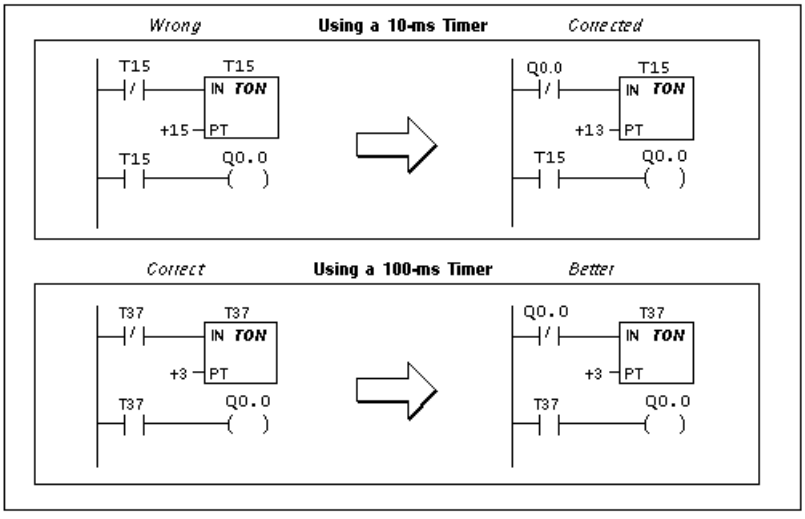
### Updating the actual timer value

The effect of the various ways in which actual timer values are updated depends upon how the timers are used. For example, consider the timer instruction shown in the diagram below.

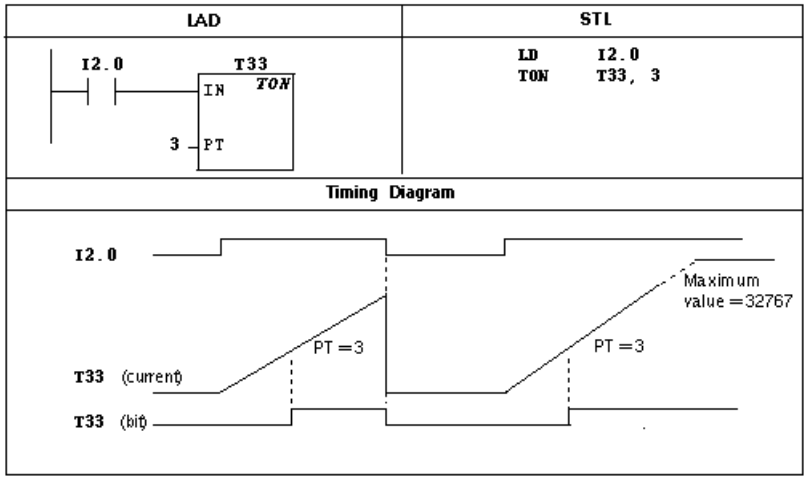
- In the case where the 10-ms timer is used, Q0.0 is never turned on, because the timer bit T15 is turned on from the top of the scan cycle to the point where the timer box is executed. Once the timer box has been executed, the timer's actual value and its timer bit are set to zero. When the NO contact T15 is executed, T15 is off and Q0.0 is turned off.
- In the case where the 100-ms timer is used, Q0.0 is always turned on for one scan cycle whenever the timer's actual value reaches the preset value.

By using the NC contact Q0.0 instead of the timer bit as the enable input to the timer box, the output Q0.0 is guaranteed to be turned on for one scan cycle each time the timer reaches the default.

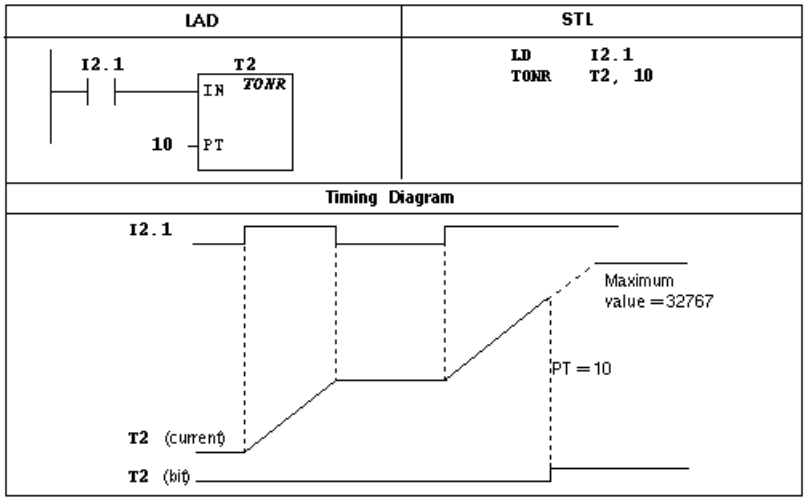
### Example of automatically triggered one-shot timer



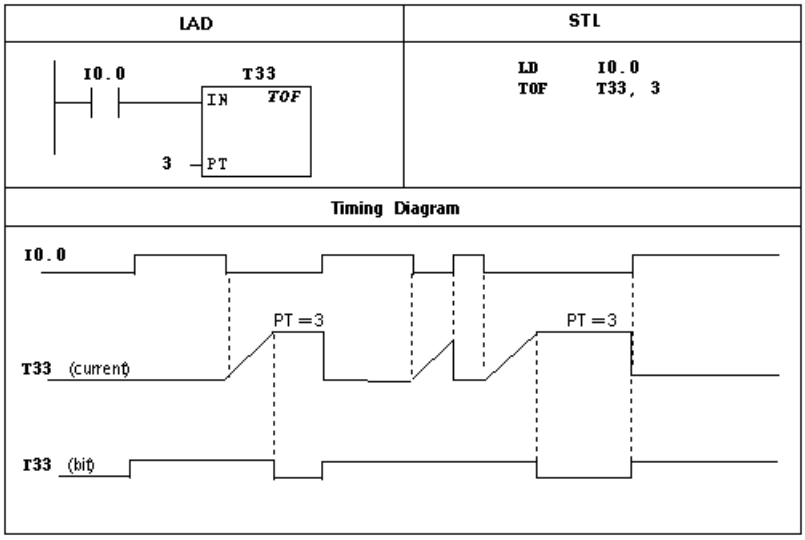

### Example of ON-delay timer



**Example of retentive ON-delay timer**




**Example of OFF-delay timer**




## 6.1.16 Description of the counter instructions

The up-counter (CTU) counts up from the actual value of the counter on a positive edge at the count up input. The counter is reset when the reset input turns on or when the reset instruction is executed. The counter stops upon reaching the maximum value (32,767).

The up-down counter (CTUD) counts up on a positive edge at the count up input and counts down on a positive edge at the count down input. The counter is reset when the reset input turns on or when the reset instruction is executed. Upon reaching the maximum value (32,767), the next positive edge at the count up input causes the actual count to wrap around to the minimum value (-32,767). Likewise on reaching the minimum value (-32,767),

the next positive edge at the count down input causes the actual count to wrap around to the maximum value (32,767).

The up and up-down counters have an actual value that maintains the actual counter value. They also have a preset value (PV) that is compared to the actual value whenever the counter instruction is executed. When the actual value is greater than or equal to the preset value, the counter bit (C bit) turns on. Otherwise, the C bit turns off.

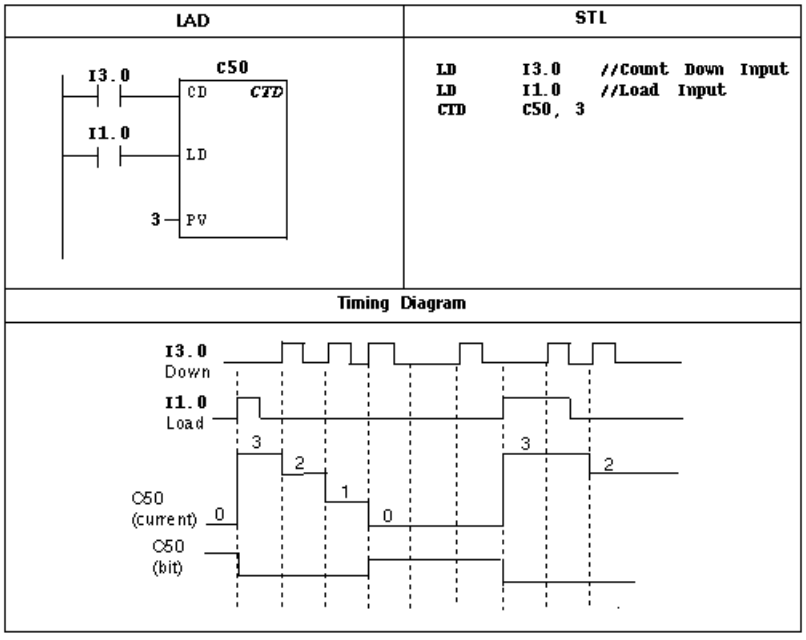
The down-counter (CTD) counts down from the actual value of the counter on a positive edge at the count down input. The counter resets the counter bit (Cxxx) and loads the actual value to the preset value (PV), if the load input (LD) is switched on. The counter stops upon reaching zero and the counter bit (C bit) turns on.

When you reset a counter using the reset instruction, the counter bit is reset and the actual counter value is set to zero. <F0> Use the counter number to reference both the actual value and the C bit of that counter.

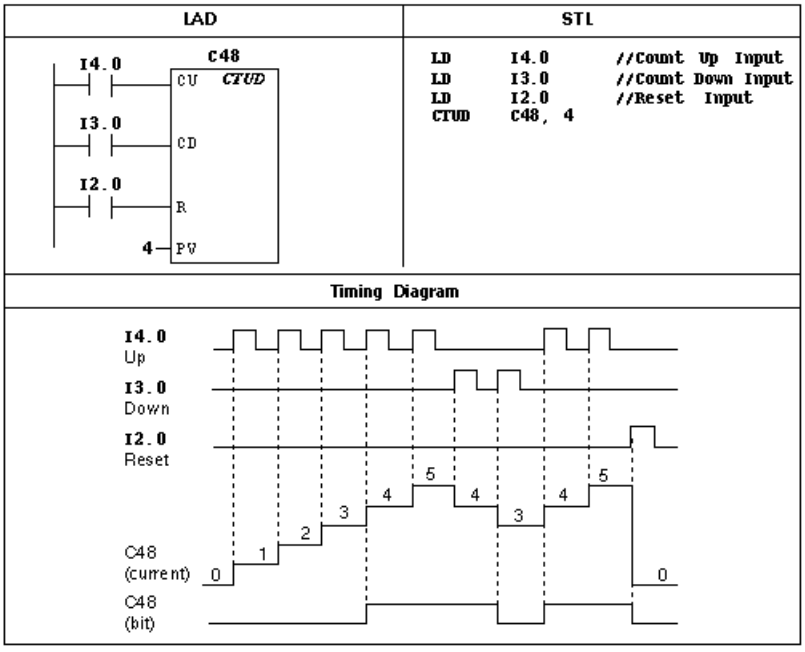
**Note**

Since each counter has its own actual value, you may not assign the same number to several counters (up-counters, up-down counters, and down-counters with the same number have access to the same actual value).

**Example of the CTD instruction in LAD and STL**



**Example of the CTUD instruction in LAD and STL**




## 6.1.17 Programming subprograms

### Getting started

| | |
|---|---|
| CALL instruction (Page 191) | |

These help topics are explained in the following:

### Using subprograms

Subprograms help you to partition your program. The instructions used in your main program determine the execution times of the specific subroutines. When the main program calls the subprogram for execution, the subprogram carries out its program to the end. Then, the system returns control to the main program at the point in the network from which the subprogram was called.

Subprograms are used to segment or divide your program into smaller, more manageable units. You can take advantage of this benefit when debugging and performing maintenance on your program. By working with smaller units, you can debug and troubleshoot these areas, as well as the entire program, faster and more easily. The automation system can also be more efficiently used by only calling the units that are needed, as all units may not have to be executed in every scan cycle.

Finally, subprograms can also be portable if your subprogram references only its parameters and the local memory. Make the subprogram portable by avoiding any use of global variables/ symbols (absolute addresses in I, Q, M, SM, V, T, C, AC memory areas). If your subprogram

has no calling parameters (IN, OUT or IN_OUT) or uses only local variables in the local data memory, you can export the subprogram and import it into another project.

To use a subprogram in your program, you must perform three tasks:

- Create the subprogram
- Define its parameters (if any) in the local variable table of the subprogram
- Call the subprogram from the appropriate POU (e.g. your OB1 block or another subprogram)

The Programming Tool automatically generates a subprogram call box instruction after the subprogram is created and its calling parameters are defined. The call instruction has the correct number and type of input/output parameters, based on your local variable table declarations for that subprogram. After a subprogram is created, it appears in the instruction tree.

To insert the subprogram call into another POU, drag the subprogram symbol from the instruction tree and drop into the required POU.

### How to create a subprogram

How to create a subprogram:

- From the Edit menu, choose the command **Insert > Subprogram**.
  - OR -
- From the instruction tree, right-click the "Program Block" symbol and select the command **Insert > Subprogram** from the shortcut menu.
  - OR -
- From the Program Editor window, right-click and select the command **Insert > Subprogram** from the shortcut menu.

The actual program organization unit (POU) is then no longer displayed in the program editor, but a new subprogram is instead. A new tab is inserted at the bottom of the program editor, which contains the new subprogram.

Now you can program the new subprogram or return to the POU which you were working on previously:

- If you want to assign the parameters of the subprogram now, you can define them using the local variable table (Page 229)for that subprogram.

  **Note**

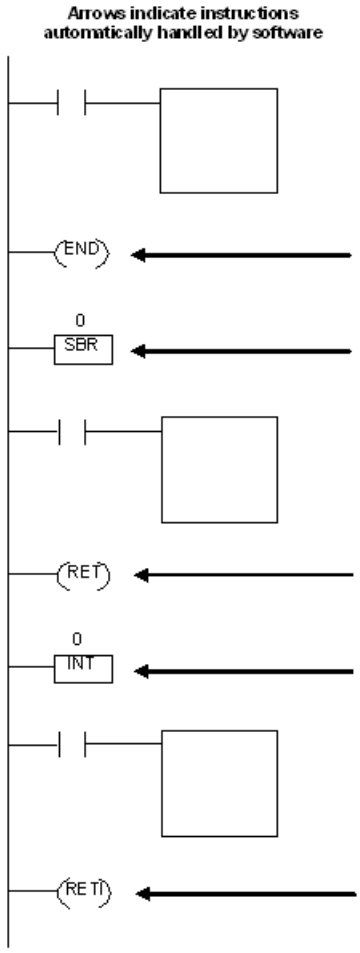
  Note that each POU in your program has its own local variable table. You must define the local variables for this subprogram in the local variable table, which will be displayed if you have opened the tab for the subprogram. Make sure that you always work on the correct tab when you are editing the local variable table.

- If you want to write the logic for the subprogram, as long as the tab is open you just need to start working in the program editor.
- If you want to work with another program organization unit, open the tab for the desired POU so that it is displayed in the program editor.

**Do not use a RET instruction to terminate your subprograms**

You may not insert the END instruction into a subprogram.

The editor automatically inserts the instructions to absolutely terminate program organization units (END for OB1, RET for SBR, and RETI for INT). An example is shown below.




## How to call a subprogram

After you insert a new subprogram and define its parameters (if any) in the local variable table of that subprogram, you can place a call to the subprogram in another POU of your program. (You can call a subprogram from OB1 or another subprogram; you cannot call the subprogram from itself.)

### LAD

For LAD programs, a customized subprogram call box instruction is generated after you assign parameters to the subprogram in the local variable table of the subprogram. This CALL instruction automatically includes the correct number and type of input, output, and in-out parameters for the subprogram.

To insert the CALL instruction in a POU of your LAD program:

1. Open the desired POU in the Program Editor window and scroll to the network where you wish to insert the subprogram call.
2. In the instruction tree, double-click to open the "Subprograms" directory. You can either drag the appropriate CALL instruction from the instruction tree and drop it on the appropriate cell of the network in the Program Editor window, or place your cursor on the corresponding cell in the Program Editor window and then double-click the CALL instruction in the instruction tree.
3. Edit the parameters of the CALL instruction in your program and assign valid operands to each parameter. Valid operands are as follows: memory addresses, constants, global symbols, local variables from the POU where the CALL instruction has been placed (not local variables from the subprogram that is being called).

**Note**

If you insert a call to a subprogram and then later modify the local variable table of that subprogram, the call becomes invalid. You must delete the invalid call and replace it with an up-to-date CALL instruction that reflects the correct parameters.

**STL**

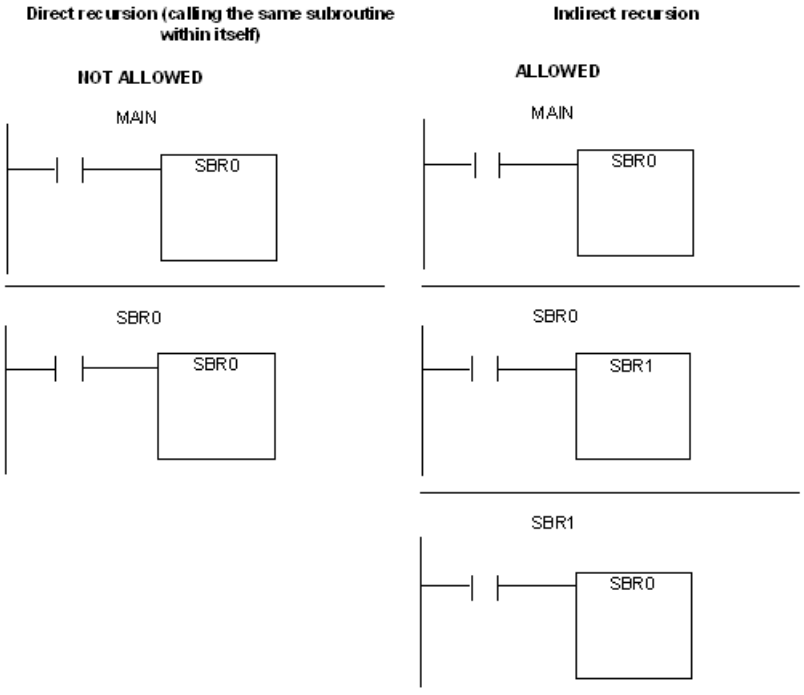
The CALL instruction (Page 191)is used to call a subprogram from an STL program.

When a subprogram is called, the entire logic stack is saved, the topmost stack value is set to 1, all other stack values are set to 0, and the called subprogram is processed. When this subprogram is completed, the stack is restored with the values saved at the point of call, and the calling program is then processed further.

Accumulators are common to subprograms and the calling program. No save or restore operation is performed on accumulators due to subprogram use.

### Nesting and recursion

You can nest subprograms (place a subprogram call within a subprogram) to a depth of eight. Direct recursion is not allowed. Example of direct recursion: SBR0 cannot be called from SBR0 (a subprogram cannot call itself). However, indirect recursion is permitted. Examples are illustrated below. Recursion should be used with caution.




## 6.1.18 Programming interrupt programs

| | |
|---|---|
| Programming of interrupt instructions (Page 131) | |

### Supporting CPUs

INT

828D
828D Step 2

### Using interrupt programs

Create an interrupt program:

- You will find these in the instruction tree under **Libraries > Interrupt programs**. You can make use of them by double-clicking or via copy and paste (in the project tree).

The actual program organization unit (POU) is then no longer displayed in the program editor, but a new interrupt program is instead. A new tab is inserted at the bottom of the program editor, which contains the new interrupt program.

- INT_0: servo-synchronous version
  For extremely fast responses to events for which such responses are absolutely essential.

  **Note**

  This interrupt program may contain a maximum of 500 instructions.

- INT_100: executed before MAIN, after reading the inputs
  Especially for marshaling data that is accessed (ready only) in the subsequent scan cycle.
- INT_101: executed after MAIN, before writing the outputs
  Especially for marshaling data that was previously written in the scan cycle, but which is to be output to other or additional outputs.

Now you can program the new interrupt program or return to the POU which you were working on previously:

- Note that each POU in your program has its own local variable table (Page 229). You must define the local variables for this interrupt program in the local variable table, which will be displayed if you have opened the tab for the interrupt program. Make sure that you always work on the correct tab when you are editing the local variable table.
- If you want to write the logic for the interrupt program, as long as the tab is open you just need to start working in the program editor.
- If you want to work with another program organization unit, open the tab for the desired POU so that it is displayed in the program editor.

The CPU processes interrupts outside the main(OB1). Only one program is ever active to process interrupts. If an interrupt program is presently being processed, then this program is terminated. It cannot be interrupted.

**Note**

Restrictions when using interrupt programs:

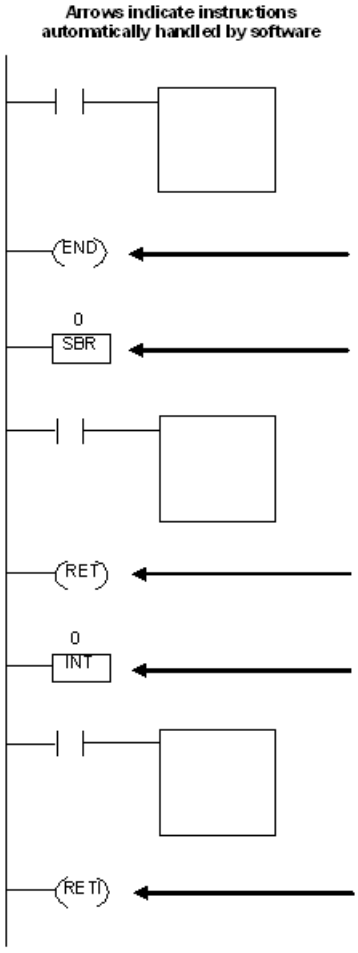
- You may not insert the END instruction into an interrupt program.

There are commands with direct access for fast responses to I/O signals in the interrupt program:

| | |
|---|---|
| NCI_CONTACT | \|I\| |
| NOI_CONTACT | \|/I\| |
| OUTI_COIL | (I) |
| SETI_COIL | (SI) |
| RESETI_COIL | (RI) |
| RBI_BOX | MOV_BIR |
| WBI_BOX | MOV_BIW |

**Note**

The editor automatically inserts the instructions to absolutely terminate program organization units (END for OB1, RET for SBR, and RETI for INT). An example is shown below.




## 6.1.19 Programming a local variable table

### Local variables

Each POU in your program has its own local variable table, with 60 bytes of local data memory. These local variable tables allow you to define variables that are restricted in scope: a local variable is only valid inside the POU where it was created. By contrast, global symbols, which are valid in every POU, can only be defined in the symbol table. In cases where you use the same symbolic name (e.g. INPUT1) for a global symbol and a local variable, the local definition takes precedence inside the POU where the local variable has been defined and the global definition is used in the other POUs.

You assign a declaration type (TEMP, IN, IN_OUT or OUT) and a data type (see Data types (Page 209)), but not a memory address, when you define a variable in a local variable table; the program editor automatically assigns addresses in the local data memory for all local variables.

These help topics are explained in the following:

## Declaration types for local variables

The type of local variable assignment you can make depends on the POU where you are defining the variable. The main program (OB1), interrupt programs, and subprograms can use temporary (TEMP) variables. Temporary variables are only available while the block is being executed. These addresses are then free to be overwritten again once the block is completed. Subprograms can also use call parameters (IN, IN_OUT, OUT).

| Type of declaration | Description |
|---|---|
| IN | The input parameter that is provided by the calling POU. |
| OUT | The output parameter that is returned by the calling POU. |
| IN_OUT | Parameter, the value of which is provided by the calling POU, is modified by the subprogram, and is then returned to the calling POU. |
| TEMP | Temporary variable that is briefly stored in the local data stack. Once the POU has been fully processed, the value of the temporary variable is no longer available. Temporary variables do not buffer the value following processing of the POU. |

## Data type checking for local variables

When you pass local variables to a subprogram as parameters, the data type that you have assigned in the local variable table of that subroutine must match the data type of the value in the calling POU.

**Example:**

1. You call SBR0 from OB1, using a global symbol called INPUT1 as an input parameter of the subprogram.
2. Inside the local variable table of SBR0, you have defined a local variable called FIRST as an input parameter.
3. When OB1 calls the SBR0 subprogram, the value of INPUT1 is passed to FIRST.
4. The data types of INPUT1 and FIRST must match.
5. If INPUT1 is REAL and FIRST is REAL, the data types match. If INPUT1 is REAL but FIRST is INT, the data types do not match and the program cannot be compiled properly until this error is corrected.

### Does your program need to use local variables?

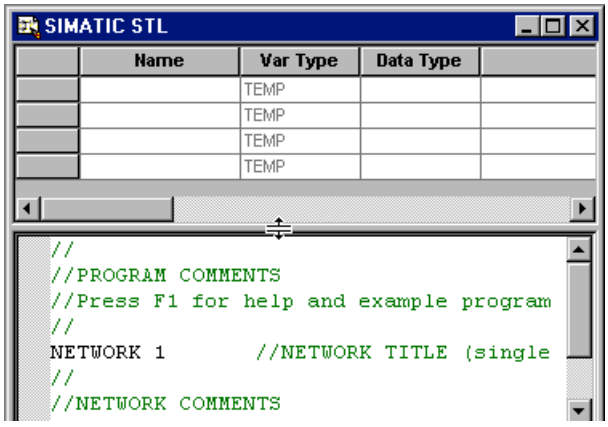
There are two reasons to use a local variable:

- You want to create portable subprograms that do not make references to absolute addresses or global symbols.
- You want to use temporary variables (local variables declared as TEMP) to perform calculations in order to free up PLC memory.

If these descriptions do not fit your situation, you do not need to use local variables; in that case, you only need to define global symbols in the symbol table.

### Show/hide the local variable table

If you drag the horizontal splitter cursor up to the top of the Program Editor window, the local variable table is no longer visible, but it is still there. Drag the splitter cursor down again to make the local variable table visible again.




### How to enter local variables in the local variable table

**Note**

It is most efficient to make assignments in the local variable table before using local variables in your program. When you use symbolic names in your program, the program editor checks first the local variable table of the appropriate POU, then the symbol table.

If the symbolic name has not been defined in either of the two tables, then the program editor treats the name as a global symbol. The program editor underlines the symbol using a green wavy line: Undefined_Local_Var. The Program Editor does not read the table again if you change the local variable table at a later time. In such a case, in order to be able to use the symbolic name as a local variable, you must manually insert a hash character # in your program in front of the name: #Undefined_Local_Var.

**How to enter the first local variable**

To make an assignment in a local variable table, follow the procedure below:

1. Ensure that the correct program organization unit (POU) is displayed in the Program Editor window by double-clicking, if necessary, on the symbol of the POU in the project tree. (Since every POU has its own local variable table, you need to make sure that you are making assignments in the correct variable table.)
2. If the local variable table is hidden, drag down the splitter cursor to display it. (See: How to hide/show the local variable table.)
3. If you are defining an assignment for OB1 or an interrupt program, the local variable table contains only variables of the type TEMP. If you are defining an assignment for a subprogram, the local variable table contains IN, IN_OUT, OUT, and TEMP variables.
   Choose a row that has the right declaration type for the kind of variable that you want to define and type a name for the variable in the "Name" field. You do not need to preface the name with a hash character # in the local variable table. Hash characters # are only used to precede local variables in the program code.

   **Note**

   - Local names are permitted to contain a maximum of 23 alphanumeric characters and underscores. They are also permitted to contain extended characters (ASCII 128 to ASCII 255). The first character is restricted to alpha and extended characters only. It is illegal to use keywords (Page 281) as symbolic names, or to use names that begin with a number or contain characters that are not alphanumeric or in the extended character set.
   - Local variable table variable names are downloaded to and stored in the CPU memory. The use of longer variable names may reduce the memory available to store your program.

4. Click the mouse pointer in the "Data type" field and use the list box to select an appropriate data type for the local variable.

   **Note**

   When you assign local variables as parameters for subprograms, you must ensure that the data type that you assign to the local variable does not conflict with the operand being used in the subprogram call. (See: Data type check.)

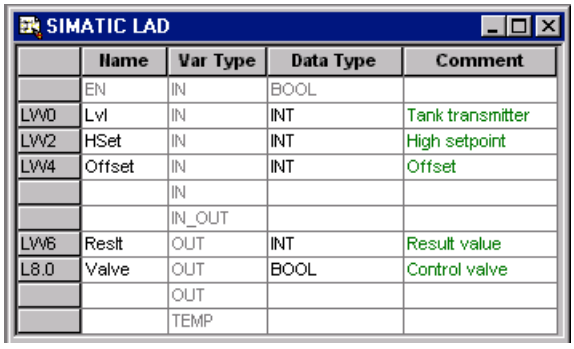
After you supply a value for the "Name" and "Data type" fields, the program editor automatically assigns an address in the local data memory to the local variable.

**How to enter additional local variables in the local variable table**

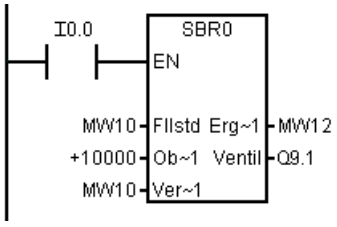
For OB1 and interrupt programs, the local variable table displays a set of rows predefined for TEMP variables. This is the only declaration type that you can use in OB1 and interrupt programs. To add more rows to the table, simply click a field in the last row and use the ENTER key to move through that row and down - a new row is automatically generated.

For subprograms, the local variable tables display a set of rows that are predefined with declaration types in this sequence: IN, IN_OUT, OUT, and TEMP. You cannot change this sequence. Your local variables must follow the same order in the table that the corresponding operands will use when you make operand assignments to the call instruction for the subprogram. If you want to add additional local variables, you must right-click an existing row and use the shortcut menu to insert another new variable of the same declaration type. Choose **Insert > Row** to insert the new row above the selected row, or **Insert > Below Row** to insert the new row below the selected row.

### Local variable table example




| | Name | Var Type | Data Type | Comment |
|---|---|---|---|---|
| | EN | IN | BOOL | |
| LW0 | Lvl | IN | INT | Tank transmitter |
| LW2 | HSet | IN | INT | High setpoint |
| LW4 | Offset | IN | INT | Offset |
| | | IN | | |
| | | IN_OUT | | |
| LW6 | Reslt | OUT | INT | Result value |
| L8.0 | Valve | OUT | BOOL | Control valve |
| | | OUT | | |
| | | TEMP | | |




### See also:

Subprogram (Page 223)
Data types (Page 209)
Keywords (Page 281)

## 6.1.20 Save data in variable memory with the help of a data block

If you wish to work with data blocks, then select a CPU (Page 469) that supports data blocks.

### CPUs that support data blocks

DB

828D
828D Step 2
808D-PPU14x (only special data blocks)
808D-PPU15x (as of PLC 07.03)
808D-PPU16x (as of PLC 07.03)

### Procedure

Use one of the following methods to access the data block:

- Click the button of the data block  in the "Navigation" toolbar (Page 500).
- Select the menu command **View > Data Block**.
- Click the button of the data block  in the project tree (Page 447).

These help topics are explained in the following:

### Data block properties

The data block structures are, just like the POUs, part of the project. They are compiled, saved, imported or exported with the project. They are loaded into/from the PLC with the project. The data blocks themselves only contain actual values. Actual values may be loaded into/from the PLC independently of the project. It is therefore essential that the structure of the data block to be loaded in the PLC is exactly the same as that of the project opened in the Programming Tool. Data blocks are listed like POUs in their own symbol table.

When changing the CPU, existing data blocks are not lost. However, if you select a CPU in which data blocks are not available, then please note that the variables from the data block are now displayed with their absolute address (e.g. DB9000.DBB0 or VB90000000). Whether this V range exists is first checked when the PLC is brought into the RUN state. In this case, the program cannot be executed.

To rename your data blocks and change their properties (Page 452), right-click the corresponding data block in the operation tree. Select "Properties". The "Data Block Properties" dialog box opens. Here, you can change the name, block number, and data class (Page 243), as well as add an author and comments. You can also activate the property "Non-retain" for data blocks. This ensures that every time the PLC is brought into the RUN state, the data block in the PLC is preset with the initial values. Data blocks can be write-protected, that is, the actual values of the data block cannot be changed with the Programming Tool. This property cannot be changed.

**Note**

If you just want to rename your data block, right-click the object you want to rename in the operation tree and select "Rename".

You can also easily edit your data block in another program (e.g. Microsoft Excel) (see Cut, Copy and Paste section in the data block editor).

### Assigning addresses and initial values in the data block

When you create a variable, you assign its name and data type. The default initial value is set to zero/OFF, however, this can also be changed. You can optionally insert comments. The Programming Tool automatically assigns the address. Each address is aligned according to the size of its data type, which means that gaps can occur. The actual values are edited in the data view. However, you may assign the initial values to all actual values.

The maximum size of a data block is limited (512 bytes), if it is exceeded then the Programming Tool marks the excess variable addresses using a red wavy line. The Programming Tool returns an error when compiling the project.

### Assigning actual values

Actual values are saved, exported or imported in your project file as part of your data block. You can only assign individual actual values in the data view. The structure of the variable (name, data type, initial value, and comment) is write-protected in the data view. You can only change the actual value and therefore specifically set initial values, for example. After loading the data block (only actual values), these values become effective in the PLC.

The structure of the data block in the project must match the structure of the data block in the PLC. If you have modified the structure of the data block, then the complete project has to be loaded again.

You can access the data view in the following ways:

- Click the Display data block values button in the toolbar.
- Select the menu command **View > View Data Block Values**.

### Accepting initial values

If you accept your initial values, then all of the actual values of your data block are overwritten. If you have not selected an initial value for a variable, then the Programming Tool sets it to zero/ OFF. Other data blocks will not be changed.

In order to reset all actual values in the actual data block to the initial values,

- Right-click in the table > Accept Initial Values.
- Select the menu command **Edit > Accept Initial Values**.

### Displaying and changing actual values

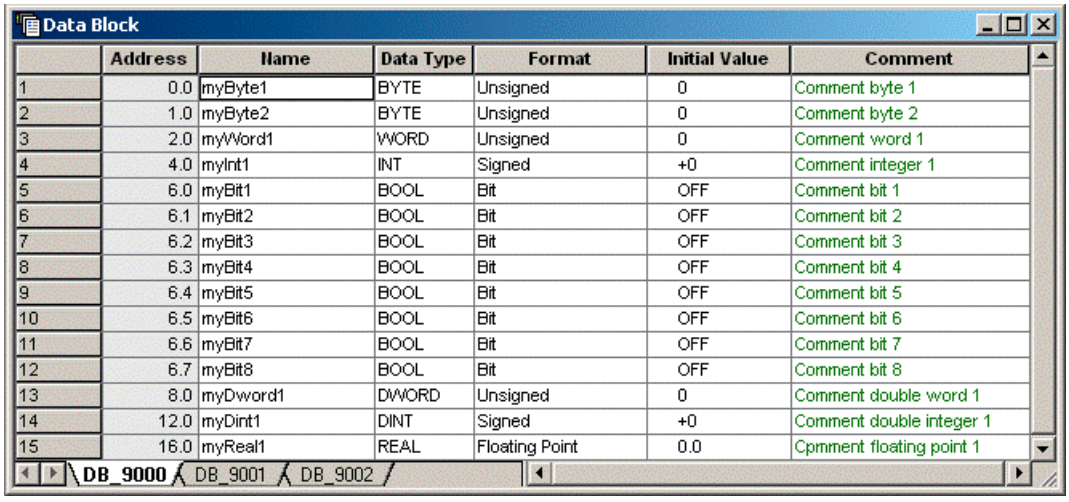
You can view the actual values of a data block in the PLC with the data block status (Page 485) and, if the data block is not write-protected, change them too. The structure of the data block in the project must match the structure of the data block in the PLC. If you have modified the structure of the data block, then the complete project has to be loaded again. When the data block status is activated, the structure of the variable (name, data type, initial value, and comment) is write-protected. If you switch on the data block status, the values in the column "Actual Value" are regularly updated with the actual values of the PLC.

After you have entered values in the column "New Value", write the required changes to the PLC using the command "Write All" .

You can activate the data block status in the following ways:

- Click the "Data block status" button in the toolbar.
- Select the menu command **Debug > Data Block Status**.

## Example for a data block



Data Block

| | Address | Name | Data Type | Format | Initial Value | Comment |
|---|---|---|---|---|---|---|
| 1 | 0.0 | myByte1 | BYTE | Unsigned | 0 | Comment byte 1 |
| 2 | 1.0 | myByte2 | BYTE | Unsigned | 0 | Comment byte 2 |
| 3 | 2.0 | myWord1 | WORD | Unsigned | 0 | Comment word 1 |
| 4 | 4.0 | myInt1 | INT | Signed | +0 | Comment integer 1 |
| 5 | 6.0 | myBit1 | BOOL | Bit | OFF | Comment bit 1 |
| 6 | 6.1 | myBit2 | BOOL | Bit | OFF | Comment bit 2 |
| 7 | 6.2 | myBit3 | BOOL | Bit | OFF | Comment bit 3 |
| 8 | 6.3 | myBit4 | BOOL | Bit | OFF | Comment bit 4 |
| 9 | 6.4 | myBit5 | BOOL | Bit | OFF | Comment bit 5 |
| 10 | 6.5 | myBit6 | BOOL | Bit | OFF | Comment bit 6 |
| 11 | 6.6 | myBit7 | BOOL | Bit | OFF | Comment bit 7 |
| 12 | 6.7 | myBit8 | BOOL | Bit | OFF | Comment bit 8 |
| 13 | 8.0 | myDword1 | DWORD | Unsigned | 0 | Comment double word 1 |
| 14 | 12.0 | myDint1 | DINT | Signed | +0 | Comment double integer 1 |
| 15 | 16.0 | myReal1 | REAL | Floating Point | 0.0 | Cpmment floating point 1 |

DB_9000 DB_9001 DB_9002

## Addressing

### Direct addressing

- **Absolute address input**
  Input in absolute format, the address of a variable in your data block comprises the data block number (e.g. DB9000), a period, and the address of the variable. However, you can also directly access the variable via a V address (e.g. V9000000.0). The display of the absolute address depends on how the address display is set and this can be selected in the menu **View > DB Address View (Ctrl + B)**.

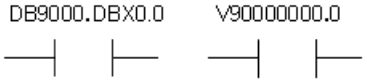



Figure 6-3 Absolute addressing using the data block number

- **Symbolic address input**
  In the symbol table, you can assign an additional name to the variables from your data block. If you wish to use your variable, then you no longer have to specify the complete address - a name in the symbol table is sufficient.
  If you have not listed your variable from the data block in your symbol address, then the symbolic address comprises the name of the corresponding data block (e.g. DB_9000) and the name of the variable in the data block (e.g. Ax1).

| Address | Name of the variables in the data block | Entry in the symbol table | Absolute representation | Symbolic representation |
|---|---|---|---|---|
| VB90000000 | VarName | FeedCorrAx1 | VB90000000 | FeedCorrAx1 |
| or | | | | |
| DB9000.DBB0 | VarName | FeedCorrAx1 | DB9000.DBB0 | FeedCorrAx1 |
| VB90000000 | VarName | | VB90000000 | NameDB.VarName |
| or | | | | |
| DB9000.DBB0 | VarName | | DB9000.DBB0 | NameDB.VarName |

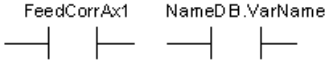



Figure 6-4 Symbolic address representation

The switch between the absolute and symbolic representation is carried out using the menu command **View > Symbolic Addressing (Ctrl + Y)**.

### Indirect addressing

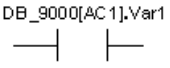
You may be able to simplify the programming if data blocks with the same structure are used in your program. You can indirectly address your data blocks using the accumulators (AC0 to AC3). The accumulators are used to inform the program which data block is to be handled. The value in the AC is then treated as an index.

For example, for axis DBs, the program text can be somewhat reduced by not having to write a dedicated program for each axis, but instead accessing the appropriate axis via the various data blocks and the index (AC). As this value is treated as an index, it starts at 0 for the first axis. Indirect addressing is only possible using ACs. The address cannot be broken down, as

the contents of the accumulator are not known offline. This also means that the absolute address cannot be determined. V addresses cannot be used for indirect addressing.

**Note**

Accumulators are used by both subprograms and the calling program. Calling a subprogram does not cause the accumulators to be saved or restored.




| Absolute input | Symbolic input |
|---|---|
| DB3800[AC1].DBX2.1 | ToAxis[AC1].ControlEnable |
| DB9000[AC0].DBW0 | Prototyp1[AC0].MyWord1 |

**Note**

**Not permitted:**

DB3800[1].DBX2.1 ToAxis[5].ControlEnable

**Indirect addressing using V addresses is not permitted:**

V3800[5]0002.1

V380[5]0002.1

V3800[AC0]0002.1

V38[AC0]0002.1

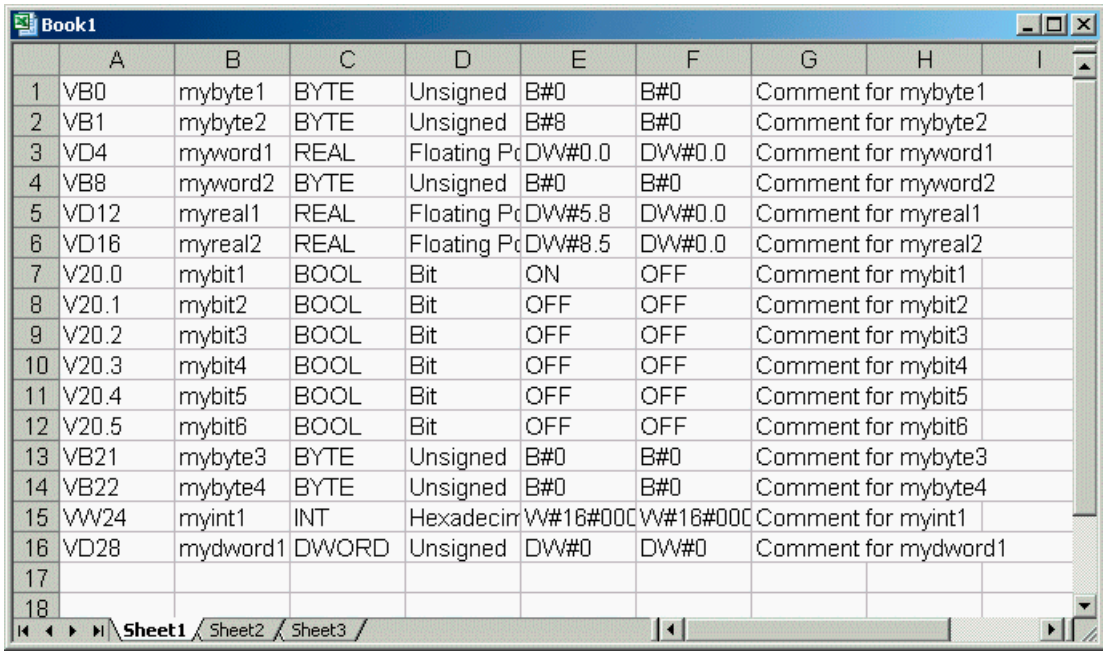
## Using Cut, Copy and Paste in the data block editor

You can save data (rows, columns, and cells) of your data block to the Microsoft clipboard in order to edit it in a different program. You can do this in one of the following ways:

- By right-clicking Cut/Copy/Paste
- By pressing the shortcut keys Ctrl+X/Ctrl+C/Ctrl+V
- By using the menu commands **Edit > Cut**, **Edit > Copy, Edit > Paste**

| | |
|---|---|
| Cut | Selected data is copied to the clipboard and is deleted |
| Copy | Selected data is copied to the clipboard and is not deleted |
| Paste | If data is in the clipboard, the data is pasted |
| Delete | Selected data block/section is deleted, it is not saved in the clipboard |

You can easily edit your data block in this way, e.g. in Microsoft Excel.



Book1

| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 1 | VB0 | mybyte1 | BYTE | Unsigned | B#0 | B#0 | Comment for mybyte1 | | |
| 2 | VB1 | mybyte2 | BYTE | Unsigned | B#8 | B#0 | Comment for mybyte2 | | |
| 3 | VD4 | myword1 | REAL | Floating Pc | DW#0.0 | DW#0.0 | Comment for myword1 | | |
| 4 | VB8 | myword2 | BYTE | Unsigned | B#0 | B#0 | Comment for myword2 | | |
| 5 | VD12 | myreal1 | REAL | Floating Pc | DW#5.8 | DW#0.0 | Comment for myreal1 | | |
| 6 | VD16 | myreal2 | REAL | Floating Pc | DW#8.5 | DW#0.0 | Comment for myreal2 | | |
| 7 | V20.0 | mybit1 | BOOL | Bit | ON | OFF | Comment for mybit1 | | |
| 8 | V20.1 | mybit2 | BOOL | Bit | OFF | OFF | Comment for mybit2 | | |
| 9 | V20.2 | mybit3 | BOOL | Bit | OFF | OFF | Comment for mybit3 | | |
| 10 | V20.3 | mybit4 | BOOL | Bit | OFF | OFF | Comment for mybit4 | | |
| 11 | V20.4 | mybit5 | BOOL | Bit | OFF | OFF | Comment for mybit5 | | |
| 12 | V20.5 | mybit6 | BOOL | Bit | OFF | OFF | Comment for mybit6 | | |
| 13 | VB21 | mybyte3 | BYTE | Unsigned | B#0 | B#0 | Comment for mybyte3 | | |
| 14 | VB22 | mybyte4 | BYTE | Unsigned | B#0 | B#0 | Comment for mybyte4 | | |
| 15 | VW24 | myint1 | INT | Hexadecim | W#16#000 | W#16#000 | Comment for myint1 | | |
| 16 | VD28 | mydword1 | DWORD | Unsigned | DW#0 | DW#0 | Comment for mydword1 | | |
| 17 | | | | | | | | | |
| 18 | | | | | | | | | |

Sheet1 Sheet2 Sheet3

## Using special data blocks

The Programming Tool offers you the possibility to use pre-configured data blocks for special applications. These special data blocks can be found in the operation tree under **Libraries > Special Data Blocks**. These data blocks have a fixed structure. You can make use of them by double-clicking or via copy and paste (in the operation tree).

## Resolving errors

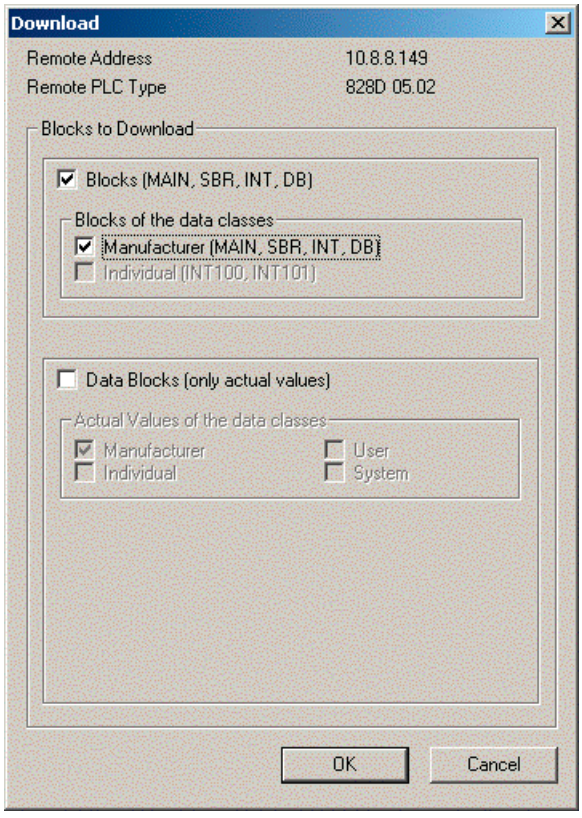
The Programming Tool marks errors made during data input as is the case in the LAD editor or the symbol table. For example, the use of invalid syntax or invalid values can cause an error. You must correct all of the errors that occur before you can compile (Page 459) without any errors and load the program into the PLC.

If any errors occur during compilation of the data block, then they are displayed in the output window. Position the cursor on an error message in the output window and double-click it so that you see the row in the data block with the error.

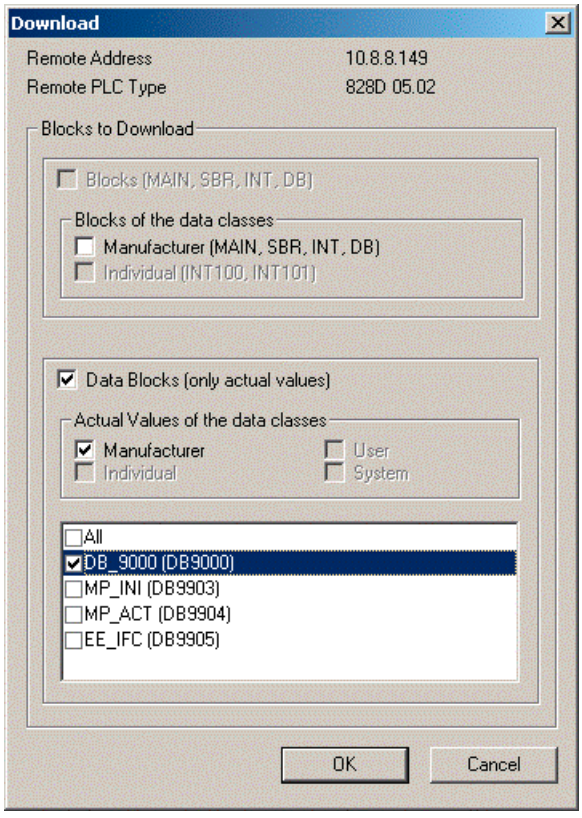
## Loading the data block

### Loading the data block into the PLC

If you edit the structure of a data block, you need to subsequently load the program block (project) into the PLC (Page 382). Your changes will only become effective after the modified project has been loaded into the PLC. Actual values and comments are not loaded into the PLC with the project. If the PLC identifies that a data block is new or has been changed, then it sets the initial values for this data block as the first actual values.

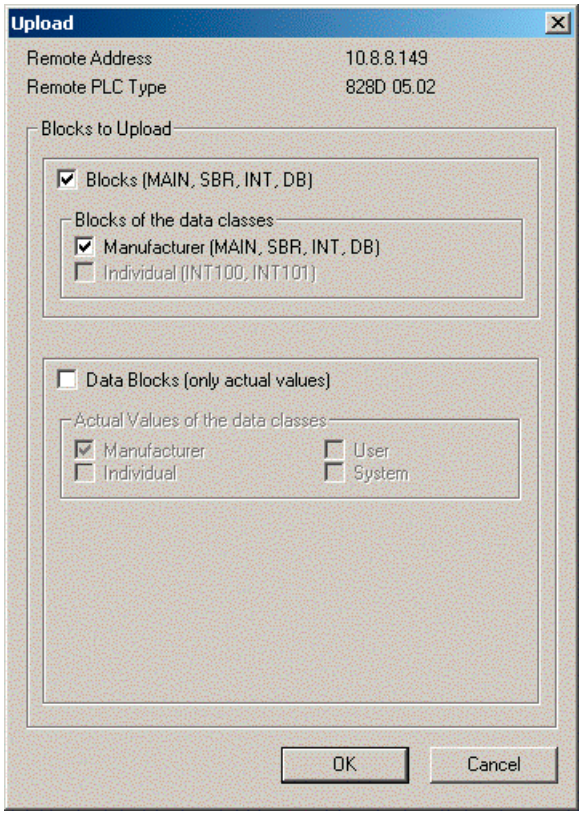



If you edit the actual values of a data block, you need to subsequently load the data block (actual values only) into the PLC (Page 382) so that these actual values become valid. The structure of the data block in the PLC must match the structure of the data block in the project.

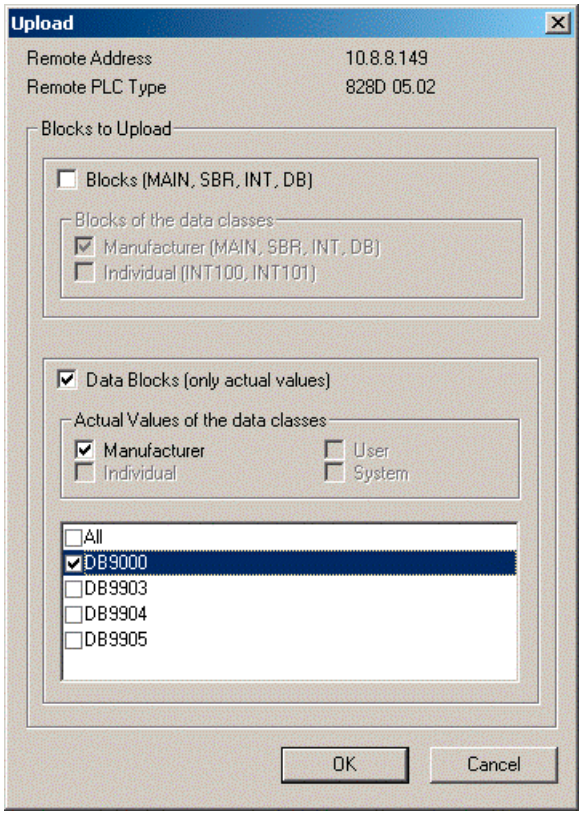



**Loading a data block from the PLC**

You must first open a project in the Programming Tool before you can load the program block (project) and therefore the structure of the data blocks contained in it from the PLC (Page 382).




If you only want to load the actual values of a data block from the PLC, then first open the corresponding project in the Programming Tool or load it from the PLC. Now you can load the data block (only actual values) from the PLC (Page 382). You cannot load the data block if the structure of the data block in the PLC does not match that of the data block of the project that has been opened (or if the project that has been opened does not have a data block).




### See also

Display the data block status (Page 74)

Constants (Page 271)

Select a target CPU (model and version) (Page 334)

Communication (Page 443)

Getting started (Page 17)

Information on using the software (Page 375)

## 6.1.21 Data classes

### Supporting CPUs

828D

828D Step 2

### Assignment of blocks into data classes

Assigning blocks in data classes is used to efficiently upgrade or commission a Sinumerik system.

A distinction is made between four data classes:

| | |
|---|---|
| • System (S) | is allocated by Siemens AG. |
| • Manufacturer (M) | is allocated by the machine manufacturer during construction (CPU-type specifications) |
| • Individual (I) | is allocated by the machine manufacturer during first commissioning (values for this specific machine) |
| • User (U) | is set by the end user (values associated with the particular process) |

#### Assigning a block to a data class

You assign a new data class to a data block in its Properties dialog (Page 452). In the project tree, right-click the corresponding data block and select "Properties". You cannot assign a different data class to your program blocks.

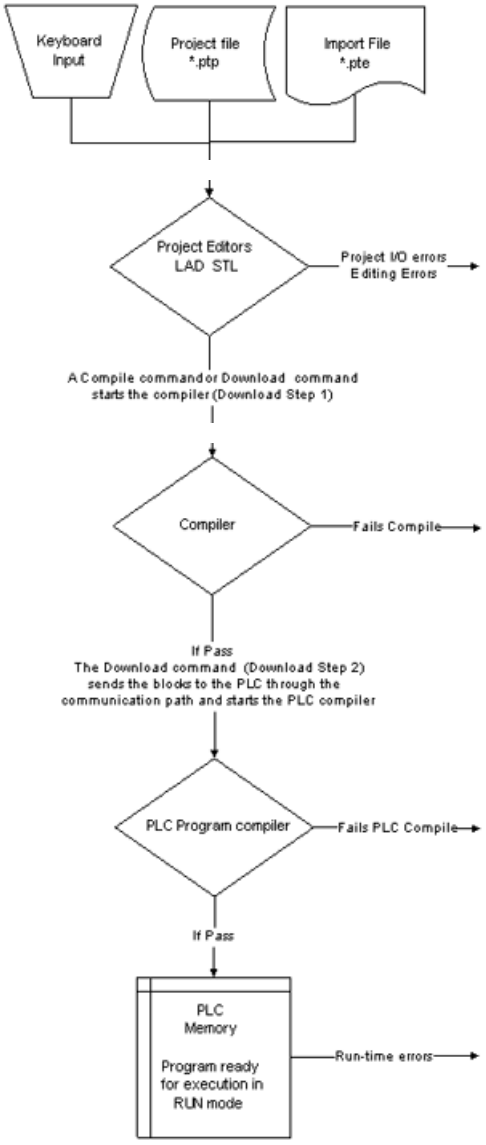
In their Properties dialog, these data classes can be assigned to blocks:

| Block | Default setting | Can be changed |
|---|---|---|
| Main program | Manufacturer | No |
| Subprogram | Manufacturer | No |
| Interrupt program INT0<br>Interrupt programs INT100, INT101 | Manufacturer<br>Individual | No<br>No |
| Data block | Manufacturer | User<br>Individual |

## 6.1.22 Compilation of projects

Use one of the following methods to start the Programming Tool Project Compiler:

- Click the Compile button  or select the menu command **PLC > Compile** to compile all project components (program block, data block).




Opening projects (Page 379)

Range checking (Page 209)

Selecting the CPU type (Page 469)

All compilation errors in the Programming Tool are listed in the Output Window (Page 451). Double-click the error and the editor scrolls to the error location. Use the menu command PLC > Type (Page 469) to set a specific model and version of the CPU.

There are errors that indicate that the project is exceeding the space available on the target system.

For example, "ERROR 820: The compiled program block or the entire project (including comments, data blocks, etc.) is too large for the current target system."

In this case, all data counts towards the overall size of the project, including program blocks, data blocks, tables, comments, etc.

To enable successful compilation of the project, gradually delete non-essential data until the storage requirements are met again.

**See also:**

Information on using the software (Page 375)

First steps: Contents (Page 17)

## 6.1.23 Display of cross references of the memory used

### Call

Use one of the following methods to view the Cross Reference window:

- Select the menu command **View > Cross Reference**.
- Click the button for cross-references  in the "Navigation" toolbar (Page 500).

To access the Cross Reference table, the Byte Usage table or the Bit Usage table, click the appropriate tab at the bottom of the "Cross Reference" window:


These help topics are explained in the following:

### Cross Reference table

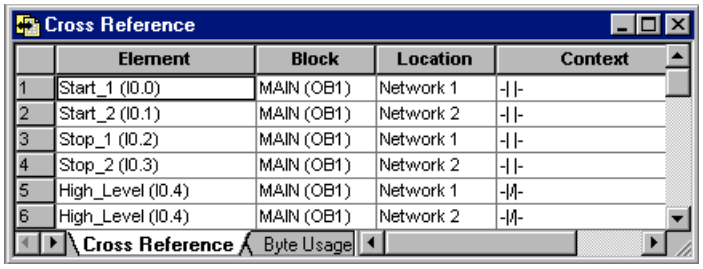
**Note**

You must compile your program in order to view the Cross Reference table.

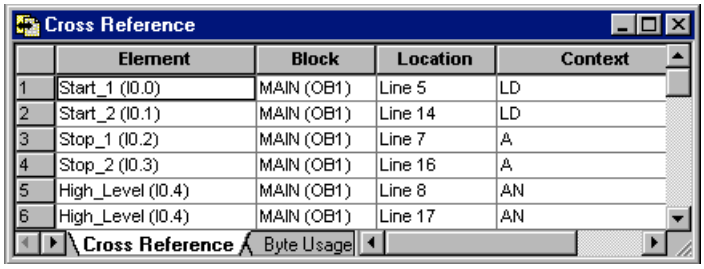
Use the Cross Reference table when you want to know whether a symbolic name or an address is already in use in your program, and where it is used. The Cross Reference table lists all operands used in the program and lists all occurrences of the operands with POU, network or line, as well as the instruction.

| | |
|---|---|
| **Element** | Refers to the operands used in your program. You can toggle between symbolic and absolute view to change the representation of all operands (select the menu command **View > Symbolic Addressing**). |
| **Block** | Refers to the POU where the operand is used. |
| **Address** | Refers to the line or network where the operand is used. |
| **Context** | Refers to the program instruction where the operand is used. |

*Example of an LAD cross reference list*




*Example of an STL cross reference list*




## Byte Usage table

**Note**

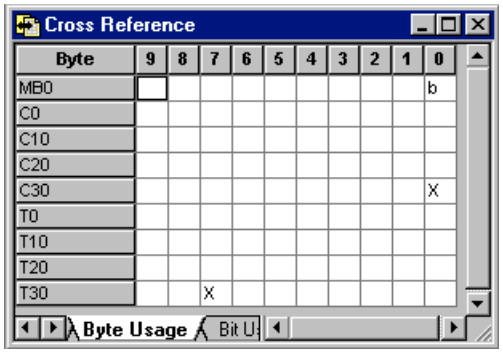
You must compile your program in order to view the Byte Usage table.

The Byte Usage table allows you to see which bytes, from which memory areas, have been used in your program. It also helps you recognize duplicate assignment errors.

| | |
|---|---|
| **b** | Indicates that a bit of memory has been assigned. |
| **B** | Indicates that a byte of memory has been assigned. |
| **W** | Indicates that a word (16 bits) of memory has been assigned. |
| **D** | Indicates that a double word (32 bits) of memory has been assigned. |
| **X** | Used for timers and counters. |

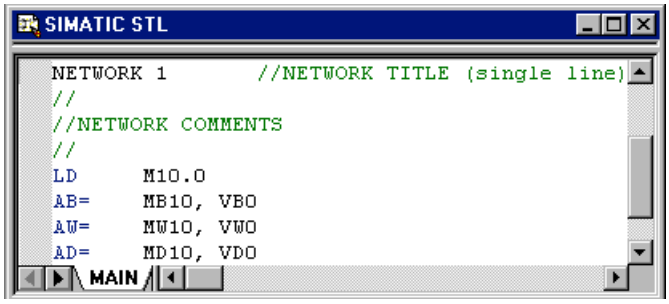
## Interpreting the Byte Usage table

This example of a Byte Usage table shows that its associated program makes use of the following addresses: one bit within MB0; counter C30; timer T37.

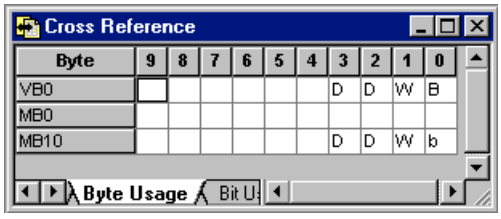


### Recognizing duplicate assignment errors

This example program contains overlapping address assignments at MB10.0.




The Byte Usage table can be examined to identify improper assignments. Since a double word requires four bytes, there should be four adjacent Ds in the row for VB140000000. Likewise, since a word requires two bytes, there should be two adjacent Ws for VB14000000. The row for MB10 reflects the same problems, plus the fact that MB10.0 was used in more than one assignment.




### Bit Usage table

**Note**

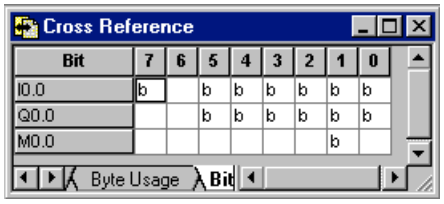
You must compile your program in order to view the Bit Usage table.

The Bit Usage table allows you to see which memory addresses, down to the bit level, have been used in your program. It also helps you recognize duplicate assignment errors.

| | |
|---|---|
| **b** | Indicates that a bit of memory has been assigned. |
| **B** | Indicates that a byte of memory has been assigned. |
| **W** | Indicates that a word (16 bits) of memory has been assigned. |
| **D** | Indicates that a double word (32 bits) of memory has been assigned. |
| **X** | Used for timers and counters. |

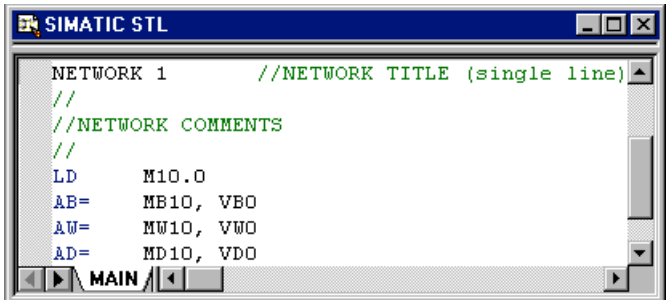
## Interpreting the Bit Usage table

This example of a Bit Usage table shows that its associated program makes use of the following addresses: from byte I0, bits 0, 1, 2, 3, 4, 5, and 7; from byte Q0, bits 0, 1, 2, 3, 4, and 5; from byte M0, bit 1.

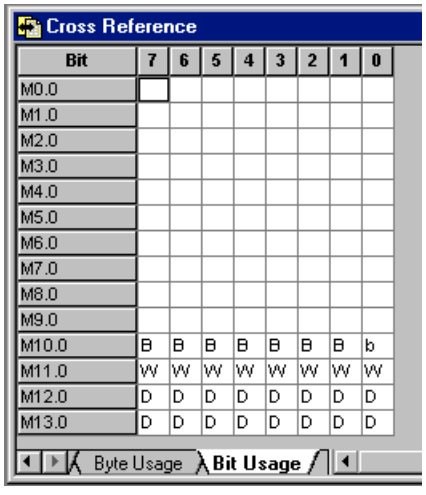



## Recognizing duplicate assignment errors

This example program contains overlapping address assignments at MB10.0.




The Bit Usage table can be examined to identify improper assignments. In a properly assigned program, there would never be a bit value in the midst of the byte. BBBBBBBb is invalid, whereas BBBBBBBB would be valid. The same is true of the word assignments (there should be 16 adjacent Ws) and the double word assignments (there should be 32 adjacent Ds).



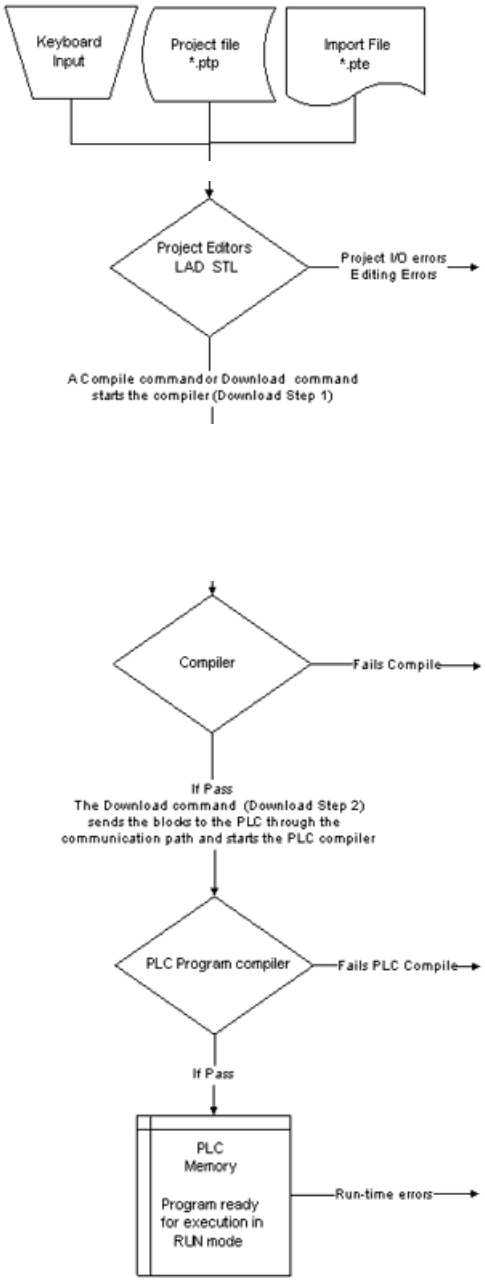
**See also:**

Communication configuration (Page 443)
Selecting the CPU type (Page 469)
Changing between symbolic and absolute addressing mode (Page 444)
First steps: Contents (Page 17)
Information on using the software (Page 375)

## 6.1.24 Downloading a program to the PLC

Use one of the following methods to download project components from the Programming Tool to the PLC:

- Click the  button.
- Select the menu command **File > Download**.
- Press the **Ctrl+D** shortcut key.




Opening projects (Page 379)

Range checking (Page 209)

Selecting the CPU type (Page 469)

All compilation errors in the Programming Tool are listed in the Output Window (Page 451). Double-click the error and the editor scrolls to the error location. Use the menu command PLC > CPU Type (Page 469)to set a specific model and version of the CPU.

Communication error

To download (editor to PLC) or upload (PLC to editor), PLC communication must be operating properly. Make sure your network hardware and PLC connecting cable are working.

The PLC Compiler verifies that the PLC hardware supports all program instructions, ranges, and the structure.

Use the menu command PLC > Information (Page 461) to view the first compilation error found. Use the menu command PLC > CPU Type (Page 469)to set a specific model and version of the CPU. This moves as many errors as possible to the Output Window where they are easier to locate and fix.

**Note**

**Saving retentive data**

The user-specific retentive data must be selected to be able to save them in the user project.

You can find information on this in the Chapter "Saving retentive data" (Page 345).

**See also:**

Communications overview (GS 4.1) (Page 50)

Testing the communications network (GS 4.2) (Page 51)

Downloading a program to the CPU (GS 4.3) (Page 52)

Uploading from the PLC (File > Upload from CPU) (Page 382)

Uploading from a PLC (GS 6.3) (Page 78)

Correcting compilation and loading errors (GS 4.4) (Page 55)

Information on using the software (Page 375)

First steps: Contents (Page 17)

## 6.1.25 Monitoring the status during runtime

### Diagnostics functions

Once you have established communication between your programming device, on which the PLC Programming Tool is installed, and a PLC and have loaded a program into the PLC, you can work with the diagnostic functions of the PLC Programming Tool and test new programs as well as monitor programs that are already being processed.

These help topics are explained in the following:

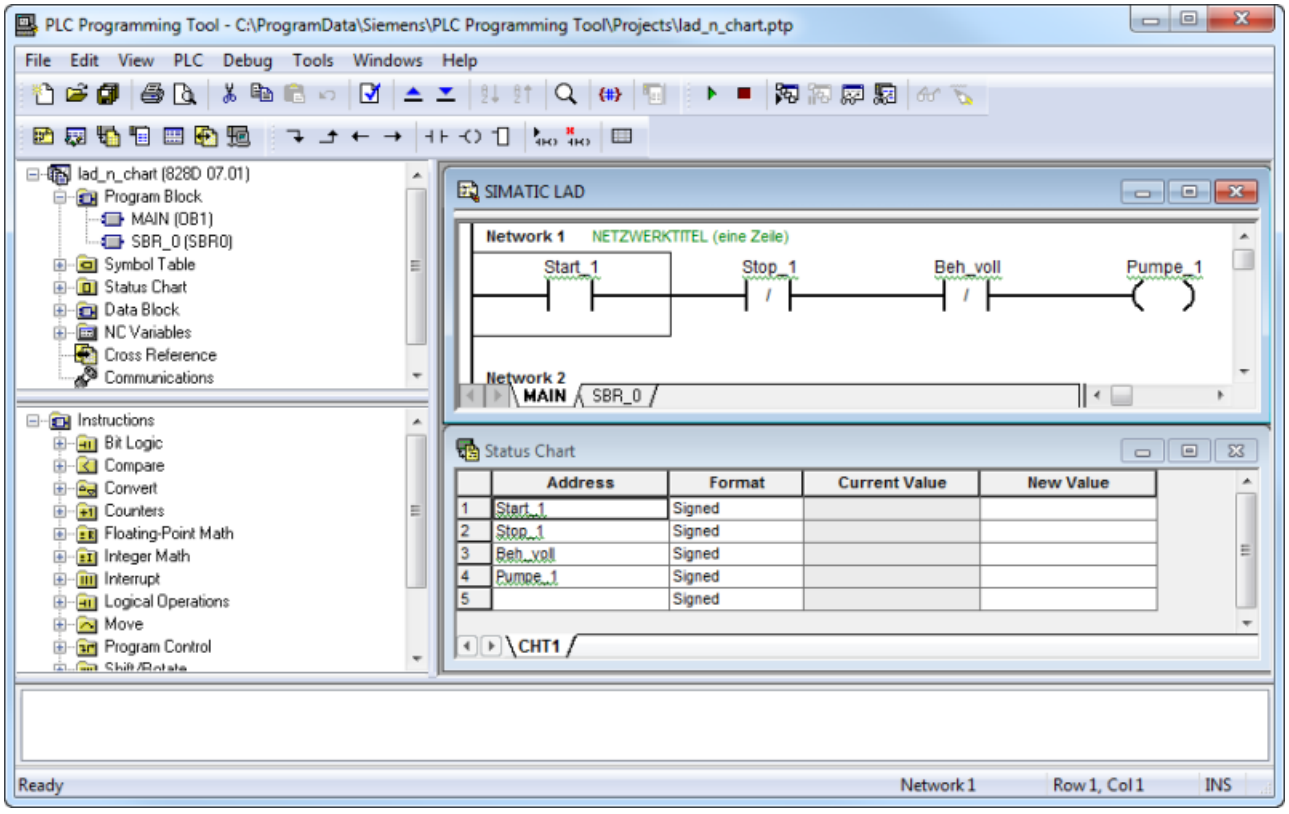
### What is "status"?

"Status" refers to the display of the actual values of operands while executing the program in the PLC. You can display status information in a status chart, with the data block status or by enabling the program status in the program editor.

In the status chart in each case the values are recorded at the end of a scan cycle (end-of-scan status). It is not possible to display the status of the local data memory and the accumulators.
With the data block status, the actual values are also recorded at the end of a scan cycle in each case.

If the execution status is activated for the program status on the "LAD Status" tab (Tools > Options) (Page 492), the status values are recorded at the time of execution of the instructions. The states of the local data memory and the accumulators are displayed. The status values are updated and displayed only if the PLC is in the RUN mode.

An example for status information in the status chart and in the program editor of the PLC Programming Tool is shown in the following diagram:




**Note**

Please note that unnecessary project components (e.g. the project tree, the instruction tree, and the output window) have been omitted, in order to provide more space to display the required components (LAD program editor, status chart, function bar to test and symbol for the status chart in the navigation bar). Using the menu "View", you can set-up the environment in the PLC Programming Tool corresponding to your particular tasks, i.e. you can display just the project components that you require.

## Supporting CPUs

The execution status is not available for all CPUs. The following table shows which CPUs are available:

| | |
|---|---|
| 828D (as of PLC 05.04) | 808D-PPU14x |
| 828D Step 2 | 808D-PPU15x |
| | 808D-PPU16x |

### Execution status for unsupported CPUs

If the execution status is activated for a CPU that does not support this status, the end-of-scan status is automatically executed.

If the execution status is not activated, the end-of-scan status is executed. For this, the values are recorded at the end of a scan cycle in each case. It is not possible to display the status of the local data memory and the accumulators.

### Preconditions of the status update

Before you can update the status to monitor and test your program, you must execute the following tasks:

- Your program must be able to be compiled error-free.
- You must have set up communication between the PLC Programming Tool and the PLC.
- Your program must have been loaded error-free into the PLC.

  **Note**

  The time stamps of your project in the Programming Tool and in the PLC must match, so that you can enable the status. The comparison of the time stamps (carried out automatically if you attempt to enable the status) ensures that the representation of the project status in the Programming Tool exactly reflects the status of the program in the PLC. If the time stamps do not match, you can compare the programs via the dialog box "Time Stamps Do Not Match". Look at the drop-down list box in order to find out which networks are different.

- After you have loaded the program into the PLC, then you should again bring this into the RUN mode. Otherwise, the address status is displayed, however, the PLC cannot execute the program, so you are not shown the logic operations that you expect.

### Operating state of the PLC

The type of monitoring and test functions that you can execute depends on the mode of your PLC.

Even if your program is not executed in the STOP mode, the operating system of the PLC still monitors the PLC (status of RAM and I/O) and transfers the data status to the Programming Tool. If the PLC is in the STOP mode, then you can execute the following functions:

- You can display the actual values of the operands as the end-of-scan status in the chart status, the data block status or the program status. (This is the same as the function "Single Read", as the program is not executed.)
- In the chart status and the data block status, you can write values.
- You can execute a certain number of scan cycles and display the effect in a status chart, in the data block status and/or in the program status.

If the PLC is in the RUN mode, you cannot execute the functions "First Scan" or "Multiple Scans". You can write values in a status chart, you can also execute the following functions (not in the STOP mode):

- In the chart status, you can carry out continuous updates. (If you wish to only execute one update, you must disable the chart status so that you can execute the command "Single Read".)
- In the data block status, you can carry out continuous updates.
- You can execute continuous updates in the program status.

## Communication and scan cycle

In a continuous scan cycle, the PLC reads the inputs, it executes the program, writes to the outputs, and executes system functions as well as communication. This scan cycle runs with an extremely high speed of many times per second. Even if the PLC Programming Tool issues status requests in a fast sequence, you cannot monitor each individual event that takes place in the PLC. When using the program status or the chart status, if you read data values from a PLC program, the data is scanned by taking samples (spot check). The update rate of the status values read from the PLC depends on the communication baud rate. Communicate with maximum baud rate, if you want to achieve the highest update rate.

## Different ways to update the status

### Chart status

Click the "Chart Status" button or select the menu command **Debug > Chart Status** to view continuous updates of the status when the PLC is in the RUN mode. Disable the chart status and use the function "Single Read" if you need to update the status just once and you do not want to switch the PLC into the STOP mode. If you switch the PLC into the STOP mode and enable the chart status, you receive a status update too. Furthermore, you can use the functions "Multiple Scans" and "First Scan" while you are viewing a status chart. In the chart status, the end-of-scan status is always used.

### Data Block Status

Click the "Data Block Status" button or select the menu command **Debug > Data block status** to view continuous updates of the current values of a data block when the PLC is in the RUN mode. If you switch the PLC into the STOP mode and enable the data block status, you receive a current value update too. Furthermore, you can use the functions "Multiple Scans" and "First Scan" while you are viewing a data block status. With the data block status, the actual values are always recorded at the end of a scan cycle.

**Program status**

Click the "Program Status" button or select the menu command **Debug > Program Status** to display the status of the data in the PLC in the program editor. The status data is recorded in the mode previously selected on the "LAD Status" tab (Tools > Options) (Page 492):

- Execution status (execution status activated)
  In the execution status in LAD, the value of operands and the signal flow for execution of the individual instructions during the execution of the program in the scan cycle are displayed. The execution status shows the values of the operands at the time of execution of the instruction; this may be overwritten again by subsequent program instructions. All of the displayed data values of the PLC are recorded in a single program scan cycle. The states of the local data memory and the accumulators are displayed. The status values are only updated and displayed if the PLC is in the RUN mode.

  **Note**

  The execution status may only be run as the program status once for each CPU at any one time. If you attempt to run the execution status in a different application, such as in a second Programming Tool, which is connected to the same PLC, an error will be displayed.

- End-of-scan status (execution status deactivated or CPU does not support execution status)
  The end-of-scan status shows the status results that are read at the end of the program scan cycle. These results may not indicate all changes to the values of the addresses of data in the PLC, because subsequent program instructions write a value and overwrite it again before the end of the program scan cycle is reached. The end-of-scan status shows the data values that are scanned at the end of multiple scan cycles. This results from the different speeds between the fast scan cycle of the PLC and the relatively slow transmission of the PC status data. The states of the local data memory and the accumulators are not displayed. If you switch the PLC into the STOP mode, you receive a status update too.

You can receive the status values of the program status continuously or as a snapshot.

- **Continuous:**
  Open the program editor window and enable the program status in order to view continuous updates of the program execution status when the PLC is in the RUN mode.

  **Note**

  "Continuous" should not be understood as "real time"; it means that the programming device scans the PLC for status information continuously and displays it on your screen, updating the display as quickly as your communication setup permits. Some rapidly fluctuating values may not be recorded and displayed on your screen. It is also possible that the values change too quickly for you to read them.

- **Snapshot:**
  If the PLC is in the STOP mode, you can use the menu command **Debug > Multiple Scans...** to display the status values of one or more scan cycles, or, with the menu command **Debug > First Scan**, to display the status values of the first scan cycle.

With the button "Pause Program Status" or the menu command Debug > Pause Program Status you can pause the status display of the program status. This can be necessary if values change too quickly for you to read them. In the execution status this can also make a single execution of a program section, e.g. a subprogram, visible. To do this, set the networks which are not being processed in the program editor and pause the status display with the

execution status activated. The next time the networks are executed in the program editor, the status display is updated. The status update then stops.

### Simulating process conditions

You can simulate process conditions by writing new values to operands: To do so, use the status chart.

### Checking cross references and the elements used

If you test your program, you may wish to supplement, delete or change the parameters.

In the window "Cross Reference", you can determine how the parameters are presently assigned in your program. This helps you to avoid assigning values twice.

### Loading changes in your program into the CPU

If you test a program with the program status and thus determine that you want to change the program, you must disable the program status before you can change the program. Once you have loaded the changed program into the CPU, you can enable the program status to check whether the changes are correct.

### See also

Displaying the status in the editors (GS 5.2) (Page 60)

Displaying the status in the status chart (GS 5.3) (Page 68)

Executing a certain number of scan cycles (GS 5.4) (Page 73)

Displaying the data block status (GS 5.5) (Page 74)

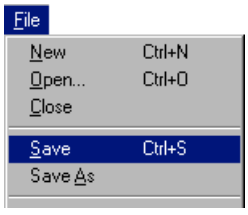
Downloading a program to the CPU (GS 4.3) (Page 52)

Cross-references and elements used (Page 439) (Ensure that program changes do not produce duplicate assignment.)

RUN/STOP modes of the PLC (Page 457)

## 6.1.26 Saving your work

You can save your work by using the Save button on the toolbar or the **Save** and **Save as** options from the File menu.



- "Save" allows you to quickly save any changes to your work. (The first time you save a project, however, you are prompted to confirm or modify the default choices of name and directory for your current project.)
- "Save as" allows you to modify the name and/or directory location of your current project.

By default, the Programming Tool supplies the name "Project1.ptp" for your project when you first create it. You can accept or modify this name. If you accept it, the default name for the next project is automatically incremented to "Project2.ptp."

The default directory location for the Programming Tool projects is a folder called "Projects". This is located in the directory in which you have installed the Programming Tool. You do not have to accept the default location.

**Note**

**Naming projects**

You can add a version identification to project names.

The maximum length of the file name without file extension is 46 characters.

For additional information, see the following: Version identification (Page 82)

## 6.2 Reading the statement list

### 6.2.1 Bookmarks in the STL program

You can set bookmarks in an STL program to make it easy to move back and forth between designated (bookmarked) lines of a long program:

| | |
|---|---|
|  | **Toggle Bookmark**: Sets or removes an STL bookmark at the program line designated by the current cursor location. |
|  | **Next Bookmark**: If you have set more than one bookmark, you can move down to the next bookmarked line of your program by clicking this button. |
|  | **Previous Bookmark**: If you have set more than one bookmark, you can move down to the previous bookmarked line of your program by clicking this button. |
|  | **Clear All Bookmarks**: Click this button to remove all the current bookmarks in your STL program. |

```
//NETWORK COMMENTS
//
LD      I0.0
=       Q0.0
```

**See also:**

Information on using the software (Page 375)

First steps: Contents (Page 17)

# Instruction sets in LAD and STL

# 7

## 7.1 ENO usage

ENO is a boolean output from LAD boxes. ENO allows you to connect boxes in series (horizontal connection) rather than in parallel (vertical connection). If a signal flow is present at a box at input EN and the box is being executed without any error, then the ENO output transfers the signal flow to the next element. If an error has occurred when the box is being executed, then the signal flow is ended at the box that caused the error.
The ENO output has the same status as the input (EN=ENO) – unless the instruction was not executed error-free.

**Example:**

The BCD_I instruction supplies ENO=0 for an invalid BCD range.

## 7.2 Supported instructions

### Instructions for 808 and 802 series CPUs

| | 808D PPU14x | 802Dsl TM plus | 802Dsl TM pro | 802Dsl TM value | 802Dsl CU pro | 802Dsl NG plus | 802Dsl NG pro | SINUMERIK 802D | SINUMERIK 802SC |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |
| BCD_I | + | + | + | + | + | + | + | - | - |
| I_BCD | + | + | + | + | + | + | + | - | - |
| | | | | | | | | | |
| -\|<>B\|- | + | + | + | + | + | + | + | - | - |
| -\|>B\|- | + | + | + | + | + | + | + | - | - |
| -\|<B\|- | + | + | + | + | + | + | + | - | - |
| -\|<>I\|- | + | + | + | + | + | + | + | - | - |
| -\|>I\|- | + | + | + | + | + | + | + | - | - |
| -\|<I\|- | + | + | + | + | + | + | + | - | - |
| -\|<>D\|- | + | + | + | + | + | + | + | - | - |
| -\|>D\|- | + | + | + | + | + | + | + | - | - |
| -\|<D\|- | + | + | + | + | + | + | + | - | - |
| -\|<>R\|- | + | + | + | + | + | + | + | - | - |
| -\|>R\|- | + | + | + | + | + | + | + | - | - |
| -\|>=R\|- | + | + | + | + | + | + | + | - | - |
| | | | | | | | | | |
| -(I) | - | - | - | - | - | - | - | - | - |
| -\|/I\|- | - | - | - | - | - | - | - | - | - |
| -(SI) | - | - | - | - | - | - | - | - | - |
| -(RI) | - | - | - | - | - | - | - | - | - |
| -(RETI) | - | - | - | - | - | - | - | - | - |
| | | | | | | | | | |
| MOV_BIW | - | - | - | - | - | - | - | - | - |
| MOV_BIR | - | - | - | - | - | - | - | - | - |

### Instructions for 828 series CPUs

828 series CPUs support all instructions.

## 7.3 Mnemonics

| Description | Mnemonics |
|---|---|
| Ladder logic | LAD |
| Statement list | STL |
| Input | I |
| Output | Q |
| Variable memory | V |
| Data block | DB |
| Internal bit memory | M |
| Timer | T |
| Counters | C |
| Special bit memories | SM |
| Accumulator | AC |
| Local data memory | L |

## 7.4 LAD instructions

You can find information on the LAD instructions at the following link. LAD instructions (Page 111)

# Use of the PLC memory

# 8

## 8.1 Memory types and properties

.

| Area | Description | Access in bit format | Access in byte format | Access in word format | Access in double word format | Can be retentive |
|---|---|---|---|---|---|---|
| I | Digital inputs and process image input | Read/write | Read/write | Read/write | Read/write | No |
| Q | Digital outputs and process image output | Read/write | Read/write | Read/write | Read/write | No |
| M | Internal bit memory | Read/write | Read/write | Read/write | Read/write | No |
| SM | Special bit memory (SM0.0 - SM0.6 are write-protected) | Read/write | Read/write | Read/write | Read/write | No |
| V<br>DB | Variable memory<br>Data block | Read/write | Read/write | Read/write | Read/write | Yes |
| T | Actual values of timers and timer bits | Timer bit<br>Read/write | | Act. value of timer<br>Read/write | No | No |
| C | Actual values of the counters and counter bits | Counter bit<br>Read/write | No | Act. value of timer<br>Read/write | | No |
| AC | Accumulators | No | Read/write | Read/write | Read/write | No |
| L | Local data memory | Read/write | Read/write | Read/write | Read/write | No |

## 8.2 Addressing

**Direct addressing**

In direct addressing, you specify the memory area and the address. Example: V14000010 refers to the address 14000010 in the variable memory.

You can access different memory areas in the CPU (V, I, Q, M, and SM) as bytes, words or double words. To specify whether an address is to be accessed as a byte, word or double word, use the size designator following the area identifier. For example, to specify that address 14000000 is to be accessed as a byte, enter VB14000000. To access it as a double word, enter VD14000000.

To access a bit in a memory area, enter the area identifier, the byte address, and the number of the bit. Separate the byte from the bit using a decimal point. Your notation would follow this format: VB14000000.7. This address accesses the last bit (bit 7) in byte 14000000.

**See also:**

Addressing overview (GS 2.2) (Page 26)

Symbol table (Page 413)

DB address representation (Page 446)

## 8.3 Bit access

To access a bit, specify the address of the bit, which consists of an area identifier, the byte, and the number of the bit. Zero is the first address for all data areas. The byte number is followed by a decimal point to distinguish the byte number from the bit number. The bit number is a decimal number from 0 through 7.

Example: I0.0

Memory areas of the automation system (Page 262)

## 8.4 Byte, word, and double word access

To access a byte, word or double word, specify the address, which consists of an area identifier, a letter signifying data size, and the address number.

**Example:**

VB14000000 accesses byte 14000000 in the variable memory.
VW14000000 accesses bytes 14000000 and 14000001 in the variable memory.
VD14000000 accesses bytes 14000000, 14000001, 14000002 and 14000003 in the variable memory.

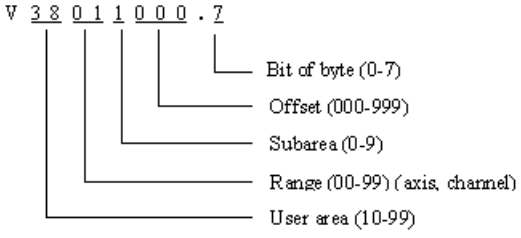
The individual memory areas are described in the help topic: Memory areas of the automation system (Page 262)

## 8.5 Memory areas of the CPU

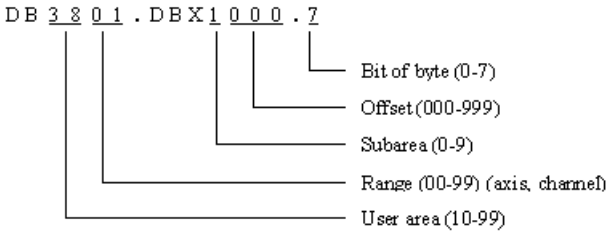
If you execute instructions which communicate with the PLC, then the Programming Tool recognizes your CPU model. When you create your program, you must ensure that you only use inputs and outputs and memory areas which are valid for the CPU in which the program is to be loaded.

If you attempt to load a program that accesses inputs and outputs or memory areas that do not lie in the range for your CPU, you will receive an error message.

**Address formation in the variable memory**



| Access | Address | Explanation |
|---|---|---|
| Bit | V38011000.7 | Bit 7 of the byte with offset 0 in subrange 1 for axis 2 in user range 38 |
| Byte | VB38010000 | Byte with offset 0 in subrange 0 for axis 2 in user range 38 |
| Word | VW45000002 | Word with offset 2 in subrange 0 in range 0 in user range 45 |
| Double word | VD25003004 | Double word with offset 4 in subrange 3 in range 0 in user range 25 |




| Access | Address | Explanation |
|---|---|---|
| Bit | DB3801.DBX1000.7 | Bit 7 of the byte with offset 0 in subrange 1 for axis 2 in user range 38 |
| Byte | DB3801.DBB0 | Byte with offset 0 in subrange 0 for axis 2 in user range 38 |
| Word | DB4500.DBW2 | Word with offset 2 in subrange 0 in range 0 in user range 45 |
| Double word | DB2500.DBD3004 | Double word with offset 4 in subrange 3 in range 0 in user range 25 |

**Note**

The permitted offset for an address is dependent on the access:

- Bit or byte access: any offset permitted,
  byte-size variables are placed one beside another seamlessly in a DB.
- Word access: offset must be divisible by 2,
  word-size variables (2 bytes) are always saved on even offsets.
- Double word access: offset must be divisible by 4,
  double word-size variables (4 bytes) are always saved on offsets which are divisible by 4.

**Address ranges in the variable memory**

**Note**

The available address ranges in the variable memory are described in the PLC user interface. The user interface is an interface comprising data blocks that the firmware creates on the PLC. It is used to exchange data between the PLC on one side and the NCK and HMI on the other side.

These DBs neither have to be loaded from the CPU nor into the CPU because they are created by the firmware and therefore belong to the system.

User data blocks are exclusively created by the user. If you intend to indirectly access blocks having the same structure, these should have subsequent numbers.

The structure of special data blocks is predefined by the system and stored in the PLC Programming Tool in "Libraries". Whether they are integrated in the UP, assigned to a data class, and loaded into the CPU is the sole responsibility of the user and their concept. For example when working without tool management, the relevant DBs need not be integrated.

| Access: | Type of memory | 828D Step 2 | 828D |
|---|---|---|---|
| Bit (byte.bit) | V | 10000000.0 - 79999999.7 [1]<br>90000000.0 - 90630511.7 [2]<br>90640000.0 - 91270511.7 (as of ab PLC 08.04) [2]<br>99000000.0 - 99120503.7 [3] | 10000000.0 - 79999999.7 [1]<br>90000000.0 - 90630511.7 [2]<br>99000000.0 - 99100503.7 [3] |
| | I | 0.0 - 255.7 [4]<br>256.0 - 256.3 [5]<br>512.0 - 1023.7 [4] | 0.0 - 255.7 [4]<br>256.0 - 256.3 [5] |
| | Q | 0.0 - 255.7 [4]<br>256.0 - 256.3 [5]<br>512.0 - 1023.7 [4] | 0.0 - 255.7 [4]<br>256.0 - 256.3 [5] |
| | M | 0.0 - 511.7 | 0.0 - 511.7 |
| | SM | 0.0 - 0.6 | 0.0 - 0.6 |
| | T | 0 - 15 (100 ms)<br>16 - 127 (10 ms)<br>128 - 255 (10 ms, as of PLC 07.01)<br>256 - 511 (10 ms, as of PLC 08.04) | 0 - 15 (100 ms)<br>16 – 127 (10 ms) |
| | C | 0 - 63 | 0 - 63 |
| | L | 0.0 - 59.7 | 0.0 - 59.7 |

| Access: | Type of memory | 828D Step 2 | 828D |
|---|---|---|---|
| Byte | VB | 10000000 - 79999999 [1]<br>90000000 - 90630511 [2]<br>90640000 - 91270511 (as of PLC 08.04) [2]<br>99000000 - 99120503 [3] | 10000000 - 79999999 [1]<br>90000000 - 90630511 [2]<br>99000000 - 99100503 [3] |
| | IB | 0 - 255 [4]<br>512 - 1023 [4] | 0 - 255 [4] |
| | QB | 0 - 255 [4]<br>512 - 1023 [4] | 0 - 255 [4] |
| | MB | 0 - 511 | 0 - 511 |
| | SMB | 0 | 0 |
| | LB | 0 - 59 | 0 - 59 |
| | AC | 0 - 3 | 0 - 3 |
| Word | VW | 10000000 - 79999998 [1]<br>90000000 - 90630510 [2]<br>90640000 - 91270510 (as of PLC 08.04) [2]<br>99000000 - 99120502 [3] | 10000000 - 79999998 [1]<br>90000000 - 90630510 [2]<br>99000000 - 99100502 [3] |
| | IW | 0 - 254 [4]<br>512 - 1022 [4] | 0 - 254 [4] |
| | QW | 0 - 254 [4]<br>512 - 1022 [4] | 0 - 254 [4] |
| | MW | 0 - 510 | 0 - 510 |
| | T | 0 - 15 (100 ms)<br>16 - 127 (10 ms)<br>128 - 255 (10 ms, as of PLC 07.01)<br>256 - 511 (10 ms, as of PLC 08.04) | 0 - 15 (100 ms)<br>16 – 127 (10 ms) |
| | C | 0 - 63 | 0 - 63 |
| | LW | 0 - 58 | 0 - 58 |
| | AC | 0 - 3 | 0 - 3 |
| Double word | VD | 10000000 - 79999994 [1]<br>90000000 - 90630508 [2]<br>90640000 - 91270508 (as of PLC 08.04) [2]<br>99000000 - 99120500 [3] | 10000000 - 79999994 [1]<br>90000000 - 90630508 [2]<br>99000000 - 99100500 [3] |
| | ID | 0 - 252 [4]<br>512 - 1020 [4] | 0 - 252 [4] |
| | QD | 0 - 252 [4]<br>512 - 1020 [4] | 0 - 252 [4] |
| | MD | 0 - 508 | 0 - 508 |
| | LD | 0 - 56 | 0 - 56 |
| | AC | 0 - 3 | 0 - 3 |

1) PLC user interface: The available address ranges are described in the PLC user interface.

2) User data blocks: The available address ranges are dependent on which data blocks are present in the project.

3) Special data blocks: The available address ranges are dependent on which data blocks are present in the project.

4) Input or output image: The addresses which can actually be reached depend on the machine configuration.

5) Direct digital onboard inputs or outputs: The available address ranges are described in the Commissioning Manual.

| Access: | Type of memory | 808D PPU-16x | 808D PPU-15x | 808D PPU-14x |
|---|---|---|---|---|
| Bit (byte.bit) | V | 10000000.0 - 79999999.7 [1]<br>90000000.0 - 90070511.7 [2]<br>99030000.0 - 99060159.7 [3] | 10000000.0 - 79999999.7 [1]<br>90000000.0 - 90070511.7 [2]<br>99030000.0 - 99060159.7 [3] | 10000000.0 - 79999999.7 [1]<br>99030000.0 - 99060159.7 [2] |
| | I | 0.0 - 8.7 | 0.0 - 8.7 | 0.0 - 8.7 |
| | Q | 0.0 - 5.7 | 0.0 - 5.7 | 0.0 - 5.7 |
| | M | 0.0 - 255.7 | 0.0 - 255.7 | 0.0 - 255.7 |
| | SM | 0.0 - 0.6 | 0.0 - 0.6 | 0.0 - 0.6 |
| | T | 0 - 15 (100 ms)<br>16 - 63 (10 ms) | 0 - 15 (100 ms)<br>16 - 63 (10 ms) | 0 - 15 (100 ms)<br>16 - 63 (10 ms) |
| | C | 0 - 63 | 0 - 63 | 0 - 63 |
| | L | 0.0 - 59.7 | 0.0 - 59.7 | 0.0 - 59.7 |
| Byte | VB | 10000000 - 79999999 [1]<br>90000000 - 90070511 [2]<br>99030000 - 99060159 [3] | 10000000 - 79999999 [1]<br>90000000 - 90070511 [2]<br>99030000 - 99060159 [3] | 10000000 - 79999999 [1]<br>99030000 - 99060159 [2] |
| | IB | 0 - 8 | 0 - 8 | 0 - 8 |
| | QB | 0 - 5 | 0 - 5 | 0 - 5 |
| | MB | 0 - 255 | 0 - 255 | 0 - 255 |
| | SMB | 0 | 0 | 0 |
| | LB | 0 - 59 | 0 - 59 | 0 - 59 |
| | AC | 0 - 3 | 0 - 3 | 0 - 3 |
| Word | VW | 10000000 - 79999998 [1]<br>90000000 - 90070510 [2]<br>99030000 - 99060158 [3] | 10000000 - 79999998 [1]<br>90000000 - 90070510 [2]<br>99030000 - 99060158 [3] | 10000000 - 79999998 [1]<br>99030000 - 99060158 [2] |
| | IW | 0 - 6 | 0 - 6 | 0 - 6 |
| | QW | 0 - 4 | 0 - 4 | 0 - 4 |
| | MW | 0 - 254 | 0 - 254 | 0 - 254 |
| | T | 0 - 15 (100 ms)<br>16 - 63 (10 ms) | 0 - 15 (100 ms)<br>16 - 63 (10 ms) | 0 - 15 (100 ms)<br>16 - 63 (10 ms) |
| | C | 0 - 63 | 0 - 63 | 0 - 63 |
| | LW | 0 - 58 | 0 - 58 | 0 - 58 |
| | AC | 0 - 3 | 0 - 3 | 0 - 3 |
| Double word | VD | 10000000 - 79999994 [1]<br>90000000 - 90070508 [2]<br>99030000 - 99060156 [3] | 10000000 - 79999994 [1]<br>90000000 - 90070508 [2]<br>99030000 - 99060156 [3] | 10000000 - 79999994 [1]<br>99030000 - 99060156 [2] |
| | ID | 0 - 4 | 0 - 4 | 0 - 4 |
| | QD | 0 | 0 | 0 |
| | MD | 0 - 252 | 0 - 252 | 0 - 252 |
| | LD | 0 - 56 | 0 - 56 | 0 - 56 |
| | AC | 0 - 3 | 0 - 3 | 0 - 3 |

1) PLC user interface: The available address ranges are described in the PLC user interface.

2) User data blocks: The available address ranges are dependent on which data blocks are present in the project.

3) Special data blocks: The available address ranges are dependent on which data blocks are present in the project.

| Access: | Type of memory | 802 Dsl TM pro<br>802 Dsl NG pro<br>802 Dsl CU pro | 802 Dsl TM plus<br>802 Dsl NG plus |
|---|---|---|---|
| Bit (byte.bit) | V | 10000000.0 - 79999999.7 [1] | 10000000.0 - 79999999.7 [1] |
| | I | 0.0 - 26.7 | 0.0 - 26.7 |
| | Q | 0.0 - 17.7 | 0.0 - 17.7 |
| | M | 0.0 - 383.7 | 0.0 - 383.7 |
| | SM | 0.0 - 0.6 | 0.0 - 0.6 |
| | T | 0 - 15 (100 ms)<br>16 - 63 (10 ms) | 0 - 15 (100 ms)<br>16 - 39 (10 ms) |
| | C | 0 - 63 | 0 - 31 |
| | L | 0.0 - 59.7 | 0.0 - 59.7 |
| Byte | VB | 10000000 - 79999999 [1] | 10000000 - 79999999 [1] |
| | IB | 0 - 26 | 0 - 26 |
| | QB | 0 - 17 | 0 - 17 |
| | MB | 0 - 383 | 0 - 383 |
| | SMB | 0 | 0 |
| | LB | 0 - 59 | 0 - 59 |
| | AC | 0 - 3 | 0 - 3 |
| Word | VW | 10000000 - 79999998 [1] | 10000000 - 79999998 [1] |
| | IW | 0 - 24 | 0 - 24 |
| | QW | 0 - 16 | 0 - 16 |
| | MW | 0 - 383 | 0 - 383 |
| | T | 0 - 15 (100 ms)<br>16 - 63 (10 ms) | 0 - 15 (100 ms)<br>16 - 63 (10 ms) |
| | C | 0 - 63 | 0 - 31 |
| | LW | 0 - 58 | 0 - 58 |
| | AC | 0 - 3 | 0 - 3 |
| Double word | VD | 10000000 - 79999994 [1] | 10000000 - 79999994 [1] |
| | ID | 0 - 20 | 0 - 20 |
| | QD | 0 -12 | 0 - 12 |
| | MD | 0 - 380 | 0 - 380 |
| | LD | 0 - 56 | 0 - 56 |
| | AC | 0 - 3 | 0 - 3 |

1) PLC user interface: The available address ranges are described in the PLC user interface.

| Access: | Type of memory | 802 Dsl TM value | SINUMERIK 802D | SINUMERIK 802 SC |
|---|---|---|---|---|
| Bit (byte.bit) | V | 10000000.0 - 79999999.7 [1] | 10000000.0 - 79999999.7 [1] | 10000000.0 - 79999999.7 [1] |
| | I | 0.0 - 26.7 | 0.0 - 17.7 | 0.0 - 7.7 |
| | Q | 0.0 - 17.7 | 0.0 - 11.7 | 0.0 - 7.7 |
| | M | 0.0 - 255.7 | 0.0 - 383.7 | 0.0 - 127.7 |
| | SM | 0.0 - 0.6 | 0.0 - 0.6 | 0.0 - 0.6 |
| | T | 0 - 15 (100 ms)<br>16 - 63 (10 ms) | 0 - 15 (100 ms)<br>16 - 39 (10 ms) | 0 - 15 (100 ms) |
| | C | 0 - 31 | 0 - 31 | 0 - 31 |
| | L | 0.0 - 59.7 | 0.0 - 59.7 | 0.0 - 59.7 |
| Byte | VB | 10000000 - 79999999 [1] | 10000000 - 79999999 [1] | 10000000 - 79999999 [1] |
| | IB | 0 - 26 | 0 - 17 | 0 - 7 |
| | QB | 0 - 17 | 0 - 11 | 0 - 7 |
| | MB | 0 - 255 | 0 - 383 | 0 - 127 |
| | SMB | 0 | 0 | 0 |
| | LB | 0 - 59 | 0 - 59 | 0 - 59 |
| | AC | 0 - 3 | 0 - 3 | 0 - 3 |
| Word | VW | 10000000 - 79999998 [1] | 10000000 - 79999998 [1] | 10000000 - 79999998 [1] |
| | IW | 0 - 24 | 0 - 16 | 0 - 6 |
| | QW | 0 - 16 | 0 - 10 | 0 - 6 |
| | MW | 0 - 254 | 0 - 382 | 0 - 126 |
| | T | 0 - 15 (100 ms)<br>16 - 63 (10 ms) | 0 - 15 (100 ms)<br>16 - 63 (10 ms) | 0 - 15 (100 ms)<br>- |
| | C | 0 - 31 | 0 - 31 | 0 - 31 |
| | LW | 0 - 58 | 0 - 58 | 0 - 58 |
| | AC | 0 - 3 | 0 - 3 | 0 - 3 |
| Double word | VD | 10000000 - 79999994 [1] | 10000000 - 79999994 [1] | 10000000 - 79999994 [1] |
| | ID | 0 - 20 | 0 - 14 | 0 - 4 |
| | QD | 0 -12 | 0 - 8 | 0 - 4 |
| | MD | 0 - 252 | 0 - 380 | 0 - 124 |
| | LD | 0 - 56 | 0 - 56 | 0 - 56 |
| | AC | 0 - 3 | 0 - 3 | 0 - 3 |

1) PLC user interface: The available address ranges are described in the PLC user interface.

**See also:**

Addressing overview (GS 2.2) (Page 26)

Memory areas and their properties (Page 261)

Addressing (Page 261)

DB address representation (Page 446)

## 8.6 Data types

There is one circumstance under which you must use data types when programming in the Programming Tool:

If you assign values in the local variable table, then you must specify a data type for every local variable.

By explicitly specifying a data type for a value, you give the Programming Tool clear instructions on how much memory must be assigned for the value (e.g. the value 100 can be stored as BYTE, WORD or DWORD) and how the value is to be represented (e.g. should 0 be interpreted as BOOL or as a numeric value?).

The instructions and parameterized subprograms are recognized using a precise definition. This definition is also called the signature. For all standardized instructions, the data types permissible for the operands of the instruction are specified in the signature. For parameterized subprograms, the signature of the subprogram is generated by the user in the local variable table.

The Programming tool software implements simple data type checking. If a data type is specified for a local or global variable, the software checks that the data type of the operand corresponds to the signature of the instruction.

| Elementary data types | Description | Memory area |
|---|---|---|
| BOOL (bit) | Boolean | 0 to 1 |
| BYTE | Byte, Unsigned | 0 to 255 |
| WORD<br><br>INT (integer) | Integer number (16 bits) without sign<br><br>Integer number (16 bits) with sign | 0 to 65535<br><br>-32768 to +32767 |
| DWORD (Double Word)<br><br>DINT (Double Integer) | Integer number (32 bits) without sign<br><br>Integer number (32 bits) with sign | 0 to 4294967295<br><br>-2147483648 to +2147483647 |
| REAL | 32-bit floating point | +/- 1.175495E-38<br>to +/- 3.402823E+38$^{+38}$ |

| Complex data types | Description | Memory areas | |
|---|---|---|---|
| TON | Switch-on delay | 100 ms | T0-T15 |
| | | 10 ms | From T16 |
| TOF | Switch-off delay | 100 ms | T0-T15 |
| | | 10 ms | From T16 |
| CTU | Up counter | Depending on CPU | |
| CTD | Down counter | Depending on CPU | |
| CTUD | Up-down counter | Depending on CPU | |
| SR | Bistable function block:<br>priority set | N/A | |
| RS | Bistable function block:<br>priority reset | N/A | |

**The Programming Tool has two data type check stages**

1. **Elementary data type check**
   For the elementary data type check, if a symbol or a variable is assigned a data type, then all data types which correspond to the bit size of the user-defined data type are assigned automatically. If, for instance, you specify DINT as the data type, then the local variable is also automatically assigned the DWORD data type, because both data types are 32-bit types. The REAL data type is not automatically assigned, although it is also a 32-bit data type. The REAL data type is defined so that it does not have any equivalent data types: it is always unique. The elementary data type check is performed when using local variables and variables of user data blocks.

| User-defined data types | Equivalent data type |
|---|---|
| BOOL | BOOL |
| BYTE | BYTE |
| WORD | WORD, INT |
| INT | WORD, INT |
| DWORD | DWORD, INT |
| DINT | DWORD, INT |
| REAL | REAL |

2. **No data type check**
   This mode is only available for global variables where no data types can be specified. If a data type check is not active, all data types of the same size are automatically assigned to the symbol. Example: A symbol, which is assigned to address VD14000000, is assigned the following data types automatically by the programming software: DWORD, DINT, and REAL.

**Data types with defined size for global symbols**

| User-defined address | Assigned equivalent data type |
|---|---|
| V14000000.0 | BOOL |
| VB14000000 | BYTE |
| VW14000000 | WORD, INT |
| VD14000000 | DWORD, DINT, REAL |

**Advantages of the data type check**

The benefit of data type checking is to assist the user in avoiding common programming mistakes. If an instruction supports numbers with signs, the Programming Tool flags the use of unsigned numbers in instruction operands. Example: The comparison < I is an operation with sign. -1 is less than 0 for operands with sign. However, if the instruction < I supports unsigned data types, then the programming itself must ensure that the following does not occur: While a program is being executed, an unsigned value of 40,000 is actually smaller than 0 for the instruction < I. If it cannot be guaranteed that the unsigned numbers for signed instructions do not exceed the positive and negative limiting values, then unpredictable events can occur in your program or in the mode of operation of the control.

**Working with instructions to convert the data type**

Conversion instructions convert one data type into another. The Programming Tool supports the following conversion instructions to transfer values between the elementary data types.

| Conversion instructions | Complete data type check, permissible addresses | | Data type check, permissible addresses | |
|---|---|---|---|---|
| BYTE in INT | IN:<br>OUT: | BYTE<br>INT | IN:<br>OUT: | BYTE<br>WORD, INT |
| INT in BYTE | IN:<br>OUT: | INT<br>BYTE | IN:<br>OUT: | WORD, INT<br>BYTE |
| INT in DINT | IN:<br>OUT: | INT<br>DINT | IN:<br>OUT: | WORD, INT<br>DWORD, DINT |
| DINT in INT | IN:<br>OUT: | DINT<br>INT | IN:<br>OUT: | DWORD, DINT<br>WORD, INT |
| DINT in REAL | IN:<br>OUT: | DINT<br>REAL | IN:<br>OUT: | DWORD, DINT<br>REAL |
| REAL in DINT (ROUND) | IN:<br>OUT: | REAL<br>DINT | IN:<br>OUT: | REAL<br>DWORD, DINT |

## 8.7 Constants

**Range of constants**

| | Range without sign | | Range with sign | |
|---|---|---|---|---|
| **Size of the data** | **Decimal:** | **Hexadecimal:** | **Decimal:** | **Hexadecimal:** |
| B (Byte) | 0 - 255 | 0 - FF | -128 - +127 | 80 - 7F |
| W (Word) | 0 - 65535 | 0 - FFFF | -32768 - +32767 | 8000 - 7FFF |
| D (Double word) | 0 - 4294967295 | 0 - FFFFFFFF | -2147483648 - +2147483647 | 80000000 - 7FFFFFFF |
| **Size:** | **Decimal real number (positive)** | | **Decimal real number (negative)** | |
| D (double word) | +1.175495E-38 - +3.402823E+38 | | -1.175495E-38 - -3.402823E+38 | |

The range of ASCII characters as constants is between ASCII 32 and ASCII 255, with the exception of the characters DEL and a single quotation mark. ASCII characters outside this range must use the special $ character, as shown in the ASCII constant examples.

**Format identifier of constants**

In many instructions, your program can use constants in the byte, word or double word format. Format identifiers specify how a constant value is to be displayed (in the binary, decimal, hexadecimal or ASCII format).

Constants in the program are considered as decimal numbers if a format identifier is not specified:

2# for dual numbers
16# for hexadecimal numbers

**Example for binary constants**

| | Numerical basis | Separator | Constant |
|---|---|---|---|
| Example: 2#1101 | 2 | # | 1101 |

**Example for hexadecimal constants**

| | Numerical basis | Separator | Constant |
|---|---|---|---|
| Example: 16#3FB2 | 16 | # | 3FB2 |

## 8.8 Data blocks

If you wish to work with data blocks, then select a CPU (Page 469) that supports data blocks.

### CPUs that support data blocks

DB

828D
828D Step 2
808D-PPU14x (only special data blocks)
808D-PPU15x (as of PLC 07.03)
808D-PPU16x (as of PLC 07.03)

### Procedure

Use one of the following methods to access the data block:

- Click the button of the data block in the "Navigation" toolbar (Page 500).
- Select the menu command **View > Data Block**.
- Click the button of the data block in the project tree (Page 447).

These help topics are explained in the following:

### Data block properties

The data block structures are, just like the POUs, part of the project. They are compiled, saved, imported or exported with the project. They are loaded into/from the PLC with the project. The data blocks themselves only contain actual values. Actual values may be loaded into/from the PLC independently of the project. It is therefore essential that the structure of the data block to be loaded in the PLC is exactly the same as that of the project opened in the Programming Tool. Data blocks are listed like POUs in their own symbol table.

When changing the CPU, existing data blocks are not lost. However, if you select a CPU in which data blocks are not available, then please note that the variables from the data block are now displayed with their absolute address (e.g. DB9000.DBB0 or VB90000000). Whether this V range exists is first checked when the PLC is brought into the RUN state. In this case, the program cannot be executed.

To rename your data blocks and change their properties (Page 452), right-click the corresponding data block in the operation tree. Select "Properties". The "Data Block Properties" dialog box opens. Here, you can change the name, block number, and data class (Page 243), as well as add an author and comments. You can also activate the property "Non-retain" for data blocks. This ensures that every time the PLC is brought into the RUN state, the data block in the PLC is preset with the initial values. Data blocks can be write-protected, that is, the actual values of the data block cannot be changed with the Programming Tool. This property cannot be changed.

**Note**

If you just want to rename your data block, right-click the object you want to rename in the operation tree and select "Rename".

You can also easily edit your data block in another program (e.g. Microsoft Excel) (see Cut, Copy and Paste section in the data block editor).

### Assigning addresses and initial values in the data block

When you create a variable, you assign its name and data type. The default initial value is set to zero/OFF, however, this can also be changed. You can optionally insert comments. The Programming Tool automatically assigns the address. Each address is aligned according to the size of its data type, which means that gaps can occur. The actual values are edited in the data view. However, you may assign the initial values to all actual values.

The maximum size of a data block is limited (512 bytes), if it is exceeded then the Programming Tool marks the excess variable addresses using a red wavy line. The Programming Tool returns an error when compiling the project.

### Assigning actual values

Actual values are saved, exported or imported in your project file as part of your data block. You can only assign individual actual values in the data view. The structure of the variable (name, data type, initial value, and comment) is write-protected in the data view. You can only change the actual value and therefore specifically set initial values, for example. After loading the data block (only actual values), these values become effective in the PLC.

The structure of the data block in the project must match the structure of the data block in the PLC. If you have modified the structure of the data block, then the complete project has to be loaded again.

You can access the data view in the following ways:

- Click the Display data block values  button in the toolbar.
- Select the menu command **View > View Data Block Values**.

### Accepting initial values

If you accept your initial values, then all of the actual values of your data block are overwritten. If you have not selected an initial value for a variable, then the Programming Tool sets it to zero/OFF. Other data blocks will not be changed.

In order to reset all actual values in the actual data block to the initial values,

- Right-click in the table > Accept Initial Values.
- Select the menu command **Edit > Accept Initial Values**.

## Displaying and changing actual values

You can view the actual values of a data block in the PLC with the data block status (Page 369) and, if the data block is not write-protected, change them too. The structure of the data block in the project must match the structure of the data block in the PLC. If you have modified the structure of the data block, then the complete project has to be loaded again. When the data block status is activated, the structure of the variable (name, data type, initial value, and comment) is write-protected. If you switch on the data block status, the values in the column "Actual Value" are regularly updated with the actual values of the PLC.

After you have entered values in the column "New Value", write the required changes to the PLC using the command "Write All"  .

You can activate the data block status in the following ways:

- Click the "Data block status"  button in the toolbar.
- Select the menu command **Debug > Data Block Status**.

## Example for a data block

Data Block

| | Address | Name | Data Type | Format | Initial Value | Comment |
|---|---|---|---|---|---|---|
| 1 | 0.0 | myByte1 | BYTE | Unsigned | 0 | Comment byte 1 |
| 2 | 1.0 | myByte2 | BYTE | Unsigned | 0 | Comment byte 2 |
| 3 | 2.0 | myWord1 | WORD | Unsigned | 0 | Comment word 1 |
| 4 | 4.0 | myInt1 | INT | Signed | +0 | Comment integer 1 |
| 5 | 6.0 | myBit1 | BOOL | Bit | OFF | Comment bit 1 |
| 6 | 6.1 | myBit2 | BOOL | Bit | OFF | Comment bit 2 |
| 7 | 6.2 | myBit3 | BOOL | Bit | OFF | Comment bit 3 |
| 8 | 6.3 | myBit4 | BOOL | Bit | OFF | Comment bit 4 |
| 9 | 6.4 | myBit5 | BOOL | Bit | OFF | Comment bit 5 |
| 10 | 6.5 | myBit6 | BOOL | Bit | OFF | Comment bit 6 |
| 11 | 6.6 | myBit7 | BOOL | Bit | OFF | Comment bit 7 |
| 12 | 6.7 | myBit8 | BOOL | Bit | OFF | Comment bit 8 |
| 13 | 8.0 | myDword1 | DWORD | Unsigned | 0 | Comment double word 1 |
| 14 | 12.0 | myDint1 | DINT | Signed | +0 | Comment double integer 1 |
| 15 | 16.0 | myReal1 | REAL | Floating Point | 0.0 | Cpmment floating point 1 |

DB_9000 DB_9001 DB_9002

## Addressing

### Direct addressing

- **Absolute address input**
  Input in absolute format, the address of a variable in your data block comprises the data block number (e.g. DB9000), a period, and the address of the variable. However, you can also directly access the variable via a V address (e.g. V9000000.0). The display of the absolute address depends on how the address display is set and this can be selected in the menu **View > DB Address View (Ctrl + B)**.

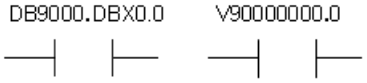



Figure 8-1 Absolute addressing using the data block number

- **Symbolic address input**
  In the symbol table, you can assign an additional name to the variables from your data block. If you wish to use your variable, then you no longer have to specify the complete address - a name in the symbol table is sufficient.
  If you have not listed your variable from the data block in your symbol address, then the symbolic address comprises the name of the corresponding data block (e.g. DB_9000) and the name of the variable in the data block (e.g. Ax1).

| Address | Name of the variables in the data block | Entry in the symbol table | Absolute representation | Symbolic representation |
|---|---|---|---|---|
| VB90000000 | VarName | FeedCorrAx1 | VB90000000 | FeedCorrAx1 |
| or | | | | |
| DB9000.DBB0 | VarName | FeedCorrAx1 | DB9000.DBB0 | FeedCorrAx1 |
| VB90000000 | VarName | | VB90000000 | NameDB.VarName |
| or | | | | |
| DB9000.DBB0 | VarName | | DB9000.DBB0 | NameDB.VarName |

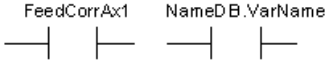



Figure 8-2 Symbolic address representation

The switch between the absolute and symbolic representation is carried out using the menu command **View > Symbolic Addressing (Ctrl + Y)**.

### Indirect addressing

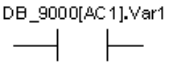
You may be able to simplify the programming if data blocks with the same structure are used in your program. You can indirectly address your data blocks using the accumulators (AC0 to AC3). The accumulators are used to inform the program which data block is to be handled. The value in the AC is then treated as an index.

For example, for axis DBs, the program text can be somewhat reduced by not having to write a dedicated program for each axis, but instead accessing the appropriate axis via the various data blocks and the index (AC). As this value is treated as an index, it starts at 0 for the first axis. Indirect addressing is only possible using ACs. The address cannot be broken down, as

the contents of the accumulator are not known offline. This also means that the absolute address cannot be determined. V addresses cannot be used for indirect addressing.

**Note**

Accumulators are used by both subprograms and the calling program. Calling a subprogram does not cause the accumulators to be saved or restored.




| Absolute input | Symbolic input |
|---|---|
| DB3800[AC1].DBX2.1 | ToAxis[AC1].ControlEnable |
| DB9000[AC0].DBW0 | Prototyp1[AC0].MyWord1 |

**Note**

**Not permitted:**

DB3800[1].DBX2.1 ToAxis[5].ControlEnable

**Indirect addressing using V addresses is not permitted:**

V3800[5]0002.1

V380[5]0002.1

V3800[AC0]0002.1

V38[AC0]0002.1

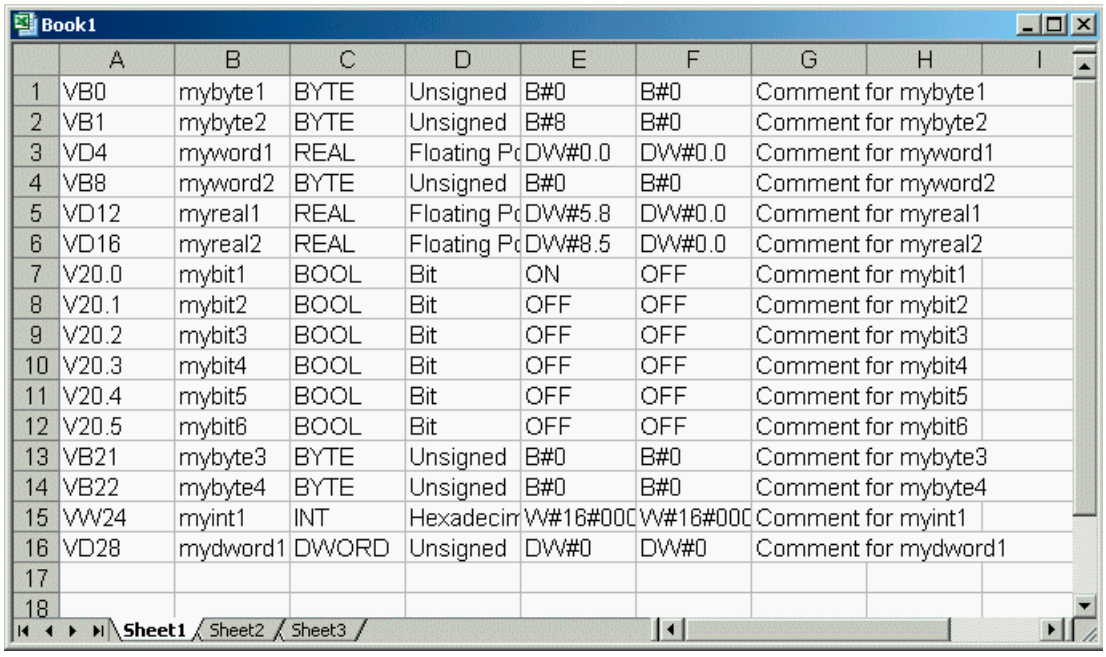
## Using Cut, Copy and Paste in the data block editor

You can save data (rows, columns, and cells) of your data block to the Microsoft clipboard in order to edit it in a different program. You can do this in one of the following ways:

- By right-clicking Cut/Copy/Paste
- By pressing the shortcut keys Ctrl+X/Ctrl+C/Ctrl+V
- By using the menu commands **Edit > Cut**, **Edit > Copy, Edit > Paste**

| | |
|---|---|
| Cut | Selected data is copied to the clipboard and is deleted |
| Copy | Selected data is copied to the clipboard and is not deleted |
| Paste | If data is in the clipboard, the data is pasted |
| Delete | Selected data block/section is deleted, it is not saved in the clipboard |

You can easily edit your data block in this way, e.g. in Microsoft Excel.



Book1

| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 1 | VB0 | mybyte1 | BYTE | Unsigned | B#0 | B#0 | Comment for mybyte1 | | |
| 2 | VB1 | mybyte2 | BYTE | Unsigned | B#8 | B#0 | Comment for mybyte2 | | |
| 3 | VD4 | myword1 | REAL | Floating Pc | DW#0.0 | DW#0.0 | Comment for myword1 | | |
| 4 | VB8 | myword2 | BYTE | Unsigned | B#0 | B#0 | Comment for myword2 | | |
| 5 | VD12 | myreal1 | REAL | Floating Pc | DW#5.8 | DW#0.0 | Comment for myreal1 | | |
| 6 | VD16 | myreal2 | REAL | Floating Pc | DW#8.5 | DW#0.0 | Comment for myreal2 | | |
| 7 | V20.0 | mybit1 | BOOL | Bit | ON | OFF | Comment for mybit1 | | |
| 8 | V20.1 | mybit2 | BOOL | Bit | OFF | OFF | Comment for mybit2 | | |
| 9 | V20.2 | mybit3 | BOOL | Bit | OFF | OFF | Comment for mybit3 | | |
| 10 | V20.3 | mybit4 | BOOL | Bit | OFF | OFF | Comment for mybit4 | | |
| 11 | V20.4 | mybit5 | BOOL | Bit | OFF | OFF | Comment for mybit5 | | |
| 12 | V20.5 | mybit6 | BOOL | Bit | OFF | OFF | Comment for mybit6 | | |
| 13 | VB21 | mybyte3 | BYTE | Unsigned | B#0 | B#0 | Comment for mybyte3 | | |
| 14 | VB22 | mybyte4 | BYTE | Unsigned | B#0 | B#0 | Comment for mybyte4 | | |
| 15 | VW24 | myint1 | INT | Hexadecim | W#16#000 | W#16#000 | Comment for myint1 | | |
| 16 | VD28 | mydword1 | DWORD | Unsigned | DW#0 | DW#0 | Comment for mydword1 | | |
| 17 | | | | | | | | | |
| 18 | | | | | | | | | |

Sheet1 Sheet2 Sheet3

## Using special data blocks

The Programming Tool offers you the possibility to use pre-configured data blocks for special applications. These special data blocks can be found in the operation tree under **Libraries > Special Data Blocks**. These data blocks have a fixed structure. You can make use of them by double-clicking or via copy and paste (in the operation tree).

## Resolving errors

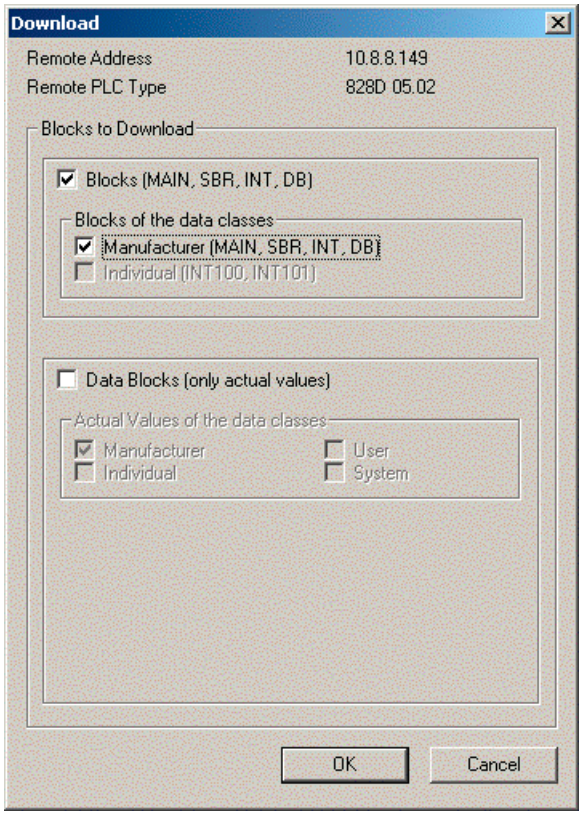
The Programming Tool marks errors made during data input as is the case in the LAD editor or the symbol table. For example, the use of invalid syntax or invalid values can cause an error. You must correct all of the errors that occur before you can compile (Page 244) without any errors and load the program into the PLC.

If any errors occur during compilation of the data block, then they are displayed in the output window. Position the cursor on an error message in the output window and double-click it so that you see the row in the data block with the error.

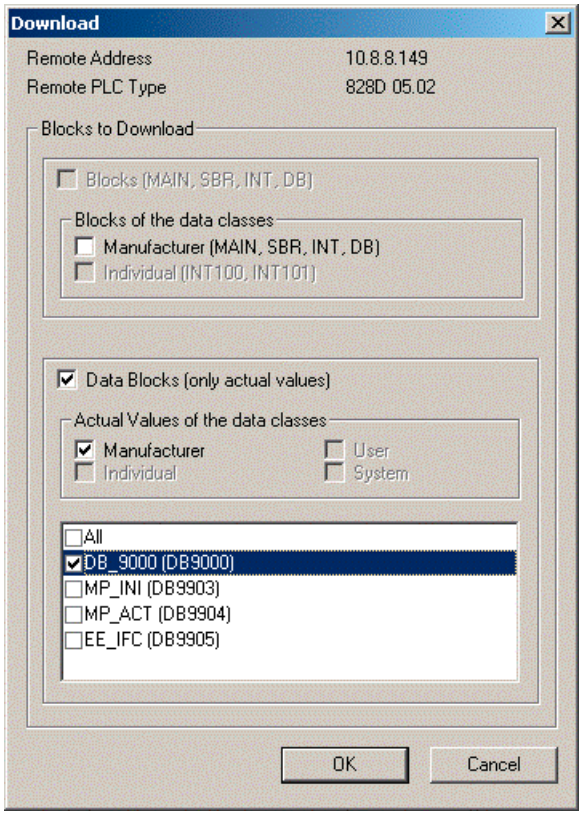
## Loading the data block

### Loading the data block into the PLC

If you edit the structure of a data block, you need to subsequently load the program block (project) into the PLC (Page 250). Your changes will only become effective after the modified project has been loaded into the PLC. Actual values and comments are not loaded into the PLC with the project. If the PLC identifies that a data block is new or has been changed, then it sets the initial values for this data block as the first actual values.

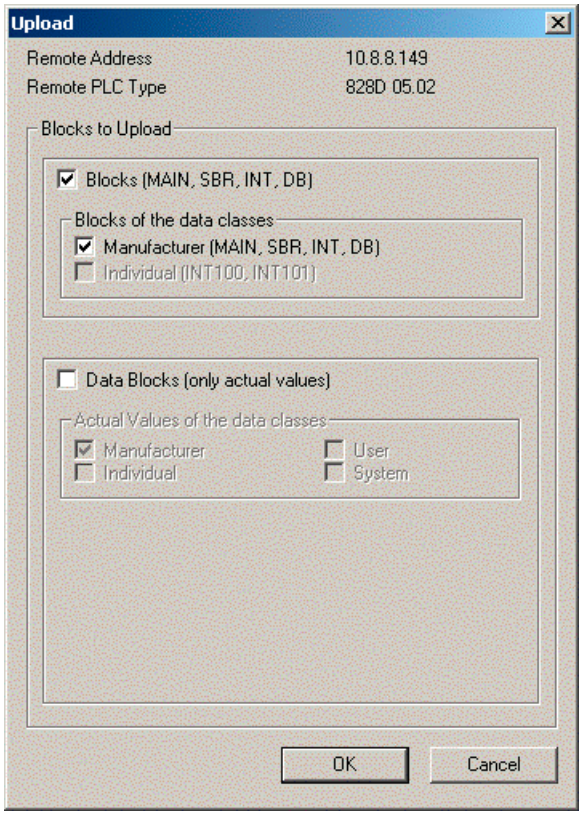



If you edit the actual values of a data block, you need to subsequently load the data block (actual values only) into the PLC (Page 250) so that these actual values become valid. The structure of the data block in the PLC must match the structure of the data block in the project.

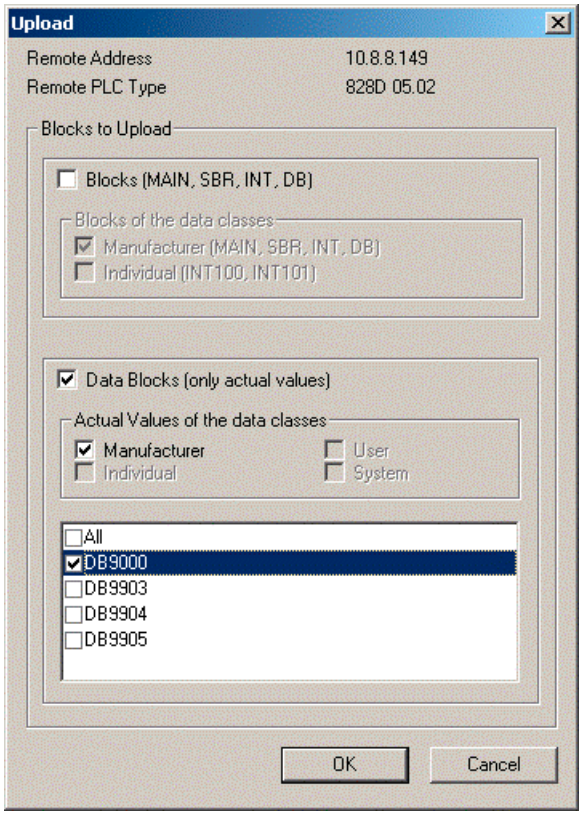



**Loading a data block from the PLC**

You must first open a project in the Programming Tool before you can load the program block (project) and therefore the structure of the data blocks contained in it from the PLC (Page 353).




If you only want to load the actual values of a data block from the PLC, then first open the corresponding project in the Programming Tool or load it from the PLC. Now you can load the data block (only actual values) from the PLC (Page 353). You cannot load the data block if the structure of the data block in the PLC does not match that of the data block of the project that has been opened (or if the project that has been opened does not have a data block).




**See also**

Display the data block status (Page 74)

Constants (Page 271)

Select a target CPU (model and version) (Page 334)

Communication (Page 443)

Getting started (Page 17)

Information on using the software (Page 375)

## 8.9 Keywords

These keywords are reserved. You cannot assign them as symbolic names or POU names.

**Note**

All keywords are listed below in capital letters. However, keywords are not case-sensitive: lower case letters, upper case letters or a mix of both cases is accepted.

| | |
|---|---|
| ANY | Not supported - reserved for future use |
| ARRAY | Not supported - reserved for future use |
| BEGIN | |
| BOOL | |
| BYTE | |
| CHAR | |
| DATE | Not supported - reserved for future use |
| DATE_AND_TIME | Not supported - reserved for future use |
| DINT | |
| DT | Not supported - reserved for future use |
| DWORD | |
| END_FUNCTION | |
| END_FUNCTION_BLOCK | |
| END_ORGANIZATION_BLOCK | |
| END_STRUCT | |
| END_TYPE | |
| END_VAR | |
| FALSE | |
| FUNCTION | |
| FUNCTION_BLOCK | |
| INT | |
| NETWORK | |
| OF | Not supported - reserved for future use |
| OFF | |
| ON | |
| ORGANIZATION_BLOCK | |
| REAL | |
| STRING | Not supported - reserved for future use |
| STRUCT | Not supported - reserved for future use |
| TIME | Not supported - reserved for future use |
| TIME_OF_DAY | Not supported - reserved for future use |
| Title | |
| TOD | Not supported - reserved for future use |
| TRUE | |
| UDT | Not supported - reserved for future use |
| VAR | |
| VAR_IN_OUT | |
| VAR_INPUT | |
| VAR_OUTPUT | |
| WORD | |
| DB | |

| | |
|---|---|
| FC | Not supported - reserved for future use |
| FB | Not supported - reserved for future use |
| SFB | |
| OB | |
| SDB | |
| SBR | |
| SUBROUTINE_BLOCK | |
| END_SUBROUTINE_BLOCK | |
| INTERRUPT_BLOCK | |
| END_INTERRUPT_BLOCK | |

All statement list instruction mnemonics are keywords.

All memory type mnemonics with size indicators are keywords.

# Special bit memories and their functions

# 9

## 9.1 SMB 0 (read-only special bit memory, predefined)

### 9.1.1 SMB0 status bits

Special bit memory SMB0 (SM0.0 - SM0.6) provides seven bits that are updated by the Programming Tool at the end of each scan cycle. You can implement various functions in your program with these bits.

| Special bit memories (read only) | Description |
|---|---|
| SM0.0 | This bit is always switched on. |
| SM0.1 | This bit is switched on in the first scan cycle. It is used, for example, to call an initialization subprogram. |
| SM0.2 | This bit is switched on for the length of a scan cycle if retentive data has become lost. It can be used either as an error bit memory or as a mechanism for calling up special startup sequences. |
| SM0.3 | This bit is switched on for the length of a scan cycle if the RUN mode has been set after switching on (power on). This permits a warm-up time for the system before starting operation. |
| SM0.4 | This bit provides a sampling time that is switched on for 30 seconds and switched off for 30 seconds, for a scan cycle time of 1 minute. This way you have an easy to program delay or a cycle time of 1 minute. |
| SM0.5 | This bit provides a sampling time that is switched on for 0.5 seconds and switched off for 0.5 seconds, for a scan cycle time of 1 second. This way you have an easy to program delay or a cycle time of 1 second. |
| SM0.6 | This bit represents a clock cycle. It is switched on for one scan cycle and switched off for the next scan cycle. So you can use this bit as a cycle counter input. |

# How to perform common tasks

# 10

## 10.1 Customize the working area

The menu command View > Zoom (Page 452)or the Zoom toolbar buttons  set:

- Zoom percentage
- Grid width in pixels
- Grid height in pixels

The menu command Tools > Options (Page 488)sets:

- Tab "General"
  - Measurement system
  - Time
  - Date
  - Language
- Tab "Colors"
  - Font and color preferences
- "LAD Editing" tab
  - Editor zoom percentage
  - Grid width in pixels
  - Grid height in pixels
  - Format field widths for network symbol information tables
  - Setup operand view for symbols only (Pump_1) or symbol and address (Pump_1:Q0.0)
- "LAD Program Status" tab
  - Status view zoom percentage
  - Grid width in pixels
  - Grid height in pixels
  - Signal flow color
  - Display operands inside the instruction, outside the instruction, or show only the status value

The menu command Tools > Customize (Page 487)sets:

- Tab "Commands"
  - View or hide tooltips
  - Choose 3D or flat buttons
  - Configure toolbar buttons
- Tab "Add-On Tools"
  - Add your favorite applications to the menu "Tools"

# 10.2 Edit a program

## 10.2.1 Data classes

### Supporting CPUs

828D
828D Step 2

### Assignment of blocks into data classes

Assigning blocks in data classes is used to efficiently upgrade or commission a Sinumerik system.

A distinction is made between four data classes:

| | |
|---|---|
| • System (S) | is allocated by Siemens AG. |
| • Manufacturer (M) | is allocated by the machine manufacturer during construction (CPU-type specifications) |
| • Individual (I) | is allocated by the machine manufacturer during first commissioning (values for this specific machine) |
| • User (U) | is set by the end user (values associated with the particular process) |

#### Assigning a block to a data class

You assign a new data class to a data block in its Properties dialog (Page 452). In the project tree, right-click the corresponding data block and select "Properties". You cannot assign a different data class to your program blocks.

In their Properties dialog, these data classes can be assigned to blocks:

| Block | Default setting | Can be changed |
|---|---|---|
| Main program | Manufacturer | No |
| Subprogram | Manufacturer | No |

| Block | Default setting | Can be changed |
|---|---|---|
| Interrupt program INT0<br>Interrupt programs INT100, INT101 | Manufacturer<br>Individual | No<br>No |
| Data block | Manufacturer | User<br>Individual |

## 10.2.2 NC variable editor

### CPUs that support freely configurable PLC interfaces

828D (06.00 or higher)
828D Step 2 (06.01 or higher)

### NC variable selection

To open the NC variable selection, proceed in one of the following ways:

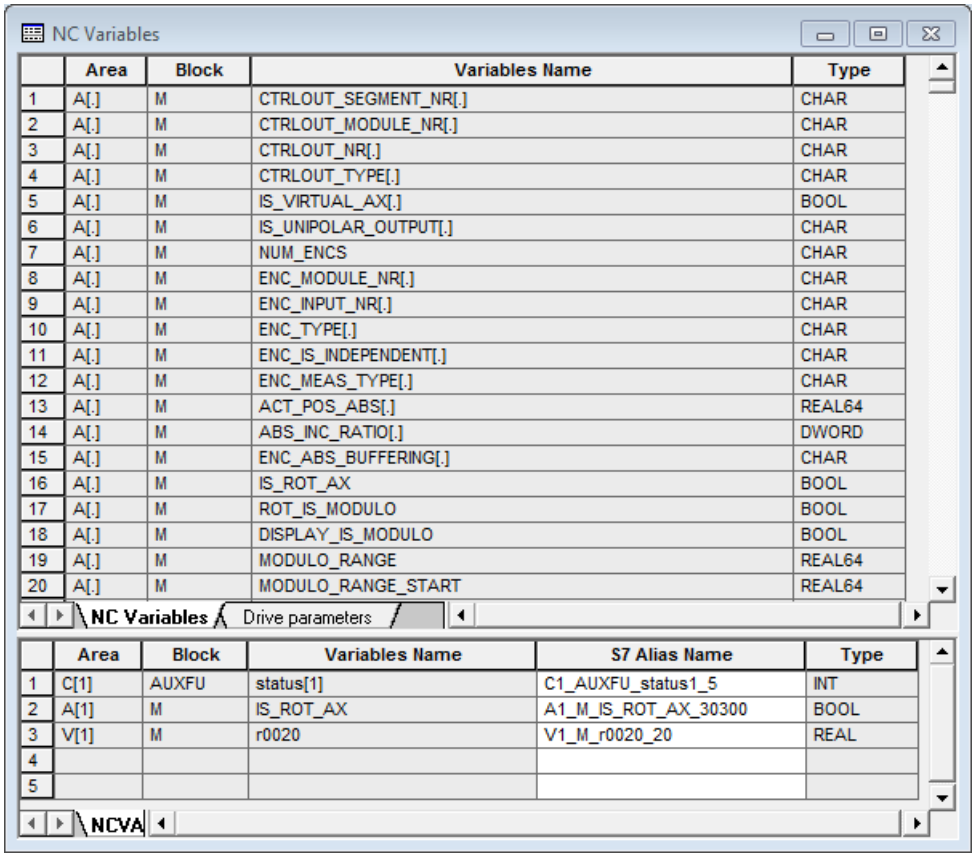
- Click the button of the NC variables  in the "Navigation" toolbar (Page 500).
- Select the **View > NC Variables** menu command.
- In the project tree (Page 447) under **NC Variables > NC Variables** List, double-click the variable table "NC Variables".

#### Installation

The NC variable selection window is divided into two sections.

- The selected variables list is shown in the upper section. It is possible to switch between the variables lists "NC Variables" and "Drive Parameters" via the tabs located at the bottom of the section.
- The lists of the selected variables are shown in the lower section. It is likewise possible to switch between these lists by using the tabs at the bottom of the section.

By using the horizontal separation bar between the sections, the size of the sections can be changed or a section can be hidden.




| | Area | Block | Variables Name | Type |
|---|---|---|---|---|
| 1 | A[.] | M | CTRLOUT_SEGMENT_NR[.] | CHAR |
| 2 | A[.] | M | CTRLOUT_MODULE_NR[.] | CHAR |
| 3 | A[.] | M | CTRLOUT_NR[.] | CHAR |
| 4 | A[.] | M | CTRLOUT_TYPE[.] | CHAR |
| 5 | A[.] | M | IS_VIRTUAL_AX[.] | BOOL |
| 6 | A[.] | M | IS_UNIPOLAR_OUTPUT[.] | CHAR |
| 7 | A[.] | M | NUM_ENCS | CHAR |
| 8 | A[.] | M | ENC_MODULE_NR[.] | CHAR |
| 9 | A[.] | M | ENC_INPUT_NR[.] | CHAR |
| 10 | A[.] | M | ENC_TYPE[.] | CHAR |
| 11 | A[.] | M | ENC_IS_INDEPENDENT[.] | CHAR |
| 12 | A[.] | M | ENC_MEAS_TYPE[.] | CHAR |
| 13 | A[.] | M | ACT_POS_ABS[.] | REAL64 |
| 14 | A[.] | M | ABS_INC_RATIO[.] | DWORD |
| 15 | A[.] | M | ENC_ABS_BUFFERING[.] | CHAR |
| 16 | A[.] | M | IS_ROT_AX | BOOL |
| 17 | A[.] | M | ROT_IS_MODULO | BOOL |
| 18 | A[.] | M | DISPLAY_IS_MODULO | BOOL |
| 19 | A[.] | M | MODULO_RANGE | REAL64 |
| 20 | A[.] | M | MODULO_RANGE_START | REAL64 |

| | Area | Block | Variables Name | S7 Alias Name | Type |
|---|---|---|---|---|---|
| 1 | C[1] | AUXFU | status[1] | C1_AUXFU_status1_5 | INT |
| 2 | A[1] | M | IS_ROT_AX | A1_M_IS_ROT_AX_30300 | BOOL |
| 3 | V[1] | M | r0020 | V1_M_r0020_20 | REAL |
| 4 | | | | | |
| 5 | | | | | |

Figure 10-1 NC variable selection window

### Editing NC variables

In order to accept the desired variable from the variables list into the current list of selected variables, double-click the row of the desired variable using the left mouse button. Alternatively, you can highlight the row of the desired variable, copy it to the clipboard using Ctrl + C, and then paste it into the current list with Ctrl + V.

As a rule, the variable still has to be parameterized. This means that the channel or axis number as well as row or column must still be supplemented. A dialog, into which the missing parameters can be entered, opens automatically.

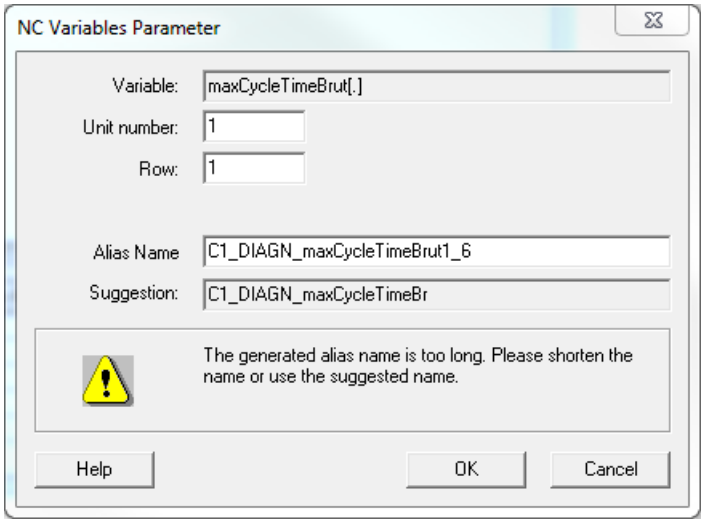



Figure 10-2 NC Variable Parameters dialog

A maximum of 42 variables per list can be saved. A maximum of three lists are available. There is a fixed assignment between the number of variables and the variables list. Variables List 1 (NCV1) contains variables 1 to 42, Variables List 2 (NCV2) contains variables 43 to 84, and Variables List 3 (NCV3) contains variables 85 to 126. The three lists exist independently of one another, i.e. Variables List 1 does not have to contain all 42 variables before variables can be entered into Variables List 2. Thus it is possible to assign the variables thematically, e.g. NCVAR_CHAN (NCV1) for NC channel variables and NCVAR_AX (NCV2) for NC axis variables.

**Note**

**Restriction for the 828D 06.xx systems**

The "828D 06.xx" systems support only 42 variables and therefore recognize only Variables List 1 (NCV1) and also only the NC_DATA1 (DB9910) data block.

In the list of selected variables, the parameters of a variable can be changed either by double-clicking on the variable, or in the menu **"Edit > Variable parameter"** or shortcut menu **right mouse button > Variable parameter**.

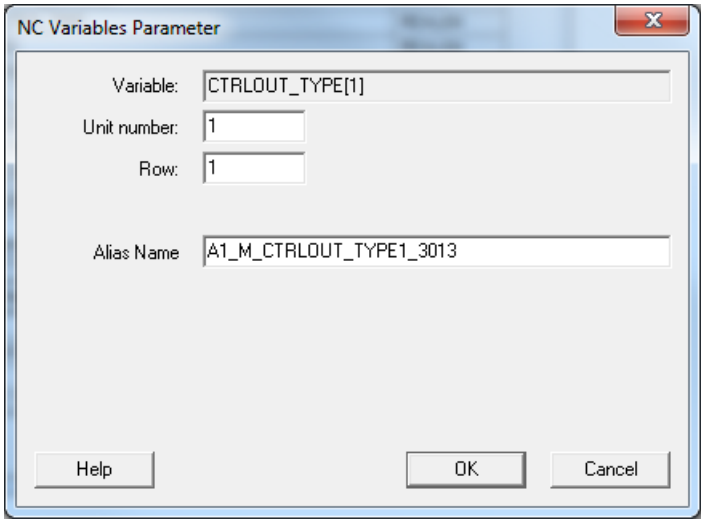



Figure 10-3 NC Variable Parameters dialog

The variable name is automatically generated from the dataset. Since the maximal length is 23 characters, the automatically generated name may be too long. In this case, an alternative name is proposed. The variable name can be edited in all cases. Independent of this, the variable name can always be edited in the list of selected variables. Refer also to Editing NC variables (Edit menu) (Page 403)

**Special case of parameterization of a variable:**

If the variable is a field, whose individual elements can be addressed via the line index, then it is sufficient to accept this variable once in the list of selected variables. In this case, a zero is entered for the line in the parameterizing dialog. If this variable is used, the user must enter the desired row, e.g. for R parameter **R parameter number + 1**, in the user interface **"Read/write NC variable"** in DB 120x RW_NCDx in addition to the variable index (DBB 1000 A_VarIdx) for the row index DBW 1002 A_RowIdx. If the row is not entered, an error message appears when reading/writing.

**Possible errors in the selection of a variable:**

In some cases, variables that do not exist for the current CPU type, e.g. **"828D"**, may also be included in the variables list. In this case, the reading or writing of the variable in the PLC user program will cause an error. In the user interface **"Read/write NC variable"** it is indicated for the variable that an error has occurred (DB120x.DBX3000.1 RW_NCDx.E_Error), and the value 10 is entered in the variables' access result (DB120x.DBD3001 RW_NCDx.E_AccRes). This means that the object does not exist. This also occurs if areas or rows described in help are used that do not exist for the current CPU type.

In addition, some variables use the "STRING" data type. This is not supported by the current CPU types. If such a variable is selected, an error message is displayed as soon as the selection is made. The variable is not transferred into the list of selected variables.

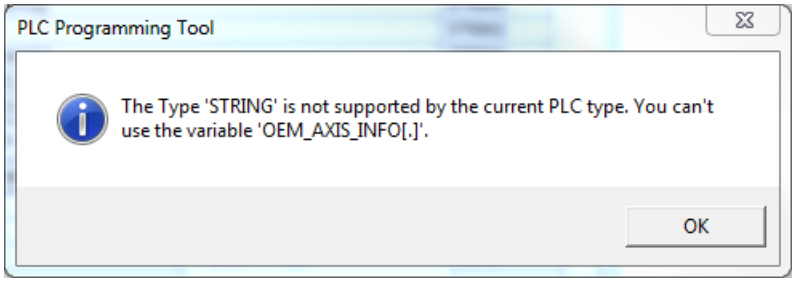



Figure 10-4 Error message data type

## Variables help

Help is available for each variable. With this help function, the variable is described and the parameter to be entered is explained. The help function can be called via the variables' parameterization dialog with the **"Help"** button.

| **30300** | IS_ROT_AX | |
|---|---|---|
| - | Rotary axis / spindle | BOOLEAN |

1: Axis: The axis is defined as a "rotary axis".
- The special functions of the rotary axis are active or can be activated by means of additional machine data according to the type of machine required
- The unit of measurement is degrees.
- The units of the axis-specific machine and setting data are interpreted as follows with the standard control setting:
- Positions in "degrees"
- Speeds in "rev/minute"
- Acceleration in "rev/second²"
- Jerk limitation in "rev/second³"

Spindle:
The machine data should always be set to "1" for a spindle, otherwise alarm 4210 "Rotary axis declaration missing" is output.

0: The axis is defined as a "linear axis".

Special cases:

Figure 10-5 Excerpt from the help function for NC variables

## "Edit" menu

Depending on the list in which the user is currently navigating, the following functions are supported by menu **"Edit"**:

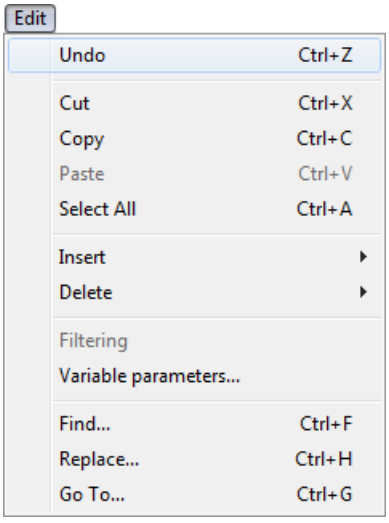


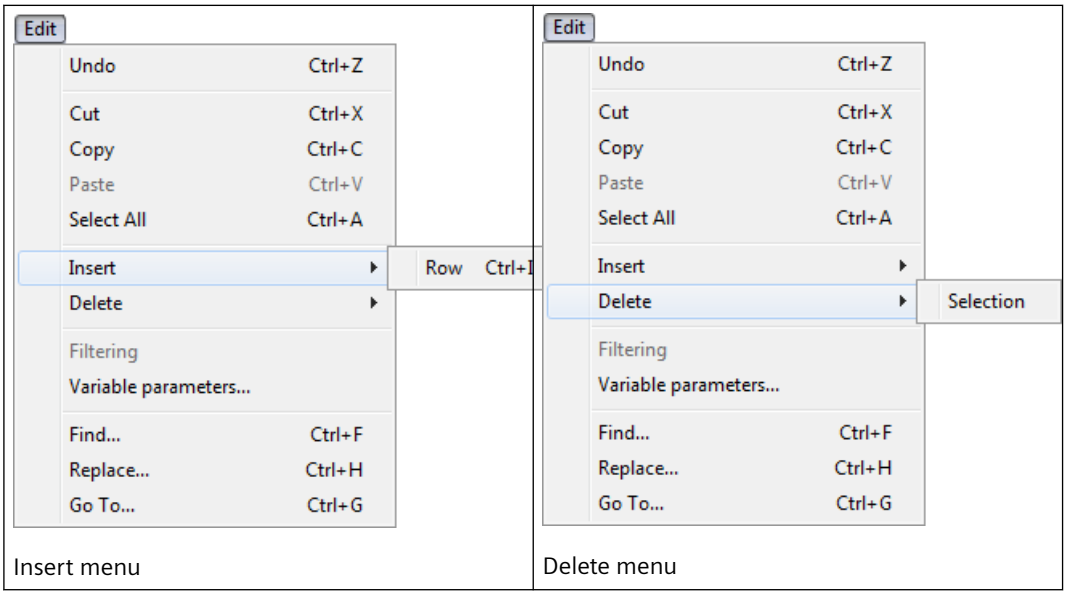

Figure 10-6 Edit menu

### List of selected variables

| Function | Key sequence |
|---|---|
| Undo | Ctrl + Z |
| Cut | Ctrl + X |
| Copy | Ctrl + C |
| Paste | CTRL + V |
| Select All | Ctrl + A |
| Insert Contents > Row | Ctrl + I |
| Delete > Selection | - |
| Variable parameters... | - |
| Find... | Ctrl + F |
| Replace... | Ctrl + H |
| Go to... | Ctrl + G |

- Changes can only be undone as long as they have not been compiled or the project has not been saved. For **"Cut"** and **"Copy"**, the data must be selected.
- The function **"Cut"** copies the selected data to the clipboard and deletes the data at the cut position.
- With **"Copy"**, the selected data is copied to the clipboard and is not deleted.
- In order to select the entire row, left-click on the row number. To select the entire content of the list, left-click on the upper left field, or use the **"Select all"** function.
- If data, e.g. entire rows from the variables list or data from the list of selected variables, exist on the clipboard, these are inserted at the cursor position using **"Paste"**. When an entire row is inserted, it is placed after the row on which the cursor is located. With the use of the clipboard, data can also be exchanged with other application, such as Microsoft Excel.

- With the use of **"Insert > Row"**, an empty row is inserted after the row on which the cursor is located.
- **"Delete > Selection"** deletes the selected data.




Insert menu | Delete menu

- **"Filter"** opens a dialog in which filter criteria can be selected. In this way it is possible to limit the very long variables list to the desired selection and thereby to make it clearer. Refer also to Filtering NC variables (Edit menu) (Page 402)
- The parameters of the variables can be changed using **"Variable parameter..."**.
- **"Find..."**, **"Replace..."** and **"Go To..."** correspond to the standard functions, but are not available in the shortcut menu. With **"Find..."** it is additionally possible to select a search by machine data numbers. **"Replace..."** is not relevant because the variables list cannot be edited. For further information: Find/Replace/Go To (Edit menu) (Page 406)
- Functions that are currently not supported are disabled.

## Options

The values of NC variables partially depend on the system of units (metric/inch). For this reason, the system of units to be used must be selected. The selected system of units is valid for all variables in the list of selected variables. The system of units is set in the **"Options"** dialog, which can be selected in the **"Tools > Options..."** menu. See also Chapter Options (Page 488).

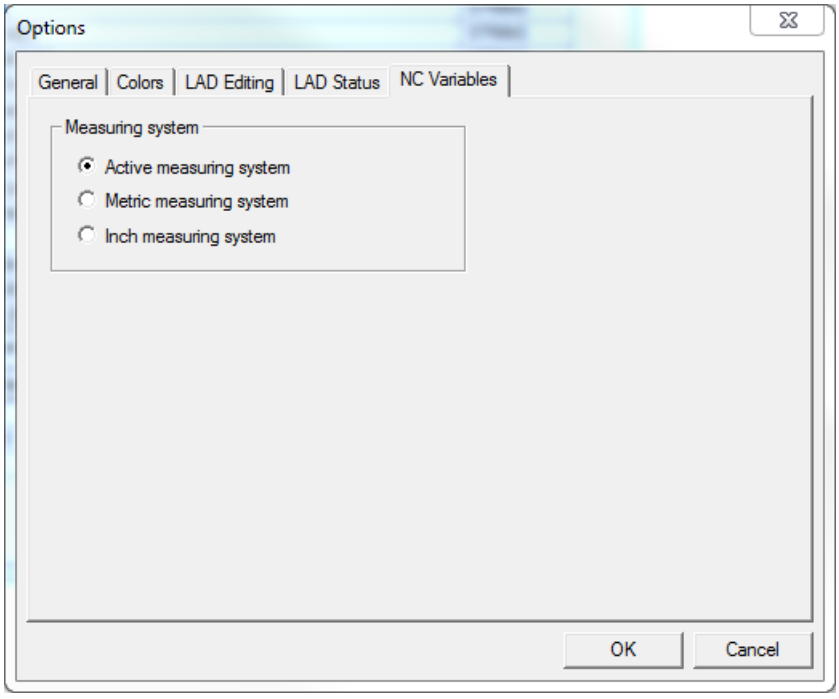



Figure 10-7 "Options" dialog

## Compiling

After one or several lists have been created with selected variables, the special data blocks NC_DATA1 (DB9910), NC_DATA2 (DB9911), and NC_DATA3 (DB9912) can be created in the menu **"PLC > Compile"**.

There is a fixed assignment:

- The information on the variables of list 1 (NCV1) is stored in the data block NC_DATA1 (DB9910).
- The information on the variables of list 2 (NCV2) is stored in the data block NC_DATA2 (DB9911).
- The information on the variables of list 3 (NCV3) is stored in the data block NC_DATA3 (DB9912).

The data blocks contain all of the information required for parameterizing the user interface "Read/write NC variable" in the PLC user program. In this way, the PLC firmware can read and write the variables that are entered into the user interface.

If a list of the selected variables is changed, a new compilation is required so that the special data blocks are updated. The compilation is performed automatically for **"Export"** and **"Download to CPU"**.

The data blocks (DB9910, DB9911, DB9912) receive the properties of the associated list of the selected variables (NCV1, NCV2, NCV3), i.e. name, list number, author, comment and data class, and are saved in the PLC user project with the attribute **"Readonly"**. In this way, the data blocks become a component of the PLC user project. They are thereby saved, opened, exported, imported, downloaded to the CPU and uploaded from the CPU with the user project, and they can be displayed in the data block editor. The content of the data blocks can be read by the PLC user program or the PLC firmware.

The lists of the selected variables are, just like the special data blocks DB9910, DB9911, and DB9912 that are available following compilation, stored in the PLC user project. In contrast to the special data blocks that are created during compilation, the lists of the selected variables can be copied, pasted, deleted, and even renamed. For this purpose, the corresponding list is selected in the instruction tree and the shortcut menu (right mouse button) is activated.

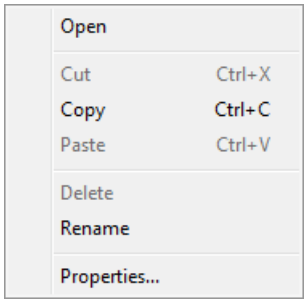



Figure 10-8 Selected NC variables shortcut menu

## "NC Variables" properties dialog

The name, list number, author, comment, and data class can be changed or entered in the properties dialog of the lists of selected variables.

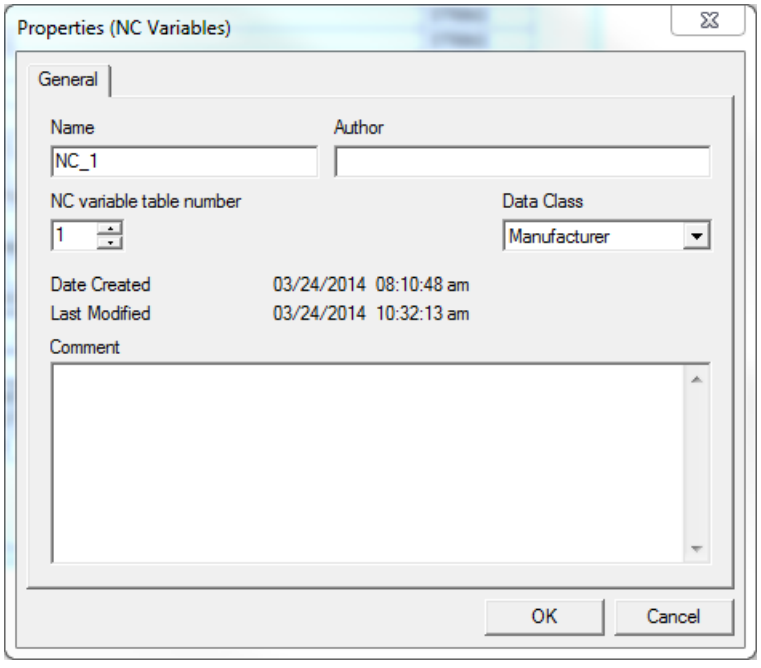



Figure 10-9 NC Variables properties dialog

## Print

The lists of the selected variables can also be printed. For this purpose, the dialog **"Print"** of the PLC Programming Tool is selected in the **"File> Print..."** menu. The print preview is available in the **"File> Print Preview"** menu.

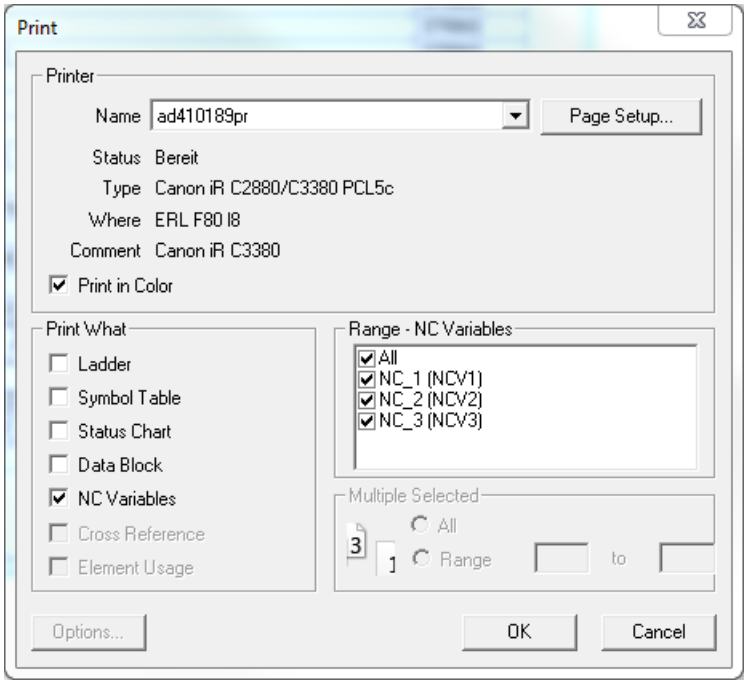



Figure 10-10 Print dialog

## Example

With the PLC user program, it is generally necessary to make use of the NCK data.

Each use case has different requirements with regard to the use of NC variables. It is possible to select NC variables and drive parameters using the function "Freely configurable PLC interface".

Using the NC variable selection, you can compile your own lists of NC variables and drive parameters. The number of lists is limited to three. A maximum of 42 variables per list can be saved. The "Read/Write NC variable" user interface in the PLC user program can thus be parameterized easily.

A maximum of eight variables can be read or written simultaneously. All of the necessary operations can be performed with the PLC Programming Tool during the creation of the project.

A particular operational sequence is necessary in order that the selected variables be available for programming:

1. Opening of the NC variable selection and compilation of the required NC variables and drive parameters.
   The lists of the selected variables can be supplemented later at any time.
2. Compilation of the lists using the **"PLC > Compile"** menu.
   In so doing, the special data blocks (DB9910, DB9911, DB9912) are created. Following this, the selected variables are available for the user program.
3. Creation of the program blocks in the LAD editor.
   The variables included in DB9910 can then be entered symbolically into the user interface **"Read/write NC variable"**, e.g. NC_DATA1.A1_M_IS_ROT_AX_30300.
4. Compilation of the project via the **"PLC > Compile"** menu.
5. Loading of the project into the PLC

**Note**

For a variable that represents a field, i.e. the individual elements can be addressed via the line index, it is sufficient to incorporate this variable once into the list of selected variables. In this case, enter a zero for the line in the parameterization dialog. If this variable is used, you must enter the desired line, e.g. for R parameter R parameter number + 1, in the user interface in addition to the variable index for the line index. This permits multiple variables to be addressed with one entry into the variables list.

**See also:**

Edit menu (Page 398)

Properties (View menu) (Page 452)

NC variables (View menu) (Page 434)

NC variables tab (Tools > Options) (Page 493)

Print (File menu) (Page 394)

## 10.2.3 Save data in variable memory with the help of a data block

If you wish to work with data blocks, then select a CPU (Page 469) that supports data blocks.

### CPUs that support data blocks


828D
828D Step 2

808D-PPU14x (only special data blocks)
808D-PPU15x (as of PLC 07.03)
808D-PPU16x (as of PLC 07.03)

### Procedure

Use one of the following methods to access the data block:

- Click the button of the data block  in the "Navigation" toolbar (Page 500).
- Select the menu command **View > Data Block**.
- Click the button of the data block  in the project tree (Page 447).

These help topics are explained in the following:

### Data block properties

The data block structures are, just like the POUs, part of the project. They are compiled, saved, imported or exported with the project. They are loaded into/from the PLC with the project. The data blocks themselves only contain actual values. Actual values may be loaded into/from the PLC independently of the project. It is therefore essential that the structure of the data block to be loaded in the PLC is exactly the same as that of the project opened in the Programming Tool. Data blocks are listed like POUs in their own symbol table.

When changing the CPU, existing data blocks are not lost. However, if you select a CPU in which data blocks are not available, then please note that the variables from the data block are now displayed with their absolute address (e.g. DB9000.DBB0 or VB90000000). Whether this V range exists is first checked when the PLC is brought into the RUN state. In this case, the program cannot be executed.

To rename your data blocks and change their properties (Page 452), right-click the corresponding data block in the operation tree. Select "Properties". The "Data Block Properties" dialog box opens. Here, you can change the name, block number, and data class (Page 243), as well as add an author and comments. You can also activate the property "Non-retain" for data blocks. This ensures that every time the PLC is brought into the RUN state, the data block in the PLC is preset with the initial values. Data blocks can be write-protected, that is, the actual values of the data block cannot be changed with the Programming Tool. This property cannot be changed.

**Note**

If you just want to rename your data block, right-click the object you want to rename in the operation tree and select "Rename".

You can also easily edit your data block in another program (e.g. Microsoft Excel) (see Cut, Copy and Paste section in the data block editor).

### Assigning addresses and initial values in the data block

When you create a variable, you assign its name and data type. The default initial value is set to zero/OFF, however, this can also be changed. You can optionally insert comments. The Programming Tool automatically assigns the address. Each address is aligned according to the size of its data type, which means that gaps can occur. The actual values are edited in the data view. However, you may assign the initial values to all actual values.

The maximum size of a data block is limited (512 bytes), if it is exceeded then the Programming Tool marks the excess variable addresses using a red wavy line. The Programming Tool returns an error when compiling the project.

### Assigning actual values

Actual values are saved, exported or imported in your project file as part of your data block. You can only assign individual actual values in the data view. The structure of the variable (name, data type, initial value, and comment) is write-protected in the data view. You can only change the actual value and therefore specifically set initial values, for example. After loading the data block (only actual values), these values become effective in the PLC.

The structure of the data block in the project must match the structure of the data block in the PLC. If you have modified the structure of the data block, then the complete project has to be loaded again.

You can access the data view in the following ways:

- Click the Display data block values  button in the toolbar.
- Select the menu command **View > View Data Block Values**.

### Accepting initial values

If you accept your initial values, then all of the actual values of your data block are overwritten. If you have not selected an initial value for a variable, then the Programming Tool sets it to zero/ OFF. Other data blocks will not be changed.

In order to reset all actual values in the actual data block to the initial values,

- Right-click in the table > Accept Initial Values.
- Select the menu command **Edit > Accept Initial Values**.

### Displaying and changing actual values

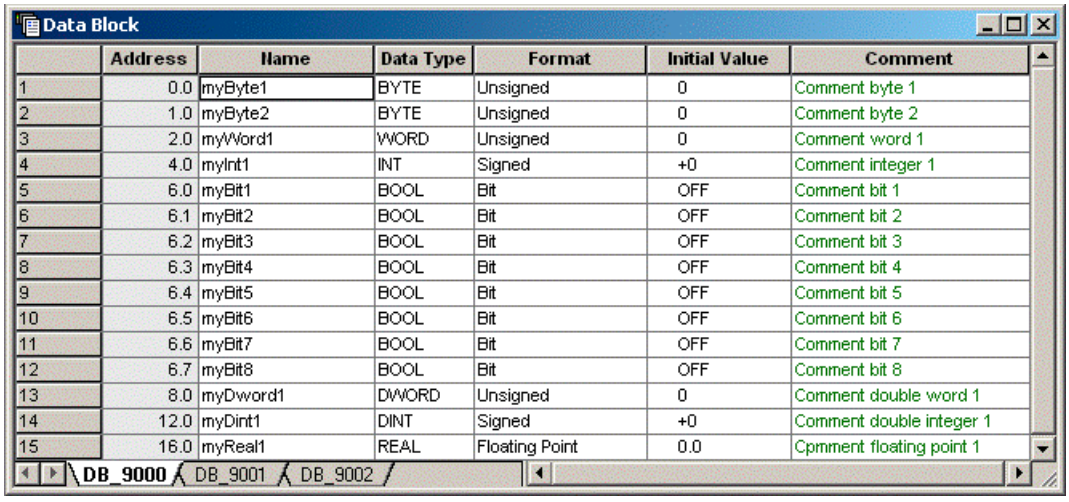
You can view the actual values of a data block in the PLC with the data block status (Page 369) and, if the data block is not write-protected, change them too. The structure of the data block in the project must match the structure of the data block in the PLC. If you have modified the structure of the data block, then the complete project has to be loaded again. When the data block status is activated, the structure of the variable (name, data type, initial value, and comment) is write-protected. If you switch on the data block status, the values in the column "Actual Value" are regularly updated with the actual values of the PLC.

After you have entered values in the column "New Value", write the required changes to the PLC using the command "Write All"  .

You can activate the data block status in the following ways:

- Click the "Data block status"  button in the toolbar.
- Select the menu command **Debug > Data Block Status**.

## Example for a data block



Data Block

| | Address | Name | Data Type | Format | Initial Value | Comment |
|---|---|---|---|---|---|---|
| 1 | 0.0 | myByte1 | BYTE | Unsigned | 0 | Comment byte 1 |
| 2 | 1.0 | myByte2 | BYTE | Unsigned | 0 | Comment byte 2 |
| 3 | 2.0 | myWord1 | WORD | Unsigned | 0 | Comment word 1 |
| 4 | 4.0 | myInt1 | INT | Signed | +0 | Comment integer 1 |
| 5 | 6.0 | myBit1 | BOOL | Bit | OFF | Comment bit 1 |
| 6 | 6.1 | myBit2 | BOOL | Bit | OFF | Comment bit 2 |
| 7 | 6.2 | myBit3 | BOOL | Bit | OFF | Comment bit 3 |
| 8 | 6.3 | myBit4 | BOOL | Bit | OFF | Comment bit 4 |
| 9 | 6.4 | myBit5 | BOOL | Bit | OFF | Comment bit 5 |
| 10 | 6.5 | myBit6 | BOOL | Bit | OFF | Comment bit 6 |
| 11 | 6.6 | myBit7 | BOOL | Bit | OFF | Comment bit 7 |
| 12 | 6.7 | myBit8 | BOOL | Bit | OFF | Comment bit 8 |
| 13 | 8.0 | myDword1 | DWORD | Unsigned | 0 | Comment double word 1 |
| 14 | 12.0 | myDint1 | DINT | Signed | +0 | Comment double integer 1 |
| 15 | 16.0 | myReal1 | REAL | Floating Point | 0.0 | Cpmment floating point 1 |

DB_9000 DB_9001 DB_9002

## Addressing

### Direct addressing

- **Absolute address input**
  Input in absolute format, the address of a variable in your data block comprises the data block number (e.g. DB9000), a period, and the address of the variable. However, you can also directly access the variable via a V address (e.g. V9000000.0). The display of the absolute address depends on how the address display is set and this can be selected in the menu **View > DB Address View (Ctrl + B)**.

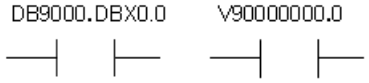



Figure 10-11 Absolute addressing using the data block number

- **Symbolic address input**
  In the symbol table, you can assign an additional name to the variables from your data block. If you wish to use your variable, then you no longer have to specify the complete address - a name in the symbol table is sufficient.
  If you have not listed your variable from the data block in your symbol address, then the symbolic address comprises the name of the corresponding data block (e.g. DB_9000) and the name of the variable in the data block (e.g. Ax1).

| Address | Name of the variables in the data block | Entry in the symbol table | Absolute representation | Symbolic representation |
|---|---|---|---|---|
| VB90000000 | VarName | FeedCorrAx1 | VB90000000 | FeedCorrAx1 |
| or | | | | |
| DB9000.DBB0 | VarName | FeedCorrAx1 | DB9000.DBB0 | FeedCorrAx1 |
| VB90000000 | VarName | | VB90000000 | NameDB.VarName |
| or | | | | |
| DB9000.DBB0 | VarName | | DB9000.DBB0 | NameDB.VarName |

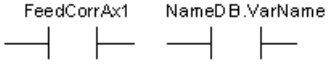



Figure 10-12 Symbolic address representation

The switch between the absolute and symbolic representation is carried out using the menu command **View > Symbolic Addressing (Ctrl + Y)**.

### Indirect addressing

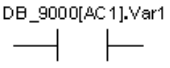
You may be able to simplify the programming if data blocks with the same structure are used in your program. You can indirectly address your data blocks using the accumulators (AC0 to AC3). The accumulators are used to inform the program which data block is to be handled. The value in the AC is then treated as an index.

For example, for axis DBs, the program text can be somewhat reduced by not having to write a dedicated program for each axis, but instead accessing the appropriate axis via the various data blocks and the index (AC). As this value is treated as an index, it starts at 0 for the first axis. Indirect addressing is only possible using ACs. The address cannot be broken down, as

the contents of the accumulator are not known offline. This also means that the absolute address cannot be determined. V addresses cannot be used for indirect addressing.

**Note**

Accumulators are used by both subprograms and the calling program. Calling a subprogram does not cause the accumulators to be saved or restored.




| Absolute input | Symbolic input |
|---|---|
| DB3800[AC1].DBX2.1 | ToAxis[AC1].ControlEnable |
| DB9000[AC0].DBW0 | Prototyp1[AC0].MyWord1 |

**Note**

**Not permitted:**

DB3800[1].DBX2.1 ToAxis[5].ControlEnable

**Indirect addressing using V addresses is not permitted:**

V3800[5]0002.1

V380[5]0002.1

V3800[AC0]0002.1

V38[AC0]0002.1

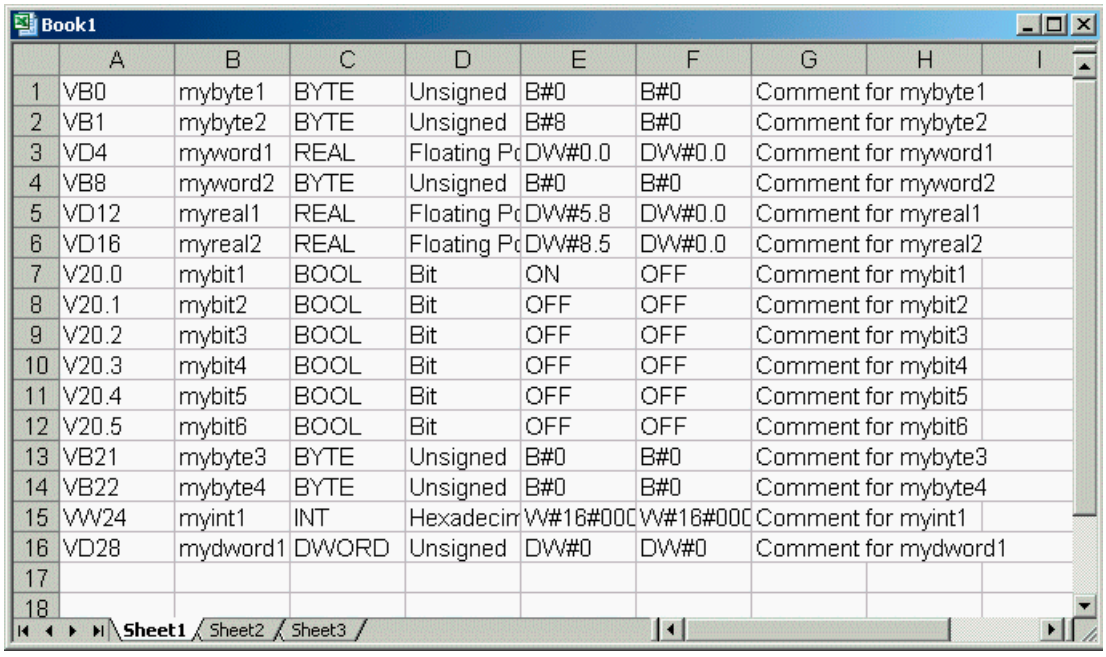
## Using Cut, Copy and Paste in the data block editor

You can save data (rows, columns, and cells) of your data block to the Microsoft clipboard in order to edit it in a different program. You can do this in one of the following ways:

- By right-clicking Cut/Copy/Paste
- By pressing the shortcut keys Ctrl+X/Ctrl+C/Ctrl+V
- By using the menu commands **Edit > Cut**, **Edit > Copy, Edit > Paste**

| | |
|---|---|
| Cut | Selected data is copied to the clipboard and is deleted |
| Copy | Selected data is copied to the clipboard and is not deleted |
| Paste | If data is in the clipboard, the data is pasted |
| Delete | Selected data block/section is deleted, it is not saved in the clipboard |

You can easily edit your data block in this way, e.g. in Microsoft Excel.



Book1

| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 1 | VB0 | mybyte1 | BYTE | Unsigned | B#0 | B#0 | Comment for mybyte1 | | |
| 2 | VB1 | mybyte2 | BYTE | Unsigned | B#8 | B#0 | Comment for mybyte2 | | |
| 3 | VD4 | myword1 | REAL | Floating Pc | DW#0.0 | DW#0.0 | Comment for myword1 | | |
| 4 | VB8 | myword2 | BYTE | Unsigned | B#0 | B#0 | Comment for myword2 | | |
| 5 | VD12 | myreal1 | REAL | Floating Pc | DW#5.8 | DW#0.0 | Comment for myreal1 | | |
| 6 | VD16 | myreal2 | REAL | Floating Pc | DW#8.5 | DW#0.0 | Comment for myreal2 | | |
| 7 | V20.0 | mybit1 | BOOL | Bit | ON | OFF | Comment for mybit1 | | |
| 8 | V20.1 | mybit2 | BOOL | Bit | OFF | OFF | Comment for mybit2 | | |
| 9 | V20.2 | mybit3 | BOOL | Bit | OFF | OFF | Comment for mybit3 | | |
| 10 | V20.3 | mybit4 | BOOL | Bit | OFF | OFF | Comment for mybit4 | | |
| 11 | V20.4 | mybit5 | BOOL | Bit | OFF | OFF | Comment for mybit5 | | |
| 12 | V20.5 | mybit6 | BOOL | Bit | OFF | OFF | Comment for mybit6 | | |
| 13 | VB21 | mybyte3 | BYTE | Unsigned | B#0 | B#0 | Comment for mybyte3 | | |
| 14 | VB22 | mybyte4 | BYTE | Unsigned | B#0 | B#0 | Comment for mybyte4 | | |
| 15 | VW24 | myint1 | INT | Hexadecim | W#16#000 | W#16#000 | Comment for myint1 | | |
| 16 | VD28 | mydword1 | DWORD | Unsigned | DW#0 | DW#0 | Comment for mydword1 | | |
| 17 | | | | | | | | | |
| 18 | | | | | | | | | |

Sheet1 Sheet2 Sheet3

## Using special data blocks

The Programming Tool offers you the possibility to use pre-configured data blocks for special applications. These special data blocks can be found in the operation tree under **Libraries > Special Data Blocks**. These data blocks have a fixed structure. You can make use of them by double-clicking or via copy and paste (in the operation tree).

## Resolving errors

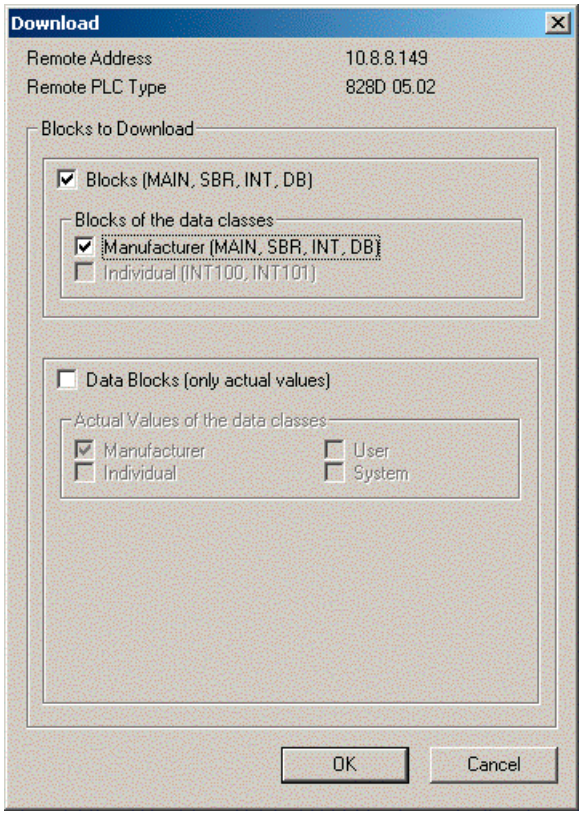
The Programming Tool marks errors made during data input as is the case in the LAD editor or the symbol table. For example, the use of invalid syntax or invalid values can cause an error. You must correct all of the errors that occur before you can compile (Page 459) without any errors and load the program into the PLC.

If any errors occur during compilation of the data block, then they are displayed in the output window. Position the cursor on an error message in the output window and double-click it so that you see the row in the data block with the error.

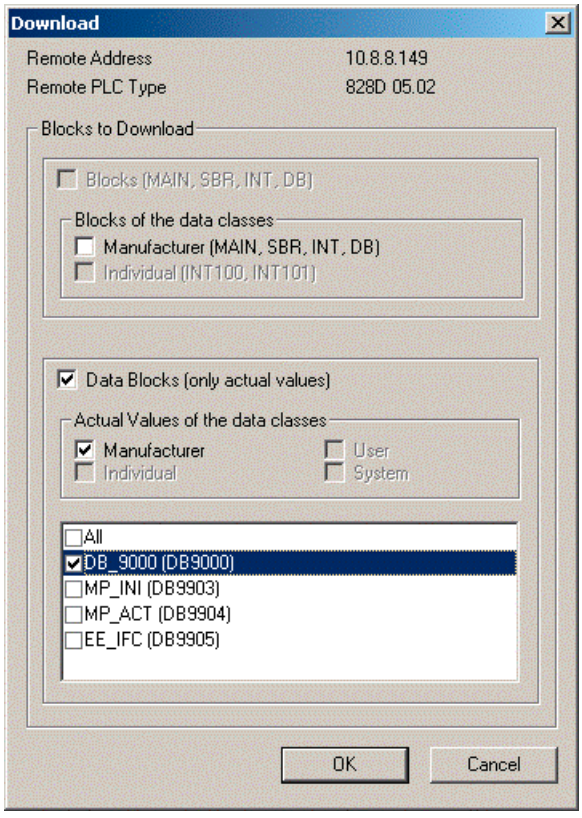
## Loading the data block

### Loading the data block into the PLC

If you edit the structure of a data block, you need to subsequently load the program block (project) into the PLC (Page 382). Your changes will only become effective after the modified project has been loaded into the PLC. Actual values and comments are not loaded into the PLC with the project. If the PLC identifies that a data block is new or has been changed, then it sets the initial values for this data block as the first actual values.

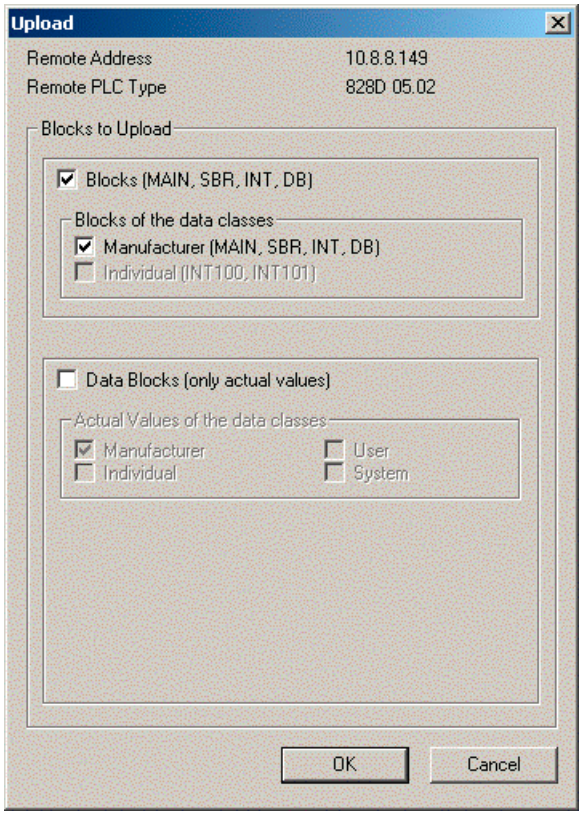



If you edit the actual values of a data block, you need to subsequently load the data block (actual values only) into the PLC (Page 382) so that these actual values become valid. The structure of the data block in the PLC must match the structure of the data block in the project.

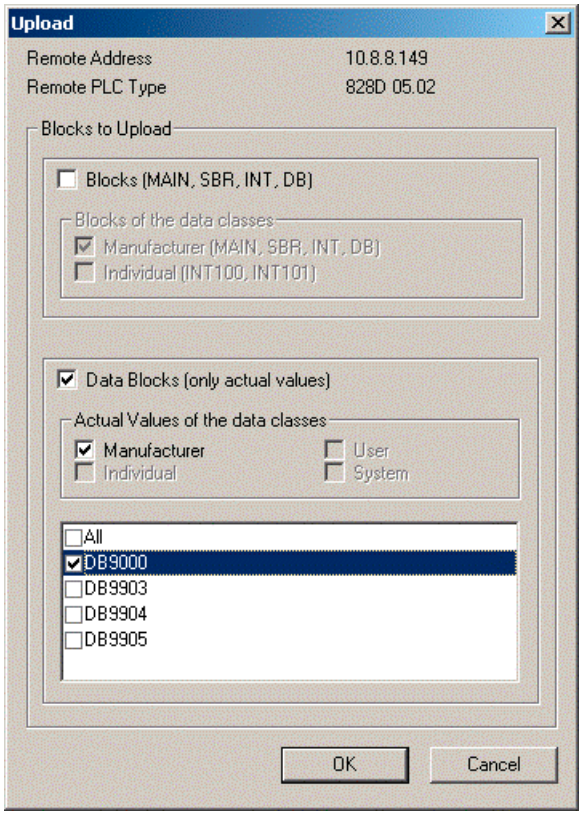



**Loading a data block from the PLC**

You must first open a project in the Programming Tool before you can load the program block (project) and therefore the structure of the data blocks contained in it from the PLC (Page 382).




If you only want to load the actual values of a data block from the PLC, then first open the corresponding project in the Programming Tool or load it from the PLC. Now you can load the data block (only actual values) from the PLC (Page 382). You cannot load the data block if the structure of the data block in the PLC does not match that of the data block of the project that has been opened (or if the project that has been opened does not have a data block).




**See also**

Display the data block status (Page 74)

Constants (Page 271)

Communication (Page 443)

Getting started (Page 17)

Information on using the software (Page 375)

## 10.2.4 Functions Find/Replace/Go To

The dialog **"Find/Replace/Go To"** corresponds to the standard dialog of the PLC Programming Tool.

Use one of the following methods to access the functions Find, Replace, and Go To:

- Select the **Edit > Find**, **Edit > Replace** or **Edit > Go To** menu command.
- Press the shortcut key **Ctrl+F**, **Ctrl+H** or **Ctrl+G**.

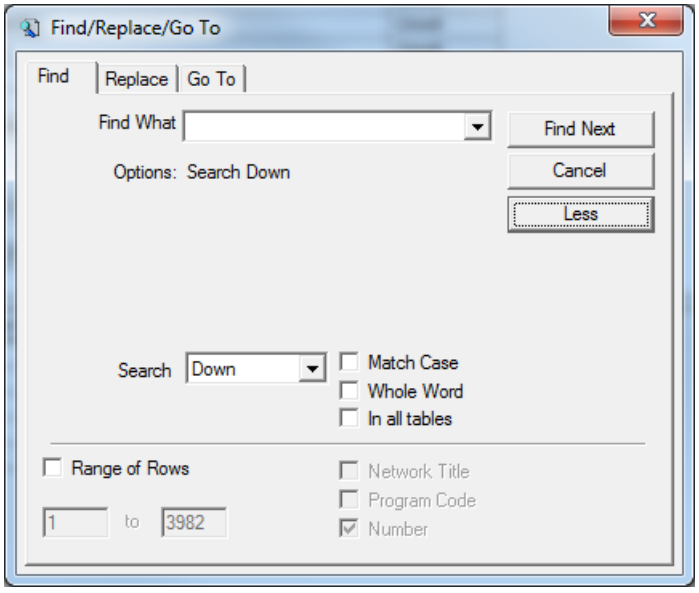



Figure 10-13 Find dialog

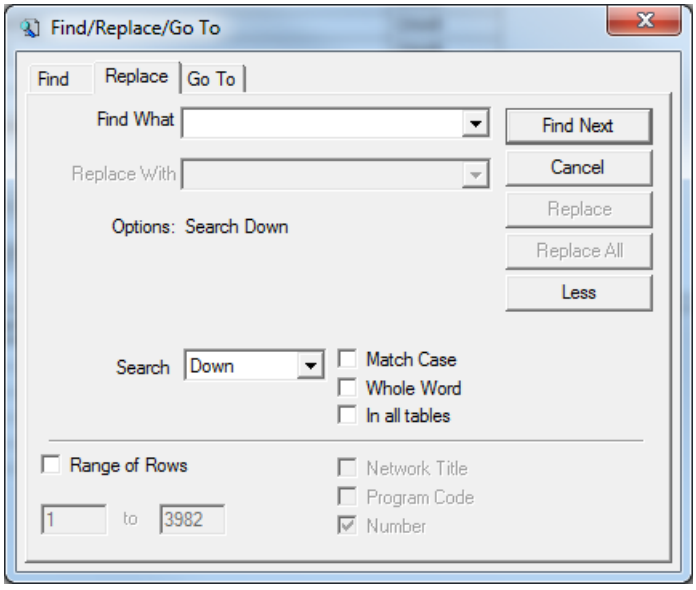



Figure 10-14 Replace dialog

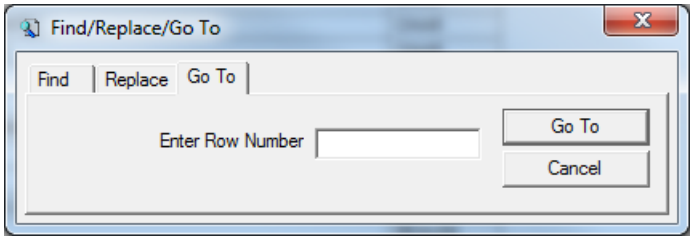



Figure 10-15 Go To dialog

Find, Replace, and Go To functions can be performed on a program, local variable table, program block, symbol table, data block or status chart. The user can also search for machine data numbers.

**See also:**

Find/Replace and Go To function (GS 3.8) (Page 46)

Information on using the software (Page 375)

First steps: Contents (Page 17)

## 10.2.5 How to use cross references and the elements used

### Call

Use one of the following methods to view the Cross Reference window:

- Select the menu command **View > Cross Reference**.
- Click the button for cross-references  in the "Navigation" toolbar (Page 500).

To access the Cross Reference table, the Byte Usage table or the Bit Usage table, click the appropriate tab at the bottom of the "Cross Reference" window:


These help topics are explained in the following:

### Cross Reference table

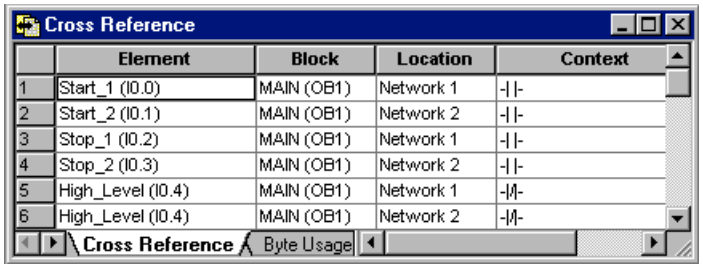
**Note**

You must compile your program in order to view the Cross Reference table.

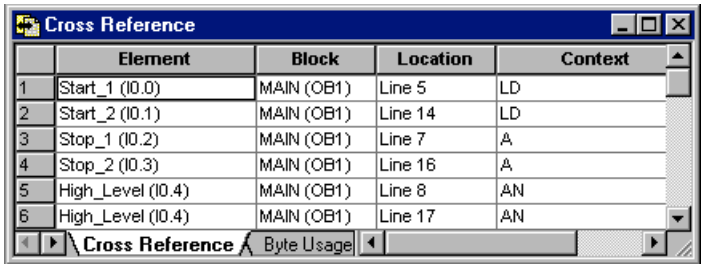
Use the Cross Reference table when you want to know whether a symbolic name or an address is already in use in your program, and where it is used. The Cross Reference table lists all operands used in the program and lists all occurrences of the operands with POU, network or line, as well as the instruction.

| | |
|---|---|
| **Element** | Refers to the operands used in your program. You can toggle between symbolic and absolute view to change the representation of all operands (select the menu command **View > Symbolic Addressing**). |
| **Block** | Refers to the POU where the operand is used. |
| **Address** | Refers to the line or network where the operand is used. |
| **Context** | Refers to the program instruction where the operand is used. |

*Example of an LAD cross reference list*




*Example of an STL cross reference list*




## Byte Usage table

**Note**

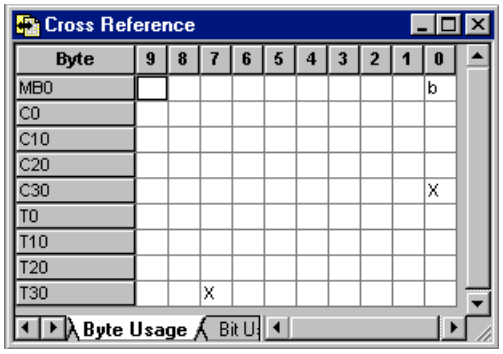
You must compile your program in order to view the Byte Usage table.

The Byte Usage table allows you to see which bytes, from which memory areas, have been used in your program. It also helps you recognize duplicate assignment errors.

| | |
|---|---|
| **b** | Indicates that a bit of memory has been assigned. |
| **B** | Indicates that a byte of memory has been assigned. |
| **W** | Indicates that a word (16 bits) of memory has been assigned. |
| **D** | Indicates that a double word (32 bits) of memory has been assigned. |
| **X** | Used for timers and counters. |

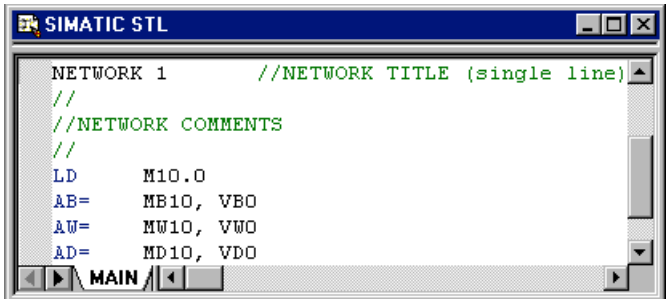
### Interpreting the Byte Usage table

This example of a Byte Usage table shows that its associated program makes use of the following addresses: one bit within MB0; counter C30; timer T37.

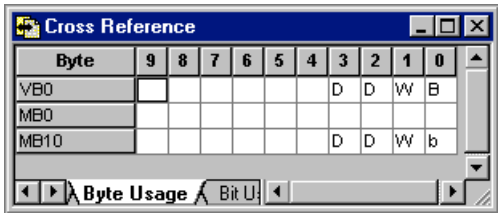



### Recognizing duplicate assignment errors

This example program contains overlapping address assignments at MB10.0.




The Byte Usage table can be examined to identify improper assignments. Since a double word requires four bytes, there should be four adjacent Ds in the row for VB140000000. Likewise, since a word requires two bytes, there should be two adjacent Ws for VB14000000. The row for MB10 reflects the same problems, plus the fact that MB10.0 was used in more than one assignment.




### Bit Usage table

**Note**

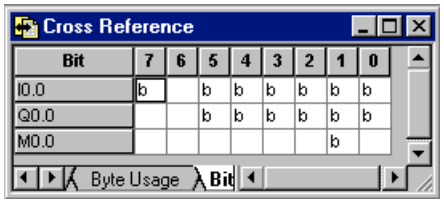
You must compile your program in order to view the Bit Usage table.

The Bit Usage table allows you to see which memory addresses, down to the bit level, have been used in your program. It also helps you recognize duplicate assignment errors.

| | |
|---|---|
| **b** | Indicates that a bit of memory has been assigned. |
| **B** | Indicates that a byte of memory has been assigned. |
| **W** | Indicates that a word (16 bits) of memory has been assigned. |
| **D** | Indicates that a double word (32 bits) of memory has been assigned. |
| **X** | Used for timers and counters. |

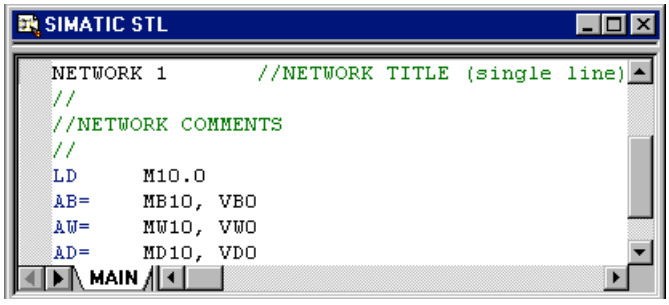
### Interpreting the Bit Usage table

This example of a Bit Usage table shows that its associated program makes use of the following addresses: from byte I0, bits 0, 1, 2, 3, 4, 5, and 7; from byte Q0, bits 0, 1, 2, 3, 4, and 5; from byte M0, bit 1.

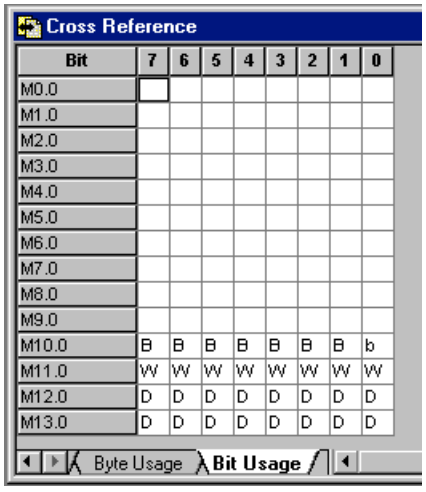



### Recognizing duplicate assignment errors

This example program contains overlapping address assignments at MB10.0.




The Bit Usage table can be examined to identify improper assignments. In a properly assigned program, there would never be a bit value in the midst of the byte. BBBBBBBb is invalid, whereas BBBBBBBB would be valid. The same is true of the word assignments (there should be 16 adjacent Ws) and the double word assignments (there should be 32 adjacent Ds).




**See also:**

Communication configuration (Page 443)

Selecting the CPU type (Page 469)

Changing between symbolic and absolute addressing mode (Page 444)

First steps: Contents (Page 17)

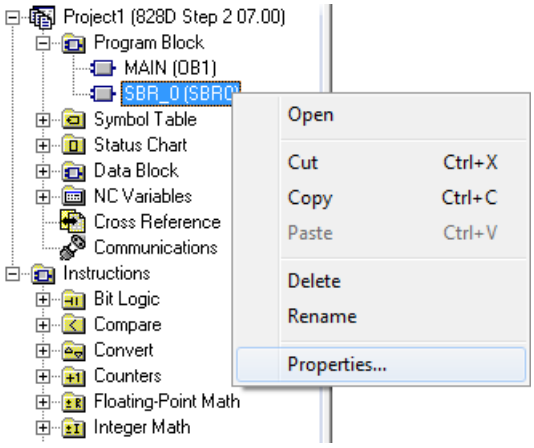
Information on using the software (Page 375)

## 10.2.6 POU password protection

The POU password protection allows you to protect your POU by means of a password. This means that the POU is invisible to other users and is encrypted when it is downloaded.

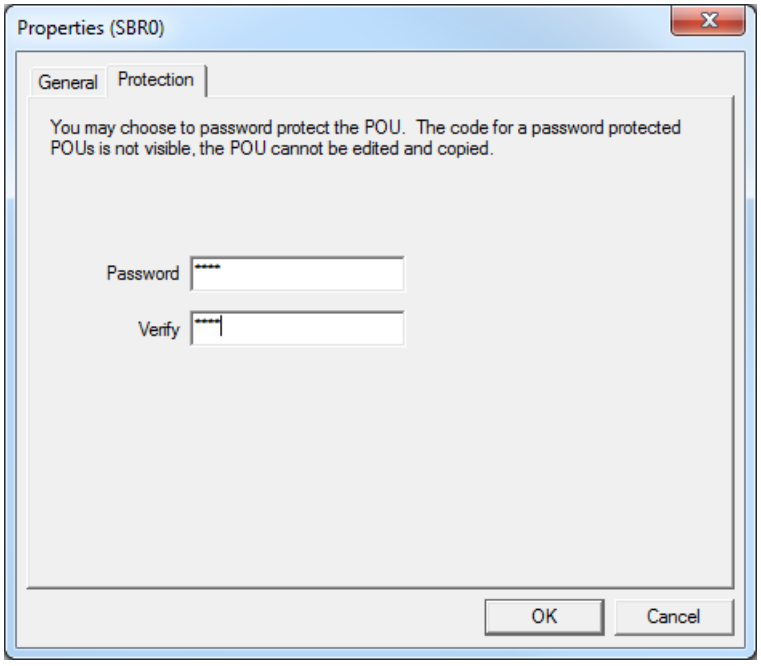
## Setting up password protection

1. Right-click on the corresponding program organization unit and then click on the "Properties" menu option.




   The Properties dialog opens.

2. Click the "Protection" tab.



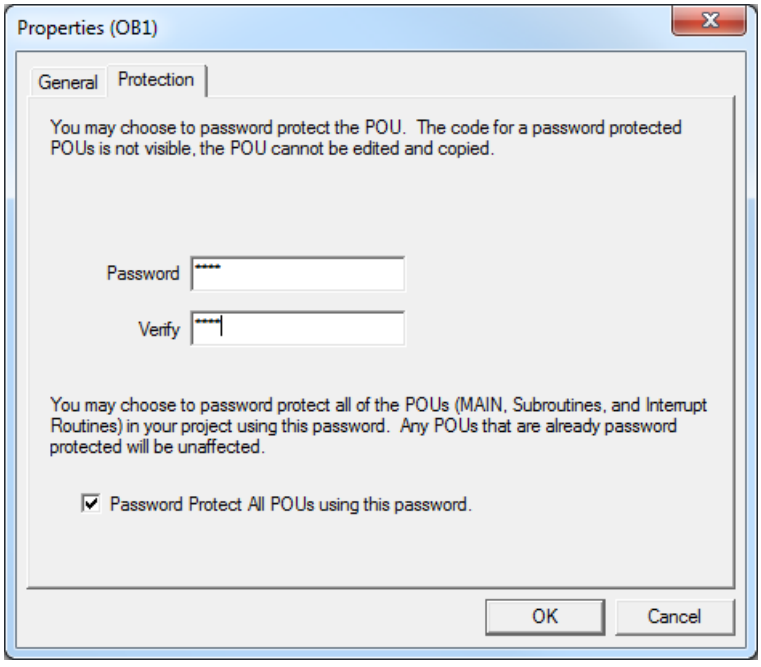
3. Enter a password and confirm it in the next field.

**Note**

**Information on password creation**

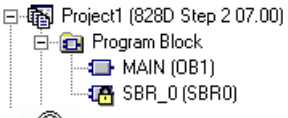
Use uppercase and lowercase letters, numbers, and special characters (ASCII). Spaces are not permitted.

4. In order to protect all POUs with this password, activate the checkbox "Protect all POUs with this password" in the properties for MAIN (OB1).




POUs that are already protected with a password are not affected by this.

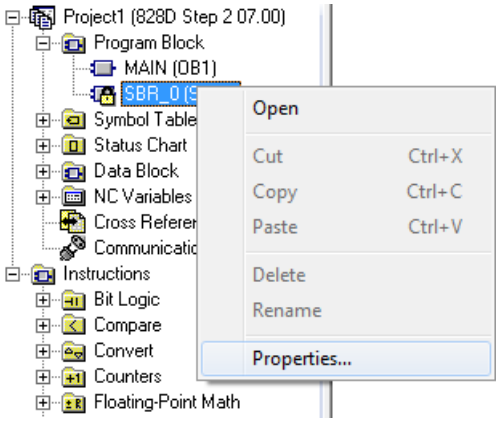
If a POU is password-protected, then it is displayed in the project tree as follows:




A  symbol is displayed in the Program Editor window, which indicates that it is password-protected.

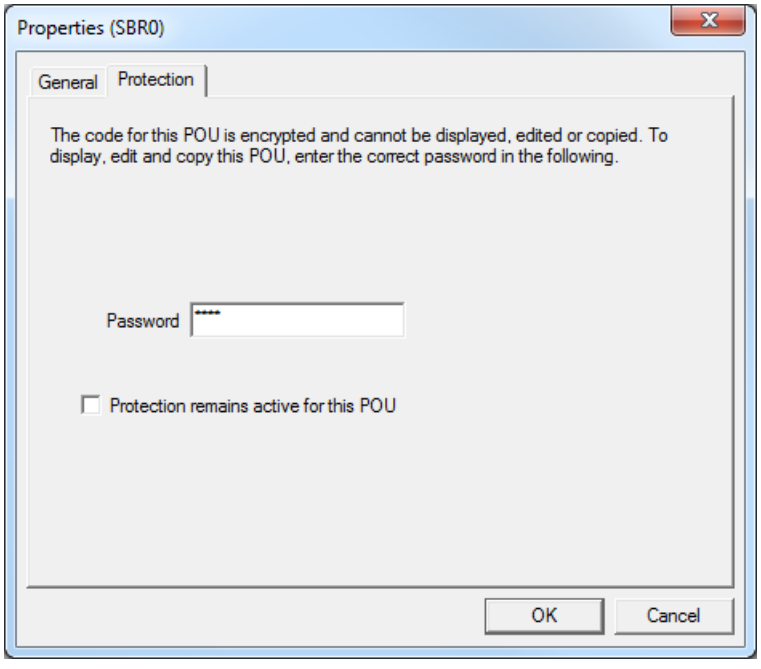
## Removing password protection

1. Right-click on the corresponding program organization unit and then click on the "Properties" menu option.




   The Properties dialog opens.

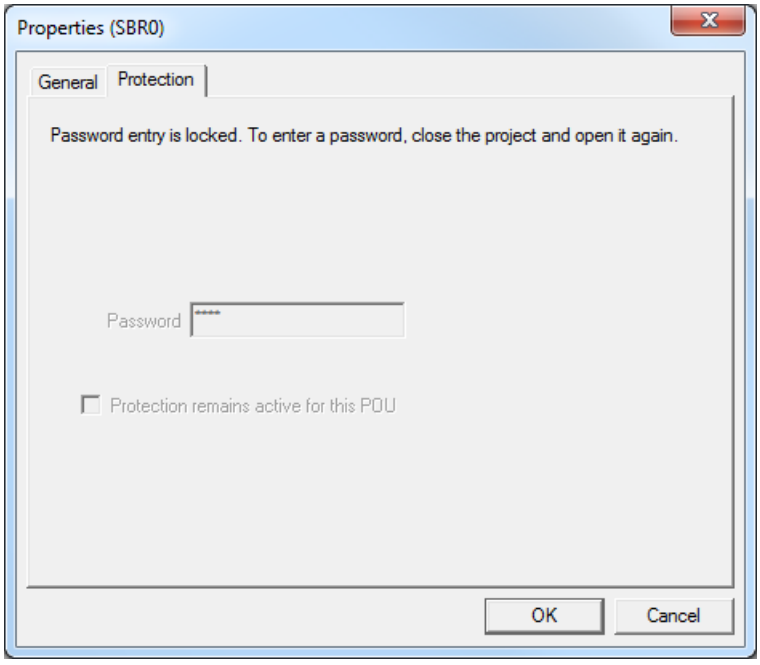
2. Click the "Protection" tab.
3. Deactivate the checkbox "Protection is maintained for this POU".
4. Enter your password, and click the "OK" button.




You must repeat this procedure for each POU that you wish to unlock.

### Repeated entry of an incorrect password

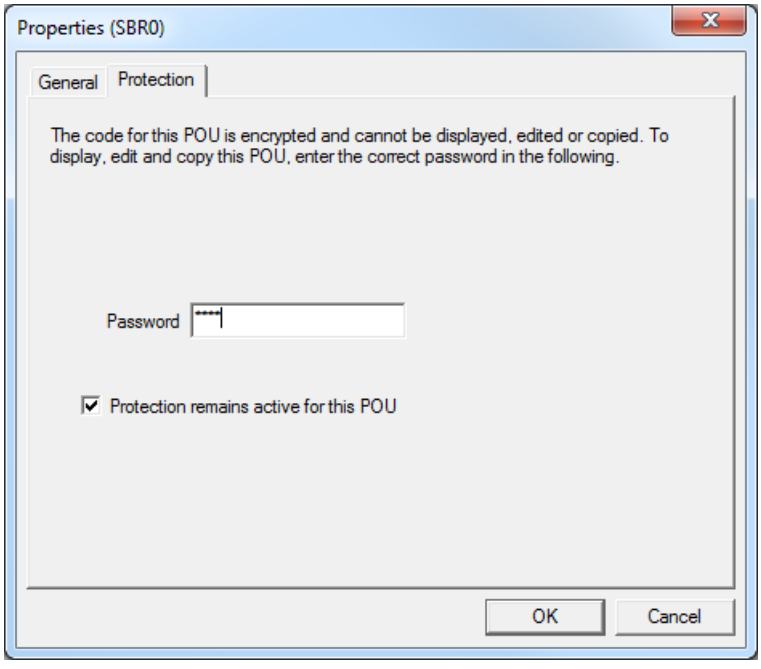
If the password is incorrectly entered three times, the password input option will be disabled. Password input will only become possible again after closing the project and then opening it again.




### Short-term deactivation of password protection

You have the possibility of temporarily deactivating password protection if you only want to temporarily view, edit or copy the POU, while maintaining the protection. As with the permanent removal of password protection, the procedure must be performed individually for each POU.

1. Open the properties of the POU that you want to view and open the "Protection" tab.




2. Enter the password.

3. Activate the checkbox "Protection is maintained for this POU".
4. Confirm with "OK".

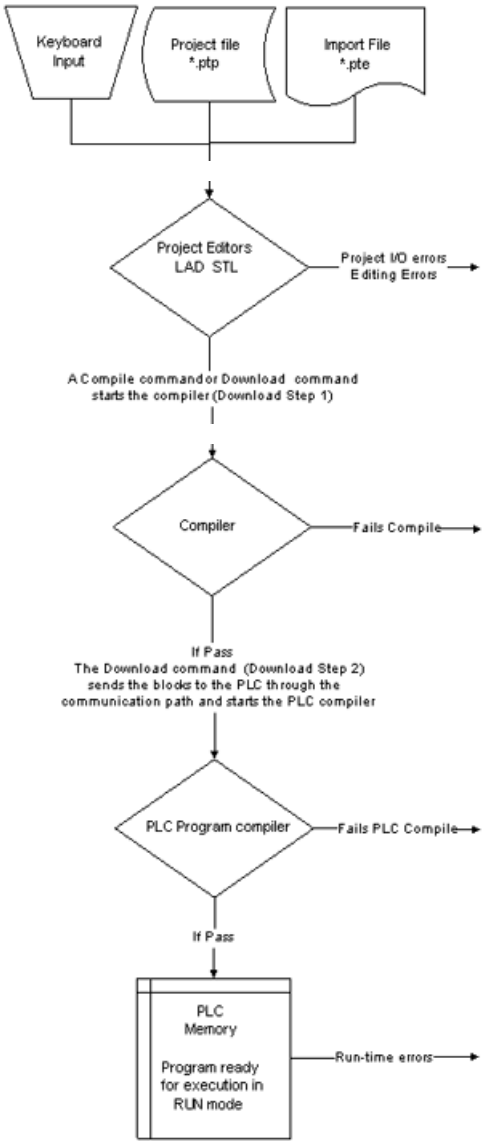
A POU for which the password has only temporarily been canceled appears in the project tree as follows:


The password protection will be automatically reactivated after the project is closed.

## 10.2.7 Error-free compilation

Use one of the following methods to start the Programming Tool Project Compiler:

- Click the Compile button or select the menu command **PLC > Compile** to compile all project components (program block, data block).




Opening projects (Page 379)

Range checking (Page 209)

Selecting the CPU type (Page 334)

All compilation errors in the Programming Tool are listed in the Output Window (Page 451). Double-click the error and the editor scrolls to the error location. Use the menu command PLC > Type (Page 469) to set a specific model and version of the CPU.

There are errors that indicate that the project is exceeding the space available on the target system.

For example, "ERROR 820: The compiled program block or the entire project (including comments, data blocks, etc.) is too large for the current target system."

In this case, all data counts towards the overall size of the project, including program blocks, data blocks, tables, comments, etc.

To enable successful compilation of the project, gradually delete non-essential data until the storage requirements are met again.

**See also:**

Information on using the software (Page 375)

First steps: Contents (Page 17)

## 10.2.8 Configuring PLC user alarms

Their are 1000 extended PLC user alarms available, which have alarm numbers 701000 to 701999. Alarm response, clear (delete) criterion and channel assignment of the alarm response is configured in special data block (DB9913). This is in the operation tree under "Instructions" > "Libraries" > "Special data blocks > "ALARM_INI (DB9913)", and must be loaded into the project using a double-click or copy/paste so that the PLC user alarms are available.

The activations for these alarms are saved in subrange 5 of the DB1600 (ALARM) (DB1600.DBX4000.0 to DB1600.DBX4124.7).

**Note**

The extended PLC user alarms are only available if the compatibility mode was deactivated in the "CPU type" (Page 469)dialog.

The maximum expansion is always assumed in the initial values of DB9913. This means that 2 channels are always preassigned for the channel assignment. For a single-channel system, the preassignments made by the PLC for the second channel are ignored.

This data block has data class "Manufacturer" - and attributes "Read-Only" and "Non-Retain". This means that the initial values for "Power On" or STOP-RUN transition are accepted as actual values. If alarm response, delete criteria and channel assignment are modified, i.e. the initial values have changed, then the PLC rejects a "Loading in RUN mode" and an error message is output. This means that before loading, the PLC is brought into the STOP state. This ensures that the modified alarm response, delete criteria or channel assignment becomes effective.

The value can be directly entered in the initial value in the data block; "Hexadezimal" is the standard format. Alternatively, the values can be selected in the Bit editor (Page 405).

### Supporting CPUs

828D Step 2 (from PLC 07.01)

### See also:

Bit editor (Page 405)

Data blocks (Page 272)

## 10.3 How to work with global symbols and local variables

### 10.3.1 Define global symbols

Open a symbol table by using one of the following methods:

- Click the button of the symbol table  in the "Navigation" toolbar (Page 500).
- Select the menu command **"View > Symbol Table"**.
- Open the directory of the symbol table in the project tree (Page 447) and double-click the  button.

## Editing in a table

### Making symbol assignments in the symbol table

To assign a symbolic name to an address, proceed as follows:

1. Open the symbol table with one of the described entry types.
2. Enter the symbolic name (e.g. Input1) in the "Name" column. The maximum symbol length is 23 characters. Use the Tab, Enter or arrow keys to confirm your work and move to the next field.

   **Note**

   - Until you assign an address to the symbol, it is shown as an undefined symbol (green wavy line). After you complete the Address column assignment, the green wavy line is removed.
   - If you have selected simultaneous symbolic and absolute view of addresses for your project, lengthy symbolic names are truncated with a tilde (~) in the LAD Program Editor window. You can place your mouse pointer over the truncated name to see the entire name displayed in a tooltip.

3. Enter the address (e.g. I0.0) in the "Address" column.
4. Enter a comment (optional: 79 characters maximum).

**Note**

In the Programming Tool, you can create multiple symbol tables. You cannot, however, use any character string more than once as symbolic name - neither within the same table nor distributed over multiple tables.

By contrast, it is permissible to use the same name in various local variable tables.

### Use of quotation marks is not supported

Global symbol names must not be set in double quotation marks. The symbol table interprets them as illegal syntax (the symbol is displayed in red until the quotation marks are removed).

### Inserting additional rows

Proceed as follows to insert additional rows into a symbol table:

- Select the menu command **"Edit > Insert > Row"**. A new row is inserted above the current location of the cursor in the symbol table.
- Right-click a field of the symbol table. In the shortcut menu select the command **"Insert > Row"**. A new row is inserted above the current location of the cursor.
- You can move the cursor to a field in the last row and press the down arrow key in order to insert a row at the bottom of the symbol table.

### Defining symbols

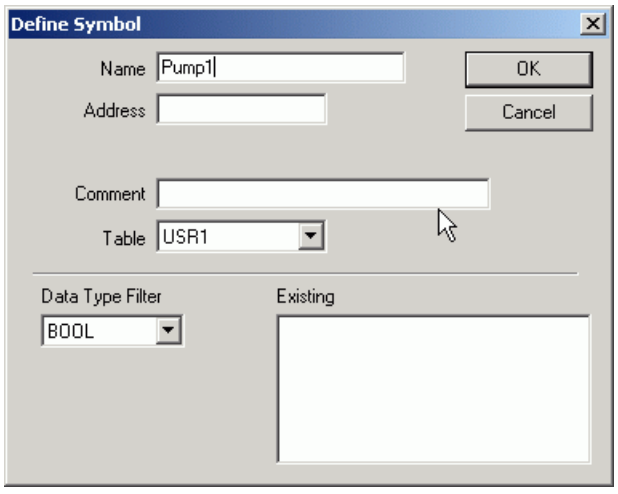
The "Define Symbol" command allows you to select an existing symbol from a list, and it also allows you to create new symbol assignments while you work in the Program Editor window or in a status chart, without having to open up the symbol table.

**Note**

You cannot use the "Define Symbol" command unless you are working in the Symbolic Addressing view. (You can check this in the "View" menu: A check mark next to the "Symbolic Addressing" option means that this view is turned on.)

Using the "Define Symbol" command:

1. Enter at least one character of the desired symbolic name in the Program Editor window or a status chart address field and press the Enter key.
2. Right-click the incomplete (or incorrect) symbolic name, then select "Define Symbol" from the shortcut menu.
3. Choose an existing symbol from the list or define a new symbol in the dialog box "Define Symbol".




4. Click "OK" to confirm your work and close the dialog box or click "Cancel" to cancel.

**Note**

If you have selected simultaneous symbolic and absolute view of addresses for your project, lengthy symbolic names are truncated with a tilde (~) in the LAD Program Editor window. You can place your mouse pointer over the truncated name to see the entire name displayed in a tooltip.

### Sorting by column

You can sort the table by the "Name" or "Address" column. You can sort a column in ascending or descending (alphabetical) order.

To sort a column in a particular order, select the column by clicking in the body. Avoid clicking on the column header ("Name" or "Address").

- To sort the column in ascending order, from A to Z, click the Sort ascending button or choose the menu command **"View > Sort Ascending"**.
- To sort the column in descending order, from Z to A, click the Sort descending button or choose the menu command **"View > Sort Descending"**.

To reverse the sort, click the column header ("Name" or "Address"). When you click the column header, you toggle between ascending and descending sorts.

## Working with multiple tables

### Creating additional symbol tables

By default, the Symbol Table window displays one tab for user-defined symbolic names (USR1). If you have renamed any POUs, the Symbol Table window shows a tab called "POU Symbols".

There are several ways to create additional symbol tables for user-defined symbolic names:

- In the instruction tree, right-click the "Symbol Table" directory and, in the shortcut menu, select the command **Insert Symbol Table**.
- Open the window "Symbol Table" and call the menu "Edit" or right-click and select the command **Insert Contents > Table**.

**Note**

After you have inserted a new symbol table, a new tab 

is displayed at the lower edge of the "Symbol Table" window.

### Moving between symbol tables

If you have created more than one symbol table, you can access the tables by any of the following methods:

- If the Symbol Table window is open, you just need to click the 

tab of the desired symbol table at the bottom of the Symbol Table window to move between tables.
- From the instruction tree, expand the Symbol Table directory icon and double-click the desired symbol table to view the tab for that table within the Symbol Table window.

### "POU Symbols" tab

If you have assigned symbolic names for any of the POUs or other components (such as a status chart, data block or symbol table) in your project, the symbolic name assignments are listed on the "POU Symbols" tab of the Symbol Table window.

This tab is read-only; you cannot edit the assignments from here.

If you want to change an assignment, you must edit the Properties dialog box for the component in question. From the Project branch of the instruction tree, right-click the component to bring up the dialog box "Properties".

All symbolic assignments must use valid syntax. If you violate the guidelines for making a symbolic name assignment, the Programming Tool reports an error when you try to compile your program.

## Symbolic and absolute addressing

### Toggling between symbolic and absolute addressing mode

After you make symbol and absolute address associations in a symbol table, you can toggle the display between symbolic and absolute addressing. To do this, proceed in one of the following ways:

- Select the menu command **"View > Symbolic Addressing"** to toggle symbolic addressing on and off
- Use the **<Ctrl+Y>** shortcut key to toggle symbolic addressing on and off

A check mark in front of the command "Symbolic Addressing" means that symbolic addressing is on. By default, symbolic addressing is on when you create the first project.

### Viewing symbolic and absolute addresses simultaneously

To view symbolic addresses and absolute addresses simultaneously in an LAD program, use the menu command **"Tools > Options"** and open the tab "LAD Editing". Select the option button "Display symbol and address".

**Note**

Symbolic addresses are only displayed in your project when the Symbolic Addresses view is on. Otherwise, even if you select "Display symbol and address", only the absolute address is shown.

If you have selected simultaneous symbolic and absolute view of addresses for your project, lengthy symbolic names are truncated with a tilde (~) in the LAD Program Editor window. You can place your mouse pointer over the truncated name to see the entire name displayed in a tooltip.

## Symbols

### Understanding the symbol scope

In the symbol table you can assign symbolic names to the addresses of inputs and outputs and to addresses in the automation system memory. When you define a symbol in the symbol table, the symbol has a global scope. This means you can use the name of the symbol in any POU as a reference to the data at the address of that symbol. By contrast, if you assign a symbolic name using a local variable table, the local variable is restricted in scope to the POU where it was defined.

### Memory types

You can create symbolic names for the following memory areas: I, Q, M, SM, V, S, C, T.

### Entry errors

- Illegal syntax - red text
  Example:

| | Name | Address |
|---|---|---|
| 1 | 2name | I0.0 |
| 2 | | VBB0 |

The first character cannot be a digit.
Invalid address, VB14000000 would be correct
Reserved keywords cannot be used as symbols

- Illegal use - red wavy line
  Example:

| | Name | Address |
|---|---|---|
| 1 | Pump1 | I0.0 |
| 2 | Pump1 | I0.0 |

Duplicate name
Duplicate address

- Undefined symbol - green wavy line
  Example:

| | Name | Adresse |
|---|---|---|
| 1 | Pumpe1 | |
| 2 | Pumpe2 | VB14000000 |

Name with invalid address

### Note

- Symbolic names are permitted to contain alphanumeric characters and underscores. They are also permitted to contain extended characters (ASCII 128 to ASCII 255). The first character is restricted to alpha and extended characters only.
- When naming global symbols it is not permissible to use double quotation marks. The symbol table interprets them as illegal syntax (the symbol is displayed in red until the quotation marks are removed).
- It is illegal to use keywords (Page 281) as symbolic names, or to use names that begin with a number or contain characters that are not alphanumeric or in the extended character set.
- The maximum permissible length for a symbolic name is 23 characters.
- If you attempt to create a symbol assignment to correct the "Undefined Symbol" error in your program, but do not press the TAB key, ENTER key or an ARROW key after typing the Address value and before you return to your program, the error is still displayed. To resolve this problem, return to the Symbol Table window, place the cursor in the Address field, and press the TAB, ENTER or an ARROW key to complete the address assignment.

### Tips for creating symbols

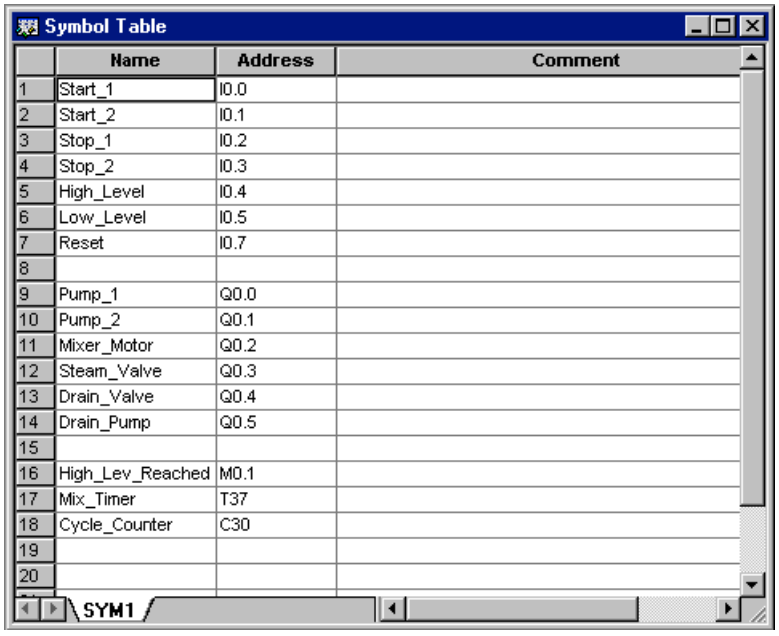
- You can make symbol assignments at any time: before, during or after the creation of program logic.
- You can document your symbols by making remarks in the "Comments" column.
- You can use the command "Define Symbol" to select from a list or create new symbolic names as you program.

- You can sort the values in the symbol table by clicking on the various table column headers.
- You can resize columns by dragging on their edges.
- You can use the menu command File > Print to print the symbol table.
- Once you have created symbolic assignments, you can display your address assignments and symbolic comments for each network in the LAD program editor by enabling the function "Symbol Information Table".

**Examples of tables**

Instruction operands are defined to accept one specific data type. For information about data types, see Data Types (Page 269).

This is an example of a symbol table:




## See also:

Local variable table (Page 229)

Symbolic addressing (Page 444)

Symbol information tables (LAD) (Page 445)

Overview of the addressing (Page 26)

Direct addressing (Page 261)

DB address representation (Page 446)

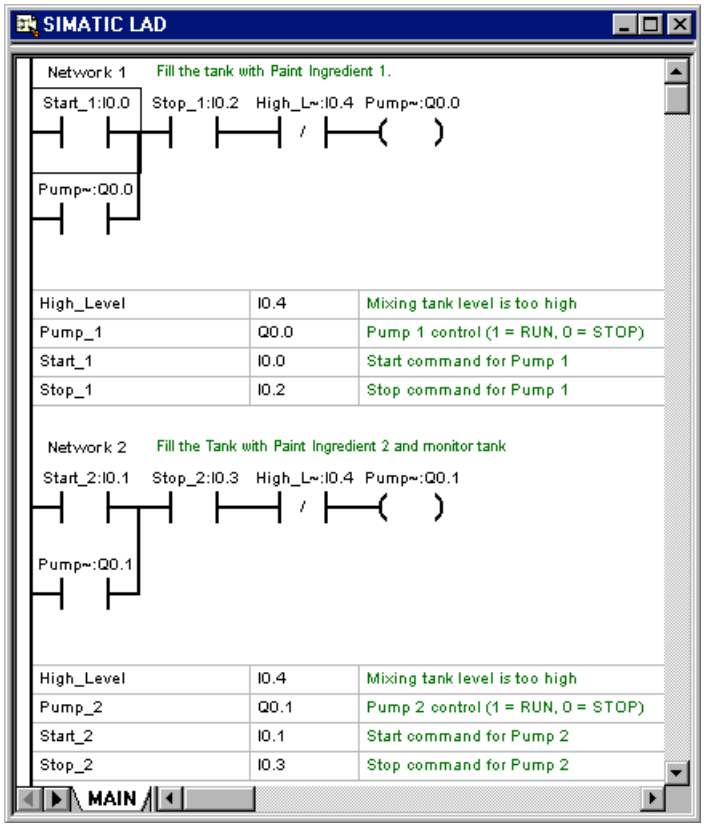
Keywords (Page 281)

## 10.3.2 Display the network symbol information tables (LAD editors)

Use one of the following methods to view or hide symbol information tables in LAD:

- Click the button in the toolbar.
- Select the menu command **"View > Symbol Information Table"**.
- Press the **<Ctrl+T>** shortcut key.

When you view symbol information tables, the symbolic names, absolute addresses, and comments are displayed below each network in your LAD program. The table lists information for any symbols in that network; networks that do not contain global symbols do not display a symbol information table.




**Tips:**

- Use the arrow keys to scroll through the symbol information table.
- You can resize the table by dragging the splitter cursor from the column edges.
- Use the button "Options" in the dialog box "Print" to enable printing of symbol information tables when you print your program.
- To conserve vertical space on the screen and view more networks of your program, you can toggle the symbol information tables on and off by using the menu command **"View > Symbol Information Table"**.

**Troubleshooting**

If you do not see symbol information tables beneath the networks in your program, consider these points:

1. Are symbol information tables enabled? Check this in the menu "View": The menu command "Symbol Information Table" should have a check mark beside it.
2. Are symbols (or the absolute addresses that correspond to the symbols) used in the network? Networks that do not contain symbolic names do not display a symbol information table.
3. Have you entered the symbol in the symbol table? No address or comment information is available for a symbol until it is fully defined.

**See also:**

Symbol table (Page 413)

"LAD Editing" tab (Tools > Options) (Page 490) (Here you can specify how the symbol information table is displayed when a program is processed.)

"LAD Program Status" tab (Tools > Options) (Page 492) (Here you can specify how the symbol information table is displayed when you display the program status.)

Information on using the software (Page 375)

First steps: Contents (Page 17)

## 10.3.3 Define local variables

### Local variables

Each POU in your program has its own local variable table, with 60 bytes of local data memory. These local variable tables allow you to define variables that are restricted in scope: a local variable is only valid inside the POU where it was created. By contrast, global symbols, which are valid in every POU, can only be defined in the symbol table. In cases where you use the same symbolic name (e.g. INPUT1) for a global symbol and a local variable, the local definition takes precedence inside the POU where the local variable has been defined and the global definition is used in the other POUs.

You assign a declaration type (TEMP, IN, IN_OUT or OUT) and a data type (see Data types (Page 209)), but not a memory address, when you define a variable in a local variable table; the program editor automatically assigns addresses in the local data memory for all local variables.

These help topics are explained in the following:

## Declaration types for local variables

The type of local variable assignment you can make depends on the POU where you are defining the variable. The main program (OB1), interrupt programs, and subprograms can use temporary (TEMP) variables. Temporary variables are only available while the block is being executed. These addresses are then free to be overwritten again once the block is completed. Subprograms can also use call parameters (IN, IN_OUT, OUT).

| Type of declaration | Description |
|---|---|
| IN | The input parameter that is provided by the calling POU. |
| OUT | The output parameter that is returned by the calling POU. |
| IN_OUT | Parameter, the value of which is provided by the calling POU, is modified by the subprogram, and is then returned to the calling POU. |
| TEMP | Temporary variable that is briefly stored in the local data stack. Once the POU has been fully processed, the value of the temporary variable is no longer available. Temporary variables do not buffer the value following processing of the POU. |

## Data type checking for local variables

When you pass local variables to a subprogram as parameters, the data type that you have assigned in the local variable table of that subroutine must match the data type of the value in the calling POU.

**Example:**

1. You call SBR0 from OB1, using a global symbol called INPUT1 as an input parameter of the subprogram.
2. Inside the local variable table of SBR0, you have defined a local variable called FIRST as an input parameter.
3. When OB1 calls the SBR0 subprogram, the value of INPUT1 is passed to FIRST.
4. The data types of INPUT1 and FIRST must match.
5. If INPUT1 is REAL and FIRST is REAL, the data types match. If INPUT1 is REAL but FIRST is INT, the data types do not match and the program cannot be compiled properly until this error is corrected.

## Does your program need to use local variables?

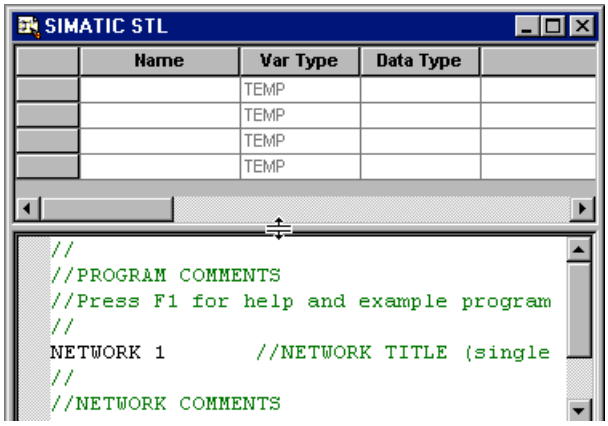
There are two reasons to use a local variable:

- You want to create portable subprograms that do not make references to absolute addresses or global symbols.
- You want to use temporary variables (local variables declared as TEMP) to perform calculations in order to free up PLC memory.

If these descriptions do not fit your situation, you do not need to use local variables; in that case, you only need to define global symbols in the symbol table.

## Show/hide the local variable table

If you drag the horizontal splitter cursor up to the top of the Program Editor window, the local variable table is no longer visible, but it is still there. Drag the splitter cursor down again to make the local variable table visible again.




## How to enter local variables in the local variable table

**Note**

It is most efficient to make assignments in the local variable table before using local variables in your program. When you use symbolic names in your program, the program editor checks first the local variable table of the appropriate POU, then the symbol table.

If the symbolic name has not been defined in either of the two tables, then the program editor treats the name as a global symbol. The program editor underlines the symbol using a green wavy line: Undefined_Local_Var. The Program Editor does not read the table again if you change the local variable table at a later time. In such a case, in order to be able to use the symbolic name as a local variable, you must manually insert a hash character # in your program in front of the name: #Undefined_Local_Var.

### How to enter the first local variable

To make an assignment in a local variable table, follow the procedure below:

1. Ensure that the correct program organization unit (POU) is displayed in the Program Editor window by double-clicking, if necessary, on the symbol of the POU in the project tree. (Since every POU has its own local variable table, you need to make sure that you are making assignments in the correct variable table.)
2. If the local variable table is hidden, drag down the splitter cursor to display it. (See: How to hide/show the local variable table.)

3. If you are defining an assignment for OB1 or an interrupt program, the local variable table contains only variables of the type TEMP. If you are defining an assignment for a subprogram, the local variable table contains IN, IN_OUT, OUT, and TEMP variables.
   Choose a row that has the right declaration type for the kind of variable that you want to define and type a name for the variable in the "Name" field. You do not need to preface the name with a hash character # in the local variable table. Hash characters # are only used to precede local variables in the program code.

   **Note**

   - Local names are permitted to contain a maximum of 23 alphanumeric characters and underscores. They are also permitted to contain extended characters (ASCII 128 to ASCII 255). The first character is restricted to alpha and extended characters only. It is illegal to use keywords (Page 281) as symbolic names, or to use names that begin with a number or contain characters that are not alphanumeric or in the extended character set.
   - Local variable table variable names are downloaded to and stored in the CPU memory. The use of longer variable names may reduce the memory available to store your program.

4. Click the mouse pointer in the "Data type" field and use the list box to select an appropriate data type for the local variable.

   **Note**

   When you assign local variables as parameters for subprograms, you must ensure that the data type that you assign to the local variable does not conflict with the operand being used in the subprogram call. (See: Data type check.)

After you supply a value for the "Name" and "Data type" fields, the program editor automatically assigns an address in the local data memory to the local variable.

**How to enter additional local variables in the local variable table**

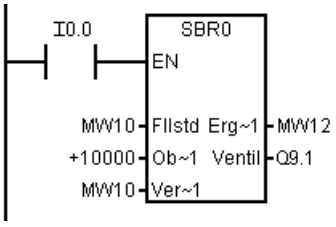
For OB1 and interrupt programs, the local variable table displays a set of rows predefined for TEMP variables. This is the only declaration type that you can use in OB1 and interrupt programs. To add more rows to the table, simply click a field in the last row and use the ENTER key to move through that row and down - a new row is automatically generated.

For subprograms, the local variable tables display a set of rows that are predefined with declaration types in this sequence: IN, IN_OUT, OUT, and TEMP. You cannot change this sequence. Your local variables must follow the same order in the table that the corresponding operands will use when you make operand assignments to the call instruction for the subprogram. If you want to add additional local variables, you must right-click an existing row and use the shortcut menu to insert another new variable of the same declaration type. Choose **Insert > Row** to insert the new row above the selected row, or **Insert > Below Row** to insert the new row below the selected row.

### Local variable table example

SIMATIC LAD

| | Name | Var Type | Data Type | Comment |
|---|---|---|---|---|
| | EN | IN | BOOL | |
| LW0 | Lvl | IN | INT | Tank transmitter |
| LW2 | HSet | IN | INT | High setpoint |
| LW4 | Offset | IN | INT | Offset |
| | | IN | | |
| | | IN_OUT | | |
| LW6 | Reslt | OUT | INT | Result value |
| L8.0 | Valve | OUT | BOOL | Control valve |
| | | OUT | | |
| | | TEMP | | |




**See also:**

Subprogram (Page 223)

Data types (Page 269)

Keywords (Page 281)

## 10.4 Set up the PLC

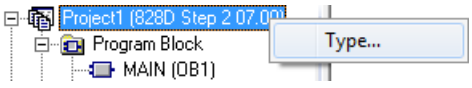
### 10.4.1 Read information about the PLC

To view PLC information such as CPU model and version numbers, mode, and scan rate, select the menu command **PLC > Information**.

The **Reset scan rates** button allows you to refresh the times in the three Scan rate fields.

### 10.4.2 Select a target CPU (model and version)

In order to call up the "CPU Type" dialog, select the menu command **"PLC > Type"**, or open the shortcut menu of the project name:



When specifying the CPU type, the PLC Programming Tool can check instructions and parameters, which helps prevent errors as you create your program. If you have specified a CPU type for your project, the instruction tree shows any instructions that are not valid for your PLC with a red x: .

If you open a new project and begin editing a program without having specified the CPU type, you can program instructions, addresses and functions for the PLC system in the PLC Programming Tool that are not supported by all CPUs. If you use instructions, addresses or functions in your project that are not supported by your target CPU, when you try to download the project, it will be rejected by the PLC.

### Specifying a CPU type selection

You can select a PLC by picking a model from the drop-down list box in the dialog box "CPU Type". If you are working, for example, with a "828D Step 2", the list box displays the following:


In addition to the CPU type, the firmware version is displayed in the list box.

When you specify the CPU type, the range and function restrictions for the latest firmware release of the "828D Step 2" are considered. However, you may have an early CPU firmware release that does not support functions available in the latest CPU firmware release. In order to ensure that both the CPU type and the version of the firmware are taken into consideration during the range checks, you can have the PLC Programming Tool read the CPU type information directly from the PLC.

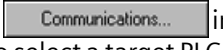
**Note**

You do not need to configure communications when you are specifying the CPU type selection: You only need to have communications established if you want the PLC Programming Tool to read the remote CPU type for you.

### Read-out of the removed CPU type by the PLC Programming Tool

In order to read the CPU type and the product version of the firmware, click the button in the "CPU Type" dialog.


**Note**

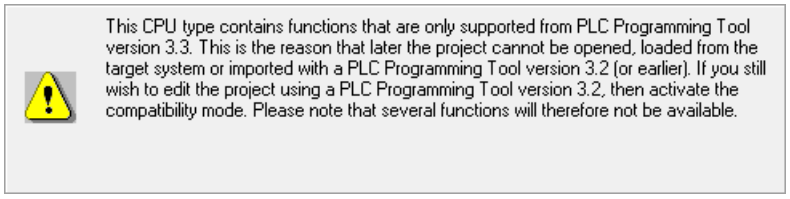
You must already have established communications successfully in order to read the CPU type and firmware information. You can use the button 

in the dialog box "CPU Type" either to establish communication with a PLC or to select a target PLC when you have more than one PLC connected to a communication network.

### Compatibility

Some CPU types include new functions that are only supported with the PLC Programming Tool Version 3.3 or higher. Projects in which these CPU types are selected can only be opened with a PLC Programming Tool Version 3.3.

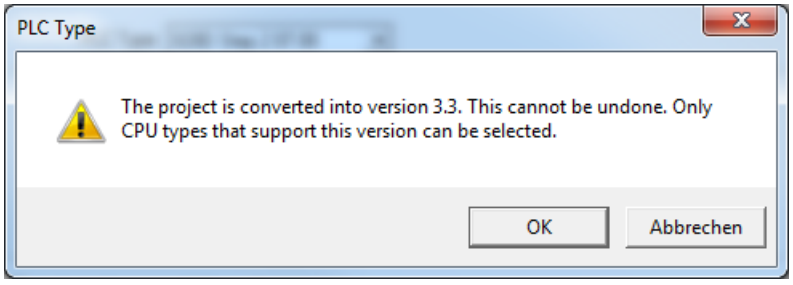
If you have selected such a CPU, a corresponding warning appears in the dialog:



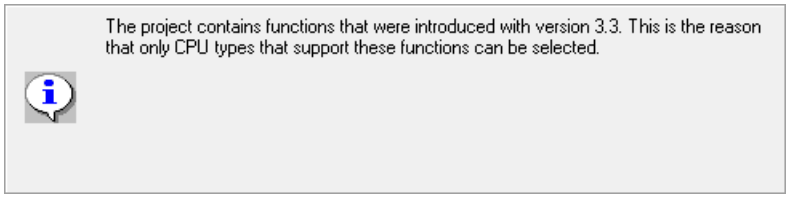

For these CPU types, the compatibility mode can be activated when the selection is made. The project can thus continue to be opened using the PLC Programming Tool Version 3.2 (or earlier).


Some functions are not supported in compatibility mode.

If you accept the CPU type without activating compatibility mode, the project will be converted into Version 3.3. This cannot be undone. A corresponding warning follows.




If you created a project with a CPU type that supports the new functions, but have not activated the compatibility mode, it will only be possible in this project to choose between CPU types that support these functions. In such a case, a corresponding note will be displayed in the dialog.




**See also:**

Communication (Page 443)

Compiling (Page 459)

Memory areas of the automation system (Page 262)

Supported instructions (Page 259)

## 10.4.3 Project comparison

### General information on the block comparison

The comparison between two PLC user projects can be performed in two ways. You can compare program blocks (POUs) and user data blocks (actual values) of the current project either

- with the PLC project in a controller (online), or
- with the blocks of another project (offline).

In this manner, the block information of the projects is directly compared.

Two PLC user projects may be compared with one another for the following reasons:

- For putting a machine/series of machines into operation, it is necessary to determine the differences in comparison with an earlier version of the PLC project.
- Whether or not the PLC project was modified in the machine should be tested during the service call.
- The machine dealer checks whether the PLC project in the supplied machine corresponds to the project that he wants to deliver to his customer.

The comparison always provides the same results, independent of the PLC Programming Tool version used for creating the respective PLC user project.

When comparing two projects, the CPU types for these projects must be consistent. If the CPU types do not concur across both projects, it will be inquired whether the CPU type of the comparison project should be adopted for the current project.

The project comparison dialog opens when the comparison is started. In this dialog you select the data classes whose blocks you want to compare. You can compare program blocks and data blocks separately. Following selection of the data classes, the program and user data blocks to be compared can be selected from the block lists. For the user data blocks, the comparison of the actual values can also be activated. Other project components such as symbol tables cannot be compared.

The results of the comparison are displayed in the output window. All compared blocks are listed. Differences between the blocks are shown below the block name. Here, the position specifications refer to the current project. For program blocks, the position data is provided via the network number and, if available, row number and column number. For data blocks, the position data is provided via the row numbers. At the end of the list of the program or data blocks, information on how many blocks were compared and how many differences were found is provided. If you double-click on the line of a difference in the output window, the corresponding position in the editor of the current project will open.

**Special features of the comparison**

- Empty networks in the program blocks or empty rows in the data blocks are ignored during the comparison.
- If too many differences are found in a network or a row, these are not all listed, and a message indicating that the network or the row is different is displayed.
- Block properties are factored in.
- Different names and data classes are captured.

- With regard to the data blocks, differences in the "non-retain" and "write-protected" properties are also covered.
- Creation date and date of last change are not compared.
- Differences in block properties have no position data. For this reason, no reaction occurs to a double-click on the row with the difference in the output window.

### Online comparison via PLC > Compare...

1. Start the online comparison via the menu **"PLC > Compare..."**.

   **Note**

   If a connection to the address set under "Communication" cannot be set up, a corresponding error message will be displayed.

   If the CPU type of the open project does not concur with that of the remote PLC, a query is displayed as to whether the CPU type of the remote PLC should be adopted for the opened project.

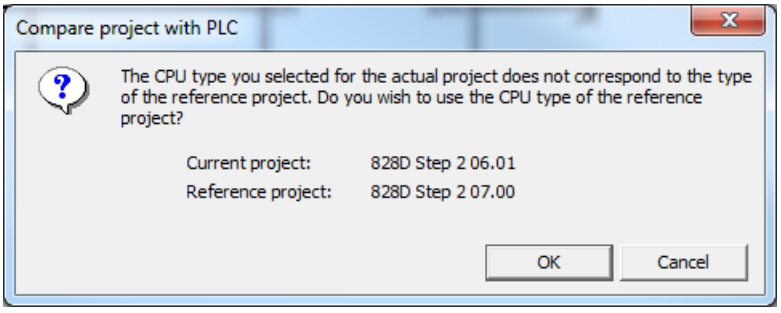



Figure 10-16 Taking over the CPU type

   In this dialog, the address and type of the remote PLC are displayed. If the address does not point to the desired PLC, click on the **"Cancel"** button to cancel the comparison; you can then modify the address under **"View> Communication"**.

2. Click on the **"OK"** button to import the CPU type and to start the comparison dialog.
   The **"Compare project with PLC"** dialog opens.

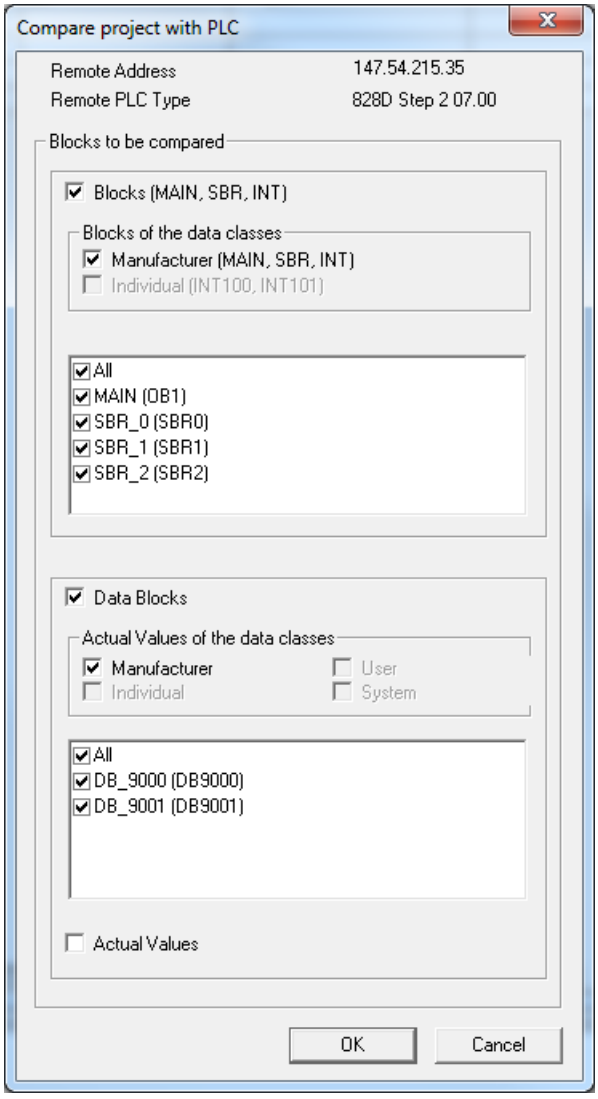



Figure 10-17 "PLC > Compare dialog"

3. In this comparison dialog, you can select the blocks that are to be compared.
   Program and data blocks are distinguished. A pre-selection can be made via the selection of data classes. By default, all blocks are selected.
4. Click on the **"OK"** button to start the comparison.

5. The result of the comparison is displayed in the output window. In this manner, all blocks compared are listed, even if there are no differences. Differences are displayed below the respective block name.

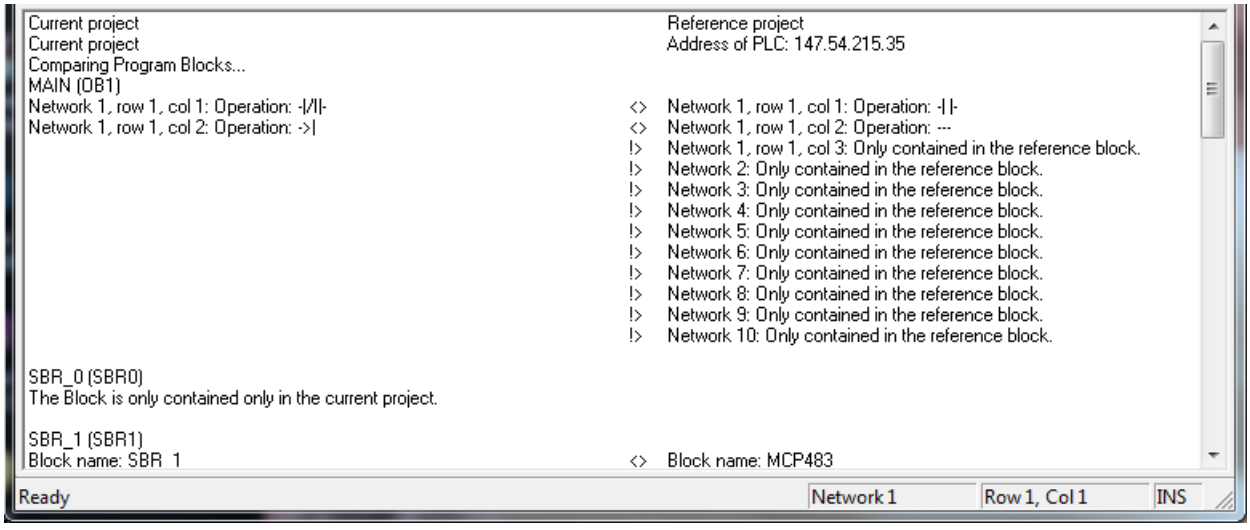
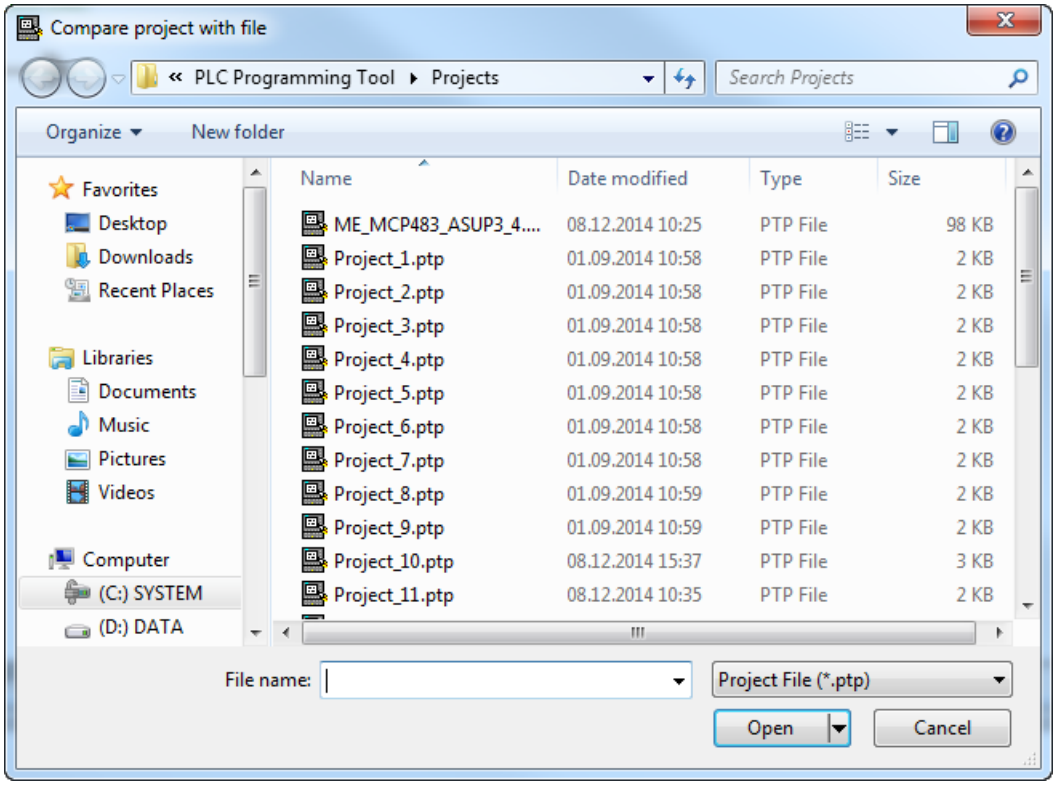



Figure 10-18 Comparison result in the output window

6. Double-click on a row with a difference in the output window to open the relevant row in the editor of the current project.

### Offline comparison via File > Compare...

1. Start the offline comparison via the **"File > Compare..."** menu.
   The **"Compare project with file"** dialog is opened.




2. In this dialog, select the project that you want to compare with the project that is already open.
3. Click on the **"Open"** button.
   If the CPU type of the current project does not concur with that of the comparison project, the question as to whether the CPU type of the comparison project should be taken over for the current project follows.

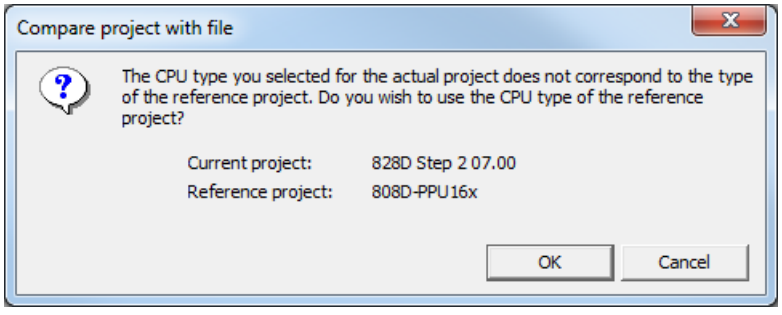



Figure 10-19 Taking over the CPU type

4. Click on the **"OK"** button to take over the CPU type and to start the comparison dialog.
   In the dialog that now opens, the name including path of the file to be compared is displayed.

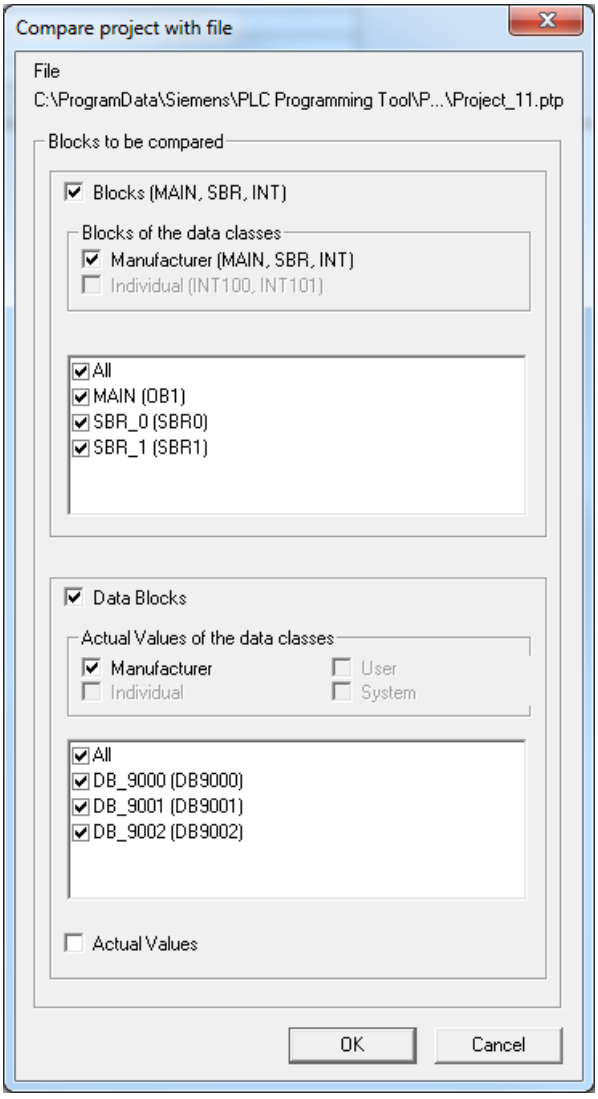



Figure 10-20 "File > Compare" dialog

5. In this comparison dialog, you can select the blocks that are to be compared.
   Program and data blocks are distinguished. A pre-selection can be made via the selection of data classes. By default, all blocks are selected.

6. Click on the **"OK"** button to start the comparison.
   The result of the comparison is displayed in the output window. All blocks compared are listed, even if no differences are present. Differences are displayed below the respective block name.

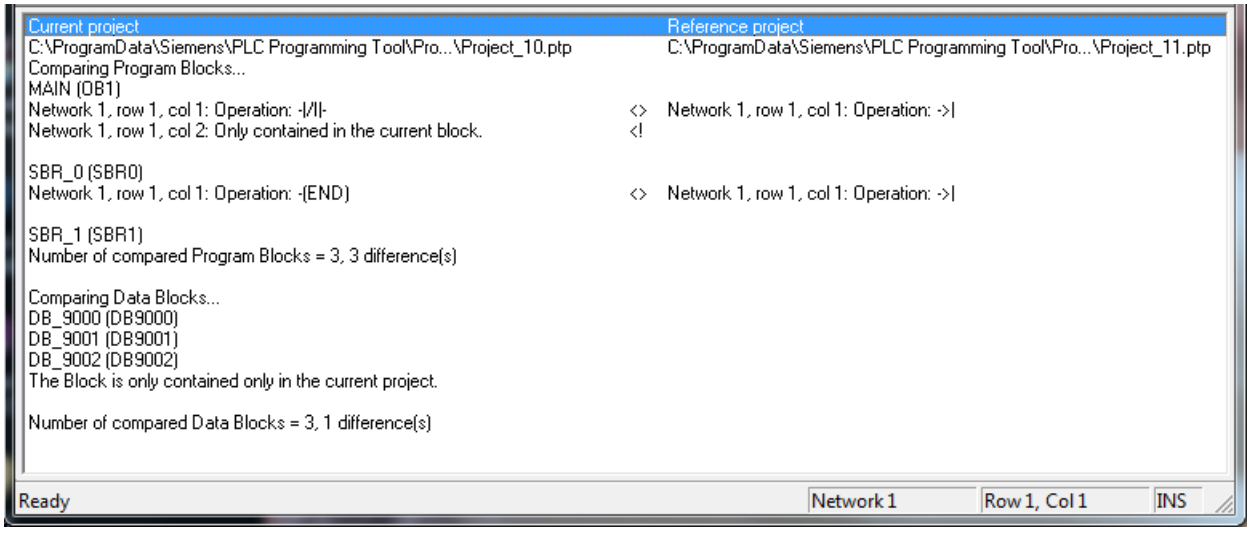



Figure 10-21 Comparison result in the output window

7. Double-click on a row with a difference in the output window to open the relevant row in the editor of the current project.

## Saving the comparison results

The comparison results can be saved and thus also be printed.

1. Right-click in the output window.

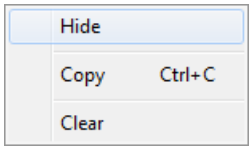



Figure 10-22 Shortcut menu of the output window

2. Select the "Copy" option.
   Alternatively, you can also click on the output window with the left mouse button and press "Ctrl + C".
   In this way, the entire content of the output window is copied as text to the clipboard.
3. Now insert the text in any text editor, text editing program, or in Microsoft Excel.
   The results can then be printed out from these programs.

## See also:

Information on using the software (Page 375)

First steps: Contents (Page 17)

Output window (View menu) (Page 451)

# 10.5 Saving retentive data (DB1400)

## System data block

In the commissioning archive or data class archives, the actual values from the DB1400 are saved in the "Individual" data class. The DB 1400 is a data block of the user interface and contains the user-specific retentive data. The data blocks of the PLC user interface belong to the "System" data class and are not, in contrast to the user data blocks, stored in the PLC user project. Hence, the actual data block values of the PLC user interface can also not be stored in the PLC user project.

It is necessary to back up the retentive data if, for example, a software upgrade is implemented on the existing equipment and the actual status of the tool management is to be maintained.

To store the actual data block values of the PLC user interface in the PLC user project nevertheless, a system data block is used. The PLC creates this system data block in order to back up the retentive data. The system data block can "view" the actual DB1400 values and is assigned to the "Individual" data class. If the system data block is uploaded from the PLC, the actual DB1400 values are automatically read. If the system data block is once again downloaded to the PLC, the data are automatically rewritten to the DB1400.

If the system data block is uploaded from the PLC, the actual DB1400 values are automatically read. If the system data block is once again downloaded to the PLC, the data are automatically rewritten to the DB1400.

## CPUs that support system data blocks

828D
828D Step 2

## Saving retentive data

If the CPU employed supports this function, the selection of the retentive data and their data classes is displayed in the **"Download to CPU"**, **"Upload from CPU"**, **"Export"** and **"Import"** dialogs. By default, the selection of the retentive data is activated for "Upload from CPU" and "Import". By default, the function is deactivated for "Download to CPU" and "Export". If the system data block is not present, the selection is disabled.

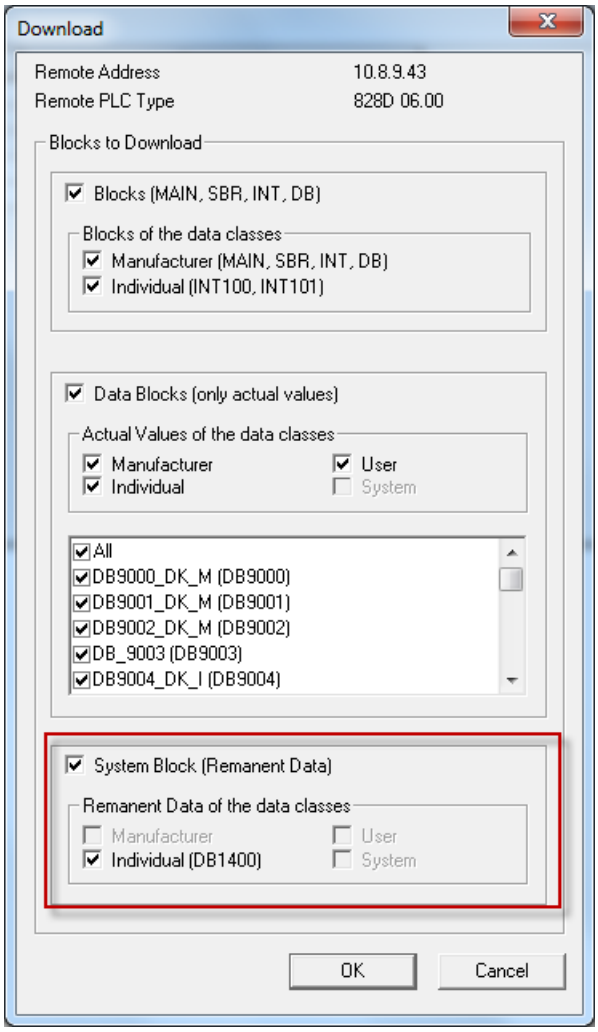



Figure 10-23 Downloading to CPU

The retentive data is backed up in the **"Individual"** data class in the data class archive. Data class archives can created or read by both the HMI operate and the PLC Programming Tool.

The system data block cannot be opened in the PLC Programming Tool and, in consequence, its content cannot be viewed. If the retentive data was selected in the dialog, this system block is stored in the PLC user project following "Upload from CPU" or "Import".

If the system data block is present and if the retentive data was selected in the dialog, the system data block will be read from the PLC user project with "Download to CPU" or "Export". The system data block is then downloaded to the PLC or is written to the archive file.

During storage, the system data block is saved with the retentive data in the PLC user project. If a system data block is present, it will be displayed in the instruction tree. Using the shortcut menu (right mouse button), the retentive data can be deleted or the Properties dialog can be opened.

Saving is possible both in the HMI operate and with the PLC Programming Tool.

**Note**

**Use of data class archives in the HMI**

It is also possible to create and view data class archives in the HMI.

## User interface

In contrast to the HMI operate, the data for these data class archives are not read directly from the PLC, but rather, the PLC user project is exported. These data must be selected when uploading from the PLC, e.g. via the **File> Upload from CPU...** menu, so that the PLC user project contains all of the desired data. The loading of the retentive DB1400 data is a separate selection point.

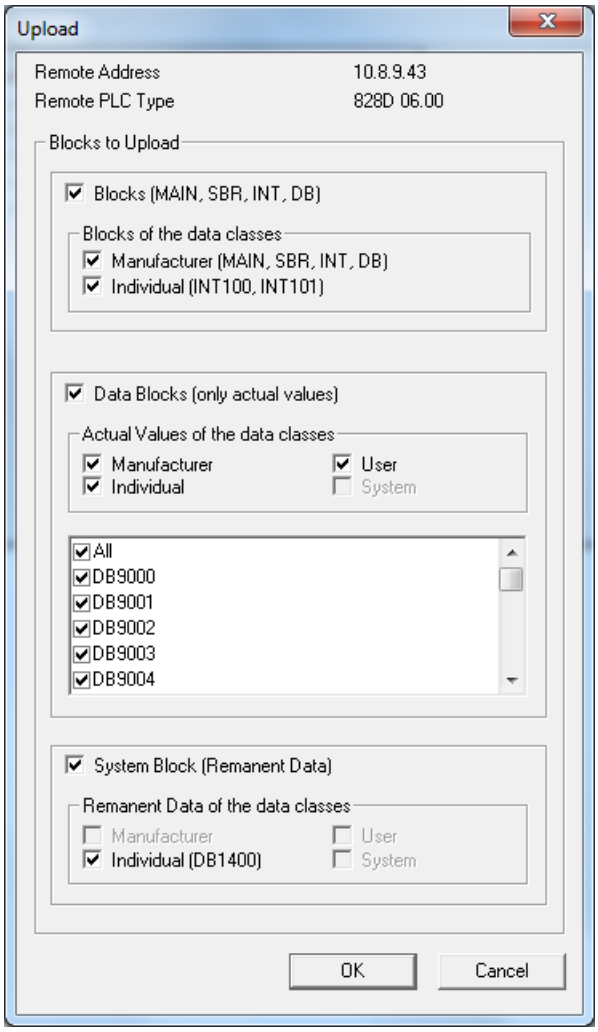



Figure 10-24 PLC Programming Tool Upload from CPU

For the creation of a data class archive, the PLC user project must be exported via the **File > Export...** menu. The exporting of the retentive DB1400 data is also a separate selection point.

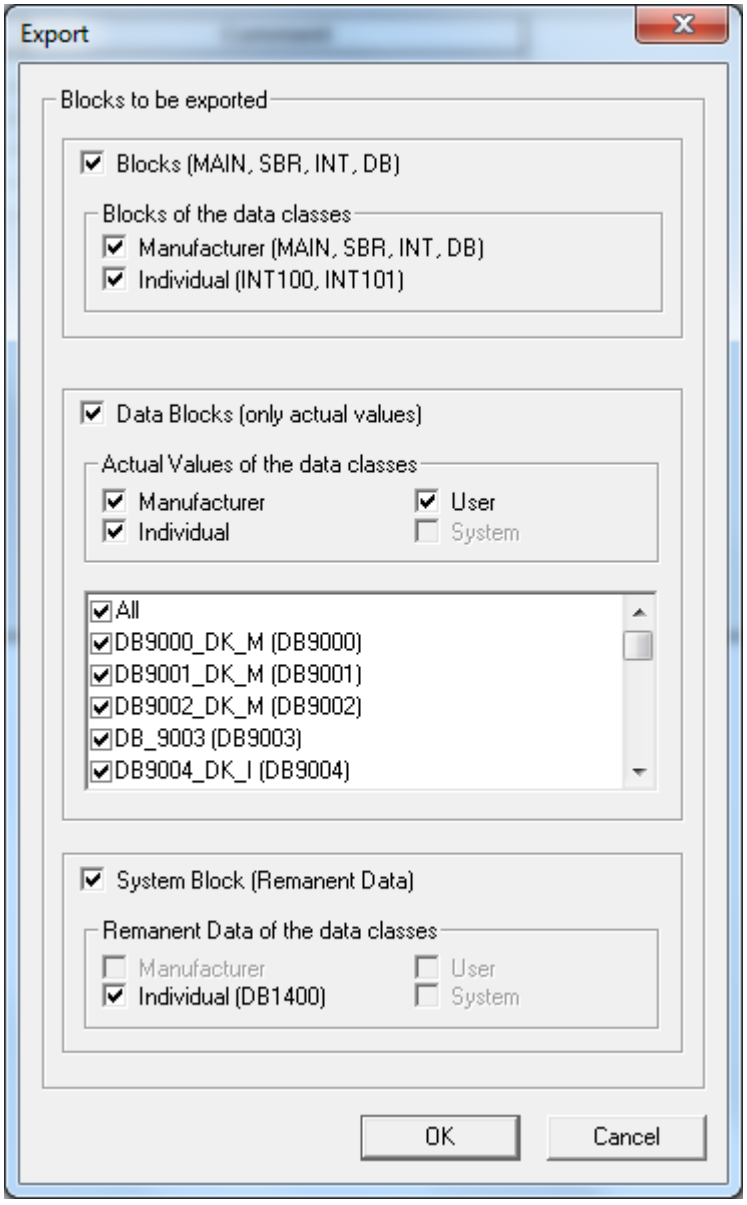



Figure 10-25 PLC Programming Tool Export

With the PLC Programming Tool, data class archives can also be imported. During import, a new PLC user project is created. The data is not directly written to the PLC. To upload the data from the archive to the PLC, the project has to be uploaded to the PLC following import. The data must be selected upon import so that the PLC user project includes all of the desired data. You open the Import dialog via the **File > Import...** menu. The import of the retentive DB1400 data is a separate selection point.

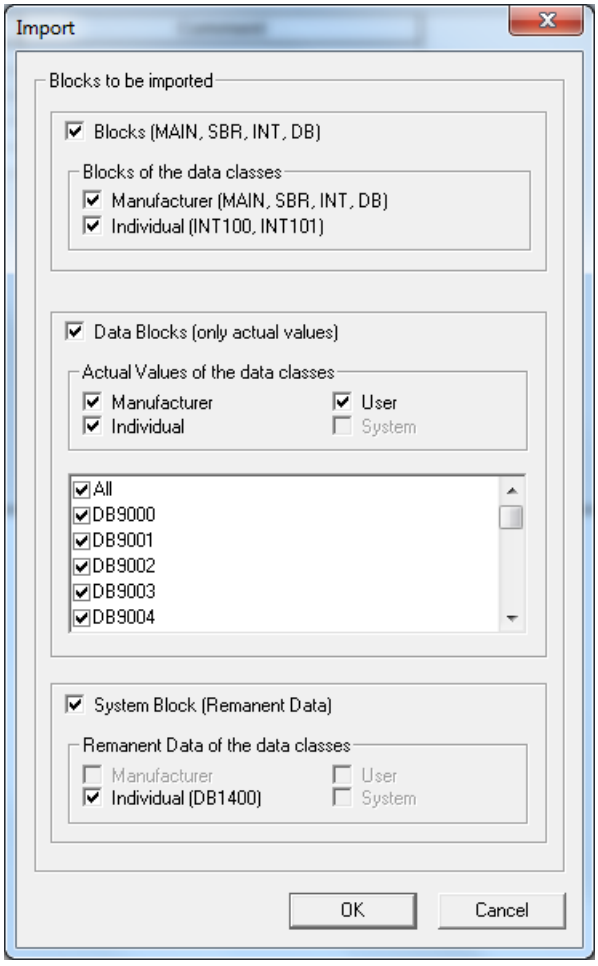



Figure 10-26 PLC Programming Tool Import

Data class archives that were created with the HMI operate can also be read in the PLC Programming Tool. Only the "PLC" component is taken into account.

In order that the data from the archive be active in the PLC, it is necessary to download the imported PLC user project to via the **File > Download to CPU...** menu. The loading of the retentive DB1400 data is a separate selection point.

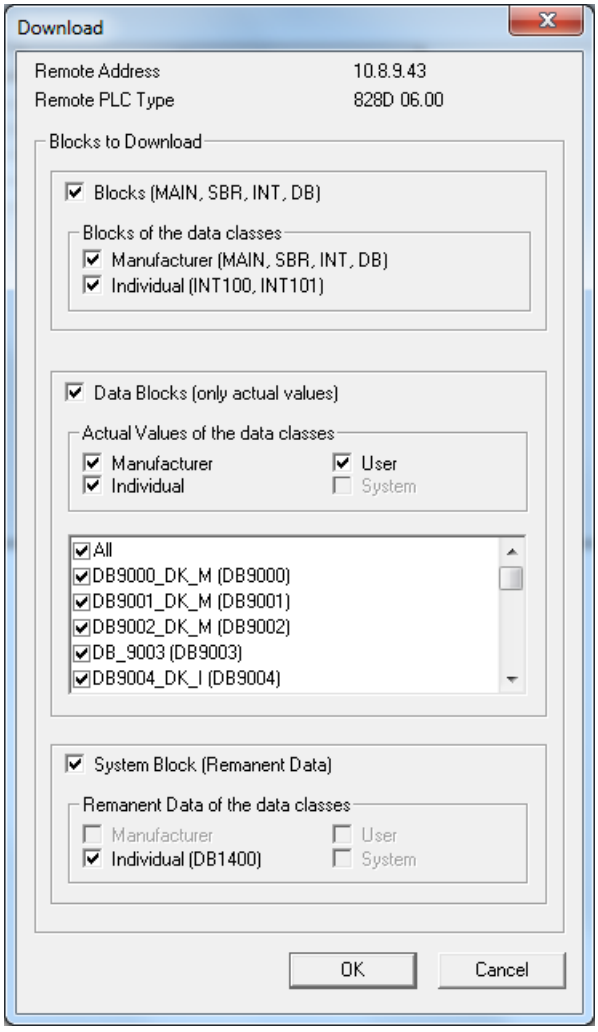



Figure 10-27 PLC Programming Tool Download to CPU

The system data block with the retentive data is stored in the PLC user project. It is displayed in the project tree. Using the shortcut menu (right mouse button), the retentive data can be deleted or the Properties dialog can be opened. The data itself cannot be displayed or edited.

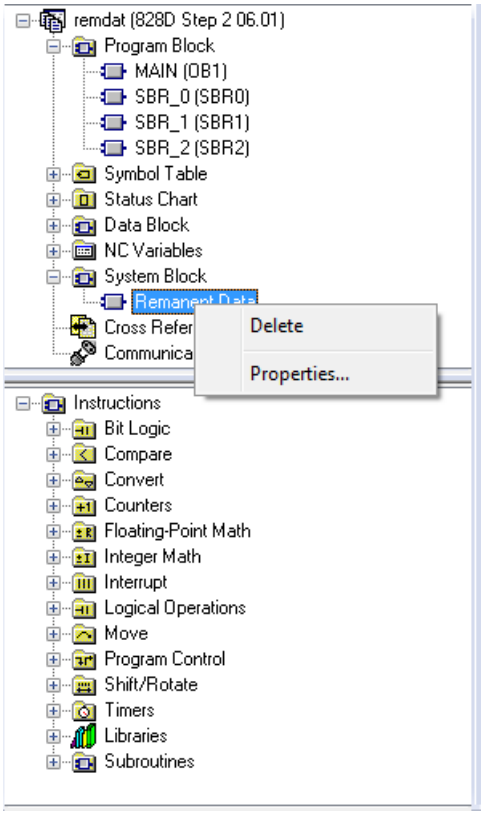



Figure 10-28 Instruction tree

In the Properties dialog, the creation date shows when the retentive data were backed up.

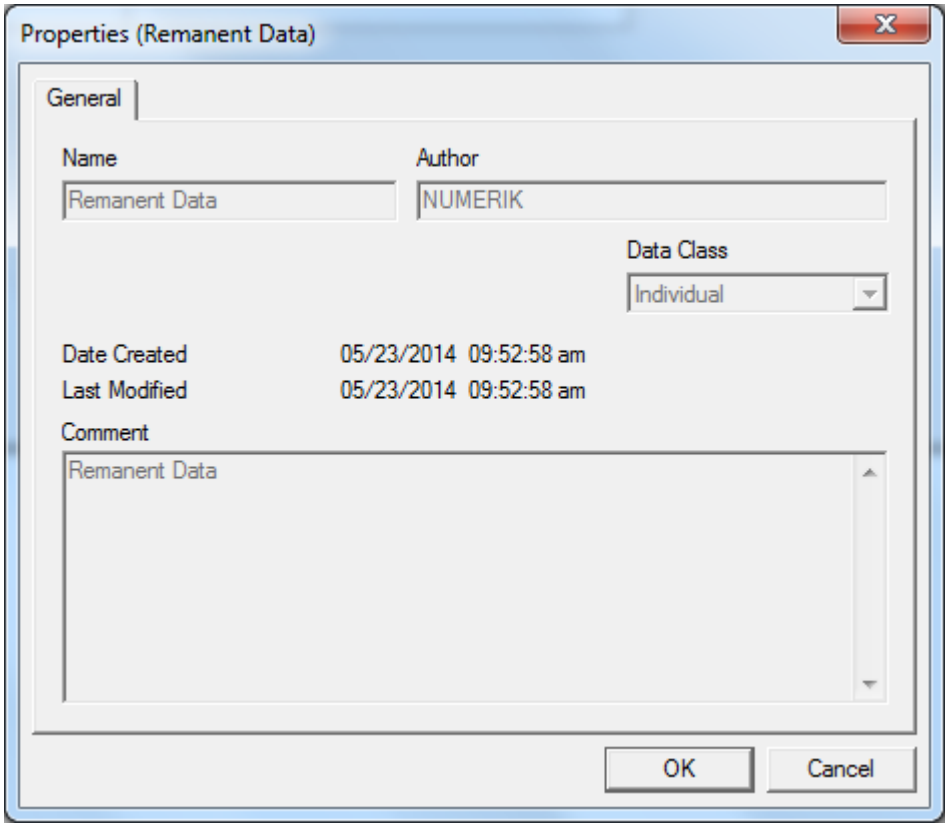



Figure 10-29 Properties dialog (retentive data)

**See also:**

Import (File menu) (Page 380)

Export (File menu) (Page 381)

Uploading from the CPU (File menu) (Page 382)

Downloading to the CPU (File menu) (Page 382)

## 10.6 Load your program into and from the PLC

### 10.6.1 Upload

Use one of the following methods to upload project components from the PLC to a Programming Tool program editor:

- Click the button.
- Select the menu command **File > Upload**.
- Press the **Ctrl+U** shortcut key.

To upload (PLC to editor), PLC communication must be operating properly. Make sure your network hardware and PLC connecting cable are working.

Select the blocks you want to upload (program block, data block, system data block). The program components you select for uploading are copied from the PLC to the currently open project. You can then save the program.

**Note**

**Saving retentive data**

The user-specific retentive data must be selected to be able to save them in the user project.

You can find information on this in the Chapter "Saving retentive data" (Page 345).

**See also:**

Uploading from a PLC (GS 6.3) (Page 78)

Communications overview (GS 4.1) (Page 50)

Testing the communications network (GS 4.2) (Page 51)

Downloading a program to the CPU (GS 4.3) (Page 52)

Downloading to the PLC (File > Download to CPU) (Page 382)

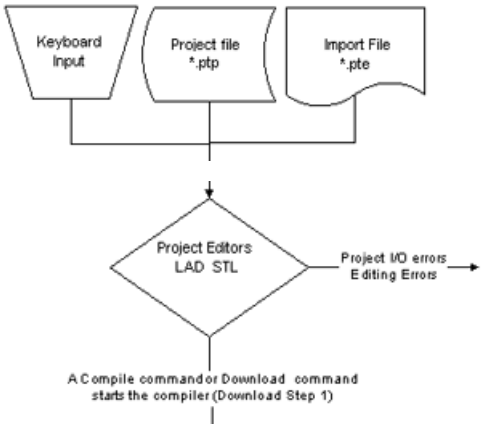
Information on using the software (Page 375)

First steps: Contents (Page 17)

## 10.6.2 Download

Use one of the following methods to download project components from the Programming Tool to the PLC:

- Click the  button.
- Select the menu command **File > Download**.
- Press the **Ctrl+D** shortcut key.

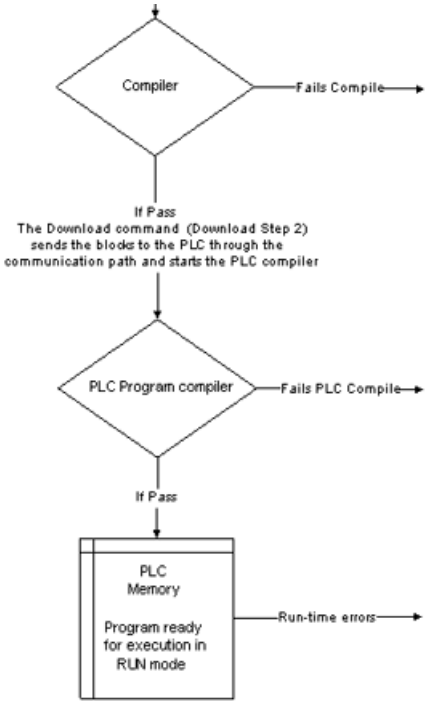



Opening projects (Page 379)

Range checking (Page 209)

Selecting the CPU type (Page 469)

All compilation errors in the Programming Tool are listed in the Output Window (Page 451). Double-click the error and the editor scrolls to the error location. Use the menu command PLC > CPU Type (Page 469)to set a specific model and version of the CPU.




Communication error

To download (editor to PLC) or upload (PLC to editor), PLC communication must be operating properly. Make sure your network hardware and PLC connecting cable are working.

The PLC Compiler verifies that the PLC hardware supports all program instructions, ranges, and the structure.

Use the menu command PLC > Information (Page 334) to view the first compilation error found. Use the menu command PLC > CPU Type (Page 469)to set a specific model and version of the CPU. This moves as many errors as possible to the Output Window where they are easier to locate and fix.

**Note**

**Saving retentive data**

The user-specific retentive data must be selected to be able to save them in the user project.

You can find information on this in the Chapter "Saving retentive data" (Page 345).

**See also:**

Communications overview (GS 4.1) (Page 50)

Testing the communications network (GS 4.2) (Page 51)
Downloading a program to the CPU (GS 4.3) (Page 52)
Uploading from the PLC (File > Upload from CPU) (Page 382)
Uploading from a PLC (GS 6.3) (Page 78)
Correcting compilation and loading errors (GS 4.4) (Page 55)
Information on using the software (Page 375)
First steps: Contents (Page 17)

# 10.7 Monitor and test your program

## 10.7.1 Overview of test and monitoring functions

### Diagnostics functions

Once you have established communication between your programming device, on which the PLC Programming Tool is installed, and a PLC and have loaded a program into the PLC, you can work with the diagnostic functions of the PLC Programming Tool and test new programs as well as monitor programs that are already being processed.

These help topics are explained in the following:

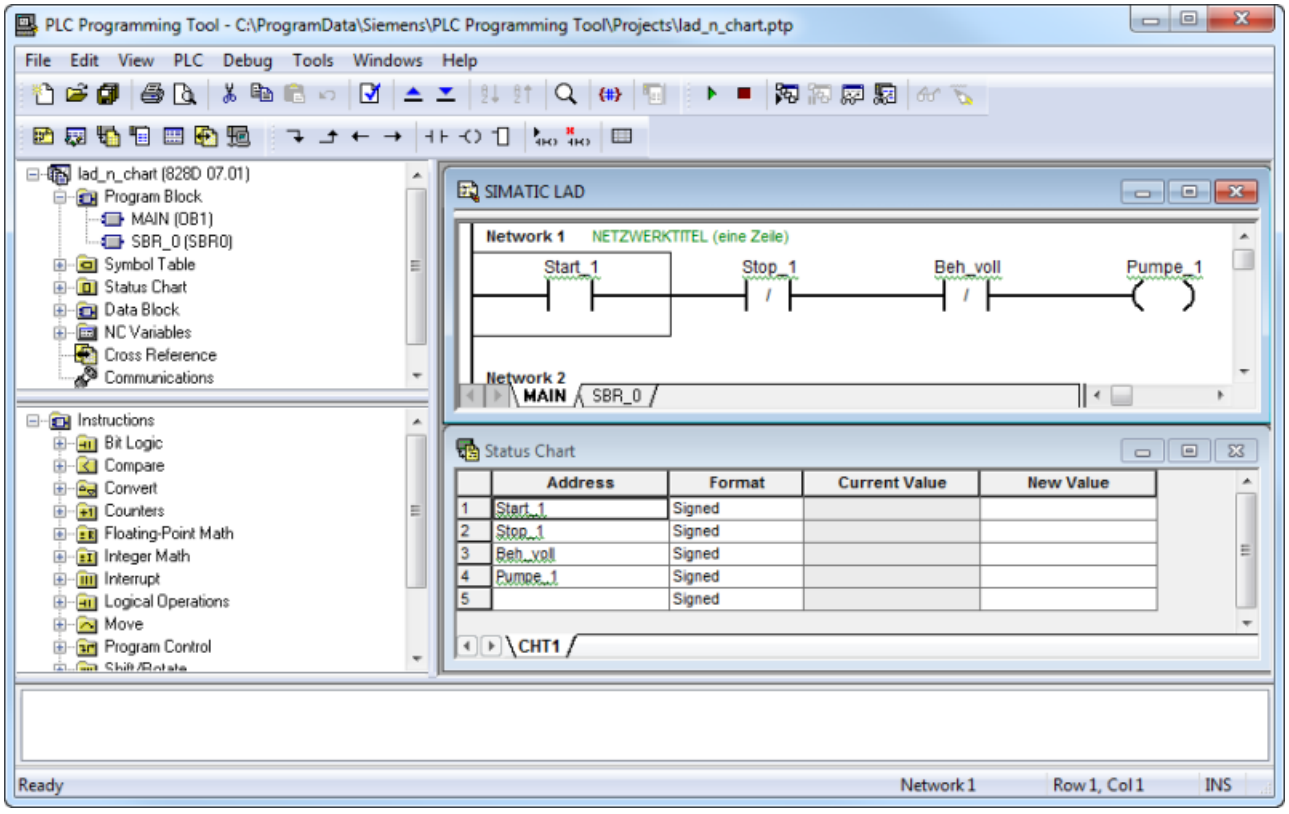
### What is "status"?

"Status" refers to the display of the actual values of operands while executing the program in the PLC. You can display status information in a status chart, with the data block status or by enabling the program status in the program editor.

In the status chart in each case the values are recorded at the end of a scan cycle (end-of-scan status). It is not possible to display the status of the local data memory and the accumulators.
With the data block status, the actual values are also recorded at the end of a scan cycle in each case.

If the execution status is activated for the program status on the "LAD Status" tab (Tools > Options) (Page 492), the status values are recorded at the time of execution of the instructions. The states of the local data memory and the accumulators are displayed. The status values are updated and displayed only if the PLC is in the RUN mode.

An example for status information in the status chart and in the program editor of the PLC Programming Tool is shown in the following diagram:




**Note**

Please note that unnecessary project components (e.g. the project tree, the instruction tree, and the output window) have been omitted, in order to provide more space to display the required components (LAD program editor, status chart, function bar to test and symbol for the status chart in the navigation bar). Using the menu "View", you can set-up the environment in the PLC Programming Tool corresponding to your particular tasks, i.e. you can display just the project components that you require.

## Supporting CPUs

The execution status is not available for all CPUs. The following table shows which CPUs are available:

| | |
|---|---|
| 828D (as of PLC 05.04) | 808D-PPU14x |
| 828D Step 2 | 808D-PPU15x |
| | 808D-PPU16x |

### Execution status for unsupported CPUs

If the execution status is activated for a CPU that does not support this status, the end-of-scan status is automatically executed.

If the execution status is not activated, the end-of-scan status is executed. For this, the values are recorded at the end of a scan cycle in each case. It is not possible to display the status of the local data memory and the accumulators.

### Preconditions of the status update

Before you can update the status to monitor and test your program, you must execute the following tasks:

- Your program must be able to be compiled error-free.
- You must have set up communication between the PLC Programming Tool and the PLC.
- Your program must have been loaded error-free into the PLC.

  **Note**

  The time stamps of your project in the Programming Tool and in the PLC must match, so that you can enable the status. The comparison of the time stamps (carried out automatically if you attempt to enable the status) ensures that the representation of the project status in the Programming Tool exactly reflects the status of the program in the PLC. If the time stamps do not match, you can compare the programs via the dialog box "Time Stamps Do Not Match". Look at the drop-down list box in order to find out which networks are different.

- After you have loaded the program into the PLC, then you should again bring this into the RUN mode. Otherwise, the address status is displayed, however, the PLC cannot execute the program, so you are not shown the logic operations that you expect.

### Operating state of the PLC

The type of monitoring and test functions that you can execute depends on the mode of your PLC.

Even if your program is not executed in the STOP mode, the operating system of the PLC still monitors the PLC (status of RAM and I/O) and transfers the data status to the Programming Tool. If the PLC is in the STOP mode, then you can execute the following functions:

- You can display the actual values of the operands as the end-of-scan status in the chart status, the data block status or the program status. (This is the same as the function "Single Read", as the program is not executed.)
- In the chart status and the data block status, you can write values.
- You can execute a certain number of scan cycles and display the effect in a status chart, in the data block status and/or in the program status.

If the PLC is in the RUN mode, you cannot execute the functions "First Scan" or "Multiple Scans". You can write values in a status chart, you can also execute the following functions (not in the STOP mode):

- In the chart status, you can carry out continuous updates. (If you wish to only execute one update, you must disable the chart status so that you can execute the command "Single Read".)
- In the data block status, you can carry out continuous updates.
- You can execute continuous updates in the program status.

## Communication and scan cycle

In a continuous scan cycle, the PLC reads the inputs, it executes the program, writes to the outputs, and executes system functions as well as communication. This scan cycle runs with an extremely high speed of many times per second. Even if the PLC Programming Tool issues status requests in a fast sequence, you cannot monitor each individual event that takes place in the PLC. When using the program status or the chart status, if you read data values from a PLC program, the data is scanned by taking samples (spot check). The update rate of the status values read from the PLC depends on the communication baud rate. Communicate with maximum baud rate, if you want to achieve the highest update rate.

## Different ways to update the status

### Chart status

Click the "Chart Status" button or select the menu command **Debug > Chart Status** to view continuous updates of the status when the PLC is in the RUN mode. Disable the chart status and use the function "Single Read" if you need to update the status just once and you do not want to switch the PLC into the STOP mode. If you switch the PLC into the STOP mode and enable the chart status, you receive a status update too. Furthermore, you can use the functions "Multiple Scans" and "First Scan" while you are viewing a status chart. In the chart status, the end-of-scan status is always used.

### Data Block Status

Click the "Data Block Status" button or select the menu command **Debug > Data block status** to view continuous updates of the current values of a data block when the PLC is in the RUN mode. If you switch the PLC into the STOP mode and enable the data block status, you receive a current value update too. Furthermore, you can use the functions "Multiple Scans" and "First Scan" while you are viewing a data block status. With the data block status, the actual values are always recorded at the end of a scan cycle.

**Program status**

Click the "Program Status" button or select the menu command **Debug > Program Status** to display the status of the data in the PLC in the program editor. The status data is recorded in the mode previously selected on the "LAD Status" tab (Tools > Options) (Page 492):

- Execution status (execution status activated)
  In the execution status in LAD, the value of operands and the signal flow for execution of the individual instructions during the execution of the program in the scan cycle are displayed. The execution status shows the values of the operands at the time of execution of the instruction; this may be overwritten again by subsequent program instructions. All of the displayed data values of the PLC are recorded in a single program scan cycle. The states of the local data memory and the accumulators are displayed. The status values are only updated and displayed if the PLC is in the RUN mode.

  **Note**

  The execution status may only be run as the program status once for each CPU at any one time. If you attempt to run the execution status in a different application, such as in a second Programming Tool, which is connected to the same PLC, an error will be displayed.

- End-of-scan status (execution status deactivated or CPU does not support execution status)
  The end-of-scan status shows the status results that are read at the end of the program scan cycle. These results may not indicate all changes to the values of the addresses of data in the PLC, because subsequent program instructions write a value and overwrite it again before the end of the program scan cycle is reached. The end-of-scan status shows the data values that are scanned at the end of multiple scan cycles. This results from the different speeds between the fast scan cycle of the PLC and the relatively slow transmission of the PC status data. The states of the local data memory and the accumulators are not displayed. If you switch the PLC into the STOP mode, you receive a status update too.

You can receive the status values of the program status continuously or as a snapshot.

- **Continuous:**
  Open the program editor window and enable the program status in order to view continuous updates of the program execution status when the PLC is in the RUN mode.

  **Note**

  "Continuous" should not be understood as "real time"; it means that the programming device scans the PLC for status information continuously and displays it on your screen, updating the display as quickly as your communication setup permits. Some rapidly fluctuating values may not be recorded and displayed on your screen. It is also possible that the values change too quickly for you to read them.

- **Snapshot:**
  If the PLC is in the STOP mode, you can use the menu command **Debug > Multiple Scans...** to display the status values of one or more scan cycles, or, with the menu command **Debug > First Scan**, to display the status values of the first scan cycle.

With the button "Pause Program Status" or the menu command Debug > Pause Program Status you can pause the status display of the program status. This can be necessary if values change too quickly for you to read them. In the execution status this can also make a single execution of a program section, e.g. a subprogram, visible. To do this, set the networks which are not being processed in the program editor and pause the status display with the

execution status activated. The next time the networks are executed in the program editor, the status display is updated. The status update then stops.

### Simulating process conditions

You can simulate process conditions by writing new values to operands: To do so, use the status chart.

### Checking cross references and the elements used

If you test your program, you may wish to supplement, delete or change the parameters.

In the window "Cross Reference", you can determine how the parameters are presently assigned in your program. This helps you to avoid assigning values twice.

### Loading changes in your program into the CPU

If you test a program with the program status and thus determine that you want to change the program, you must disable the program status before you can change the program. Once you have loaded the changed program into the CPU, you can enable the program status to check whether the changes are correct.

### See also

Displaying the status in the editors (GS 5.2) (Page 60)

Displaying the status in the status chart (GS 5.3) (Page 68)

Executing a certain number of scan cycles (GS 5.4) (Page 73)

Displaying the data block status (GS 5.5) (Page 74)

Downloading a program to the CPU (GS 4.3) (Page 52)

Cross-references and elements used (Page 439) (Ensure that program changes do not produce duplicate assignment.)

RUN/STOP modes of the PLC (Page 457)

## 10.7.2 RUN/STOP modes of the PLC

**Mode of operation of the program:**

If you load a program into your PLC and switch the PLC into the RUN mode, the PLC (CPU) of the PLC executes the program in a repeated scan cycle. This scan cycle is called in the interval of MD10075 PLC_CYCLE_TIME:

1. **Reading the inputs**
   The CPU reads the status of all inputs connected to the automation system. This data is stored in the input memory area, known as the process image input.
2. **Executing the optional interrupt program INT100**

3. **Executing the program**
   The CPU calls MAIN and executes the logic of the program. In this respect, it has read and write access to the process image, variables, data blocks, etc.
4. **Executing the optional interrupt program INT101**
5. **Writing to the outputs**
   At the end of the program, the CPU writes the data from the process image output to the physical outputs.
6. **Processing communication requests (NCK, HMI)**
7. **CPU self-diagnosis**

**Changing the mode of the PLC**

- Click the RUN button to place the PLC into the RUN mode. Click the STOP button to place the PLC into the STOP mode.
- Select the PLC > RUN menu command to place the PLC into the RUN mode. Select the PLC > STOP menu command to place the PLC into the STOP mode.

**Note**

In order that you can set the RUN/STOP mode using the Programming Tool, communication (Page 85) between the Programming Tool and the PLC must be established.

**PLC operating mode details:**

The PLC has two operating modes: STOP and RUN. In the STOP mode, you can create and edit your program. However, your program is not executed in the STOP mode. The program is executed in the RUN mode. Further, in the RUN mode, you can observe the mode of operation and data of the program. Test functions are available to better observe the mode of operation of the program and to identify programming errors (bugs).

The test functions to execute the first or several scan cycles can be used in the STOP mode. The RUN mode is then only assumed for the specified number of scan cycles.

Fatal errors are saved in the CPU operating system and force a mode change from the RUN to the STOP mode. If the PLC has identified a fatal error, then it can no longer be brought from the STOP mode into the RUN mode as long as the error is present. Although the operating system of the CPU stores minor errors that can be evaluated later, they do not require a transition from RUN to STOP.

In the **STOP** mode

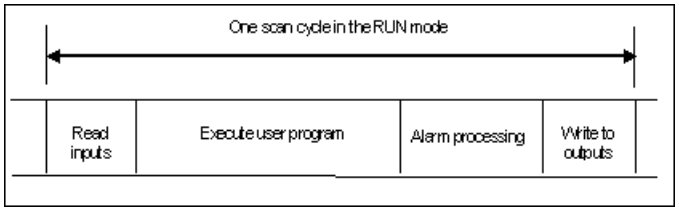
- The PLC is, in effect, in a quasi-idle state (the execution of the user program is interrupted)
- You can load the user program into and from the PLC.

If one or several devices are communicating with the PLC via the communication interface, then the PLC responds to all of the requests one after the other. The PLC does not prevent several communicating devices mutually interfering with one another. Your system design must provide the necessary safety precautions to prevent this mutual interference between communicating partners.

In the **RUN** mode, the PLC

- Reads the inputs
- Executes the user program
- Writes to the outputs
- Responds to communications requests
- Performs internal administrative tasks
- Responds to interrupt conditions

The PLC does not support fixed scan cycle times to execute the scan cycle in the RUN mode. These tasks (with the exception of the interrupts) are executed according to their priority, in the sequence that they occur. This continuous execution of the CPU tasks is called the scan cycle and is shown in the following diagram.




Each scan cycle begins by reading the actual values of the inputs and writing these values into the process image input. Input bits that have no corresponding physical input but which are in the same byte as bits with physical inputs are set to zero in the process image in each scan cycle.

After the inputs have been read, your program executes the instructions from the first through to the last one.

During the phase in which the alarms are processed in the scan cycle, all alarms are processed for which changes occurred during the scan cycle.

Finally, the values are written from the process image into the output modules. This completes one scan cycle.

**See also:**

Information on using the software (Page 375)

First steps: Contents (Page 17)

## 10.7.3 Execute a certain number of scan cycles

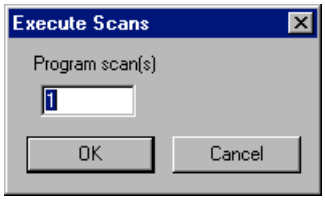
You can specify that the PLC should process a certain number of scan cycles of your program (from one scan cycle up to 65,535 scan cycles). If you specify that the PLC should execute a certain number of scan cycles, you can monitor the processing of the process variables.
In the first scan cycle, the value of SM0.1 = 1 (ON).

**Executing a single scan cycle:**

1. The PLC must be in the STOP mode. If it is not already in STOP, switch the PLC into the STOP mode (Page 361).
2. Select the menu command **Debug > First Scan**.

**Executing multiple scan cycles:**

1. The PLC must be in the STOP mode. If it is not already in STOP, switch the PLC into the STOP mode (Page 361).
2. To execute multiple scan cycles, select the menu command **Debug > Multiple Scans...**. This opens the dialog box "Execute Scans".




3. Specify how many scan cycles should be executed and confirm with "OK".

**Note**

Make sure that you switch the PLC back into the RUN mode (this requires that you click the RUN button in the toolbar or select the menu command **PLC > RUN**) if you wish to return to normal program processing.

**See also:**

Overview of test and monitoring functions (GS 5.1) (Page 55)

Displaying the status in the editors (GS 5.2) (Page 60)

Displaying the status in the status chart (GS 5.3) (Page 68)

Displaying the data block status (GS 5.5) (Page 74)

Downloading a program to the CPU (GS 4.3) (Page 52)

Cross-references and elements used (Page 439) (Ensure that program changes do not produce duplicate assignment.)

RUN/STOP modes of the PLC (Page 457)

## 10.7.4 How to work with the status chart

### Precondition

After you have loaded your program into the PLC, you can generate one or multiple status charts to monitor and test program execution.

The program is continuously executed if the PLC is in the RUN mode. You can enable the chart status so that the status values in the chart are continuously updated (not interrupted).

As an alternative, using the function "Single Read", you can generate a "snapshot" of the status values in the chart without having to enable the status chart.

While you are viewing a status chart, you can also switch the PLC into the STOP mode and only execute the first or a certain number of scan cycles, in which you monitor program execution.

**Note**

Please note that you cannot change your chart if the chart status is enabled.

Disable the chart status if you wish to edit the chart.

These help topics are explained in the following:

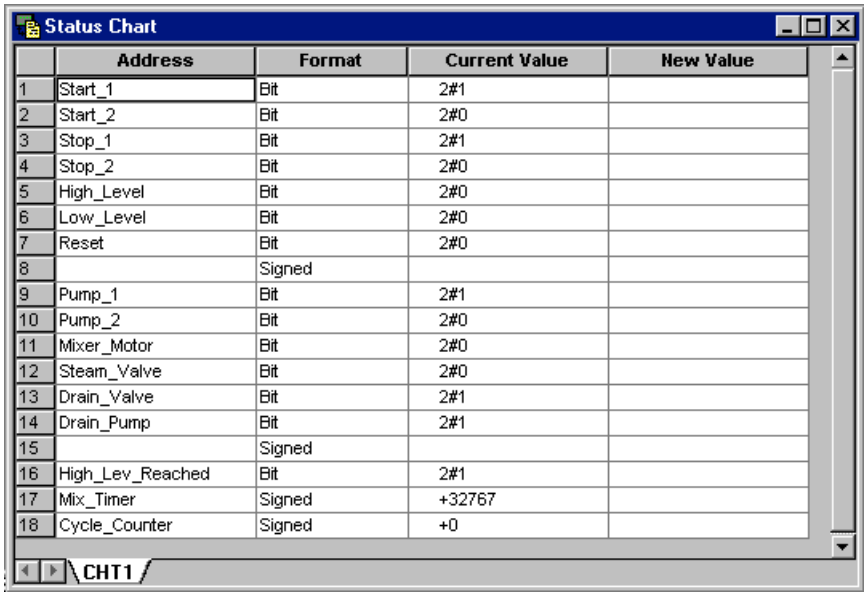
## Example for the status chart

Opening a status chart is not the same as enabling a status chart. You can open and evaluate or change a status chart: However, if you do not execute the command "Single Read" (in the menu "Debug" or in the toolbar) or enable the chart status (in the menu "Debug" or in the toolbar), no status information will be displayed in the column "Actual Value".

If you use the function "Single Read" (this is only available when the chart status is disabled) to evaluate a status chart, the actual values of the PLC are accepted and displayed in the column "Actual Value". However, the values are not updated while the PLC is executing the program.

If you enable the chart status (in the menu "Debug" or in the toolbar), the actual values of the PLC are regularly updated. If changes are received from the PLC, then the column "Actual Value" is updated.

You can assign (write) certain values in the PLC using the column "New Value".

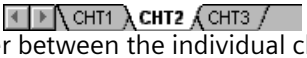



| | Address | Format | Current Value | New Value |
|---|---|---|---|---|
| 1 | Start_1 | Bit | 2#1 | |
| 2 | Start_2 | Bit | 2#0 | |
| 3 | Stop_1 | Bit | 2#1 | |
| 4 | Stop_2 | Bit | 2#0 | |
| 5 | High_Level | Bit | 2#0 | |
| 6 | Low_Level | Bit | 2#0 | |
| 7 | Reset | Bit | 2#0 | |
| 8 | | Signed | | |
| 9 | Pump_1 | Bit | 2#1 | |
| 10 | Pump_2 | Bit | 2#0 | |
| 11 | Mixer_Motor | Bit | 2#0 | |
| 12 | Steam_Valve | Bit | 2#0 | |
| 13 | Drain_Valve | Bit | 2#1 | |
| 14 | Drain_Pump | Bit | 2#1 | |
| 15 | | Signed | | |
| 16 | High_Lev_Reached | Bit | 2#1 | |
| 17 | Mix_Timer | Signed | +32767 | |
| 18 | Cycle_Counter | Signed | +0 | |

## Opening or enabling a status chart

Open a status chart to evaluate or change the contents of the chart. Enable the status chart so that the status information can be updated.

To open a status chart, proceed in one of the following ways:

- Click on the button of the status chart  in the "Navigation" toolbar (Page 500)
- Select the menu command **View > Status Chart**.
- Open the directory of the status chart in the project tree (Page 447) and double-click the symbol of a table  .
- If your project includes more than one status chart, using the tab for the  status charts at the lower edge of the window, you can switch over between the individual charts.

To enable a status chart, proceed in one of the following ways:

- If you wish to continually update the status information in the status chart, enable the chart status: Select the menu command **Debug > Chart Status** or click the appropriate button in the toolbar  .
- If you only require a "snapshot" of the values, execute the function "Single Read": Select the menu command **Debug > Single Read** or click the appropriate button in the toolbar  (if the chart status is enabled, then the function "Single Read" is deactivated).

**Tips:**

If you open a status chart, then the status is still not displayed. You must enable the status chart so that the status information is updated.

Enabling the status chart will have no effect if the chart is still empty: You must first create your status chart by entering program values (operands) in the column "Address" and entering a data type in the column "Format" for each address.

## Working with multiple status charts

You can create additional status charts in several ways:

- In the instruction tree, right-click the "Status Chart" folder and, in the shortcut menu, select the command **Insert Status Chart**.
- Open the window "Status Chart" and call the menu "Edit" or right-click and select the command **Insert Contents > Chart**.

**Note**

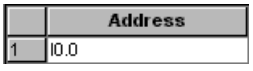
- After you have inserted a new status chart, a new tab  is displayed at the lower edge of the "Status Chart" window. If you wish to switch between the status charts, simply click the tab of the required status chart.
- Sometimes, a tab is hidden by the buttons on the right-hand side that are used to scroll. If a tab is not visible, drag the demarcation line between the tab area and the scroll buttons to display additional tabs.

## Creating a status chart

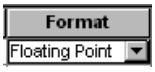
You can enter addresses in a status chart in order to monitor and control values from your program. Values of timers and counters can be displayed as bits or words. If you wish to display the value of a timer or a counter as a bit, then the status of the output is displayed (on or off). If you wish to display the value of a timer or a counter as a word, then the actual value is used.

To create a status chart, proceed as follows:

1. In the address field you enter the addresses of the desired values.
   Most of the memory types listed in the memory areas of the automation system (Page 262) are valid, with the exception of data constants and accumulators.
   To edit an address field, select the desired field using the cursor keys or the mouse. If you enter data, existing data are deleted and the new characters are entered. The field is selected if you double click with the mouse or press key "F2". You can then move the cursor using the cursor keys to the position that you wish to edit.

2. If the element involves a bit (e.g. I, Q or M), then the bit format is displayed in the second column. If the element involves a byte, word or double word, select the field in the column "Format" and double-click or press the space bar or ENTER in order to scroll through the valid formats until the correct format is displayed.

To insert a new row, open the menu "Edit" or right-click a field in the status chart so that the shortcut menu is displayed and select the command **Insert Contents > Row**. A new row is then inserted in the status chart above the cursor position. You can also move the cursor to a field in the last row and press the down arrow key in order to insert a row at the bottom of the status chart.

**Tips:**

You can select addresses in the symbol table and copy these into the status chart in order to generate your chart faster.
You can display the status a multiple number of times. You can classify the elements in logical groups to display each group in an individual chart. In this way, you avoid having to scroll through extremely long lists.

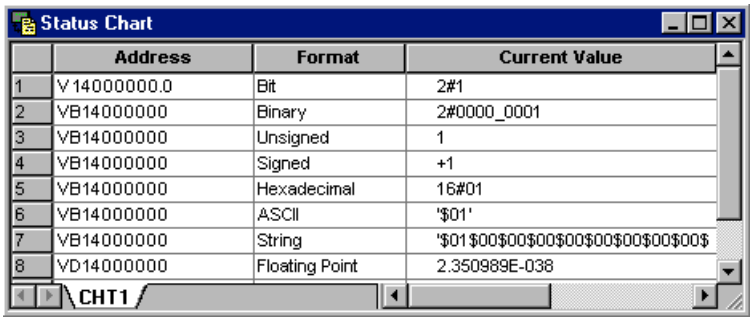
## Shortcut key combinations for editing

- To insert a new row with the following address and the same data format, select an address field and press the Enter key.
- To scroll through the possible data formats for a specific address, select the field "Data Format" and press the Enter key several times. To display all available data formats, open the drop-down list box.
- Use the TAB key to jump into the next field of the chart.
- To set the width of a column, position the mouse cursor on the edge of a column until the appearance of the cursor changes and then drag to increase or decrease the width of the column.
- To select a complete row (to cut or copy), click once the number of the row.

- To insert a new row, select a field or a row and right-click. In the shortcut menu select the command **Insert Contents > Row**. The row is inserted at the cursor position. The subsequent rows are shifted down by one row.
- To insert a row at the bottom of the status chart, move the cursor to a field in the last row and press the down arrow key.
- To delete a field or a row, select the field or the row and right-click. Select the menu command **Delete > Selection**. If you delete a row, the subsequent rows are shifted up by one row.
- To select the entire status chart, click once the upper left corner above the row numbers.

## Data formats

The data format that you assign to a value defines how the value is represented in the status chart.
In the example below, VD14000000 (and therefore VB14000000 and V14000000.0) contains the value 1.




**Tips:**

Bit and binary values are both introduced by the number 2 and the # symbol. Hexadecimal values are introduced by the number 16 and the # symbol. Bit values have one digit. Binary values have eight digits.
Signed and unsigned values use the basis 10 (decimal).

## Single read or continuous chart status

- If you only require a "snapshot" of the values, execute the function "Single Read": Select the menu command Debug > Single Read or click the appropriate button in the toolbar (if the chart status is enabled, then the function "Single Read" is deactivated).
- If you wish to continually update the status information in the status chart, enable the chart status: Select the menu command Debug > Chart Status or click the appropriate button in the toolbar .

### Working with test functions in the status chart

You access the test functions (Single Read, Write All) using the menu Debug or using the toolbar with the test functions.

| | |
|---|---|
|  | **Single Read**<br>Use Single Read if you require a "snapshot", i.e. a single update of the program status of all values. By default, the chart status continually scans the PLC for status updates. If you click a status chart and the chart status is disabled, then the button for Single Read is activated. |
|  | **Write All**<br>After you have entered the values in the column "New Value" in the status chart, write the required changes to the PLC using the command "Write All". |

### See also

Overview of test and monitoring functions (GS 5.1) (Page 55)

Displaying the status in the editors (GS 5.2) (Page 60)

Executing a certain number of scan cycles (GS 5.4) (Page 73)

Displaying the data block status (GS 5.5) (Page 74)

Downloading a program to the CPU (GS 4.3) (Page 52)

Cross-references and elements used (Page 439) (Ensure that program changes do not produce duplicate assignment.)

RUN/STOP modes of the PLC (Page 457)

## 10.7.5 Display the data block status

### Precondition

If the CPU that you are using supports data blocks, you can, after you have loaded your program into the PLC, display the current values of the data blocks in the PLC.

### CPUs that support data blocks


828D
828D Step 2
808D-PPU14x (only special data blocks)
808D-PPU15x
808D-PPU16x

## Procedure

The program is continuously executed if the PLC is in the RUN mode. You can enable the data block status so that the actual values in the data block are continuously updated (not interrupted).

While you are viewing a data block chart, you can also switch the PLC into the STOP mode and only execute the first or a certain number of scan cycles, in which you monitor program execution. The actual values of the data block are then continuously updated.

**Note**

Please note that you cannot change your data block if the status is enabled.

Disable the status if you wish to edit the data block.

## Example for the data block status

If you enable the data block status (in the menu "Debug" or in the toolbar), the values in the column "Actual Value" are regularly updated with the actual values of the PLC.

You can assign (write) certain values in the PLC using the column "New Value".

Data Block

| | Address | Name | Data Type | Format | Current Value | New Value | Comment |
|---|---|---|---|---|---|---|---|
| 1 | 0.0 | myByte1 | BYTE | Unsigned | 0 | | Comment byte 1 |
| 2 | 1.0 | myByte2 | BYTE | Unsigned | 0 | | Comment byte 2 |
| 3 | 2.0 | myWord1 | WORD | Unsigned | 0 | | Comment word 1 |
| 4 | 4.0 | myInt1 | INT | Signed | +0 | | Comment integer 1 |
| 5 | 6.0 | myBit1 | BOOL | Bit | OFF | | Comment bit 1 |
| 6 | 6.1 | myBit2 | BOOL | Bit | OFF | | Comment bit 2 |
| 7 | 6.2 | myBit3 | BOOL | Bit | OFF | | Comment bit 3 |
| 8 | 6.3 | myBit4 | BOOL | Bit | OFF | | Comment bit 4 |
| 9 | 6.4 | myBit5 | BOOL | Bit | OFF | | Comment bit 5 |
| 10 | 6.5 | myBit6 | BOOL | Bit | OFF | | Comment bit 6 |
| 11 | 6.6 | myBit7 | BOOL | Bit | OFF | | Comment bit 7 |
| 12 | 6.7 | myBit8 | BOOL | Bit | OFF | | Comment bit 8 |
| 13 | 8.0 | myDword1 | DWORD | Unsigned | 711 | | Comment double word 1 |
| 14 | 12.0 | myDint1 | DINT | Signed | +0 | | Comment double integer 1 |
| 15 | 16.0 | myReal1 | REAL | Floating Point | 0.0 | | Cpmment floating point 1 |

DB_9000 / DB_9001 / DB_9002

## Enabling the data block status

Open a data block and enable the data block status so that the actual values from the PLC are displayed. If you enable the data block status without having opened a data block, the first data block from the data block list is opened automatically.

To enable the data block status, proceed in one of the following ways:

- Click the "Data block status" button in the toolbar.
- Select the menu command **Debug > Data Block Status**.

If your project includes more than one data block, if the data block status is active then you can switch between the individual data blocks. To do this, proceed in one of the following ways:

- Switch between the individual data blocks using the tab for the data blocks  at the lower edge of the window.
- In the project tree, double-click the desired data block.

**Note**

- When the data block status is activated, the structure of the variables (name, data type, initial value, and comment) is write-protected. The format of the variables can be changed.
- The structure of the data block in the project must match the structure of the data block in the PLC. If you have modified the structure of the data block, then the complete project has to be loaded again.

## Changing the actual values in the data block status

If the data block is not write-protected, you can change the actual values in the PLC with the data block status active. To do so, enter the desired values in the column "New Value" and write the changes to the PLC with the command "Write All"  .

## See also

Overview of test and monitoring functions (GS 5.1) (Page 55)

Displaying the status in the editors (GS 5.2) (Page 60)

Displaying the status in the status chart (GS 5.3) (Page 68)

Executing a certain number of scan cycles (GS 5.4) (Page 73)

Saving data in the variable memory with the help of a data block (Page 424)

Downloading a program to the CPU (GS 4.3) (Page 52)

Cross-references and elements used (Page 439) (Ensure that program changes do not produce duplicate assignment.)

RUN/STOP modes of the PLC (Page 457)

# 10.8 Print projects

## 10.8.1 Print projects and documentation

### Access

Print a hard copy of your program and project documentation by using one of the following methods:

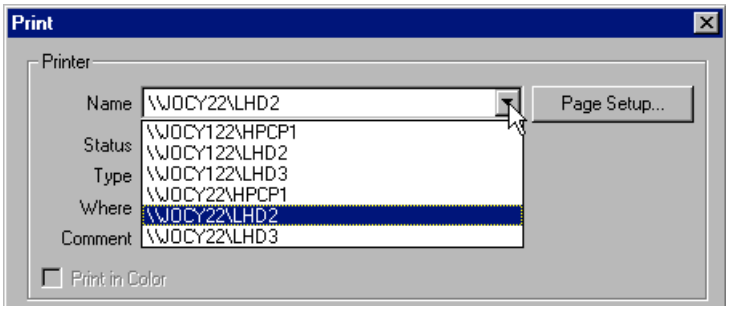
- Select the  button.
- Select the menu command **"File > Print"**.
- Press the **<Ctrl+P>** shortcut key.

### Choosing a printer and a color selection

The printers that are displayed in the list box are determined by the printers that have been configured for your personal computer / programming device.

**Note**

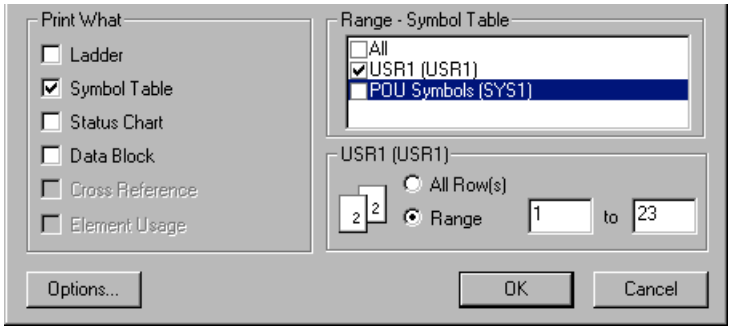
If you have a monochrome printer, make sure the "Print in Color" checkbox is deselected, otherwise network comments in the printout may be unreadable.




### Print subareas

If several Program Blocks, Symbol Tables and Status Charts exist, you can print a network area from a specific program block or a specific number of rows from a specific Symbol Table or Status Chart. To do this, activate the appropriate checkboxes and specify the elements in the area fields that you want to print.

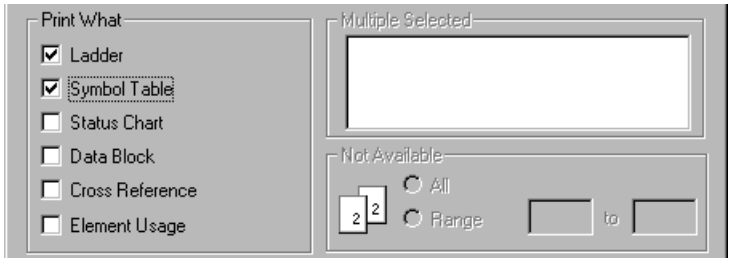
For example, you want to print out rows from the USR1 symbol table. Then activate only the Symbol Table checkbox under the "Print What" heading and only the "USR1" checkbox under "Range". Then define the range of rows to print.




**Note**

However, if you continue to select additional components of the program for printing, your range selection for the first component is **disabled** and the full range of each component will be printed.

Example: You want to check LAD and specify a range of networks. If you select an additional program component, such as Symbol Table, which you also wish to print, the range selection for Ladder becomes disabled and the Print command will print the entire range of LAD networks as well as all Symbol Table information.




## Print multiple project components

Select as many component checkboxes as you wish under the "Print What" heading. Bear in mind that you cannot make a range selection when you are printing more than one project component.

**Note**

Each time you click a checkbox, you toggle its check mark on or off. If several checkboxes are selected and you do not want them all, click the ones that you want to deselect.

## Select print options

[Print options: POU properties, data block properties, number of columns (LAD), line numbers (STL), local variable table]

From within the "Print" dialog box, click the Options... button to change the print options.

Activate or deactivate the checkbox to toggle the option on and off:

| | |
|---|---|
| Number of Columns to Print | (LAD only) If any networks in your program contain more columns than you stipulate here, the additional columns are printed out on a second page. Click anywhere in the Program Editor window to see a selection box that shows the width of a single column. For instance, an LAD contact is one column wide. |
| Properties | Print the author and time stamp properties of the program organization unit. |
| Local Variable Tables | Local variable tables for all POUs selected. |
| Network Comments | (LAD only) Print network titles and comments that were entered into the LAD Network Title / Comment editor. [STL comment lines are always printed.] |
| Print Line Numbers | (STL only) Print line numbers beside each line of your program. |
| Data block | Print the properties of the data block. |

## Printing help

You can print groups of topics from the Contents browser or individual topics.

- To print **all of the topics in a book** from the "Table of contents" browser, select the book title and click the "Print" button.
- To print **an individual topic** from the Contents browser, select the topic title and click the "Print" button.
- To print from inside an **individual topic,** use the "Print" button that is located above the topic title.

## See also:

Print (GS 6.1) (Page 76)

Print preview (Page 393)

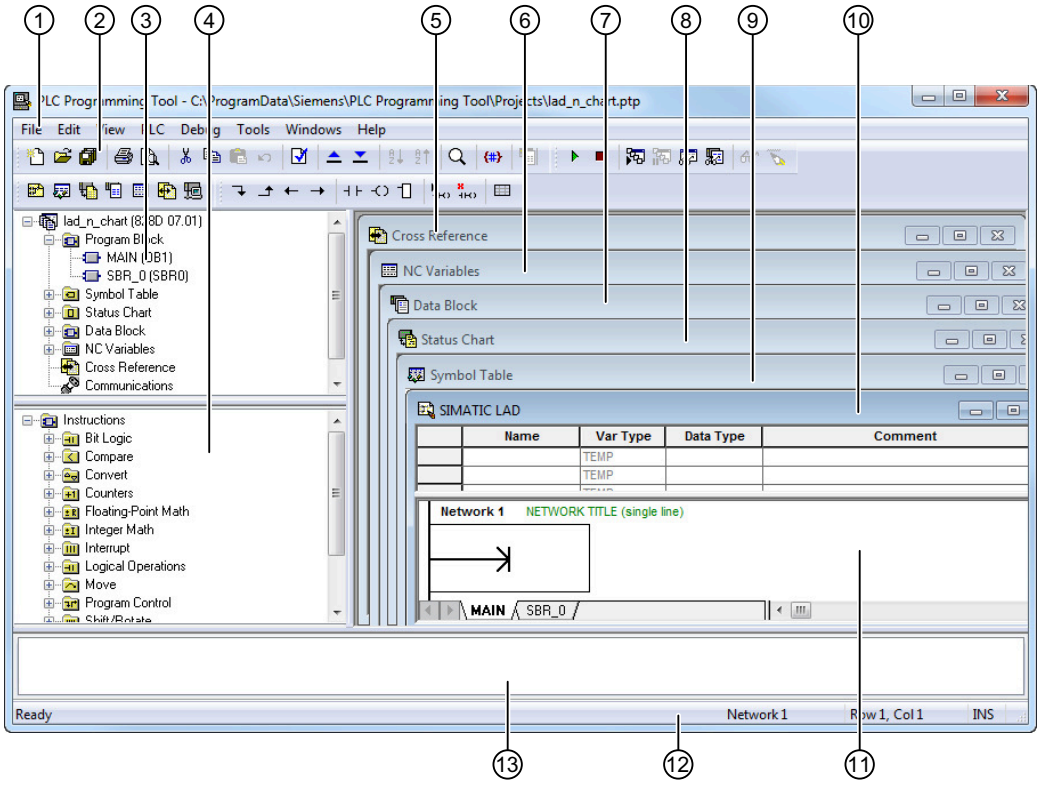
Information on using the software (Page 375)

First steps: Contents (Page 17)

# Information on using the software

# 11

## 11.1 Window elements




1. Menu bar (Page 378)
2. Toolbar (Page 499)
3. Project tree (Page 447)
4. Instruction tree (Page 449)
5. Cross-references (Page 439)
6. NC variables (Page 434)
7. Data block (Page 424)
8. Status chart (Page 420)
9. Symbol table (Page 413)
10. Program block (Page 412)
11. Local variable table (Page 229)
12. Status bar (Page 456)
13. Output window (Page 451)

## Explanation of the window elements

Menu bar (Page 378)

Allows you to perform operations using either a mouse or keystrokes.

Toolbars (Page 499)

Provides easy mouse access to the most commonly used Programming Tool operations. You can customize (Page 487) the content and appearance of each of the toolbars.

Shortcut keys (Page 496)

You can use shortcut keys in the Programming Tool to make a variety of tasks more efficient.

Project tree (Page 447)

The project tree displays all of the elements of the project. You can right-click project elements to insert additional Program Organization Units (POUs). You can right-click an individual Program Organization Unit to open, rename, delete, or edit its properties.

Instruction tree (Page 449)

Provides a tree view of all project elements and all the instructions that are available for your current program editor (LAD or STL). You can right-click project elements to insert additional Program Organization Units (POUs). You can right-click an individual Program Organization Unit to open, rename, delete, or edit its properties. You can right-click a folder or individual instruction in the Instructions portion of the tree to hide the entire tree.

Once you open an instruction folder, you can drag and drop individual instructions or double-click to automatically insert the selected instruction at the cursor location in the Program Editor window.

**Note**

In STL programs, the instruction tree is for reference only: You must enter the instructions in the LAD editor. You cannot transfer them from the instruction tree.

Local variable table (Page 229)

Contains assignments that you have made to local variables (in other words, variables that are used by your subprograms and interrupt programs). Variables created in the Local Variable Table use temporary memory. Address assignment is handled for you by the system. Use of the variable is restricted to the POU where it was created.

Program Editor Window (Page 412)

Contains the Local Variable Table and the program view for the editor (LAD (Page 111) or STL (Page 258)) that you are using for this project. You can drag the split bar to expand the program view and cover up the Local Variable Table if desired. When you create subprograms (Page 223) or interrupt programs (Page 227) in addition to the main program (OB1), tabs appear at the bottom of the Program Editor window. You can click the tabs to move between the subprograms, interrupt programs, and OB1.

Output window (Page 451)

Provides information messages when you compile your program. When the output window lists program errors, you can double-click an error message and the appropriate network is displayed in the Program Editor window.

Status bar (Page 456)

Provides information about the status of the operations that you perform in Programming Tool.

Cross-references (Page 439)

In this window you can view Cross Reference and Element Usage information for your program.

Symbol Table Window (Page 413)

Allows you to assign and edit global symbols (in other words, symbolic values that can be used in any POU, not just the POU where the symbol was created). You can create multiple symbol tables. There is also a tab in the Symbol Table for system-defined symbols that you may want to use in your program.

Status Chart Window (Page 420)

Allows you to track the status of program inputs, outputs, or variables by putting them into the chart. You can create multiple status charts in order to view elements from different portions of your program. Each status chart has its own tab in the Status Chart window.

Data Block Editor Window (Page 74)

Allows you to display and edit the content of your data block.

NC Variable Editor Window (Page 434)

Allows you to display and edit the NC variables.

## See also

Print (File menu) (Page 394)

New (File menu) (Page 378)

Open (File menu) (Page 379)

Save (File menu) (Page 379)

Print Preview (File menu) (Page 393)

Cut (Edit menu) (Page 398)

Copy (Edit menu) (Page 399)

Paste (Edit menu) (Page 400)

Undo (Edit menu) (Page 398)

Compile (Page 459)

Uploading from the CPU (File menu) (Page 382)

Downloading to the CPU (File menu) (Page 382)

Sorting table and chart columns (Page 446)

Zoom (Page 452)

RUN / STOP (Page 457)

Displaying the status in the program editor (Page 60)

Displaying the status in a status chart (Page 68)

Entering instructions in LAD (Page 36)

Insert (Edit menu) (Page 401)

Delete (Edit menu) (Page 404)

Symbol information table (Page 445)

Bookmarks in the STL program (Page 258)

Communication (Page 443)

File menu (Page 378)

Edit menu (Page 398)

View menu (Page 411)

Menu PLC (Page 457)

Menu Debug (Page 471)

Menu Tools (Page 487)

Menu Window (Page 494)

Menu Help (Page 495)

# 11.2 Main menu

## 11.2.1 File menu

### 11.2.1.1 New (File menu)

Start a new project by using one of the following methods:

- Select the  button.
- Select the menu command **"File > New"**.
- Press the **<Ctrl+N>** shortcut key.

The New command creates a set of empty project components and begins an editing session.

Only one project can be open for each instance of the Programming Tool. If you want to open two projects at the same time, you must run two instances of the Programming Tool. If two instances are open, you can copy and paste LAD program elements from one project to the other.

**See also:**

Information on using the software (Page 375)

First steps: Contents (Page 17)

### 11.2.1.2 Open (File menu)

Open an existing project by using one of the following methods:

- Select the  button.
- Select the menu command **"File > Open"**.
- Press the **<Ctrl+O>** shortcut key.

---

**Note**

- You cannot use this menu command to open a project that resides on a PLC. The project file must reside on your personal computer / programming device.
- Only one project can be open for each instance of the Programming Tool. To open two projects at the same time, you must run two instances of the Programming Tool. If two instances are open, you can copy and paste LAD program elements from one project to the other.

---

**See also:**

Projects from previous versions (Page 82)

### 11.2.1.3 Close (File menu)

Close the current project:

- Select the menu command **"File > Close"**.

The Close command closes all project components in the current editing session. You are prompted to save your work, if there are any unsaved changes. Only one project can be open for each instance of the Programming Tool.

**See also:**

Information on using the software (Page 375)

First steps: Contents (Page 17)

### 11.2.1.4 Save (File menu)

Save your work from the current editing session by using one of the following methods:

- Select the  button.
- Select the menu command **"File > Save"**.
- Press the **<Ctrl+S>** shortcut key.

This menu command saves the following project data in a single file:

- Program
- PLC configuration
- Symbol tables
- Status charts

- Data blocks
- Comments

The file has *.ptp as extension.

**See also:**

Information on using the software (Page 375)

First steps: Contents (Page 17)

### 11.2.1.5 Save As (File menu)

Save the project with a file name and in a location that you specify:

- Select the menu command **"File > Save As"**.

If you have saved your project previously, you can use this menu command to save it again. To do this, specify a different name and/or a different directory.

This menu command saves the following project data in a single file:

- Program
- PLC configuration
- Symbol tables
- Status charts
- Data blocks
- Comments

The file has *.ptp as extension.

**See also:**

Information on using the software (Page 375)

First steps: Contents (Page 17)

### 11.2.1.6 Import (File menu)

Importing a project:

- Select the menu command **File > Import**.

Use this menu command to import a project from an external source. Export files of type PTE and ARD are supported.

An ARD file contains an archive with data classes. Here you specify which data classes will be transferred.

A new or existing project must be open in order for you to use the Import command. The import action deletes existing project data and replaces it with the data from the imported file.

**Note**

Projects that were created with previous versions of the Programming Tool must be opened directly with the menu command **File > Open**.

**Note**

**Saving retentive data**

The user-specific retentive data must be selected to be able to save them in the user project.

You can find information on this in the Chapter "Saving retentive data" (Page 345).

**See also:**

Information on using the software (Page 375)

First steps: Contents (Page 17)

### 11.2.1.7 Export (File menu)

Exporting a project:

- Select the menu command **File > Export**.

Use this menu command to export a project from the Programming Tool. The export is possible only for projects that have been compiled successfully. If a project contains data classes (Page 243), it will be exported as archive in an ARD file. You can specify which data classes will be exported.

Projects that do not contain any data classes are exported in a PTE file.

A project must be open in order for you to use the Export feature. The Export command exports existing program organization units (POUs) (main program, subprograms, and interrupt programs), data blocks, and system data blocks.

**Note**

**Saving retentive data**

The user-specific retentive data must be selected to be able to save them in the user project.

You can find information on this in the Chapter "Saving retentive data" (Page 345).

**See also:**

Information on using the software (Page 375)

First steps: Contents (Page 17)

### 11.2.1.8 Uploading from the CPU (File menu)

Use one of the following methods to upload project components from the PLC to a Programming Tool program editor:

- Click the  button.
- Select the menu command **File > Upload**.
- Press the **Ctrl+U** shortcut key.

To upload (PLC to editor), PLC communication must be operating properly. Make sure your network hardware and PLC connecting cable are working.

Select the blocks you want to upload (program block, data block, system data block). The program components you select for uploading are copied from the PLC to the currently open project. You can then save the program.

**Note**

**Saving retentive data**

The user-specific retentive data must be selected to be able to save them in the user project.

You can find information on this in the Chapter "Saving retentive data" (Page 345).

**See also:**

Uploading from a PLC (GS 6.3) (Page 78)

Communications overview (GS 4.1) (Page 50)

Testing the communications network (GS 4.2) (Page 51)

Downloading a program to the CPU (GS 4.3) (Page 52)

Downloading to the PLC (File > Download to CPU) (Page 382)

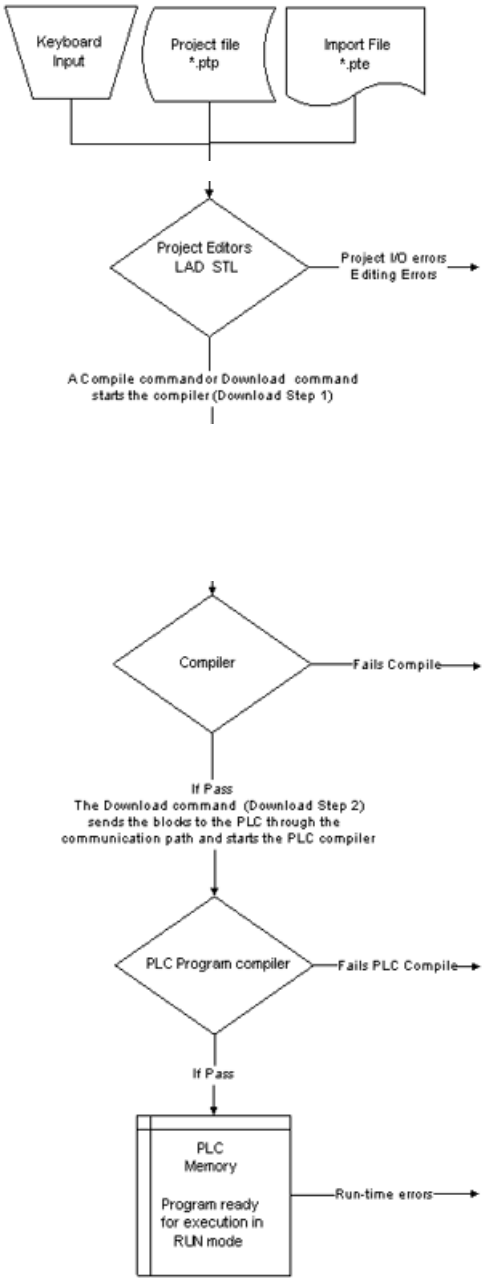
Information on using the software (Page 375)

First steps: Contents (Page 17)

### 11.2.1.9 Downloading to the CPU (File menu)

Use one of the following methods to download project components from the Programming Tool to the PLC:

- Click the  button.
- Select the menu command **File > Download**.
- Press the **Ctrl+D** shortcut key.




Opening projects (Page 379)

Range checking (Page 209)

Selecting the CPU type (Page 469)

All compilation errors in the Programming Tool are listed in the Output Window (Page 451). Double-click the error and the editor scrolls to the error location. Use the menu command PLC > CPU Type (Page 469)to set a specific model and version of the CPU.

Communication error

To download (editor to PLC) or upload (PLC to editor), PLC communication must be operating properly. Make sure your network hardware and PLC connecting cable are working.

The PLC Compiler verifies that the PLC hardware supports all program instructions, ranges, and the structure.

Use the menu command PLC > Information (Page 461) to view the first compilation error found. Use the menu command PLC > CPU Type (Page 469)to set a specific model and version of the CPU. This moves as many errors as possible to the Output Window where they are easier to locate and fix.

**Note**

**Saving retentive data**

The user-specific retentive data must be selected to be able to save them in the user project.

You can find information on this in the Chapter "Saving retentive data" (Page 345).

**See also:**

Communications overview (GS 4.1) (Page 50)

Testing the communications network (GS 4.2) (Page 51)

Downloading a program to the CPU (GS 4.3) (Page 52)

Uploading from the PLC (File > Upload from CPU) (Page 382)

Uploading from a PLC (GS 6.3) (Page 78)

Correcting compilation and loading errors (GS 4.4) (Page 55)

Information on using the software (Page 375)

First steps: Contents (Page 17)

### 11.2.1.10 Compare (File menu)

#### General information on the block comparison

The comparison between two PLC user projects can be performed in two ways. You can compare program blocks (POUs) and user data blocks (actual values) of the current project either

- with the PLC project in a controller (online), or
- with the blocks of another project (offline).

In this manner, the block information of the projects is directly compared.

Two PLC user projects may be compared with one another for the following reasons:

- For putting a machine/series of machines into operation, it is necessary to determine the differences in comparison with an earlier version of the PLC project.
- Whether or not the PLC project was modified in the machine should be tested during the service call.
- The machine dealer checks whether the PLC project in the supplied machine corresponds to the project that he wants to deliver to his customer.

The comparison always provides the same results, independent of the PLC Programming Tool version used for creating the respective PLC user project.

When comparing two projects, the CPU types for these projects must be consistent. If the CPU types do not concur across both projects, it will be inquired whether the CPU type of the comparison project should be adopted for the current project.

The project comparison dialog opens when the comparison is started. In this dialog you select the data classes whose blocks you want to compare. You can compare program blocks and data blocks separately. Following selection of the data classes, the program and user data blocks to be compared can be selected from the block lists. For the user data blocks, the comparison of the actual values can also be activated. Other project components such as symbol tables cannot be compared.

The results of the comparison are displayed in the output window. All compared blocks are listed. Differences between the blocks are shown below the block name. Here, the position specifications refer to the current project. For program blocks, the position data is provided via the network number and, if available, row number and column number. For data blocks, the position data is provided via the row numbers. At the end of the list of the program or data blocks, information on how many blocks were compared and how many differences

were found is provided. If you double-click on the line of a difference in the output window, the corresponding position in the editor of the current project will open.

**Special features of the comparison**

- Empty networks in the program blocks or empty rows in the data blocks are ignored during the comparison.
- If too many differences are found in a network or a row, these are not all listed, and a message indicating that the network or the row is different is displayed.
- Block properties are factored in.
- Different names and data classes are captured.
- With regard to the data blocks, differences in the "non-retain" and "write-protected" properties are also covered.
- Creation date and date of last change are not compared.
- Differences in block properties have no position data. For this reason, no reaction occurs to a double-click on the row with the difference in the output window.

## Online comparison via PLC > Compare...

1. Start the online comparison via the menu **"PLC > Compare..."**.

   **Note**

   If a connection to the address set under "Communication" cannot be set up, a corresponding error message will be displayed.

   If the CPU type of the open project does not concur with that of the remote PLC, a query is displayed as to whether the CPU type of the remote PLC should be adopted for the opened project.

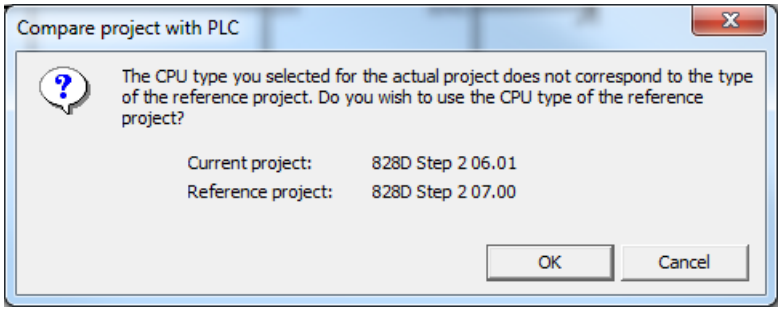



   Figure 11-1 Taking over the CPU type

   In this dialog, the address and type of the remote PLC are displayed. If the address does not point to the desired PLC, click on the **"Cancel"** button to cancel the comparison; you can then modify the address under **"View> Communication"**.

2. Click on the **"OK"** button to import the CPU type and to start the comparison dialog. The **"Compare project with PLC"** dialog opens.

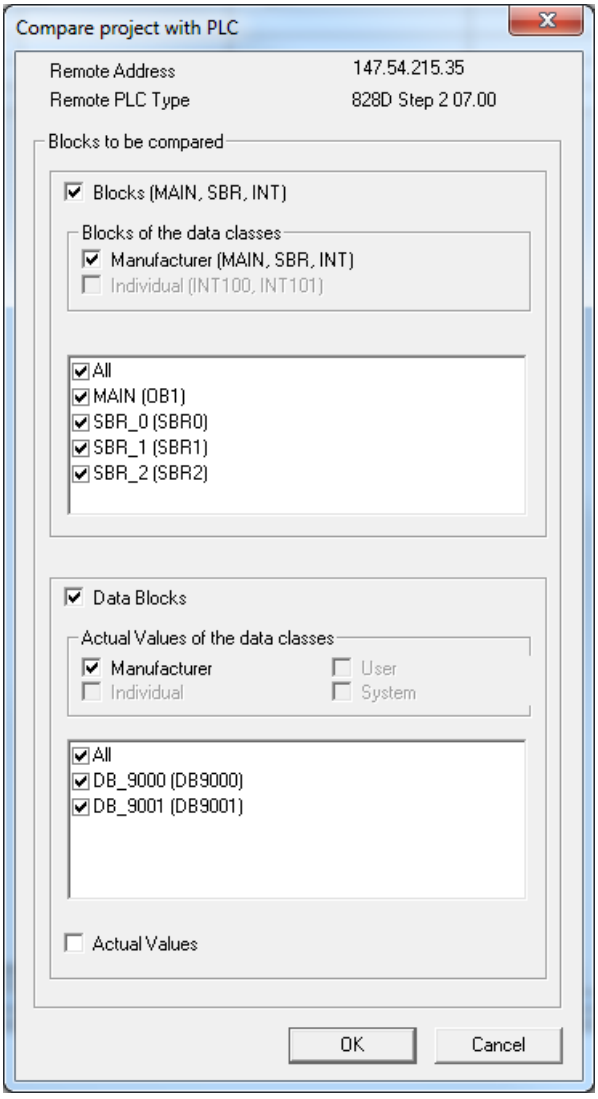



Figure 11-2 "PLC > Compare dialog"

3. In this comparison dialog, you can select the blocks that are to be compared.
   Program and data blocks are distinguished. A pre-selection can be made via the selection of data classes. By default, all blocks are selected.
4. Click on the **"OK"** button to start the comparison.

5. The result of the comparison is displayed in the output window. In this manner, all blocks compared are listed, even if there are no differences. Differences are displayed below the respective block name.

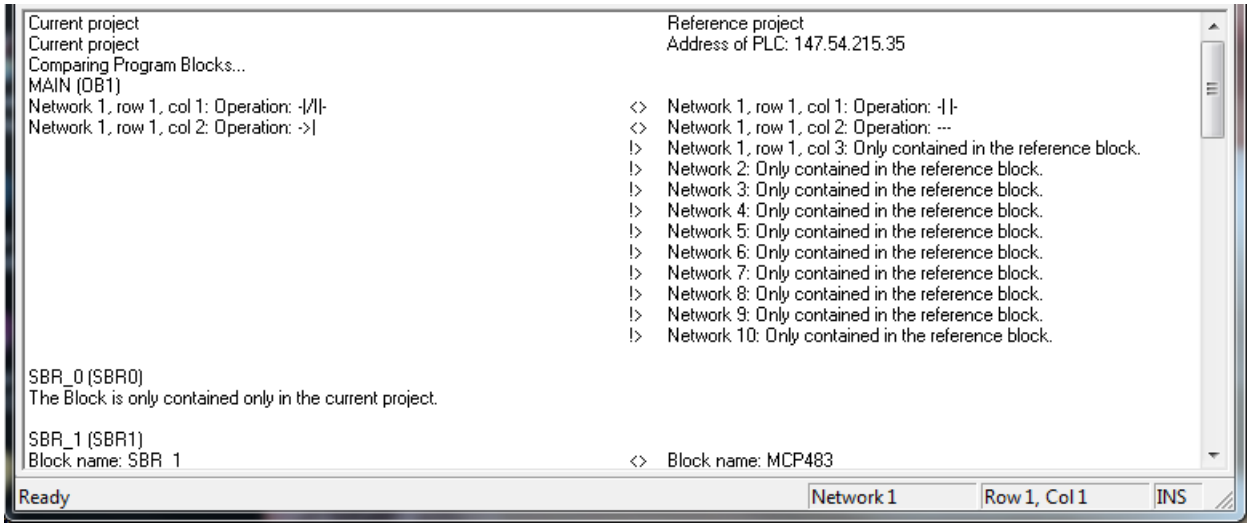
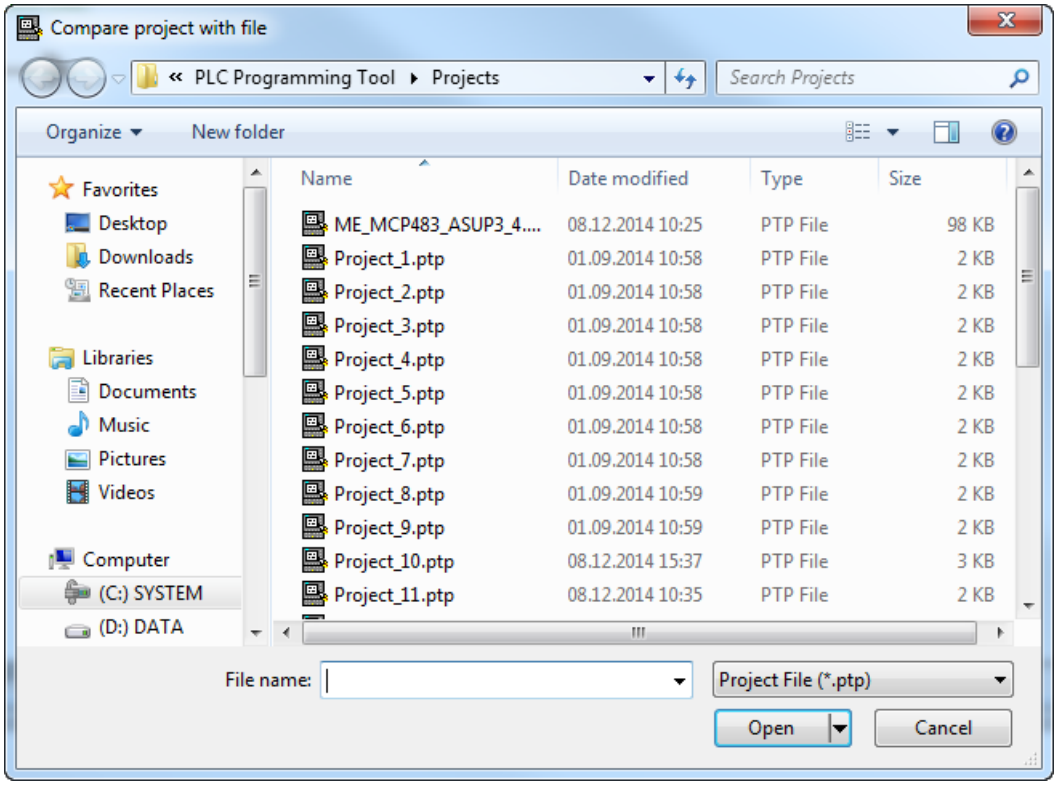



Figure 11-3 Comparison result in the output window

6. Double-click on a row with a difference in the output window to open the relevant row in the editor of the current project.

### Offline comparison via File > Compare...

1. Start the offline comparison via the **"File > Compare..."** menu.
   The **"Compare project with file"** dialog is opened.




2. In this dialog, select the project that you want to compare with the project that is already open.
3. Click on the **"Open"** button.
   If the CPU type of the current project does not concur with that of the comparison project, the question as to whether the CPU type of the comparison project should be taken over for the current project follows.

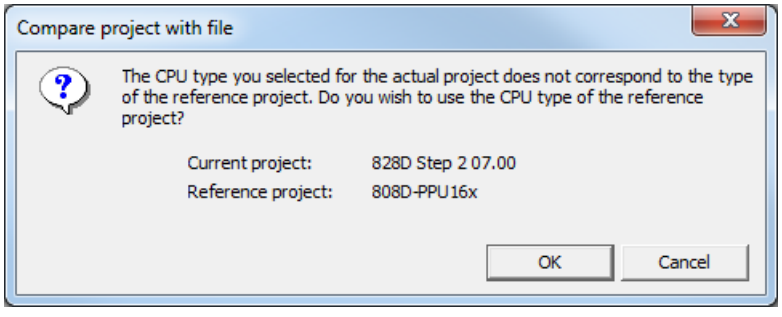



Figure 11-4 Taking over the CPU type

4. Click on the **"OK"** button to take over the CPU type and to start the comparison dialog.
   In the dialog that now opens, the name including path of the file to be compared is displayed.

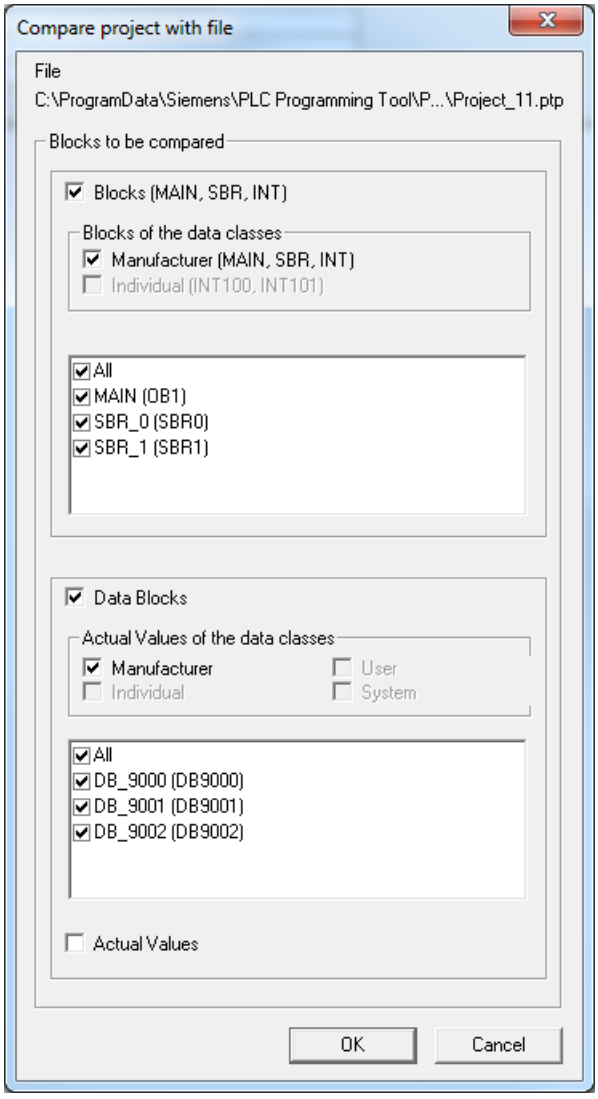



Figure 11-5 "File > Compare" dialog

5. In this comparison dialog, you can select the blocks that are to be compared.
   Program and data blocks are distinguished. A pre-selection can be made via the selection of data classes. By default, all blocks are selected.

6. Click on the **"OK"** button to start the comparison.
   The result of the comparison is displayed in the output window. All blocks compared are listed, even if no differences are present. Differences are displayed below the respective block name.

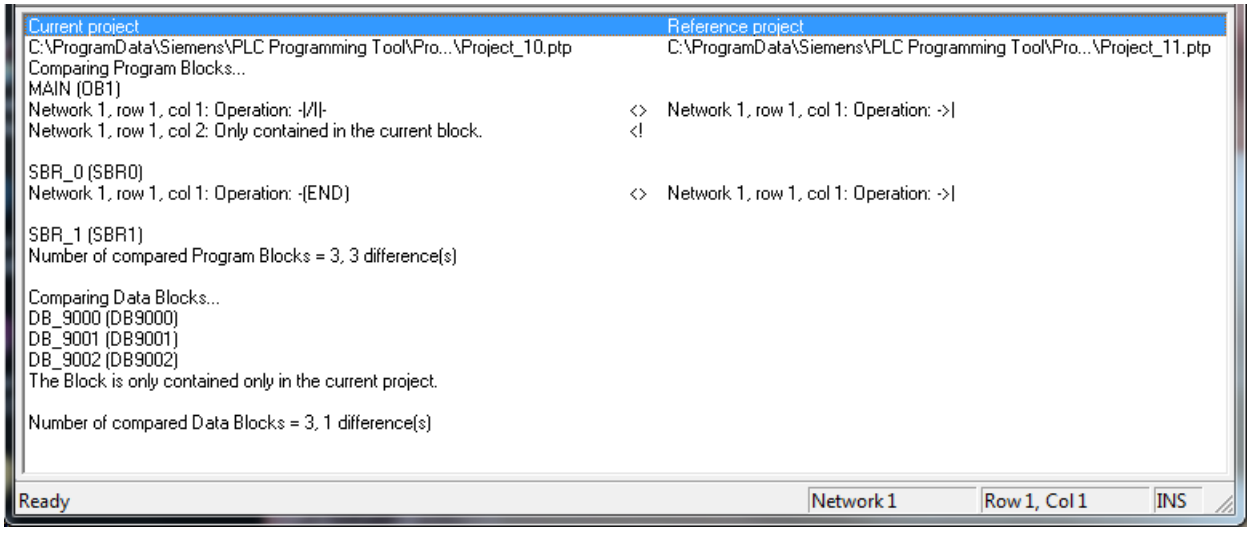



Figure 11-6 Comparison result in the output window

7. Double-click on a row with a difference in the output window to open the relevant row in the editor of the current project.

## Saving the comparison results

The comparison results can be saved and thus also be printed.

1. Right-click in the output window.

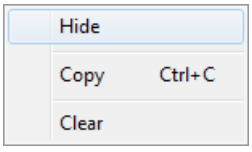



Figure 11-7 Shortcut menu of the output window

2. Select the "Copy" option.
   Alternatively, you can also click on the output window with the left mouse button and press "Ctrl + C".
   In this way, the entire content of the output window is copied as text to the clipboard.
3. Now insert the text in any text editor, text editing program, or in Microsoft Excel.
   The results can then be printed out from these programs.

## See also:

Information on using the software (Page 375)

First steps: Contents (Page 17)

Output window (View menu) (Page 451)

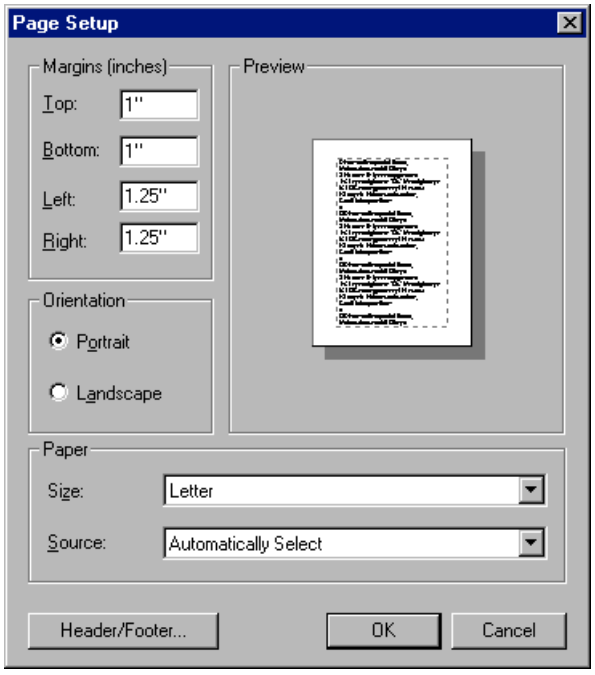
### 11.2.1.11 Setup page (File menu)

#### Page Setup

To format the pages of your project printout, select the menu command **"File > Page Setup"** or click the "Page setup" Page Setup... button from within the "Print" dialog box.

The "Page Setup" dialog box lets you set margins, orientation, and paper size/source.

You can access the "Header/Footer" dialog box from "Page Setup".




#### Header/Footer

From within the "Page Setup" dialog box, click the Header/Footer... button to configure a header or footer on your project printouts.

The "Header/Footer" dialog box provides a toolbar of items to use in headers or footers. To select an item, position your cursor in the field (header or footer) in which you want the item to appear, and click the appropriate button. For footers, you have the additional option of using an annotation box. The annotation box lets you create fields with titles that you type (for example, "Signature" or "Comments") so that you can better customize the footer.

The Header/Footer toolbar buttons are as follows:

| | |
|---|---|
|  | **Project Name** (full path) |
|  | **Object Name**<br>For new projects, the object name (POU, symbol table, status chart or data block) is automatically included in the header when you print. |
|  | **Date** |
|  | **Timer** |

| | |
|---|---|
|  | **No. of Pages** |
|  | **Left Justified** (header/footer) |
|  | **Centered** (header/footer) |
|  | **Right Justified** (header/footer) |

When you have selected your options, type in the text that you want to appear in the header or footer of your printout. You might want to insert spaces between the sections of header and footer text.
Click the "OK" button to confirm the settings or click the "Cancel" button to cancel.

To view your header and footer, select the menu command **"File > Print Preview"**.

**See also:**

Information on using the software (Page 375)

First steps: Contents (Page 17)

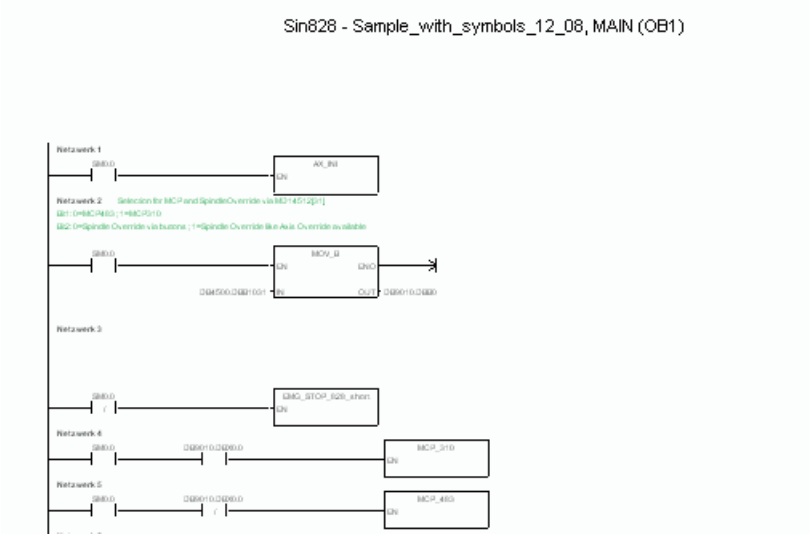
### 11.2.1.12 Print Preview (File menu)

To view the project before printing:

- Select the menu command **"File > Print Preview"** or click the "Print Preview"  button in the toolbar.

You can display the project page by page or two pages at the same time. You can use the zoom function to increase or decrease the size of the preview display. You must click the "Close" button at the top of the "Preview" window in order to return to the Program Editor view.

- To make changes to the appearance of the printout, select the menu command **"File > Page Setup"**.
- To define which components should be printed and which printer should be used, choose the menu command **"File > Print"**.




**See also:**

Information on using the software (Page 375)

First steps: Contents (Page 17)

### 11.2.1.13 Print (File menu)

#### Access

Print a hard copy of your program and project documentation by using one of the following methods:

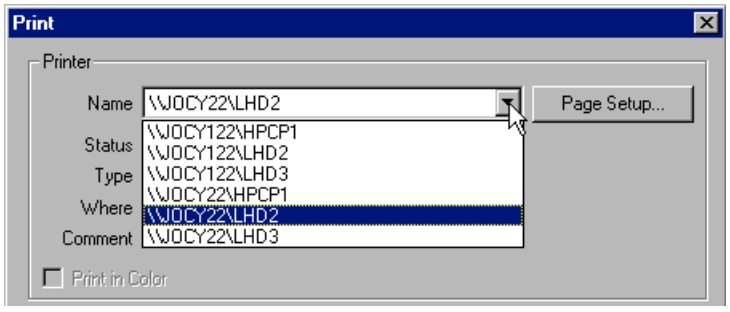
- Select the  button.
- Select the menu command **"File > Print"**.
- Press the **<Ctrl+P>** shortcut key.

#### Choosing a printer and a color selection

The printers that are displayed in the list box are determined by the printers that have been configured for your personal computer / programming device.

**Note**

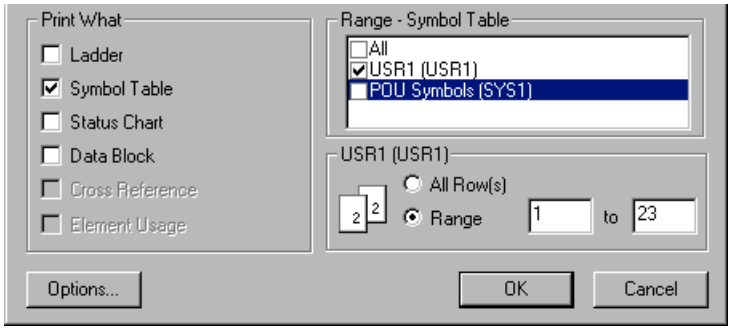
If you have a monochrome printer, make sure the "Print in Color" checkbox is deselected, otherwise network comments in the printout may be unreadable.




## Print subareas

If several Program Blocks, Symbol Tables and Status Charts exist, you can print a network area from a specific program block or a specific number of rows from a specific Symbol Table or Status Chart. To do this, activate the appropriate checkboxes and specify the elements in the area fields that you want to print.

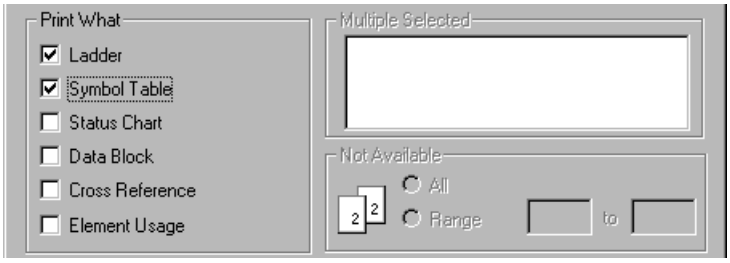
For example, you want to print out rows from the USR1 symbol table. Then activate only the Symbol Table checkbox under the "Print What" heading and only the "USR1" checkbox under "Range". Then define the range of rows to print.




**Note**

However, if you continue to select additional components of the program for printing, your range selection for the first component is **disabled** and the full range of each component will be printed.

Example: You want to check LAD and specify a range of networks. If you select an additional program component, such as Symbol Table, which you also wish to print, the range selection for Ladder becomes disabled and the Print command will print the entire range of LAD networks as well as all Symbol Table information.




### Print multiple project components

Select as many component checkboxes as you wish under the "Print What" heading. Bear in mind that you cannot make a range selection when you are printing more than one project component.

**Note**

Each time you click a checkbox, you toggle its check mark on or off. If several checkboxes are selected and you do not want them all, click the ones that you want to deselect.

### Select print options

[Print options: POU properties, data block properties, number of columns (LAD), line numbers (STL), local variable table]

From within the "Print" dialog box, click the Options... button to change the print options.

Activate or deactivate the checkbox to toggle the option on and off:

| | |
|---|---|
| Number of Columns to Print | (LAD only) If any networks in your program contain more columns than you stipulate here, the additional columns are printed out on a second page. Click anywhere in the Program Editor window to see a selection box that shows the width of a single column. For instance, an LAD contact is one column wide. |
| Properties | Print the author and time stamp properties of the program organization unit. |
| Local Variable Tables | Local variable tables for all POUs selected. |
| Network Comments | (LAD only) Print network titles and comments that were entered into the LAD Network Title / Comment editor. [STL comment lines are always printed.] |
| Print Line Numbers | (STL only) Print line numbers beside each line of your program. |
| Data block | Print the properties of the data block. |

### Printing help

You can print groups of topics from the Contents browser or individual topics.

- To print **all of the topics in a book** from the "Table of contents" browser, select the book title and click the "Print" button.
- To print **an individual topic** from the Contents browser, select the topic title and click the "Print" button.
- To print from inside an **individual topic,** use the "Print" button that is located above the topic title.

### See also:

Print (GS 6.1) (Page 76)

Print preview (Page 393)

Information on using the software (Page 375)

First steps: Contents (Page 17)

## 11.2.1.14 Recent File (File menu)

Opening a recently edited project:

- Select your file name with the menu command **"File > 1, 2, 3 or 4"**.

The file names of project files you last worked on are listed at the bottom of the File menu. You can select and open one of the projects directly without having to browse or specify a file name.

**See also:**

Information on using the software (Page 375)

First steps: Contents (Page 17)

## 11.2.1.15 Exit (File menu)

Close the Programming Tool:

- Select the menu command **"File > Exit"**.

You are prompted to save your work, if there are any unsaved changes in the project.

**See also:**

Information on using the software (Page 375)

First steps: Contents (Page 17)

## 11.2.2 Edit menu

### 11.2.2.1 Undo (Edit menu)

Reverse the last action you performed by using one of the following methods:

- Select the  button.
- Select the menu command **"Edit > Undo"**.
- Press the **<Ctrl+Z>** shortcut key.

You can issue multiple Undo commands to undo several actions in reverse order. If you issue an Open, Close, Save or Compile command, the Undo buffer is erased. Your next action is recorded as the start of a new Undo sequence.

**See also:**

Information on using the software (Page 375)

First steps: Contents (Page 17)

### 11.2.2.2 Cut (Edit menu)

Cut a selection and place it on the Windows clipboard by using one of the following methods:

- Select the  button.
- Select the menu command **"Edit > Cut"**.
- Press the **<Ctrl+X>** shortcut key.

Use the Cut command to remove a selected item and put it on the clipboard. The Cut function is only available when an item is selected.

You can cut the following items in a Programming Tool project:

- Text or data fields
- Instructions
- Single networks
- Multiple adjacent networks
- Entire POUs
- Status chart rows or columns, or entire status charts
- Symbol table rows or columns, or entire symbol tables
- Data block rows or columns, or entire data blocks

**Note**

You cannot simultaneously select multiple networks that are not adjacent. Due to the write-protected assignments in the local data memory, which must be unique in every table, no linked data area may be cut from one local variable table and copied to another. You must be in the LAD editor (**"View > Ladder"**).

**Note**

To cut, copy or delete a complete network in an LAD program, you must place the cursor on the network title. To paste a network or another item in an LAD program, you can place the cursor anywhere.

**See also:**

Information on using the software (Page 375)

First steps: Contents (Page 17)

### 11.2.2.3 Copy (Edit menu)

Copy a selection to the Windows clipboard by using one of the following methods:

- Select the  button.
- Select the menu command **"Edit > Copy"**.
- Press the **<Ctrl+C>** shortcut key.

Use the Copy command to copy a selected item to the clipboard. The Copy function is available only after you have selected an item for copying.

You can copy the following items in a Programming Tool project:

- Text or data fields
- Instructions
- Single networks
- Multiple adjacent networks
- Entire POUs
- Status chart rows or columns, or entire status charts
- Symbol table rows or columns, or entire symbol tables
- Data block rows or columns, or entire data blocks

**Note**

You cannot simultaneously select or copy multiple networks that are not adjacent. Due to the write-protected assignments in the local data memory, which must be unique in every table, no linked data area may be copied from one local variable table to another. You must be in the LAD editor (**"View > Ladder"**).

**Note**

To cut, copy or delete a complete network in an LAD program, you must place the cursor on the network title. To paste a network or another item in an LAD program, you can place the cursor anywhere.

**See also:**

Information on using the software (Page 375)

First steps: Contents (Page 17)

### 11.2.2.4 Paste (Edit menu)

Paste the contents of the Windows clipboard into the active window by using one of the following methods:

- Select the  button.
- Select the menu command **"Edit > Paste"**.
- Press the **<Ctrl+V>** shortcut key.

Use the Paste command to insert a copy of the clipboard contents above the current location of the cursor. This function is not available if the clipboard is empty.

You can paste the following items in a Programming Tool project:

- Text or data fields
- Instructions
- Single networks
- Multiple adjacent networks
- Entire POUs
- Status chart rows or columns, or entire status charts
- Symbol table rows or columns, or entire symbol tables
- Data block rows or columns, or entire data blocks

**Note**

You cannot simultaneously select or copy multiple networks that are not adjacent. Due to the write-protected assignments in the local data memory, which must be unique in every table, no linked data area may be copied from one local variable table to another. You must be in the LAD editor (**"View > Ladder"**).

**Note**

To cut, copy or delete a complete network in an LAD program, you must place the cursor on the network title. To paste a network or another item in an LAD program, you can place the cursor anywhere.

**Insert versus overstrike mode**

Press the >Insert> key to toggle between Insert mode (the default) and Overstrike mode (indicated as OVR in the status bar) when pasting. If you paste a selection in Insert mode, the existing text or graphical instruction is moved to the right to make room for the pasted selection. If you paste a selection in Overstrike mode, the pasted selection replaces the existing selection.

For LAD, if you replace (overstrike) one instruction with another instruction that has the same profile, any assignments that you made to the existing parameters are transferred to the new parameters. (That means, if the second instruction has the same number of signal flow inputs, input parameters, signal flow outputs, and output parameters as the first instruction, then the parameter assignments are retained when you overstrike the first instruction with the second.)

**See also:**

Information on using the software (Page 375)

First steps: Contents (Page 17)

### 11.2.2.5 Select All (Edit menu)

Select all text at the current cursor location by using one of the following methods:

- Select the menu command **"Edit > Select All"**.
- Press the **<Ctrl+A>** shortcut key.

**See also:**

Information on using the software (Page 375)

First steps: Contents (Page 17)

### 11.2.2.6 Insert (Edit menu)

Insert an item at the cursor location by using one of the following methods:

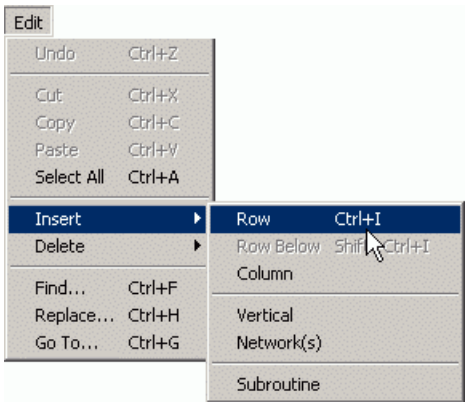
- Select the menu command **"Edit > Insert"**.
- Right-click an area in the Local Variable Table or Program Editor.
- Right-click the program block in the project tree.
- Click the Insert LAD Network  button to insert an empty network in the LAD program.

The Insert menu options that are available depend on the window component that you are editing and the location of the cursor.

**Note**

Place the cursor on a network title to cut, copy or paste a complete network in an LAD program. You can place the cursor anywhere in an LAD program and perform the network paste operation.

**LAD editor: "Insert" submenu**

| | |
|---|---|
|  | Insert row above cursor (in program or Local Variable Table)<br>Insert row below cursor (in program or Local Variable Table)<br>Insert column left of the cursor (in program)<br>Insert vertical connecting line (in program)<br>Insert network above the cursor and renumber all networks<br>Insert new subprogram into the program |

**Status chart and symbol table: "Insert" submenu**

| | |
|---|---|
| You can: | Insert a row above the cursor<br>Insert a new chart or table |

**Data block: "Insert" submenu**

| | |
|---|---|
| You can: | Insert a row below the cursor<br>Insert a new data block |

**See also:**

Information on using the software (Page 375)

First steps: Contents (Page 17)

### 11.2.2.7 Filtering NC variables (Edit menu)

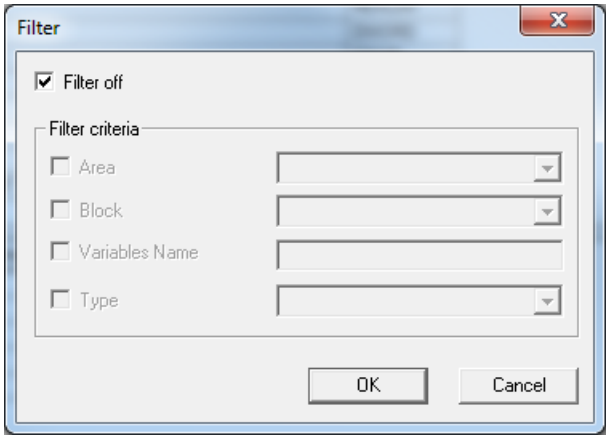
Proceed as follows to call the **"Filter"**function:

1. Open the variable table in which you wish to filter according to variables.
2. Call menu **"Edit"**, or right click in the variable table and select the**"Filter"**command.

The list of variables to be selected can be restricted by activating a filter. The following filter criteria are possible:

- Area
- Block
- Variable name
- Type

Area, block and type are selected from a list. The variable name must be entered. In this case, a part of a name is permissible. Placeholders may not be used. No distinction is made between upper and lower case.




**See also:**

NC variables (Page 434)

Information on using the software (Page 375)

First steps: Contents (Page 17)

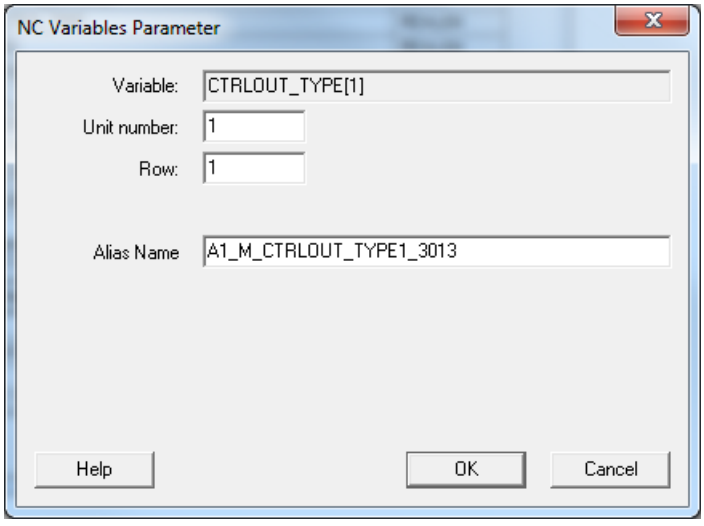
### 11.2.2.8 Editing NC variable parameters (Edit menu)

Proceed as follows to call the **"NC variables parameter"**dialog:

- The dialog is automatically opened when copying variables into the table of selected variables.
- Select a variable in the lower section of the **"NC variables"** window. Select the **"Edit"** menu, or right-click on a variable - and select the command **"Variable parameters..."**.

The parameters required for a variable can be edited in the **"NC Variable Parameters**" dialog. These can be the area number, line and column. Please refer to the user documentation for the significance of the individual parameters. Every variable automatically has an alias name. This alias name is used in the PLC user program to symbolically address variables. As the symbol length is limited, it is possible that the automatically generated alias name is too long. In this case, a shorter name is recommended. The alias name can be changed.

You can obtain additional information on a variable by clicking the **"Help"** button.




**See also:**

NC variables (Page 434)

Information on using the software (Page 375)

First steps: Contents (Page 17)

### 11.2.2.9 Delete (Edit menu)

Delete an item at the cursor location by using one of the following methods:

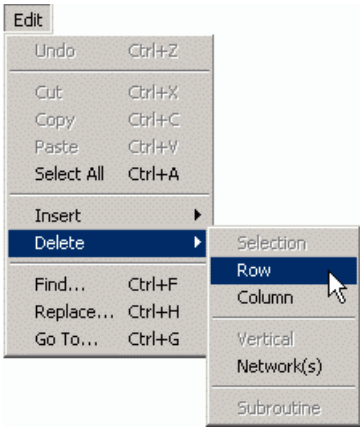
- Select the menu command **"Edit > Delete"**.
- Right-click an area in the Local Variable Table or Program Editor.
- Right-click the program block in the project tree.
- Click the LAD Delete Network button to delete an entire network in an LAD program.

The Delete menu options that are available depend on the window component that you are editing and the location of the cursor.

**Note**

To cut, copy or delete a complete network in an LAD program, you must place the cursor on the network title. To paste a network or another item in an LAD program, you can place the cursor anywhere.

**LAD editor: "Delete" submenu**




Delete row at the cursor (in program and Local Variable Table)
Delete column at the cursor (in program)
Delete vertical connecting line (cursor must be immediately to the left of the connecting line)
Delete network at the cursor and renumber all networks
Delete subprogram from program

**Status chart and symbol table: "Delete" submenu**

| | |
|---|---|
| You can: | Delete a selection |
| | Delete a chart or table |

**Data block: "Delete" submenu**

| | |
|---|---|
| You can: | Delete a selection |
| | Delete a data block |

**See also:**

Information on using the software (Page 375)

First steps: Contents (Page 17)

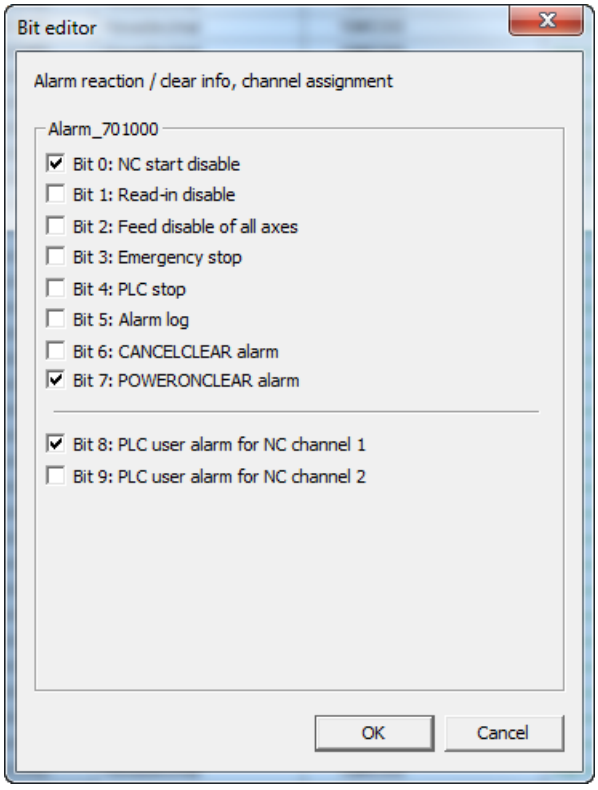
### 11.2.2.10 Bit editor (Edit menu)

Proceed as follows to call the **"Bit editor"**dialog:

- Double-click on the **"Initial value"** of the required alarm.
- Select the "**Initial value"** cell of the required alarm, and call menu**"Edit" > "Bit editor"**.
- Alternatively, you can right click on the cell of the initial value of the required user alarm, and select the **"Bit editor"**command from the shortcut menu.

The bit editor allows the configuration data to be conveniently edited. Using the bit editor, the actual alarm responses/delete criteria and channel assignments from the initial value can be displayed for a specific alarm. The values can also be selected or deselected there. The bit assignments correspond to the previous PLC user alarms.

The values are written back to the initial value by pressing "OK"




**Note**

Presently, the bit editor is only available for configuring the alarms.

### 11.2.2.11 Find / Replace / Go To (Edit menu)

The dialog **"Find/Replace/Go To"** corresponds to the standard dialog of the PLC Programming Tool.

Use one of the following methods to access the functions Find, Replace, and Go To:

- Select the **Edit > Find**, **Edit > Replace** or **Edit > Go To** menu command.
- Press the shortcut key **Ctrl+F**, **Ctrl+H** or **Ctrl+G**.

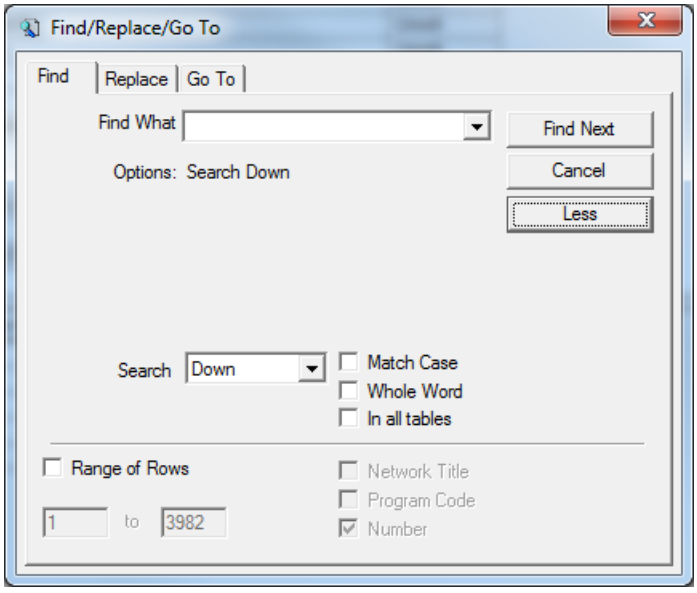



Figure 11-8 Find dialog

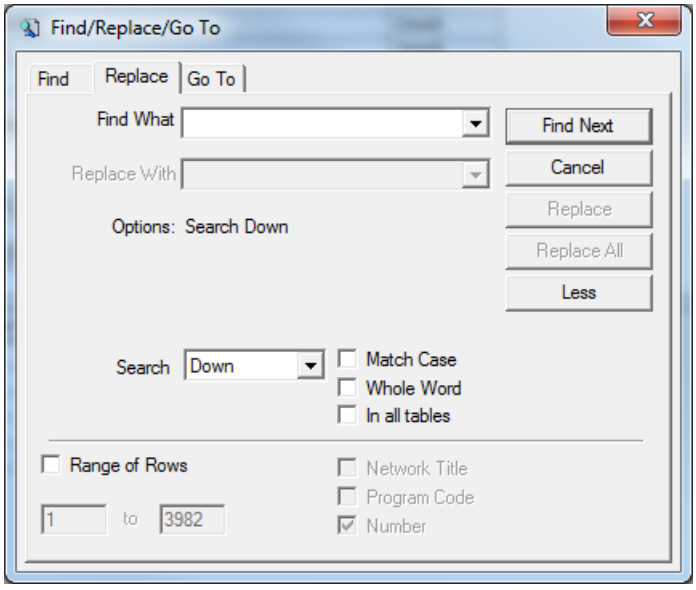



Figure 11-9 Replace dialog

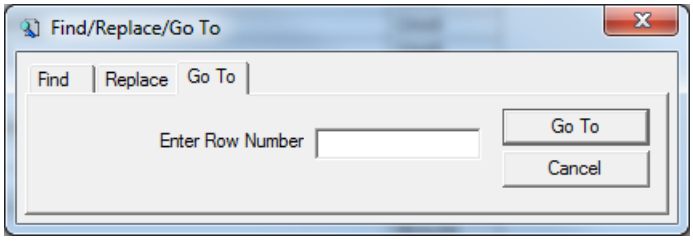



Figure 11-10 Go To dialog

Find, Replace, and Go To functions can be performed on a program, local variable table, program block, symbol table, data block or status chart. The user can also search for machine data numbers.

**See also:**

Find/Replace and Go To function (GS 3.8) (Page 46)

Information on using the software (Page 375)

First steps: Contents (Page 17)

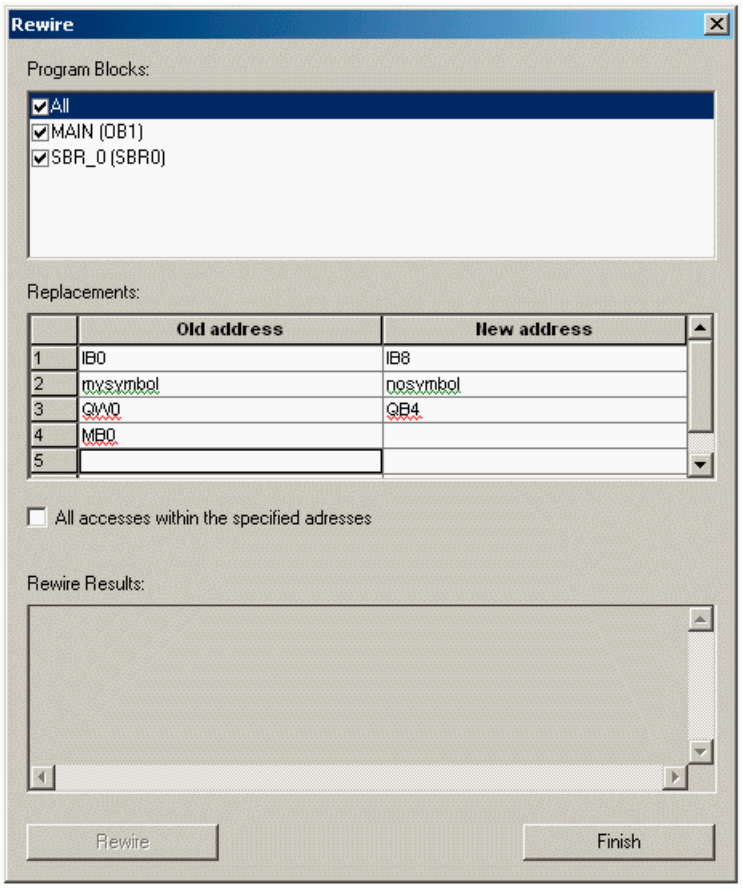
### 11.2.2.12 Rewire (Edit menu)

With the "Rewire" function, PLC user program operands can be centrally modified, e.g. IW0 to IW8. This means that user programs can be quickly adapted to a modified I/O configuration.

## Example

You are writing a PLC user program for a series of machines. As the I/O configuration of some machines differs due to the differing machine configurations, user program operands must currently be modified individually. Under certain circumstances, this can involve several hundred operands. A list of operands to be changed, e.g. inputs and outputs, can be entered for these machines via the "Rewire" dialog. This list is processed with the "Rewire" function and the operands are modified in the user program.

Use the "Rewire" dialog box to rewire operands:




## Procedure

1. **"Program blocks" list**
   Displays a list of all POUs available in the project. Select the POUs for rewiring.
2. **"Replacements" list**
   In this list, enter the old and the new operands for rewiring.
   Permissible operands include:
   - Inputs
   - Outputs
   - Bit memories
   - Special bit memories
   - Variable memories / data blocks
   - Timers
   - Counters

3. Editing the list of operands:
   The following functions are supported using the shortcut menu (right mouse button):
   - Cut (Ctrl+X)
   - Copy (Ctrl+C)
   - Paste (Ctrl+V)
   - Select All (Ctrl+A)
   - Insert Row (Ctrl+I)
   - Delete Selection

   This means that it is possible to copy the list or parts of the list from other or into other applications, e.g. Microsoft Excel.
4. Old operand
   In this column, enter the name or the address of the operand which you wish to rewire.
5. New operand
   In this column, enter the new name or the new address of the operand. Please observe that the type of the new operand corresponds to the type of the old operand, e.g. old operand IW0 and new operand IW4, not IB4, or old operand DB9000.DBB0 and new operand MB0, not MW0.
6. Check the validity of the operand
   If the name of the operand (symbol) does not exist in the open project, then this is marked with a green wavy line. If the type of the old operand does not match that of the new operand, or if only one operand was entered (old or new operand), then this operand is marked with a red wavy line.
7. **Selection box "All accesses within the specified operands"**
   If this option is activated, operand ranges (BYTE, WORD, DWORD) are rewired.
   Example: You specify IW0 and IW4 as operand ranges. In this case, the operands I0.0 – I1.7 are rewired to operands I4.0 – I5.7. Operands from the unwired range (e.g. I0.1) can then no longer be entered individually into the table.
8. **"Rewiring results" list**
   After rewiring, the results are displayed in this list. This list comprises the list of operands, "Old operand" and "New operand". These list the individual blocks and the number of wiring operations that were carried out in each block. Using the shortcut menu (right mouse button), the results can be copied into other applications, e.g. Microsoft Word.
9. **"Rewire" button**
   Press this button to start rewiring.
10. **"Exit" button**
    Press this button to exit rewiring.
11. **Activate the Rewire function:**
    The "Rewire" dialog is opened in the LAD editor in the screen shortcut menu or in the "Edit > Rewire..." menu.

**Note**

Please observe the following when rewiring:

- The name or number of a POE cannot be modified with the "Rewire" function. For this, use the functions "Rename" or "Properties..." in the POU shortcut menu in the project tree (right-click, e.g. MAIN).
- Timers can only be rewired to timers, e.g. old operand T0, new operand T16, and counters only to counters, e.g. old operand C0, new operand C1.

**See also:**

Information on using the software (Page 375)

First steps: Contents (Page 17)

## 11.2.3 View menu

### 11.2.3.1 STL

Instructions (Page 111)
Bookmarks in the STL program (Page 258)

**See also:**

Information on using the software (Page 375)

### 11.2.3.2 LAD

LAD instructions (Page 111)
Signal flow displays (Page 193)
Connecting instructions (Page 193)
Instruction parameters (Page 195)
General instructions (Page 197)
Transfer parameters (Page 197)
Entering instructions in LAD (Page 36)
Guidelines for creating networks in LAD (Page 34)
Entering comments for the program in LAD (Page 42)
Structuring a program using the Symbol Table (Page 201)
Displaying Symbol Information Tables in the LAD editor (Page 445)
Data types (Page 209)
DB address representation (Page 446)
Programming subprograms (Page 223)
Programming interrupt programs (Page 227)
Programming a Local Variable Table (Page 229)
Initializing data in the variable memory using a data block (Page 233)
Compiling projects (Page 244)
Displaying cross references of the used memory (Page 246)
Uploading a program into the PLC (Page 250)
Monitoring the status at runtime (Page 252)
Saving your work (Page 257)

**See also:**

Communication configuration (Page 443)

Selecting the CPU type (Page 469)

First steps in LAD (Page 17)

Information on using the software (Page 375)

### 11.2.3.3 Program block (View menu)

To create or change your program, edit the program block as follows:

- Click the button of the program block  in the "Navigation" toolbar (Page 500).
  This opens the POU of the main program (OB1). You can click one of the tabs "Subprogram" or "Interrupt Program" to open a different program organization unit.
- Open the program block directory in the project tree (Page 447) by clicking the button  or double-clicking the button.

  Then double-click the main program (OB1)  button, a subprogram  button or an interrupt program  button to open the desired program organization unit.
- Select the menu command **View > Program Block**.

Change the editor selection by using one of the following methods:

- Change the editor type with the menu command **View > LAD** or **STL** (only for view, no editing possible).

**See also:**

The LAD and STL program editors (Page 31)

LAD editor (help for the ladder logic) (Page 412)

STL editor (Page 411)

First steps in LAD (help topics) (Page 17)

### 11.2.3.4 Symbol table

Open a symbol table by using one of the following methods:

- Click the button of the symbol table  in the "Navigation" toolbar (Page 500).
- Select the menu command **"View > Symbol Table"**.
- Open the directory of the symbol table in the project tree (Page 447) and double-click the  button.

## Editing in a table

**Making symbol assignments in the symbol table**

To assign a symbolic name to an address, proceed as follows:

1. Open the symbol table with one of the described entry types.
2. Enter the symbolic name (e.g. Input1) in the "Name" column. The maximum symbol length is 23 characters. Use the Tab, Enter or arrow keys to confirm your work and move to the next field.

   **Note**

   - Until you assign an address to the symbol, it is shown as an undefined symbol (green wavy line). After you complete the Address column assignment, the green wavy line is removed.
   - If you have selected simultaneous symbolic and absolute view of addresses for your project, lengthy symbolic names are truncated with a tilde (~) in the LAD Program Editor window. You can place your mouse pointer over the truncated name to see the entire name displayed in a tooltip.

3. Enter the address (e.g. I0.0) in the "Address" column.
4. Enter a comment (optional: 79 characters maximum).

**Note**

In the Programming Tool, you can create multiple symbol tables. You cannot, however, use any character string more than once as symbolic name - neither within the same table nor distributed over multiple tables.

By contrast, it is permissible to use the same name in various local variable tables.

### Use of quotation marks is not supported

Global symbol names must not be set in double quotation marks. The symbol table interprets them as illegal syntax (the symbol is displayed in red until the quotation marks are removed).

### Inserting additional rows

Proceed as follows to insert additional rows into a symbol table:

- Select the menu command **"Edit > Insert > Row"**. A new row is inserted above the current location of the cursor in the symbol table.
- Right-click a field of the symbol table. In the shortcut menu select the command **"Insert > Row"**. A new row is inserted above the current location of the cursor.
- You can move the cursor to a field in the last row and press the down arrow key in order to insert a row at the bottom of the symbol table.

### Defining symbols

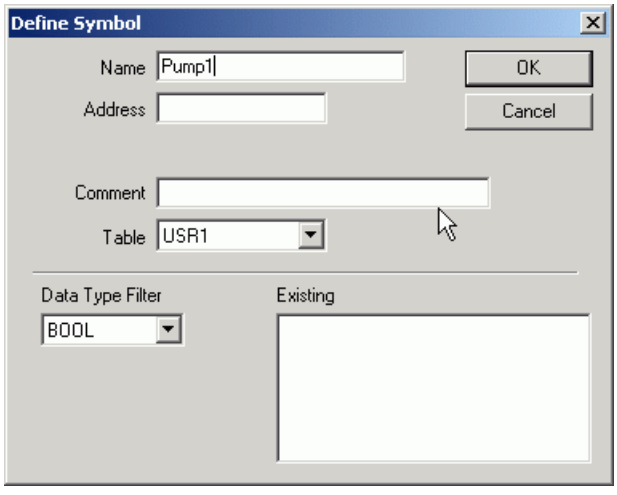
The "Define Symbol" command allows you to select an existing symbol from a list, and it also allows you to create new symbol assignments while you work in the Program Editor window or in a status chart, without having to open up the symbol table.

### Note

You cannot use the "Define Symbol" command unless you are working in the Symbolic Addressing view. (You can check this in the "View" menu: A check mark next to the "Symbolic Addressing" option means that this view is turned on.)

Using the "Define Symbol" command:

1. Enter at least one character of the desired symbolic name in the Program Editor window or a status chart address field and press the Enter key.
2. Right-click the incomplete (or incorrect) symbolic name, then select "Define Symbol" from the shortcut menu.
3. Choose an existing symbol from the list or define a new symbol in the dialog box "Define Symbol".




4. Click "OK" to confirm your work and close the dialog box or click "Cancel" to cancel.

**Note**

If you have selected simultaneous symbolic and absolute view of addresses for your project, lengthy symbolic names are truncated with a tilde (~) in the LAD Program Editor window. You can place your mouse pointer over the truncated name to see the entire name displayed in a tooltip.

#### Sorting by column

You can sort the table by the "Name" or "Address" column. You can sort a column in ascending or descending (alphabetical) order.

To sort a column in a particular order, select the column by clicking in the body. Avoid clicking on the column header ("Name" or "Address").

- To sort the column in ascending order, from A to Z, click the Sort ascending button or choose the menu command **"View > Sort Ascending"**.
- To sort the column in descending order, from Z to A, click the Sort descending button or choose the menu command **"View > Sort Descending"**.

To reverse the sort, click the column header ("Name" or "Address"). When you click the column header, you toggle between ascending and descending sorts.

### Working with multiple tables

#### Creating additional symbol tables

By default, the Symbol Table window displays one tab for user-defined symbolic names (USR1). If you have renamed any POUs, the Symbol Table window shows a tab called "POU Symbols".

There are several ways to create additional symbol tables for user-defined symbolic names:

- In the instruction tree, right-click the "Symbol Table" directory and, in the shortcut menu, select the command **Insert Symbol Table**.
- Open the window "Symbol Table" and call the menu "Edit" or right-click and select the command **Insert Contents > Table**.

**Note**

After you have inserted a new symbol table, a new tab 

is displayed at the lower edge of the "Symbol Table" window.

**Moving between symbol tables**

If you have created more than one symbol table, you can access the tables by any of the following methods:

- If the Symbol Table window is open, you just need to click the 

tab of the desired symbol table at the bottom of the Symbol Table window to move between tables.
- From the instruction tree, expand the Symbol Table directory icon and double-click the desired symbol table to view the tab for that table within the Symbol Table window.

**"POU Symbols" tab**

If you have assigned symbolic names for any of the POUs or other components (such as a status chart, data block or symbol table) in your project, the symbolic name assignments are listed on the "POU Symbols" tab of the Symbol Table window.

This tab is read-only; you cannot edit the assignments from here.

If you want to change an assignment, you must edit the Properties dialog box for the component in question. From the Project branch of the instruction tree, right-click the component to bring up the dialog box "Properties".

All symbolic assignments must use valid syntax. If you violate the guidelines for making a symbolic name assignment, the Programming Tool reports an error when you try to compile your program.

## Symbolic and absolute addressing

**Toggling between symbolic and absolute addressing mode**

After you make symbol and absolute address associations in a symbol table, you can toggle the display between symbolic and absolute addressing. To do this, proceed in one of the following ways:

- Select the menu command **"View > Symbolic Addressing"** to toggle symbolic addressing on and off
- Use the **<Ctrl+Y>** shortcut key to toggle symbolic addressing on and off

A check mark in front of the command "Symbolic Addressing" means that symbolic addressing is on. By default, symbolic addressing is on when you create the first project.

**Viewing symbolic and absolute addresses simultaneously**

To view symbolic addresses and absolute addresses simultaneously in an LAD program, use the menu command **"Tools > Options"** and open the tab "LAD Editing". Select the option button "Display symbol and address".

**Note**

Symbolic addresses are only displayed in your project when the Symbolic Addresses view is on. Otherwise, even if you select "Display symbol and address", only the absolute address is shown.

If you have selected simultaneous symbolic and absolute view of addresses for your project, lengthy symbolic names are truncated with a tilde (~) in the LAD Program Editor window. You can place your mouse pointer over the truncated name to see the entire name displayed in a tooltip.

## Symbols

**Understanding the symbol scope**

In the symbol table you can assign symbolic names to the addresses of inputs and outputs and to addresses in the automation system memory. When you define a symbol in the symbol table, the symbol has a global scope. This means you can use the name of the symbol in any POU as a reference to the data at the address of that symbol. By contrast, if you assign a symbolic name using a local variable table, the local variable is restricted in scope to the POU where it was defined.

**Memory types**

You can create symbolic names for the following memory areas: I, Q, M, SM, V, S, C, T.

**Entry errors**

- Illegal syntax - red text
  Example:

| | Name | Address |
|---|---|---|
| 1 | 2name | I0.0 |
| 2 | | VBB0 |

The first character cannot be a digit.
Invalid address, VB14000000 would be correct
Reserved keywords cannot be used as symbols

- Illegal use - red wavy line
  Example:

| | Name | Address |
|---|---|---|
| 1 | Pump1 | I0.0 |
| 2 | Pump1 | I0.0 |

Duplicate name
Duplicate address

- Undefined symbol - green wavy line
  Example:

| | Name | Adresse |
|---|---|---|
| 1 | Pumpe1 | |
| 2 | Pumpe2 | VB14000000 |

Name with invalid address

**Note**

- Symbolic names are permitted to contain alphanumeric characters and underscores. They are also permitted to contain extended characters (ASCII 128 to ASCII 255). The first character is restricted to alpha and extended characters only.
- When naming global symbols it is not permissible to use double quotation marks. The symbol table interprets them as illegal syntax (the symbol is displayed in red until the quotation marks are removed).
- It is illegal to use keywords (Page 281) as symbolic names, or to use names that begin with a number or contain characters that are not alphanumeric or in the extended character set.
- The maximum permissible length for a symbolic name is 23 characters.
- If you attempt to create a symbol assignment to correct the "Undefined Symbol" error in your program, but do not press the TAB key, ENTER key or an ARROW key after typing the Address value and before you return to your program, the error is still displayed. To resolve this problem, return to the Symbol Table window, place the cursor in the Address field, and press the TAB, ENTER or an ARROW key to complete the address assignment.

**Tips for creating symbols**

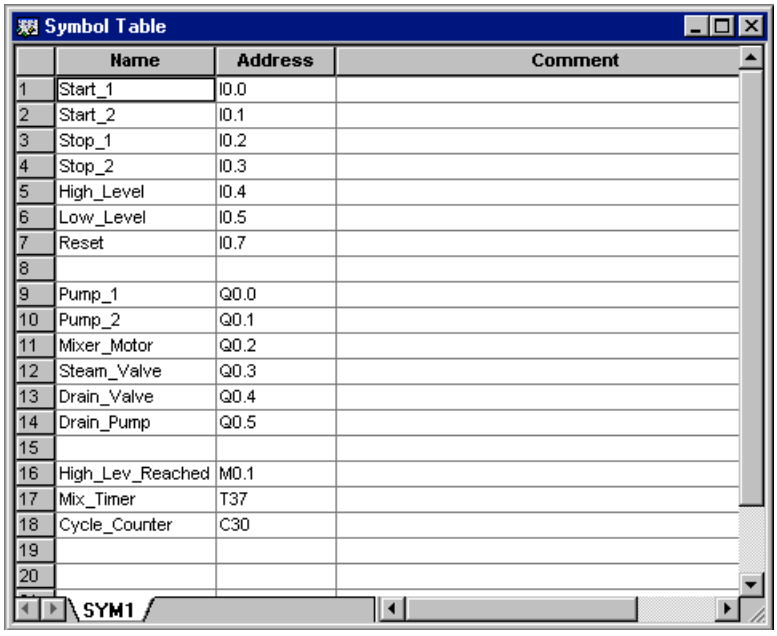
- You can make symbol assignments at any time: before, during or after the creation of program logic.
- You can document your symbols by making remarks in the "Comments" column.
- You can use the command "Define Symbol" to select from a list or create new symbolic names as you program.

- You can sort the values in the symbol table by clicking on the various table column headers.
- You can resize columns by dragging on their edges.
- You can use the menu command File > Print to print the symbol table.
- Once you have created symbolic assignments, you can display your address assignments and symbolic comments for each network in the LAD program editor by enabling the function "Symbol Information Table".

**Examples of tables**

Instruction operands are defined to accept one specific data type. For information about data types, see Data Types (Page 269).

This is an example of a symbol table:




## See also:

Local variable table (Page 229)

Symbolic addressing (Page 444)

Symbol information tables (LAD) (Page 445)

Overview of the addressing (Page 26)

Direct addressing (Page 261)

DB address representation (Page 446)

Keywords (Page 281)

### 11.2.3.5 Status chart

#### Precondition

After you have loaded your program into the PLC, you can generate one or multiple status charts to monitor and test program execution.

The program is continuously executed if the PLC is in the RUN mode. You can enable the chart status so that the status values in the chart are continuously updated (not interrupted). As an alternative, using the function "Single Read", you can generate a "snapshot" of the status values in the chart without having to enable the status chart.

While you are viewing a status chart, you can also switch the PLC into the STOP mode and only execute the first or a certain number of scan cycles, in which you monitor program execution.

**Note**

Please note that you cannot change your chart if the chart status is enabled.

Disable the chart status if you wish to edit the chart.

These help topics are explained in the following:

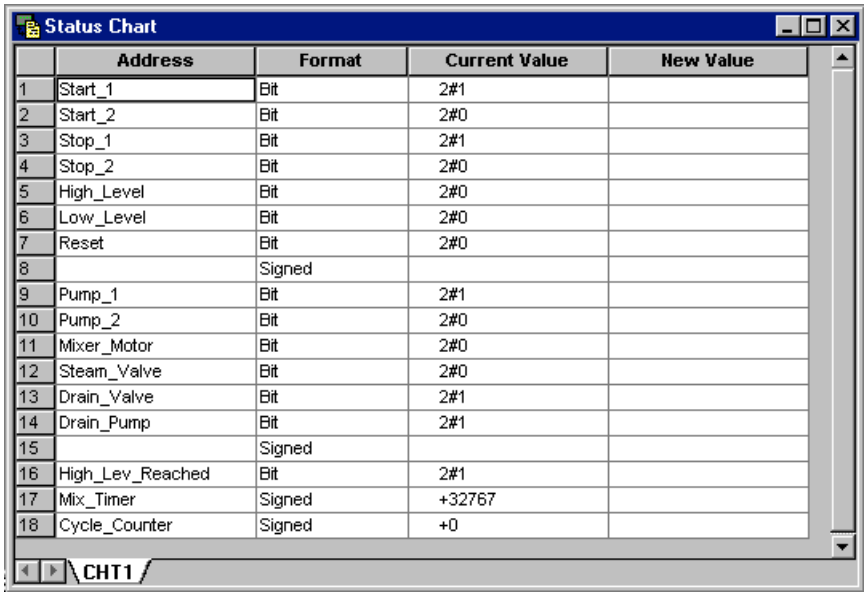
#### Example for the status chart

Opening a status chart is not the same as enabling a status chart. You can open and evaluate or change a status chart: However, if you do not execute the command "Single Read" (in the menu "Debug" or in the toolbar) or enable the chart status (in the menu "Debug" or in the toolbar), no status information will be displayed in the column "Actual Value".

If you use the function "Single Read" (this is only available when the chart status is disabled) to evaluate a status chart, the actual values of the PLC are accepted and displayed in the column "Actual Value". However, the values are not updated while the PLC is executing the program.

If you enable the chart status (in the menu "Debug" or in the toolbar), the actual values of the PLC are regularly updated. If changes are received from the PLC, then the column "Actual Value" is updated.

You can assign (write) certain values in the PLC using the column "New Value".




## Opening or enabling a status chart

Open a status chart to evaluate or change the contents of the chart. Enable the status chart so that the status information can be updated.

To open a status chart, proceed in one of the following ways:

- Click on the button of the status chart in the "Navigation" toolbar (Page 500)
- Select the menu command **View > Status Chart**.
- Open the directory of the status chart in the project tree (Page 447) and double-click the symbol of a table.
- If your project includes more than one status chart, using the tab for the CHT1 CHT2 CHT3 status charts at the lower edge of the window, you can switch over between the individual charts.

To enable a status chart, proceed in one of the following ways:

- If you wish to continually update the status information in the status chart, enable the chart status: Select the menu command **Debug > Chart Status** or click the appropriate button in the toolbar.
- If you only require a "snapshot" of the values, execute the function "Single Read": Select the menu command **Debug > Single Read** or click the appropriate button in the toolbar (if the chart status is enabled, then the function "Single Read" is deactivated).

**Tips:**

If you open a status chart, then the status is still not displayed. You must enable the status chart so that the status information is updated.

Enabling the status chart will have no effect if the chart is still empty: You must first create your status chart by entering program values (operands) in the column "Address" and entering a data type in the column "Format" for each address.

### Working with multiple status charts

You can create additional status charts in several ways:

- In the instruction tree, right-click the "Status Chart" folder and, in the shortcut menu, select the command **Insert Status Chart**.
- Open the window "Status Chart" and call the menu "Edit" or right-click and select the command **Insert Contents > Chart**.

**Note**

- After you have inserted a new status chart, a new tab 

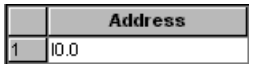
is displayed at the lower edge of the "Status Chart" window. If you wish to switch between the status charts, simply click the tab of the required status chart.
- Sometimes, a tab is hidden by the buttons on the right-hand side that are used to scroll. If a tab is not visible, drag the demarcation line between the tab area and the scroll buttons to display additional tabs.

### Creating a status chart

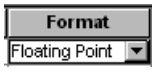
You can enter addresses in a status chart in order to monitor and control values from your program. Values of timers and counters can be displayed as bits or words. If you wish to display the value of a timer or a counter as a bit, then the status of the output is displayed (on or off). If you wish to display the value of a timer or a counter as a word, then the actual value is used.

To create a status chart, proceed as follows:

1. In the address field you enter the addresses of the desired values.
   Most of the memory types listed in the memory areas of the automation system (Page 262) are valid, with the exception of data constants and accumulators.
   To edit an address field, select the desired field using the cursor keys or the mouse. If you enter data, existing data are deleted and the new characters are entered. The field is selected if you double click with the mouse or press key "F2". You can then move the cursor using the cursor keys to the position that you wish to edit.


2. If the element involves a bit (e.g. I, Q or M), then the bit format is displayed in the second column. If the element involves a byte, word or double word, select the field in the column "Format" and double-click or press the space bar or ENTER in order to scroll through the valid formats until the correct format is displayed.


To insert a new row, open the menu "Edit" or right-click a field in the status chart so that the shortcut menu is displayed and select the command **Insert Contents > Row**. A new row is then inserted in the status chart above the cursor position. You can also move the cursor to a field in the last row and press the down arrow key in order to insert a row at the bottom of the status chart.

**Tips:**

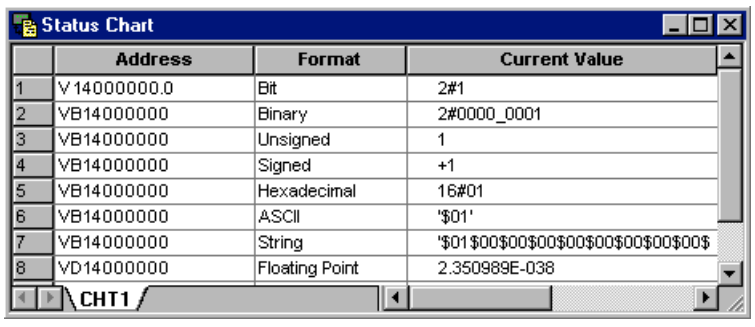
You can select addresses in the symbol table and copy these into the status chart in order to generate your chart faster.
You can display the status a multiple number of times. You can classify the elements in logical groups to display each group in an individual chart. In this way, you avoid having to scroll through extremely long lists.

## Shortcut key combinations for editing

- To insert a new row with the following address and the same data format, select an address field and press the Enter key.
- To scroll through the possible data formats for a specific address, select the field "Data Format" and press the Enter key several times. To display all available data formats, open the drop-down list box.
- Use the TAB key to jump into the next field of the chart.
- To set the width of a column, position the mouse cursor on the edge of a column until the appearance of the cursor changes and then drag to increase or decrease the width of the column.
- To select a complete row (to cut or copy), click once the number of the row.
- To insert a new row, select a field or a row and right-click. In the shortcut menu select the command **Insert Contents > Row**. The row is inserted at the cursor position. The subsequent rows are shifted down by one row.
- To insert a row at the bottom of the status chart, move the cursor to a field in the last row and press the down arrow key.
- To delete a field or a row, select the field or the row and right-click. Select the menu command **Delete > Selection**. If you delete a row, the subsequent rows are shifted up by one row.
- To select the entire status chart, click once the upper left corner above the row numbers.

## Data formats

The data format that you assign to a value defines how the value is represented in the status chart.
In the example below, VD14000000 (and therefore VB14000000 and V14000000.0) contains the value 1.



**Tips:**

Bit and binary values are both introduced by the number 2 and the # symbol. Hexadecimal values are introduced by the number 16 and the # symbol. Bit values have one digit. Binary values have eight digits.
Signed and unsigned values use the basis 10 (decimal).

### Single read or continuous chart status

- If you only require a "snapshot" of the values, execute the function "Single Read": Select the menu command Debug > Single Read or click the appropriate button in the toolbar  (if the chart status is enabled, then the function "Single Read" is deactivated).
- If you wish to continually update the status information in the status chart, enable the chart status: Select the menu command Debug > Chart Status or click the appropriate button in the toolbar  .

### Working with test functions in the status chart

You access the test functions (Single Read, Write All) using the menu Debug or using the toolbar with the test functions.

| | |
|---|---|
|  | **Single Read**<br>Use Single Read if you require a "snapshot", i.e. a single update of the program status of all values. By default, the chart status continually scans the PLC for status updates. If you click a status chart and the chart status is disabled, then the button for Single Read is activated. |
|  | **Write All**<br>After you have entered the values in the column "New Value" in the status chart, write the required changes to the PLC using the command "Write All". |

### See also

Overview of test and monitoring functions (GS 5.1) (Page 55)

Displaying the status in the editors (GS 5.2) (Page 60)

Executing a certain number of scan cycles (GS 5.4) (Page 73)

Displaying the data block status (GS 5.5) (Page 74)

Downloading a program to the CPU (GS 4.3) (Page 52)

Cross-references and elements used (Page 439) (Ensure that program changes do not produce duplicate assignment.)

RUN/STOP modes of the PLC (Page 457)

## 11.2.3.6 Data block

If you wish to work with data blocks, then select a CPU (Page 469) that supports data blocks.

### CPUs that support data blocks


| | |
|---|---|
| 828D | 808D-PPU14x (only special data blocks) |
| 828D Step 2 | 808D-PPU15x (as of PLC 07.03) |
| | 808D-PPU16x (as of PLC 07.03) |

### Procedure

Use one of the following methods to access the data block:

- Click the button of the data block  in the "Navigation" toolbar (Page 500).
- Select the menu command **View > Data Block**.
- Click the button of the data block  in the project tree (Page 447).

These help topics are explained in the following:

### Data block properties

The data block structures are, just like the POUs, part of the project. They are compiled, saved, imported or exported with the project. They are loaded into/from the PLC with the project. The data blocks themselves only contain actual values. Actual values may be loaded into/from the PLC independently of the project. It is therefore essential that the structure of the data block to be loaded in the PLC is exactly the same as that of the project opened in the Programming Tool. Data blocks are listed like POUs in their own symbol table.

When changing the CPU, existing data blocks are not lost. However, if you select a CPU in which data blocks are not available, then please note that the variables from the data block are now displayed with their absolute address (e.g. DB9000.DBB0 or VB90000000). Whether this V range exists is first checked when the PLC is brought into the RUN state. In this case, the program cannot be executed.

To rename your data blocks and change their properties (Page 452), right-click the corresponding data block in the operation tree. Select "Properties". The "Data Block Properties" dialog box opens. Here, you can change the name, block number, and data class (Page 243), as well as add an author and comments. You can also activate the property "Non-retain" for data blocks. This ensures that every time the PLC is brought into the RUN state, the data block in the PLC is preset with the initial values. Data blocks can be write-protected, that is, the actual values of the data block cannot be changed with the Programming Tool. This property cannot be changed.

**Note**

If you just want to rename your data block, right-click the object you want to rename in the operation tree and select "Rename".

You can also easily edit your data block in another program (e.g. Microsoft Excel) (see Cut, Copy and Paste section in the data block editor).

### Assigning addresses and initial values in the data block

When you create a variable, you assign its name and data type. The default initial value is set to zero/OFF, however, this can also be changed. You can optionally insert comments. The Programming Tool automatically assigns the address. Each address is aligned according to the size of its data type, which means that gaps can occur. The actual values are edited in the data view. However, you may assign the initial values to all actual values.

The maximum size of a data block is limited (512 bytes), if it is exceeded then the Programming Tool marks the excess variable addresses using a red wavy line. The Programming Tool returns an error when compiling the project.

### Assigning actual values

Actual values are saved, exported or imported in your project file as part of your data block. You can only assign individual actual values in the data view. The structure of the variable (name, data type, initial value, and comment) is write-protected in the data view. You can only change the actual value and therefore specifically set initial values, for example. After loading the data block (only actual values), these values become effective in the PLC.

The structure of the data block in the project must match the structure of the data block in the PLC. If you have modified the structure of the data block, then the complete project has to be loaded again.

You can access the data view in the following ways:

- Click the Display data block values  button in the toolbar.
- Select the menu command **View > View Data Block Values**.

### Accepting initial values

If you accept your initial values, then all of the actual values of your data block are overwritten. If you have not selected an initial value for a variable, then the Programming Tool sets it to zero/OFF. Other data blocks will not be changed.

In order to reset all actual values in the actual data block to the initial values,

- Right-click in the table > Accept Initial Values.
- Select the menu command **Edit > Accept Initial Values**.

### Displaying and changing actual values

You can view the actual values of a data block in the PLC with the data block status (Page 369) and, if the data block is not write-protected, change them too. The structure of the data block in the project must match the structure of the data block in the PLC. If you have modified the structure of the data block, then the complete project has to be loaded again. When the data block status is activated, the structure of the variable (name, data type, initial value, and comment) is write-protected. If you switch on the data block status, the values in the column "Actual Value" are regularly updated with the actual values of the PLC.

After you have entered values in the column "New Value", write the required changes to the PLC using the command "Write All"  .

You can activate the data block status in the following ways:

- Click the "Data block status" button in the toolbar.
- Select the menu command **Debug > Data Block Status**.

## Example for a data block

Data Block

| | Address | Name | Data Type | Format | Initial Value | Comment |
|---|---|---|---|---|---|---|
| 1 | 0.0 | myByte1 | BYTE | Unsigned | 0 | Comment byte 1 |
| 2 | 1.0 | myByte2 | BYTE | Unsigned | 0 | Comment byte 2 |
| 3 | 2.0 | myWord1 | WORD | Unsigned | 0 | Comment word 1 |
| 4 | 4.0 | myInt1 | INT | Signed | +0 | Comment integer 1 |
| 5 | 6.0 | myBit1 | BOOL | Bit | OFF | Comment bit 1 |
| 6 | 6.1 | myBit2 | BOOL | Bit | OFF | Comment bit 2 |
| 7 | 6.2 | myBit3 | BOOL | Bit | OFF | Comment bit 3 |
| 8 | 6.3 | myBit4 | BOOL | Bit | OFF | Comment bit 4 |
| 9 | 6.4 | myBit5 | BOOL | Bit | OFF | Comment bit 5 |
| 10 | 6.5 | myBit6 | BOOL | Bit | OFF | Comment bit 6 |
| 11 | 6.6 | myBit7 | BOOL | Bit | OFF | Comment bit 7 |
| 12 | 6.7 | myBit8 | BOOL | Bit | OFF | Comment bit 8 |
| 13 | 8.0 | myDword1 | DWORD | Unsigned | 0 | Comment double word 1 |
| 14 | 12.0 | myDint1 | DINT | Signed | +0 | Comment double integer 1 |
| 15 | 16.0 | myReal1 | REAL | Floating Point | 0.0 | Cpmment floating point 1 |

DB_9000 DB_9001 DB_9002

## Addressing

### Direct addressing

- **Absolute address input**
  Input in absolute format, the address of a variable in your data block comprises the data block number (e.g. DB9000), a period, and the address of the variable. However, you can also directly access the variable via a V address (e.g. V9000000.0). The display of the absolute address depends on how the address display is set and this can be selected in the menu **View > DB Address View (Ctrl + B)**.

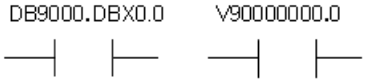



Figure 11-11 Absolute addressing using the data block number

- **Symbolic address input**
  In the symbol table, you can assign an additional name to the variables from your data block. If you wish to use your variable, then you no longer have to specify the complete address - a name in the symbol table is sufficient.
  If you have not listed your variable from the data block in your symbol address, then the symbolic address comprises the name of the corresponding data block (e.g. DB_9000) and the name of the variable in the data block (e.g. Ax1).

| Address | Name of the variables in the data block | Entry in the symbol table | Absolute representation | Symbolic representation |
|---|---|---|---|---|
| VB90000000 | VarName | FeedCorrAx1 | VB90000000 | FeedCorrAx1 |
| or | | | | |
| DB9000.DBB0 | VarName | FeedCorrAx1 | DB9000.DBB0 | FeedCorrAx1 |
| VB90000000 | VarName | | VB90000000 | NameDB.VarName |
| or | | | | |
| DB9000.DBB0 | VarName | | DB9000.DBB0 | NameDB.VarName |

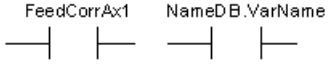



Figure 11-12 Symbolic address representation

The switch between the absolute and symbolic representation is carried out using the menu command **View > Symbolic Addressing (Ctrl + Y)**.

### Indirect addressing

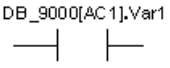
You may be able to simplify the programming if data blocks with the same structure are used in your program. You can indirectly address your data blocks using the accumulators (AC0 to AC3). The accumulators are used to inform the program which data block is to be handled. The value in the AC is then treated as an index.

For example, for axis DBs, the program text can be somewhat reduced by not having to write a dedicated program for each axis, but instead accessing the appropriate axis via the various data blocks and the index (AC). As this value is treated as an index, it starts at 0 for the first axis. Indirect addressing is only possible using ACs. The address cannot be broken down, as

the contents of the accumulator are not known offline. This also means that the absolute address cannot be determined. V addresses cannot be used for indirect addressing.

**Note**

Accumulators are used by both subprograms and the calling program. Calling a subprogram does not cause the accumulators to be saved or restored.




| Absolute input | Symbolic input |
|---|---|
| DB3800[AC1].DBX2.1 | ToAxis[AC1].ControlEnable |
| DB9000[AC0].DBW0 | Prototyp1[AC0].MyWord1 |

**Note**

**Not permitted:**

DB3800[1].DBX2.1 ToAxis[5].ControlEnable

**Indirect addressing using V addresses is not permitted:**

V3800[5]0002.1

V380[5]0002.1

V3800[AC0]0002.1

V38[AC0]0002.1

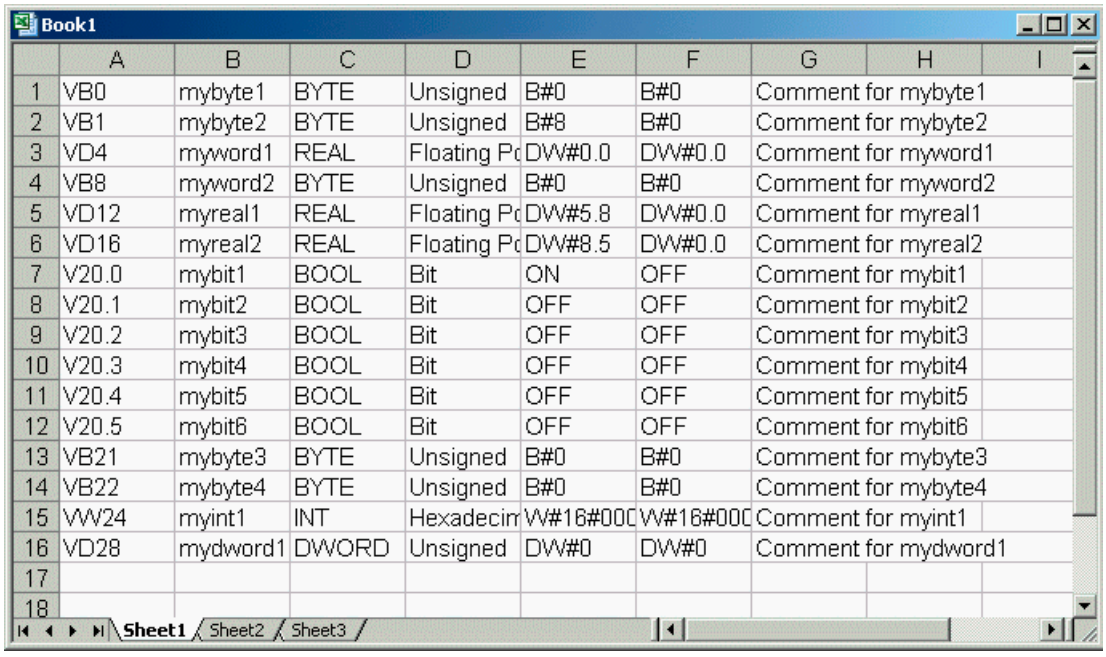
## Using Cut, Copy and Paste in the data block editor

You can save data (rows, columns, and cells) of your data block to the Microsoft clipboard in order to edit it in a different program. You can do this in one of the following ways:

- By right-clicking Cut/Copy/Paste
- By pressing the shortcut keys Ctrl+X/Ctrl+C/Ctrl+V
- By using the menu commands **Edit > Cut**, **Edit > Copy, Edit > Paste**

| | |
|---|---|
| Cut | Selected data is copied to the clipboard and is deleted |
| Copy | Selected data is copied to the clipboard and is not deleted |
| Paste | If data is in the clipboard, the data is pasted |
| Delete | Selected data block/section is deleted, it is not saved in the clipboard |

You can easily edit your data block in this way, e.g. in Microsoft Excel.



Book1

| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 1 | VB0 | mybyte1 | BYTE | Unsigned | B#0 | B#0 | Comment for mybyte1 | | |
| 2 | VB1 | mybyte2 | BYTE | Unsigned | B#8 | B#0 | Comment for mybyte2 | | |
| 3 | VD4 | myword1 | REAL | Floating Pc | DW#0.0 | DW#0.0 | Comment for myword1 | | |
| 4 | VB8 | myword2 | BYTE | Unsigned | B#0 | B#0 | Comment for myword2 | | |
| 5 | VD12 | myreal1 | REAL | Floating Pc | DW#5.8 | DW#0.0 | Comment for myreal1 | | |
| 6 | VD16 | myreal2 | REAL | Floating Pc | DW#8.5 | DW#0.0 | Comment for myreal2 | | |
| 7 | V20.0 | mybit1 | BOOL | Bit | ON | OFF | Comment for mybit1 | | |
| 8 | V20.1 | mybit2 | BOOL | Bit | OFF | OFF | Comment for mybit2 | | |
| 9 | V20.2 | mybit3 | BOOL | Bit | OFF | OFF | Comment for mybit3 | | |
| 10 | V20.3 | mybit4 | BOOL | Bit | OFF | OFF | Comment for mybit4 | | |
| 11 | V20.4 | mybit5 | BOOL | Bit | OFF | OFF | Comment for mybit5 | | |
| 12 | V20.5 | mybit6 | BOOL | Bit | OFF | OFF | Comment for mybit6 | | |
| 13 | VB21 | mybyte3 | BYTE | Unsigned | B#0 | B#0 | Comment for mybyte3 | | |
| 14 | VB22 | mybyte4 | BYTE | Unsigned | B#0 | B#0 | Comment for mybyte4 | | |
| 15 | VW24 | myint1 | INT | Hexadecim | W#16#000 | W#16#000 | Comment for myint1 | | |
| 16 | VD28 | mydword1 | DWORD | Unsigned | DW#0 | DW#0 | Comment for mydword1 | | |
| 17 | | | | | | | | | |
| 18 | | | | | | | | | |

Sheet1 Sheet2 Sheet3

## Using special data blocks

The Programming Tool offers you the possibility to use pre-configured data blocks for special applications. These special data blocks can be found in the operation tree under **Libraries > Special Data Blocks**. These data blocks have a fixed structure. You can make use of them by double-clicking or via copy and paste (in the operation tree).

## Resolving errors

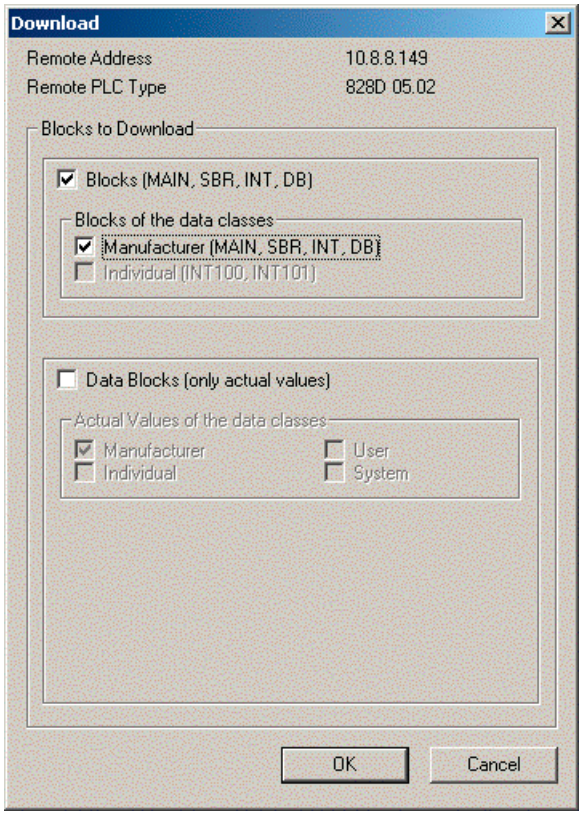
The Programming Tool marks errors made during data input as is the case in the LAD editor or the symbol table. For example, the use of invalid syntax or invalid values can cause an error. You must correct all of the errors that occur before you can compile (Page 244) without any errors and load the program into the PLC.

If any errors occur during compilation of the data block, then they are displayed in the output window. Position the cursor on an error message in the output window and double-click it so that you see the row in the data block with the error.

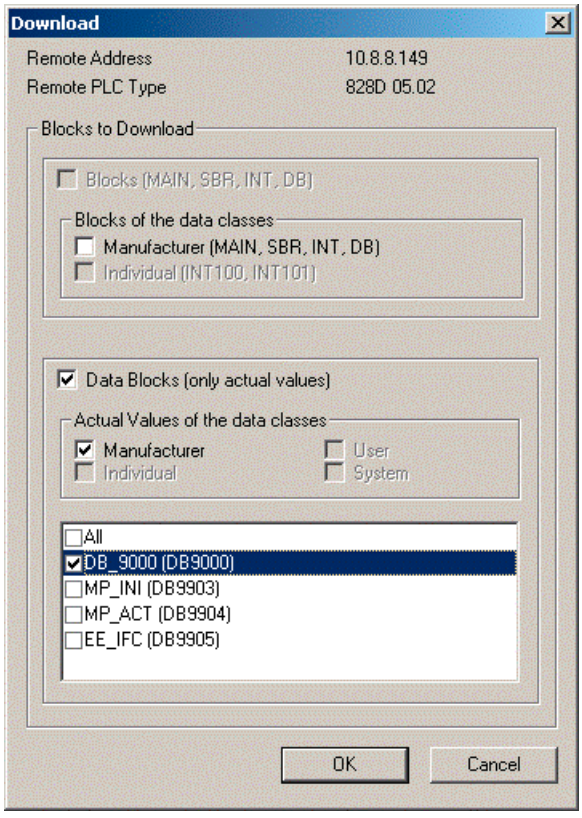
## Loading the data block

### Loading the data block into the PLC

If you edit the structure of a data block, you need to subsequently load the program block (project) into the PLC (Page 250). Your changes will only become effective after the modified project has been loaded into the PLC. Actual values and comments are not loaded into the PLC with the project. If the PLC identifies that a data block is new or has been changed, then it sets the initial values for this data block as the first actual values.

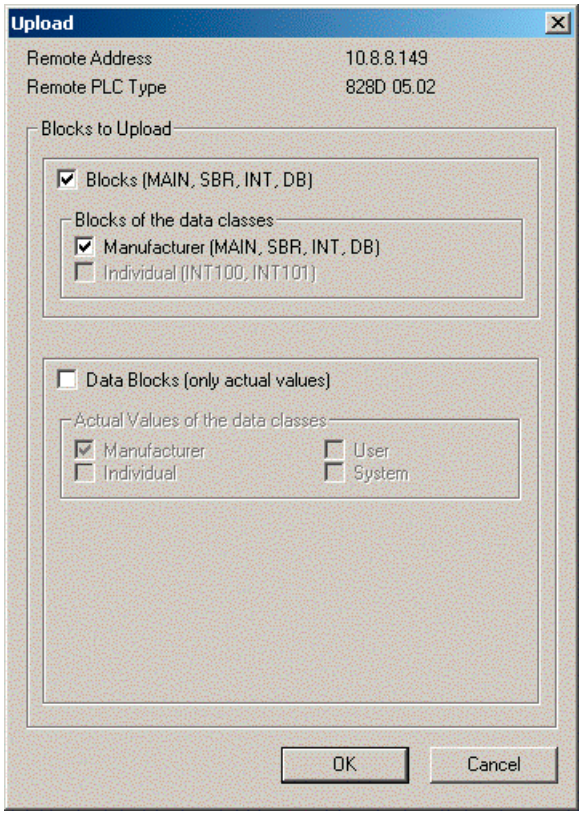



If you edit the actual values of a data block, you need to subsequently load the data block (actual values only) into the PLC (Page 250) so that these actual values become valid. The structure of the data block in the PLC must match the structure of the data block in the project.

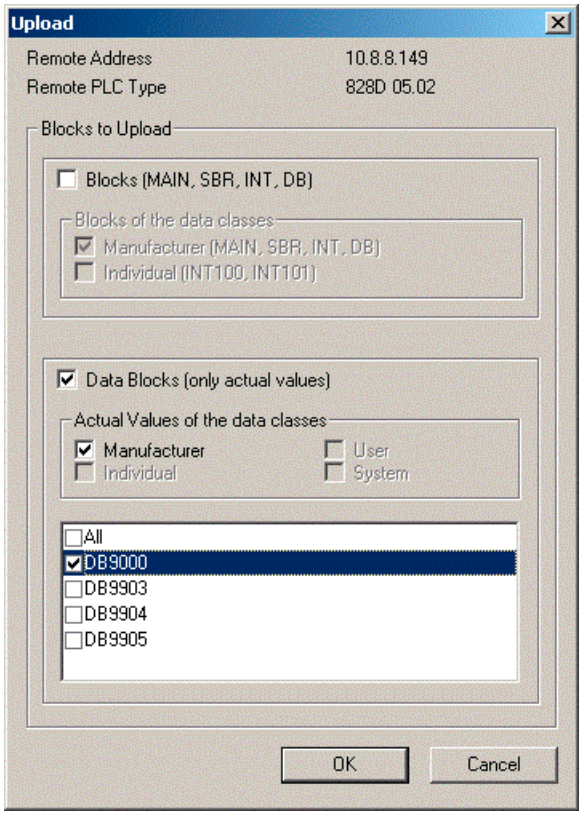



**Loading a data block from the PLC**

You must first open a project in the Programming Tool before you can load the program block (project) and therefore the structure of the data blocks contained in it from the PLC (Page 353).




If you only want to load the actual values of a data block from the PLC, then first open the corresponding project in the Programming Tool or load it from the PLC. Now you can load the data block (only actual values) from the PLC (Page 353). You cannot load the data block if the structure of the data block in the PLC does not match that of the data block of the project that has been opened (or if the project that has been opened does not have a data block).




## See also

Display the data block status (Page 74)

Constants (Page 271)

Select a target CPU (model and version) (Page 334)

Communication (Page 443)

Getting started (Page 17)

Information on using the software (Page 375)

### 11.2.3.7 NC variables (View menu)

If you wish to read or write NC variables from your user program and the CPU that you are using supports the NC variables editor, you can use it to compile a list of selected NC variables. Open the NC variables editor in one of the following ways:

- Click the button of the NC variables editor in the "Navigation" toolbar (Page 500).
- Select the "View > NC Variables" menu command.
- Click the button of the NC variables list in the project tree (Page 447).

These help topics are explained in the following:

## Properties of the NC variable lists

The following variable lists are available for CPUs that support the NC variables editor:

- NC variables
- Drive parameters

For each variable, these lists contain the following information:

- Area
- Block
- Variable name
- Type

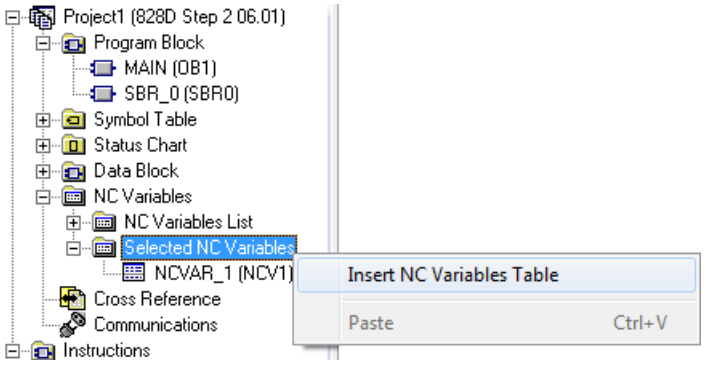
These variable lists are only in the PLC programming tool, and under certain circumstances, are dependent on the PLC programming tool version. They are not saved in the project, and therefore they are also not exported or downloaded to the PLC.

**Note**

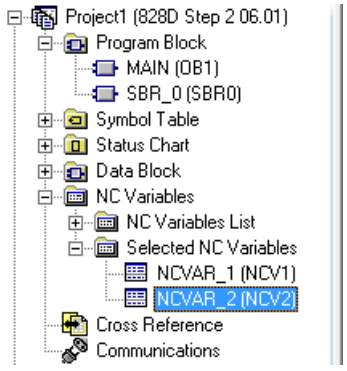
Because the PLC programming tool does not know the degree of expansion of the PLC, the lists can contain variables that cannot be read or written by the user program, or whose read values are invalid or whose written values are not effective. The user is responsible for correctly selecting the variables.

## Creating a variable table

To add an NC variable table, right-click on the "Selected NC variables" directory and then select "Insert NC variable table".




A new variable table is created below the existing table and contains consecutive numbering.




**Note**

**Maximum number of variable tables**

Please observe that up to 3 variable tables can only be created for the CPU type 828D Step 2.

## Selecting variables

Select the required variable list, by opening the corresponding tab at the lower edge of the variable table. Click the required variable or select the lines of the required variables. To copy the selected variables, select the "Edit" menu or right-click and select the Copy command or use the Ctrl+C shortcut key. Now click on a line in the table of the selected variables below or select the required lines there. To insert or overwrite the selected variables, select the "Edit" menu or right-click and select the Paste command or use the Ctrl+V shortcut key.

A double-click on a variable in the Variables Table copies it at the end of the table of the selected variables.
When variables are copied into the table of the selected variables, a dialog for editing the variable parameters opens automatically. This means that you can enter or change variable parameters.

The variable lists are very large. A filter can be activated to limit to the desired topic group. The list of the selected NC variables is part of the project. It is compiled, saved, imported or exported with the project. It is downloaded and uploaded to/from the PLC.

A maximum of 42 variables may be selected.

**Note**

If a variable whose particular type is not supported by the actual CPU types is selected, then an error message is output. The variable is not transferred into the list of selected variables.

## Editing variable parameters

The parameters required for a variable can be edited in the Variable Parameters dialog. These can be the area number, line and column. Please refer to the user documentation for the significance of the individual parameters. Every variable is automatically allocated an alias name, this can be changed. See also: "Editing NC variable parameters (Edit menu)" (Page 403).

You can obtain additional information on a variable, by clicking the "Help" button in the variable parameter dialog.

The system of units of the NC variables can be set in the options. To do this, select the menu command Tools > Options and open the "NC variables" tab. See also: "Tab "NC variables" (Tools > Options)" (Page 493).

## Cutting, copying and pasting NC variables editor

You can save the lines of your NC variable list in the Microsoft clipboard to edit them in a different program. You can do this in one of the following ways:

- By right-clicking, Cut/Copy/Paste
- By pressing the shortcut keys Ctrl+X/Ctrl+C/Ctrl+V
- Using the menu commands Edit > Cut, Edit > Copy, Edit > Paste

Cut: Selected lines are copied into the clipboard and deleted.
Copy: Selected lines are copied into the clipboard and are not deleted.
Paste: If lines are in the clipboard, then they are pasted.

You can easily edit NC variables in this way, e.g. in Microsoft Excel.

## Example of a list of selected NC variables

NC Variables

| | Area | Block | Variables Name | Type |
|---|---|---|---|---|
| 1 | A[.] | M | AA_OFF_MODE | CHAR |
| 2 | A[.] | M | ACCEL_REDUCTION_FACTOR | REAL64 |
| 3 | A[.] | M | ACCEL_REDUCTION_SPEED_POINT | REAL64 |
| 4 | A[.] | M | ACCEL_REDUCTION_TYPE | CHAR |
| 5 | A[.] | M | ACCEL_TYPE_DRIVE | BOOL |
| 6 | A[.] | M | ACT_POS_ABS[.] | REAL64 |
| 7 | A[.] | M | AC_FILTER_TIME | REAL64 |
| 8 | A[.] | M | AUTO_GET_TYPE | CHAR |
| 9 | A[.] | M | AXCONF_ASSIGN_MASTER_CHAN | CHAR |
| 10 | A[.] | M | AXIS_DIAGNOSIS | DWORD |
| 11 | A[.] | M | AX_EMERGENCY_STOP_TIME | REAL64 |
| 12 | A[.] | M | AX_INERTIA | REAL64 |
| 13 | A[.] | M | AX_JERK_ENABLE | BOOL |
| 14 | A[.] | M | AX_JERK_TIME | REAL64 |
| 15 | A[.] | M | AX_MASS | REAL64 |
| 16 | A[.] | M | AX_MOTION_DIR | DWORD |
| 17 | A[.] | M | AX_VELO_LIMIT[.] | REAL64 |
| 18 | A[.] | M | BACKLASH[.] | REAL64 |
| 19 | A[.] | M | BERO_CYCLE[.] | DWORD |
| 20 | A[.] | M | BERO_DELAY_TIME_MINUS[.] | REAL64 |

NC Variables

| | Area | Block | Variables Name | S7 Alias Name | Type |
|---|---|---|---|---|---|
| 1 | A[1] | M | IS_ROT_AX | A1_M_IS_ROT_AX_30300 | BOOL |
| 2 | B | S | opMode | B_S_opMode_3 | INT |
| 3 | C | M | MM_NUM_R_PARAM | C_M_MM_NUM_R_PARA_28050 | DWORD |
| 4 | N | M | FASTIO_DIG_NUM_OUTPUTS | N_M_FASTIO_DIG_NUM_OUTP | CHAR |
| 5 | V[1] | M | r0035 | V1_M_r0035_35 | REAL |

### Compiling NC variables

In order that in the PLC user program selected NC variables can be used, it is necessary to compile the NC variables. The DB9910 (NC_DATA) data block is created for an error-free compilation. This is write-protected, and assigned to the data class "Manufacturer". All of the necessary information so that the PLC user program can read or write NC variables is provided in this data block.

### Using NC variables

Using the data of DB9910 (NC_DATA), which was generated when compiling in the PLC programming tool and was loaded with the project into the PLC, the user interface "Read/write NC variables" can be simply parameterized in the PLC user program. Please refer to the user documentation for the description of the user interface. To specify the job, only the variable index, saved under the alias name, is entered in the variable index of the NC data interface (DB120x.DBB1000).

**Tip:**

If the variable is a field, whose individual elements can be addressed via the line index, then it is sufficient to accept this variable once in the list of selected variables. In this case, a zero is entered for the line in the parameterizing dialog. If this variable is used, in addition to the variable index (DB120x.DBB1000) for the line index (DB120x.DBW1002), the user must also enter the required line in the user interface "Read/write NC variable", for instance, for R parameter R parameter number + 1. If the line is not entered, an error message is output when reading/writing.

**Note**

For the "REAL64" data type, they represent a 64-bit floating-point value; the CPU converts the value because it supports only 32-bit floating-point values ("REAL").

### Resolving errors

The programming tool marks errors when creating variables. For example, using an alias name several times can result in an error. You must correct all of the errors that occur before you can compile without any errors and load the program into the PLC.

If any errors occur when compiling NC variables, then they are displayed in the output window. Position the cursor to an error message in the output window, and double-click it so that you see the line with error in the NC variables editor.

### Loading the data block to the PLC

If you edit NC variables, you need to subsequently download the program block into the PLC. Your changes will only become effective after the modified project has been loaded into the PLC. In addition to the list of the selected NC variables, when compiling, then the created DB9910 (NC_DATA) is also downloaded.

### Loading a data block from the PLC

You must first open a project in the Programming Tool before you can upload the program block (project) and therefore the NC variables from the PLC. In so doing, in addition to the list of the selected NC variables, the DB9910 (NC_DATA) is also downloaded.

### See also:

Communications configuration (Page 443)
Selecting the CPU type (Page 469)
First steps: Content (Page 17)
Information on using the software (Page 375)

## 11.2.3.8 Cross references

### Call

Use one of the following methods to view the Cross Reference window:

- Select the menu command **View > Cross Reference**.
- Click the button for cross-references  in the "Navigation" toolbar (Page 500).

To access the Cross Reference table, the Byte Usage table or the Bit Usage table, click the appropriate tab at the bottom of the "Cross Reference" window:


These help topics are explained in the following:

### Cross Reference table

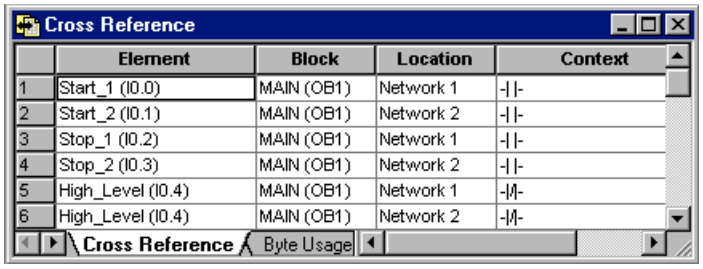
**Note**

You must compile your program in order to view the Cross Reference table.

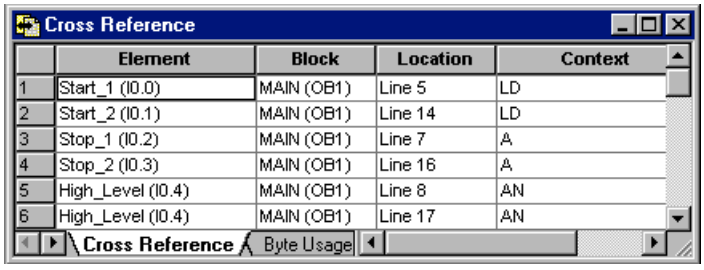
Use the Cross Reference table when you want to know whether a symbolic name or an address is already in use in your program, and where it is used. The Cross Reference table lists all operands used in the program and lists all occurrences of the operands with POU, network or line, as well as the instruction.

| | |
|---|---|
| **Element** | Refers to the operands used in your program. You can toggle between symbolic and absolute view to change the representation of all operands (select the menu command **View > Symbolic Addressing**). |
| **Block** | Refers to the POU where the operand is used. |
| **Address** | Refers to the line or network where the operand is used. |
| **Context** | Refers to the program instruction where the operand is used. |

*Example of an LAD cross reference list*




*Example of an STL cross reference list*




## Byte Usage table

**Note**

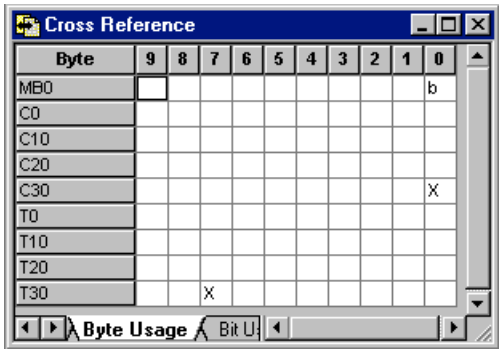
You must compile your program in order to view the Byte Usage table.

The Byte Usage table allows you to see which bytes, from which memory areas, have been used in your program. It also helps you recognize duplicate assignment errors.

| | |
|---|---|
| **b** | Indicates that a bit of memory has been assigned. |
| **B** | Indicates that a byte of memory has been assigned. |
| **W** | Indicates that a word (16 bits) of memory has been assigned. |
| **D** | Indicates that a double word (32 bits) of memory has been assigned. |
| **X** | Used for timers and counters. |

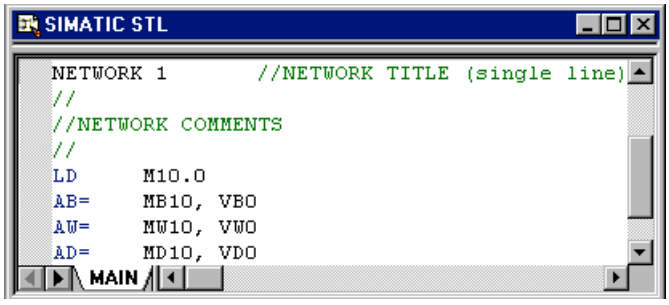
### Interpreting the Byte Usage table

This example of a Byte Usage table shows that its associated program makes use of the following addresses: one bit within MB0; counter C30; timer T37.

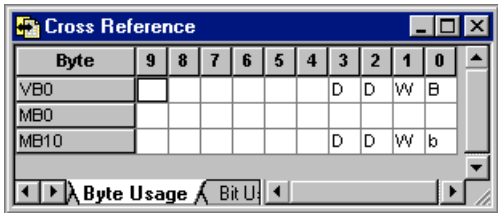



### Recognizing duplicate assignment errors

This example program contains overlapping address assignments at MB10.0.




The Byte Usage table can be examined to identify improper assignments. Since a double word requires four bytes, there should be four adjacent Ds in the row for VB140000000. Likewise, since a word requires two bytes, there should be two adjacent Ws for VB14000000. The row for MB10 reflects the same problems, plus the fact that MB10.0 was used in more than one assignment.




### Bit Usage table

**Note**

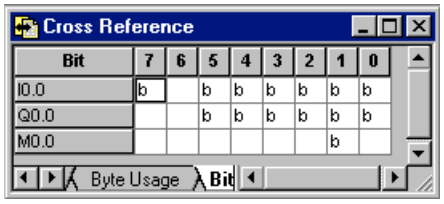
You must compile your program in order to view the Bit Usage table.

The Bit Usage table allows you to see which memory addresses, down to the bit level, have been used in your program. It also helps you recognize duplicate assignment errors.

| | |
|---|---|
| **b** | Indicates that a bit of memory has been assigned. |
| **B** | Indicates that a byte of memory has been assigned. |
| **W** | Indicates that a word (16 bits) of memory has been assigned. |
| **D** | Indicates that a double word (32 bits) of memory has been assigned. |
| **X** | Used for timers and counters. |

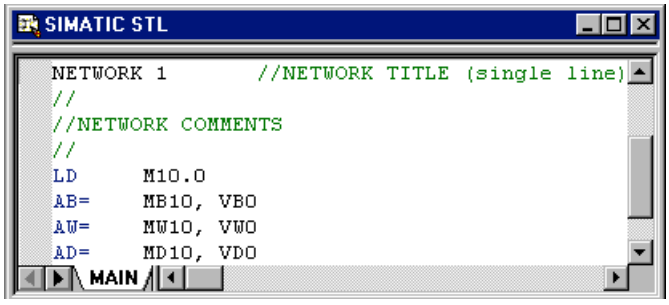
### Interpreting the Bit Usage table

This example of a Bit Usage table shows that its associated program makes use of the following addresses: from byte I0, bits 0, 1, 2, 3, 4, 5, and 7; from byte Q0, bits 0, 1, 2, 3, 4, and 5; from byte M0, bit 1.

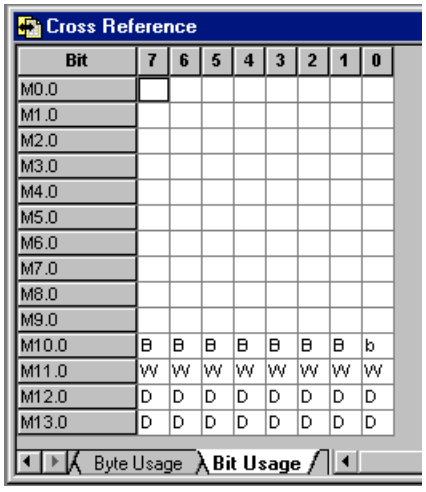



### Recognizing duplicate assignment errors

This example program contains overlapping address assignments at MB10.0.




The Bit Usage table can be examined to identify improper assignments. In a properly assigned program, there would never be a bit value in the midst of the byte. BBBBBBBb is invalid, whereas BBBBBBBB would be valid. The same is true of the word assignments (there should be 16 adjacent Ws) and the double word assignments (there should be 32 adjacent Ds).




**See also:**

Communication configuration (Page 443)

Selecting the CPU type (Page 469)

Changing between symbolic and absolute addressing mode (Page 444)

First steps: Contents (Page 17)

Information on using the software (Page 375)

### 11.2.3.9 Communication

You can set up communication at the following points:

- In the last step when installing the Programming Tool.
- In the Programming Tool.

**Setting up communication in the Programming Tool**

To display or change the communications configuration in the "Communications Connections" dialog box, perform one of the following actions:

- Click the Communication button in the "Navigation" toolbar (Page 500).
- Select the menu command **View > Communications**.
- Click the Communication button in the project tree (Page 447).

On the left-hand side, enter the address of the remote PLC directly in the remote address 2 field. This sets the PLC for the Programming Tool when you exit the "Communications Connections" dialog box.

On the right-hand side, you can reconfigure the communication by double-clicking the component symbols in the network. The "Set PG/PC interface" dialog box is displayed when you double-click the interface symbol (the top symbol).

You can also set the PLC for the Programming Tool. First double-click the Refresh button to start a query that identifies all CPUs in the network. Then click the symbol of the PLC to select the desired PLC.


The field with green frame indicates which PLC is selected for the Programming Tool.

**Further information about the communication setup:**

Options for communication (Page 85)

Installing and removing communication interfaces (Page 96)

Setting and changing parameters (Page 97)

**Note**

The Programming Tool and the STEP 7 software use the same user interface for installing, configuring and deleting modules (drivers under Windows). While displaying information in the "Communication" dialog box for the currently installed drivers, you can load and configure the Windows drivers in the "Set PG/PC Interface" dialog box. The "Set PG/PC Interface" dialog box is displayed when you double-click the interface symbol (at the uppermost edge of the "Communications Connections" dialog box). When the "Set PG/PC Interface" dialog box is displayed, press F1 or click the "Help" button in this dialog box to fetch the help for setting the PG/PC interface.

### 11.2.3.10 Symbolic Addressing Display

If you have assigned symbolic addresses in the Symbol Table or a Local Variable Table, you can toggle to switch between viewing parameter addresses with Absolute (for example, I0.0) or Symbolic (for example, Pump1) representation. To do this, proceed in one of the following ways:

- Select the menu command **"View > Symbolic Addressing"**.
- Press the **<Ctrl+Y>** shortcut key.

You can switch the address display in the Status Chart and Cross Reference windows, and the program window of the LAD and STL editors.

**See also:**

Viewing symbolic and absolute addresses simultaneously in LAD (Page 490)

Symbol information tables (Page 445)

Symbol table (Page 413)

Overview of the addressing (Page 26)

Direct addressing (Page 261)

DB address representation (Page 446)

Information on using the software (Page 375)

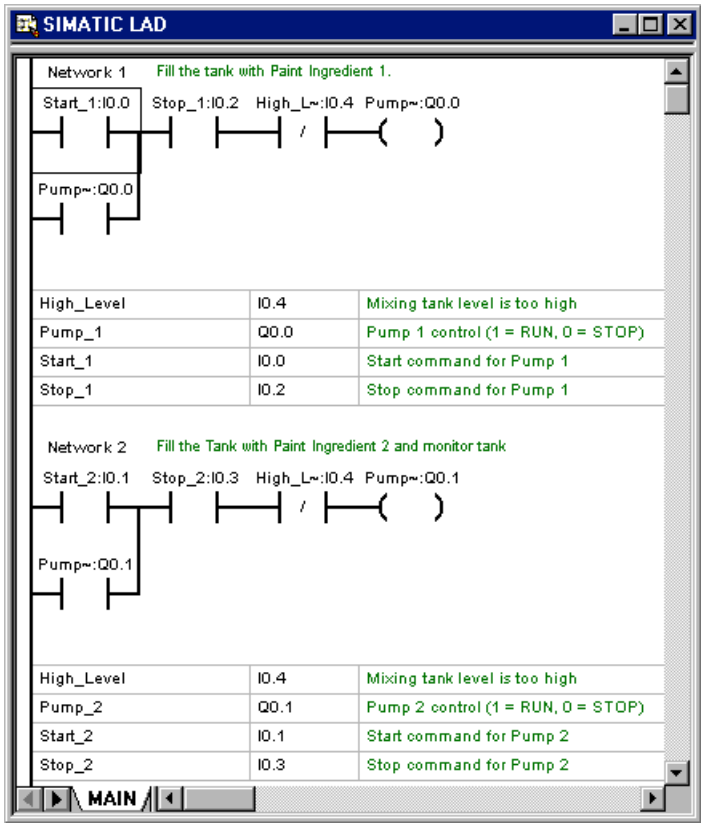
First steps: Contents (Page 17)

### 11.2.3.11 Symbol information table

Use one of the following methods to view or hide symbol information tables in LAD:

- Click the button in the toolbar.
- Select the menu command **"View > Symbol Information Table"**.
- Press the **<Ctrl+T>** shortcut key.

When you view symbol information tables, the symbolic names, absolute addresses, and comments are displayed below each network in your LAD program. The table lists information for any symbols in that network; networks that do not contain global symbols do not display a symbol information table.




**Tips:**

- Use the arrow keys to scroll through the symbol information table.
- You can resize the table by dragging the splitter cursor from the column edges.
- Use the button "Options" in the dialog box "Print" to enable printing of symbol information tables when you print your program.
- To conserve vertical space on the screen and view more networks of your program, you can toggle the symbol information tables on and off by using the menu command **"View > Symbol Information Table"**.

**Troubleshooting**

If you do not see symbol information tables beneath the networks in your program, consider these points:

1. Are symbol information tables enabled? Check this in the menu "View": The menu command "Symbol Information Table" should have a check mark beside it.
2. Are symbols (or the absolute addresses that correspond to the symbols) used in the network? Networks that do not contain symbolic names do not display a symbol information table.
3. Have you entered the symbol in the symbol table? No address or comment information is available for a symbol until it is fully defined.

**See also:**

Symbol table (Page 413)

"LAD Editing" tab (Tools > Options) (Page 490) (Here you can specify how the symbol information table is displayed when a program is processed.)

"LAD Program Status" tab (Tools > Options) (Page 492) (Here you can specify how the symbol information table is displayed when you display the program status.)

Information on using the software (Page 375)

First steps: Contents (Page 17)

### 11.2.3.12 DB Address View (View menu)

You can view the addresses of the variable memory (data block) as V addresses (e.g. "VB14000000") or DB addresses (e.g. "DB1400.DBB0"). To do this, proceed in one of the following ways:

- Select the menu command **"View > DB Address View"**.
- Press the **<Ctrl+B>** shortcut key.

You can switch the address display in the Status Chart and Cross Reference windows, and the program window of the LAD and STL editors.

**See also:**

Information on using the software (Page 375)

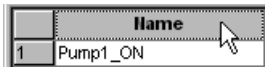
First steps: Contents (Page 17)

### 11.2.3.13 Sorting table and chart columns

Sort Symbol Table and Status Chart by "Name" or "Address" columns using one of the following methods:

- Click the  button or select the menu command **"View > Sort Ascending"** to sort the Symbol Table "Name" column from A to Z.
- Click the  button or select the menu command **"View > Sort Descending"** to sort the Symbol Table "Name" column from Z to A.

- Click the header of the "Name" column to sort Symbol Table names.




- Click the header of the "Address" column to sort Symbol Table or Status Chart addresses.


**Note**

Click again on the header of a column to re-sort in reverse order.

**See also:**

Symbol table (Page 413)

Status chart (Page 420)

Information on using the software (Page 375)

First steps: Contents (Page 17)

### 11.2.3.14 Windows (View menu)

Use the menu command **"View > Windows > Reset All"** to reset all windows to the default settings.

**See also:**

Information on using the software (Page 375)

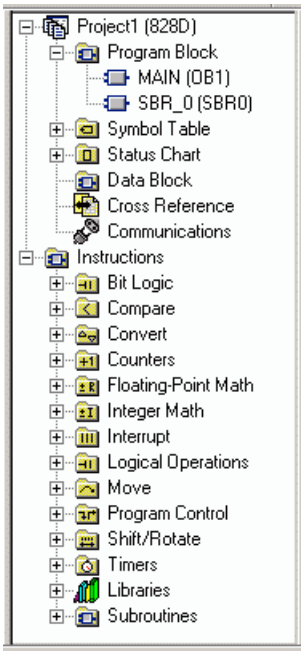
First steps: Contents (Page 17)

### 11.2.3.15 Project tree (View menu)

The project branch Project1 manages your project.

Display of the project tree:

- Use the menu command **"View > Project Tree"** to display and hide the project tree.




**Program blocks**

- Right-click the **Program Block** directory to insert new subprograms and interrupt programs.
- Open the **Program Block** directory.
- Right-click a button of a program organization unit (POU) to open the POU, rename it, delete it, add comments, or edit its properties.

**Status and symbol tables**

- Right-click on the **Status Chart** or **Symbol Table** directory to insert a new chart or table.
- Open the **Status Chart** or **Symbol Table** directory and right-click on the button of a chart or table to open, cut, copy, paste, delete, or rename it.

**Data blocks**

- Open the **Data Blocks** directory. Here, the directories **User interface** and **User data blocks** are available.
  - User interface: Open the **User Interface** directory. Right-click on the button of a data block to open it or display its properties.
  - User data blocks: Right-click on the **User Data Blocks** directory to insert a new user data block. Right-click on the newly created data block to open, cut, copy, paste, delete, rename it, or to display its properties.

**NC variables**

- Open the **NC Variables** directory. Here the directories **NC Variables List** and **Selected NC variables** are available.
  - NC Variables List: Open the **NC Variables List** directory. Click the button of the NC variables or drive parameters to open them.
  - Selected NC variables: Right-click on the **Selected NC variables** directory to insert a new NC variable table. Right-click on the newly created NC variable table to open, cut, copy, paste, delete, rename it, or to display its properties.

**System data blocks**

- Open the **System Data Block** directory. The directory is only visible if retentive data is saved in the PLC user project.
- Right-click on the **Retentive Data** button to delete the system data block or display its properties.

### Expand and collapse the project tree directories

Click once on the branch to open a directory. Alternatively, you can double-click the icon of the directory.

To close directories, double-click the directory or click once on the branch.

### To resize the project tree window

Move the cursor over the right edge, then click and drag the splitter cursor.

### See also:

Keywords (Page 281)

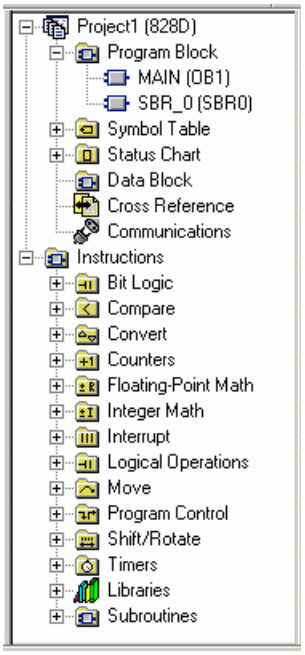
Information on using the software (Page 375)

First steps: Contents (Page 17)

## 11.2.3.16 Instruction tree (View menu)

Displaying the instruction tree:

- Use the menu command **"View > Instruction Tree"** to display and hide the instruction tree.




The Instructions directory Instructions helps you create programs.

Once you have opened an instruction directory and selected an instruction, you can:

- Drag and drop or double-click to insert instructions into a program.
- Right-click an instruction and choose Help from the shortcut menu for information about that instruction.

If you specified a CPU type for your project, the instruction tree shows any instructions that are not valid for your PLC with a red x: .

### Expand and collapse the instruction tree directories

Click once on the branch to open a directory. Alternatively, you can double-click the icon of the directory.

To close directories, double-click the directory or click once on the branch .

### To resize the instruction tree window

Move the cursor over the right edge, then click and drag the splitter cursor.

### See also:

Keywords (Page 281)

Information on using the software (Page 375)

First steps: Contents (Page 17)

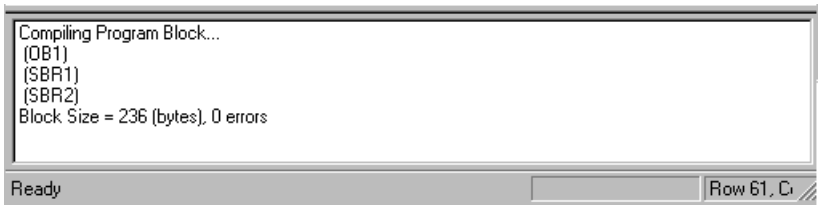
### 11.2.3.17 Output Window (View menu)

The Output Window displays the results of the block comparison, a list of the most recently compiled POUs, and any errors that occurred during the compilation.

#### Working with the output window

Open the output window with the menu command **"View > Output Window"**.

The Output Window opens below the program editor:




Errors that occur during the compilation of a program are displayed in the output window. Open the program editor window and the output window. Next, double-click on an error message in the output window. The program jumps automatically to the network in which the error appeared.

Perform a recompilation (Page 459) so that you are not warned about errors in the output window that you have already corrected in the program.

The shortcut menu allows you to hide the output window, copy the text in the output window, or delete the text. To open the shortcut menu, right-click in the output window.

The Programming Tool also lists the results of a block comparison in the output window. All compared blocks are listed in the representation, even if there are no differences. Differences are displayed below the respective block name. Here, the position specifications refer to the current project. For program blocks, these are the network numbers and, if available, row number and column number. For data blocks, these are the row numbers. At the end of the list of the program or data blocks, information on how many blocks were compared and how many differences were found is provided. Double-clicking on the row with the difference opens the relevant row in the editor of the current project.

#### To resize the Output Window

Move the cursor over the top edge, then click and drag the splitter cursor  .

#### See also:

Information on using the software (Page 375)

First steps: Contents (Page 17)

Compare (File menu) (Page 384)

### 11.2.3.18 Zoom

To set the graphical size and the grid representation in the LAD editor perform one of the following actions:

- Click the button.
- Select the menu command **"View > Zoom"**.
- Press the key combinations once to incrementally enlarge or reduce the size and keep the key combinations pressed to further increase or decrease the size.

| | |
|---|---|
| **Ctrl + Plus key**<br>(only in the keyboard numeric keypad) | Enlarge |
| **Ctrl + Minus key**<br>(only in the keyboard numeric keypad) | Reduce |

You can set a percentage value for the zoom function, and the width and height for the grid. The grid size is specified in pixels.

**See also:**

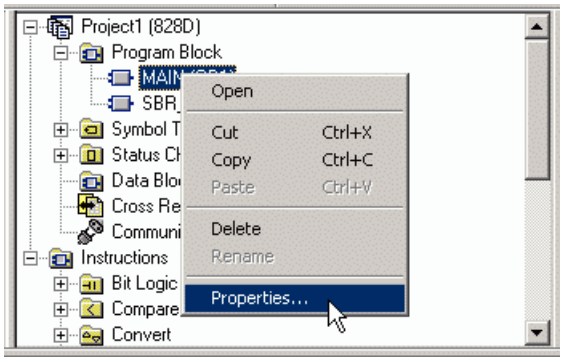
Information on using the software (Page 375)

First steps: Contents (Page 17)

### 11.2.3.19 Properties

#### Properties of program organization units

To open the menu of the "Properties" dialog box for the component, navigate in the project tree to a project component and right-click it.




Each POU and each data block of your project has a "Properties" dialog box that can be used to:

- Rename components
- Specify the author of the component
- Enter comments
- Determine data classes (Page 288)

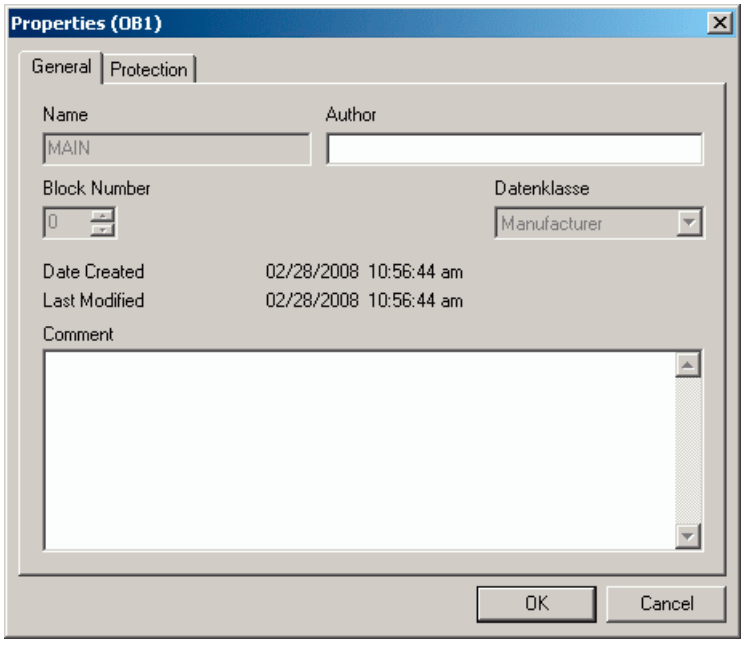
Although you can renumber data blocks and subprograms, this cannot be done for interrupt programs and the main program (OB1). When you assign a symbolic name to a POU, the symbolic name is displayed in your program code even when you have not activated symbolic addressing.

**Note**

You cannot use the default name (the POU address assigned by the Programming Tool, e.g. SBR1 for a subprogram or INT0 for an interrupt program) as symbolic name because this would cause duplicated addresses. If you violate the guidelines for making a symbolic name assignment, the Programming Tool reports an error when you try to compile your program.

If you assign symbolic names to the POUs in your project, the Symbol Table displays a separate tab ("POU symbols") that lists these assignments. You can only display this tab. You cannot edit the entries. To change the symbolic names, open the "Properties" dialog box of the associated POU.

POUs can be protected with a password (Page 315). Select the "Protection" tab.

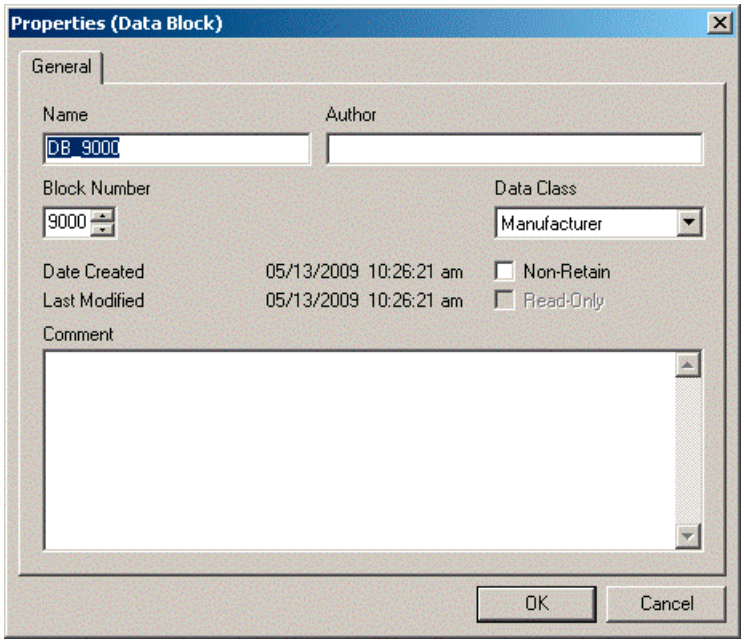



## Data block properties

If you assign symbolic names to the data blocks in your project, the Symbol Table displays a separate tab ("DB symbols") that lists these assignments. You can only display this tab. You cannot edit the entries. To change the symbolic names, open the "Properties" dialog box of the associated data block.

You can also activate the property "Non-retain" for data blocks. This ensures that every time the PLC is brought into the RUN mode, the data block in the PLC is preset with the initial values.

Data blocks can be write-protected, that is, the actual values of the data block cannot be changed with the Programming Tool. This property cannot be changed.




## Properties of NC variables

The **variables** listed in the NC variable selection window are displayed with the following information:

- Area
- Block
- Variable name
- Type

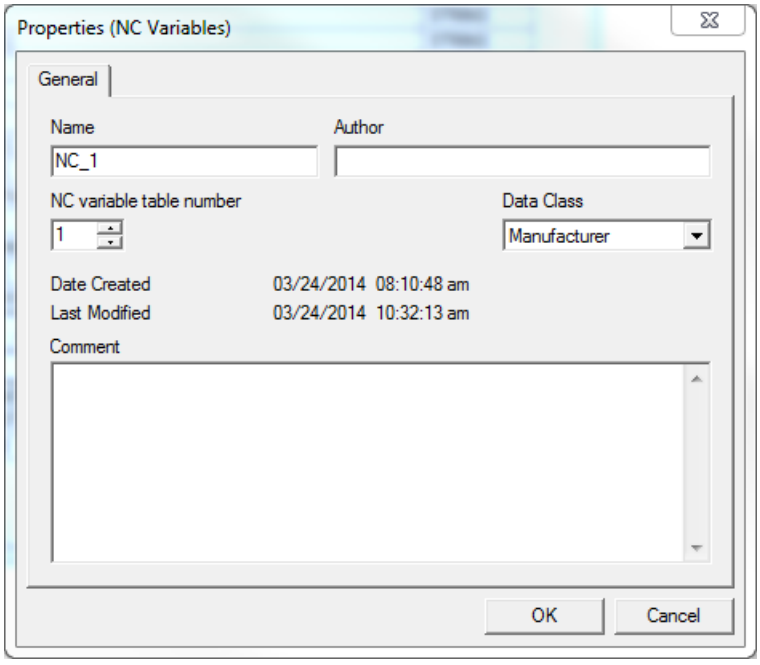
You cannot edit these entries.

In addition, you can display the properties of the **list of selected NC variables**.

The following properties can be edited:

- Name
- Author
- Number NC variable table
- Data class
- Comment

In addition, you can see the creation date as well as the date of the last change.




## Properties of system data blocks

If you saved retentive data in the PLC user project, you can see the properties of the system data blocks, but you cannot edit them.

**Note**

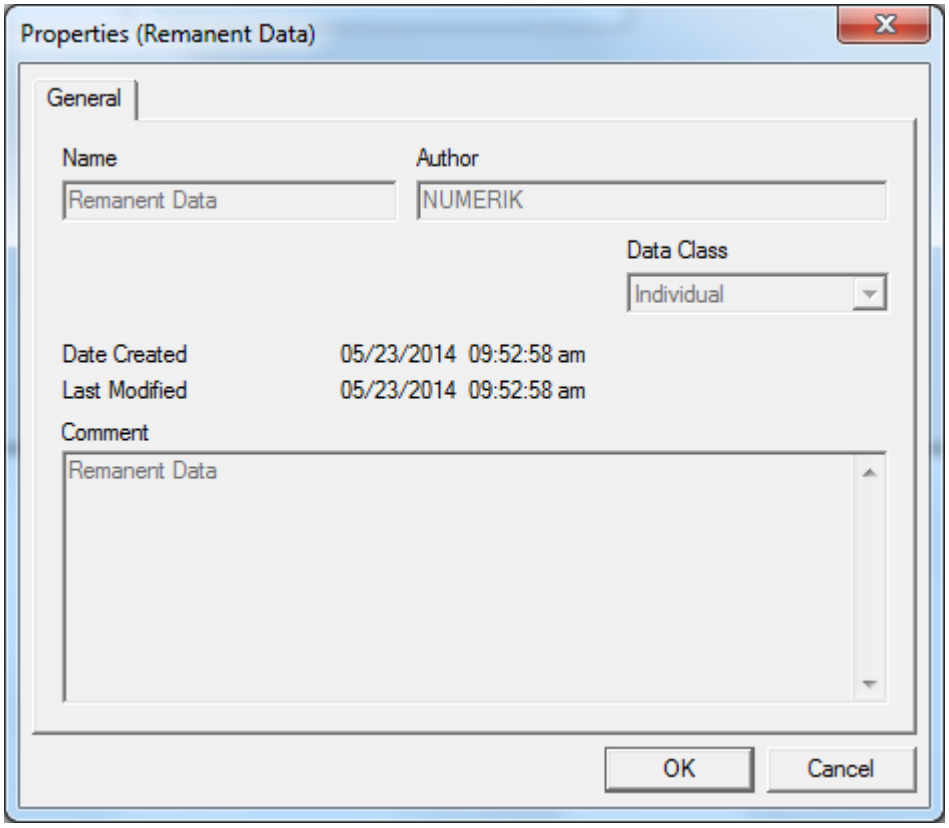
**Supporting CPUs**

Only the following CPUs support this function:

- SINUMERIK 828D
- SINUMERIK 828D Step 2

Visible properties:

- Name
- Author
- Data class
- Creation date
- Last change
- Comment




**See also:**

Information on using the software (Page 375)

First steps: Contents (Page 17)

### 11.2.3.20 Status bar (View menu)

The status bar located at the bottom of the user interface provides information about the instructions that you perform in the Programming Tool.

| | | | | |
|---|---|---|---|---|
| A brief status description | Network 45 | Line 1005 | OVR | Symbols |

**Information on the editor**

Editor information is displayed when you work in editing mode. The status bar displays, as appropriate, the following information:

- A brief status description
- The current network number
- The cursor location (line and column for the STL editor; row and column for the LAD editor)
- The current editing mode: Insert or Overstrike
- Symbols that indicate the status of background tasks: for example, saving or printing

**Information on the online status**

Online status information is available when you turn on either Program Status or Chart Status. The status bar displays, as appropriate, the following information:

| | | | | | | |
|---|---|---|---|---|---|---|
| PPI cable | 19,200 | Local: Station 0, COM1 | Remote: Station 5, port 0 | STOP | NFE 31A6 | Symbols |

- The local hardware configuration that is being used for communications
- The baud rate
- The communication address for the local station and the remote station
- The CPU operating mode

**Information on progress**

Progress information is shown when the operation underway takes a long time to complete. The status bar offers a description of the operation and a progress indicator.

| | |
|---|---|
| Description of an operation which takes a long time | Progress indicator |

**See also:**

Information on using the software (Page 375)

First steps: Contents (Page 17)

## 11.2.4 Menu PLC

### 11.2.4.1 RUN / STOP

**Mode of operation of the program:**

If you load a program into your PLC and switch the PLC into the RUN mode, the PLC (CPU) of the PLC executes the program in a repeated scan cycle. This scan cycle is called in the interval of MD10075 PLC_CYCLE_TIME:

1. **Reading the inputs**
   The CPU reads the status of all inputs connected to the automation system. This data is stored in the input memory area, known as the process image input.
2. **Executing the optional interrupt program INT100**
3. **Executing the program**
   The CPU calls MAIN and executes the logic of the program. In this respect, it has read and write access to the process image, variables, data blocks, etc.
4. **Executing the optional interrupt program INT101**
5. **Writing to the outputs**
   At the end of the program, the CPU writes the data from the process image output to the physical outputs.
6. **Processing communication requests (NCK, HMI)**
7. **CPU self-diagnosis**

**Changing the mode of the PLC**

- Click the RUN button to place the PLC into the RUN mode. Click the STOP button to place the PLC into the STOP mode.
- Select the PLC > RUN menu command to place the PLC into the RUN mode. Select the PLC > STOP menu command to place the PLC into the STOP mode.

**Note**

In order that you can set the RUN/STOP mode using the Programming Tool, communication (Page 85) between the Programming Tool and the PLC must be established.

**PLC operating mode details:**

The PLC has two operating modes: STOP and RUN. In the STOP mode, you can create and edit your program. However, your program is not executed in the STOP mode. The program is executed in the RUN mode. Further, in the RUN mode, you can observe the mode of operation and data of the program. Test functions are available to better observe the mode of operation of the program and to identify programming errors (bugs).

The test functions to execute the first or several scan cycles can be used in the STOP mode. The RUN mode is then only assumed for the specified number of scan cycles.

Fatal errors are saved in the CPU operating system and force a mode change from the RUN to the STOP mode. If the PLC has identified a fatal error, then it can no longer be brought from the STOP mode into the RUN mode as long as the error is present. Although the operating system of the CPU stores minor errors that can be evaluated later, they do not require a transition from RUN to STOP.

In the **STOP** mode

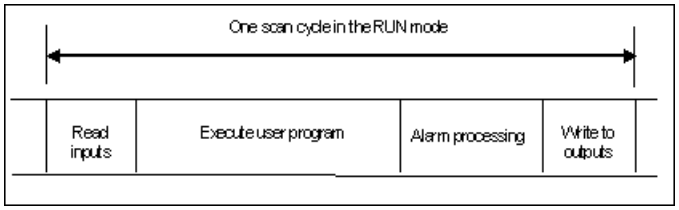
- The PLC is, in effect, in a quasi-idle state (the execution of the user program is interrupted)
- You can load the user program into and from the PLC.

If one or several devices are communicating with the PLC via the communication interface, then the PLC responds to all of the requests one after the other. The PLC does not prevent several communicating devices mutually interfering with one another. Your system design must provide the necessary safety precautions to prevent this mutual interference between communicating partners.

In the **RUN** mode, the PLC

- Reads the inputs
- Executes the user program
- Writes to the outputs
- Responds to communications requests
- Performs internal administrative tasks
- Responds to interrupt conditions

The PLC does not support fixed scan cycle times to execute the scan cycle in the RUN mode. These tasks (with the exception of the interrupts) are executed according to their priority, in the sequence that they occur. This continuous execution of the CPU tasks is called the scan cycle and is shown in the following diagram.




Each scan cycle begins by reading the actual values of the inputs and writing these values into the process image input. Input bits that have no corresponding physical input but which are in the same byte as bits with physical inputs are set to zero in the process image in each scan cycle.

After the inputs have been read, your program executes the instructions from the first through to the last one.

During the phase in which the alarms are processed in the scan cycle, all alarms are processed for which changes occurred during the scan cycle.

Finally, the values are written from the process image into the output modules. This completes one scan cycle.

**See also:**

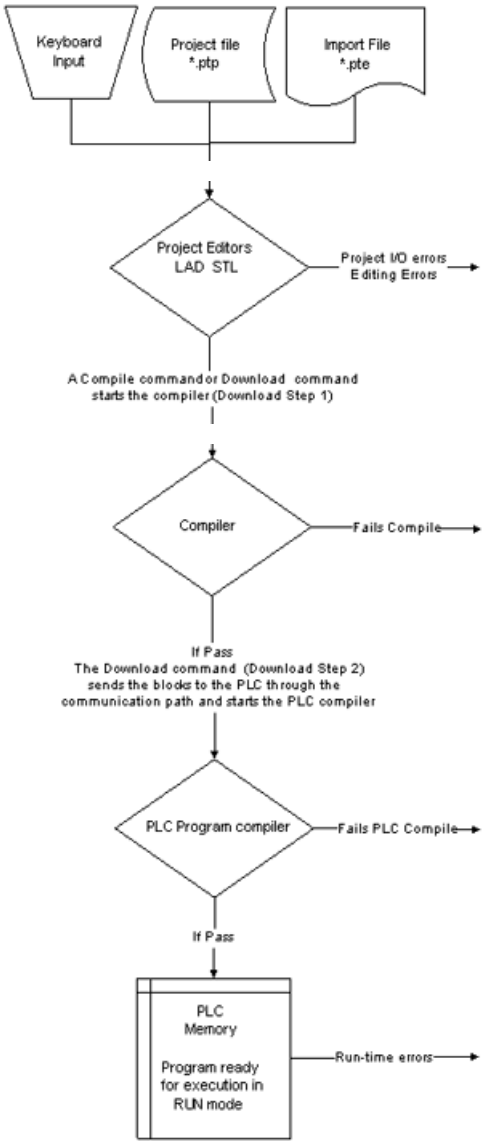
Information on using the software (Page 375)

First steps: Contents (Page 17)

### 11.2.4.2 Compile

Use one of the following methods to start the Programming Tool Project Compiler:

- Click the Compile button or select the menu command **PLC > Compile** to compile all project components (program block, data block).




Opening projects (Page 379)

Range checking (Page 209)

Selecting the CPU type (Page 469)

All compilation errors in the Programming Tool are listed in the Output Window (Page 451). Double-click the error and the editor scrolls to the error location. Use the menu command PLC > Type (Page 469) to set a specific model and version of the CPU.

There are errors that indicate that the project is exceeding the space available on the target system.

For example, "ERROR 820: The compiled program block or the entire project (including comments, data blocks, etc.) is too large for the current target system."

In this case, all data counts towards the overall size of the project, including program blocks, data blocks, tables, comments, etc.

To enable successful compilation of the project, gradually delete non-essential data until the storage requirements are met again.

**See also:**

Information on using the software (Page 375)

First steps: Contents (Page 17)

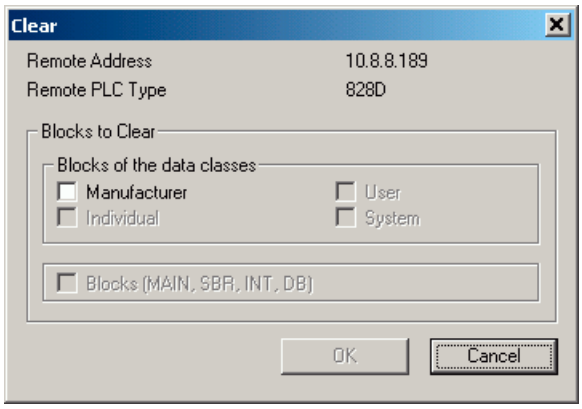
### 11.2.4.3 Delete

You can delete project components in the target system:

- Select the menu command **PLC > Delete** to delete project components.

A dialog opens that displays the connected target system and contains the selection fields for the data classes to be deleted.

Deletion is permitted only in the "STOP" mode. Deleting the main program (OB1) results in a start error on the transition to the "RUN" mode. The target system then returns into the safe "STOP" mode.




### 11.2.4.4 Information

To view PLC information such as CPU model and version numbers, mode, and scan rate, select the menu command **PLC > Information**.

The **Reset scan rates** button allows you to refresh the times in the three Scan rate fields.

### 11.2.4.5 Compare (PLC menu)

#### General information on the block comparison

The comparison between two PLC user projects can be performed in two ways. You can compare program blocks (POUs) and user data blocks (actual values) of the current project either

- with the PLC project in a controller (online), or
- with the blocks of another project (offline).

In this manner, the block information of the projects is directly compared.

Two PLC user projects may be compared with one another for the following reasons:

- For putting a machine/series of machines into operation, it is necessary to determine the differences in comparison with an earlier version of the PLC project.
- Whether or not the PLC project was modified in the machine should be tested during the service call.
- The machine dealer checks whether the PLC project in the supplied machine corresponds to the project that he wants to deliver to his customer.

The comparison always provides the same results, independent of the PLC Programming Tool version used for creating the respective PLC user project.

When comparing two projects, the CPU types for these projects must be consistent. If the CPU types do not concur across both projects, it will be inquired whether the CPU type of the comparison project should be adopted for the current project.

The project comparison dialog opens when the comparison is started. In this dialog you select the data classes whose blocks you want to compare. You can compare program blocks and data blocks separately. Following selection of the data classes, the program and user data blocks to be compared can be selected from the block lists. For the user data blocks, the comparison of the actual values can also be activated. Other project components such as symbol tables cannot be compared.

The results of the comparison are displayed in the output window. All compared blocks are listed. Differences between the blocks are shown below the block name. Here, the position specifications refer to the current project. For program blocks, the position data is provided via the network number and, if available, row number and column number. For data blocks, the position data is provided via the row numbers. At the end of the list of the program or data blocks, information on how many blocks were compared and how many differences were found is provided. If you double-click on the line of a difference in the output window, the corresponding position in the editor of the current project will open.

**Special features of the comparison**

- Empty networks in the program blocks or empty rows in the data blocks are ignored during the comparison.
- If too many differences are found in a network or a row, these are not all listed, and a message indicating that the network or the row is different is displayed.
- Block properties are factored in.
- Different names and data classes are captured.
- With regard to the data blocks, differences in the "non-retain" and "write-protected" properties are also covered.
- Creation date and date of last change are not compared.
- Differences in block properties have no position data. For this reason, no reaction occurs to a double-click on the row with the difference in the output window.

## Online comparison via PLC > Compare...

1. Start the online comparison via the menu **"PLC > Compare..."**.

   **Note**

   If a connection to the address set under "Communication" cannot be set up, a corresponding error message will be displayed.

   If the CPU type of the open project does not concur with that of the remote PLC, a query is displayed as to whether the CPU type of the remote PLC should be adopted for the opened project.

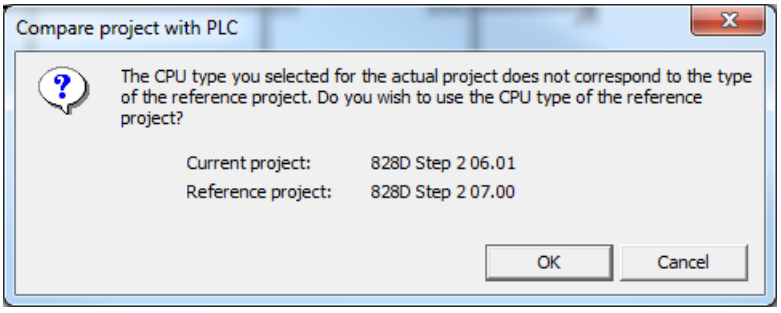



   Figure 11-13 Taking over the CPU type

   In this dialog, the address and type of the remote PLC are displayed. If the address does not point to the desired PLC, click on the **"Cancel"** button to cancel the comparison; you can then modify the address under **"View> Communication"**.

2. Click on the **"OK"** button to import the CPU type and to start the comparison dialog.
   The **"Compare project with PLC"** dialog opens.

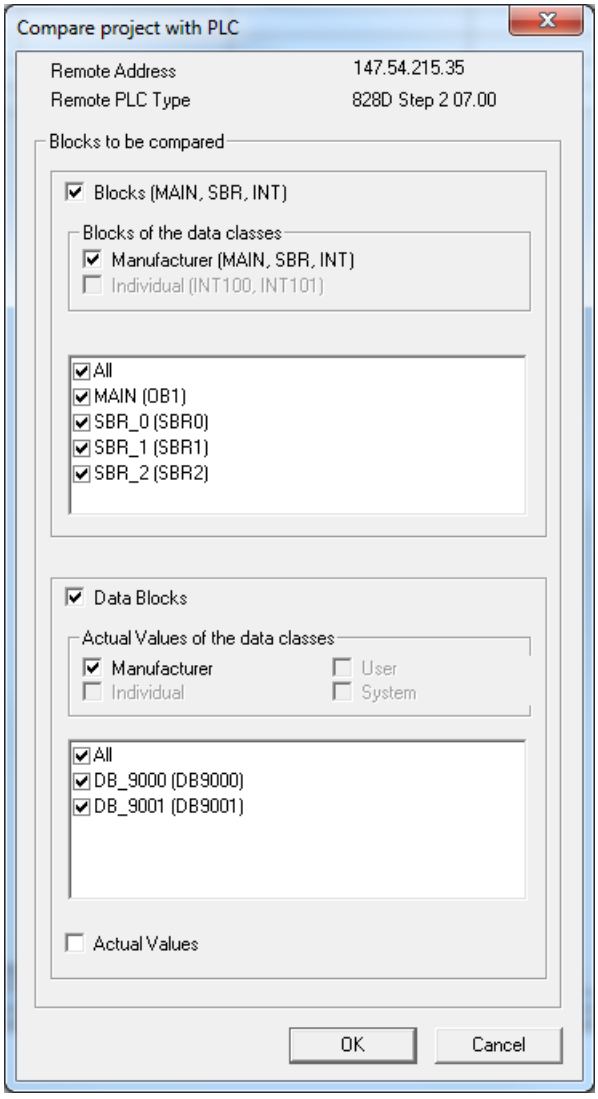



Figure 11-14 "PLC > Compare dialog"

3. In this comparison dialog, you can select the blocks that are to be compared.
   Program and data blocks are distinguished. A pre-selection can be made via the selection of data classes. By default, all blocks are selected.
4. Click on the **"OK"** button to start the comparison.

5. The result of the comparison is displayed in the output window. In this manner, all blocks compared are listed, even if there are no differences. Differences are displayed below the respective block name.

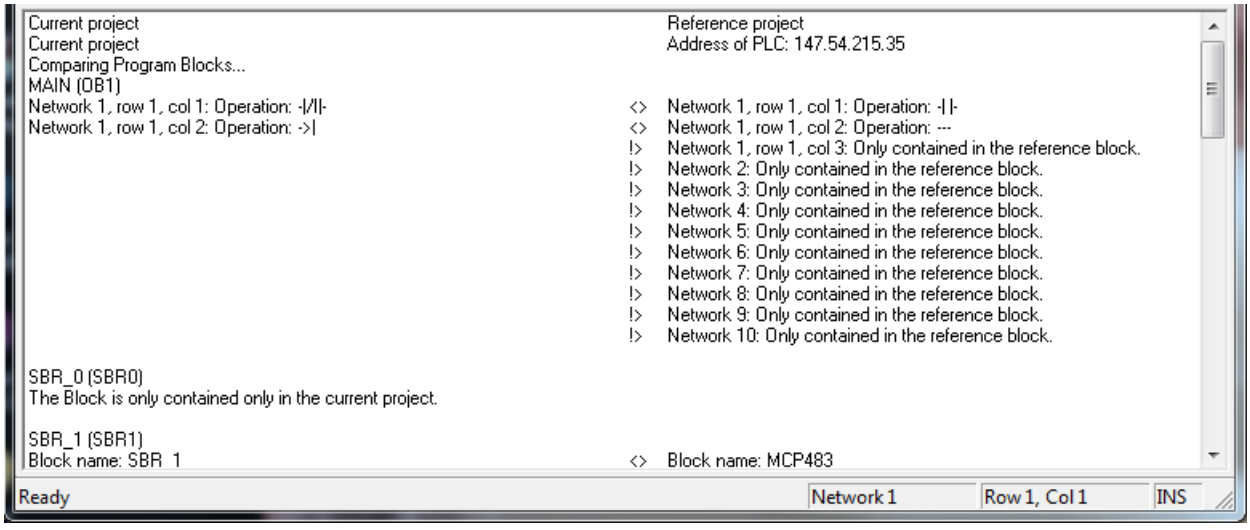
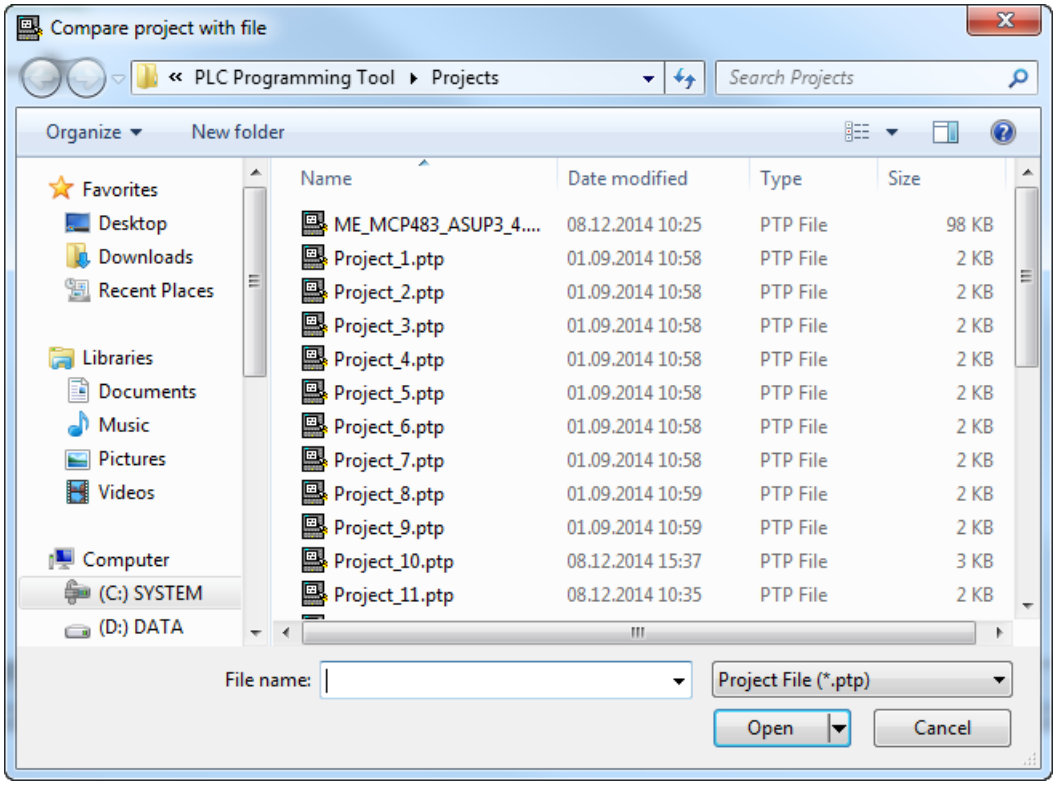



Figure 11-15 Comparison result in the output window

6. Double-click on a row with a difference in the output window to open the relevant row in the editor of the current project.

### Offline comparison via File > Compare...

1. Start the offline comparison via the **"File > Compare..."** menu.
   The **"Compare project with file"** dialog is opened.




2. In this dialog, select the project that you want to compare with the project that is already open.
3. Click on the **"Open"** button.
   If the CPU type of the current project does not concur with that of the comparison project, the question as to whether the CPU type of the comparison project should be taken over for the current project follows.

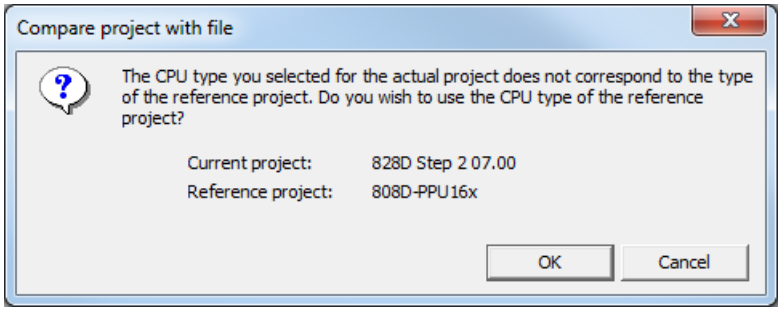



Figure 11-16 Taking over the CPU type

4. Click on the **"OK"** button to take over the CPU type and to start the comparison dialog.
   In the dialog that now opens, the name including path of the file to be compared is displayed.

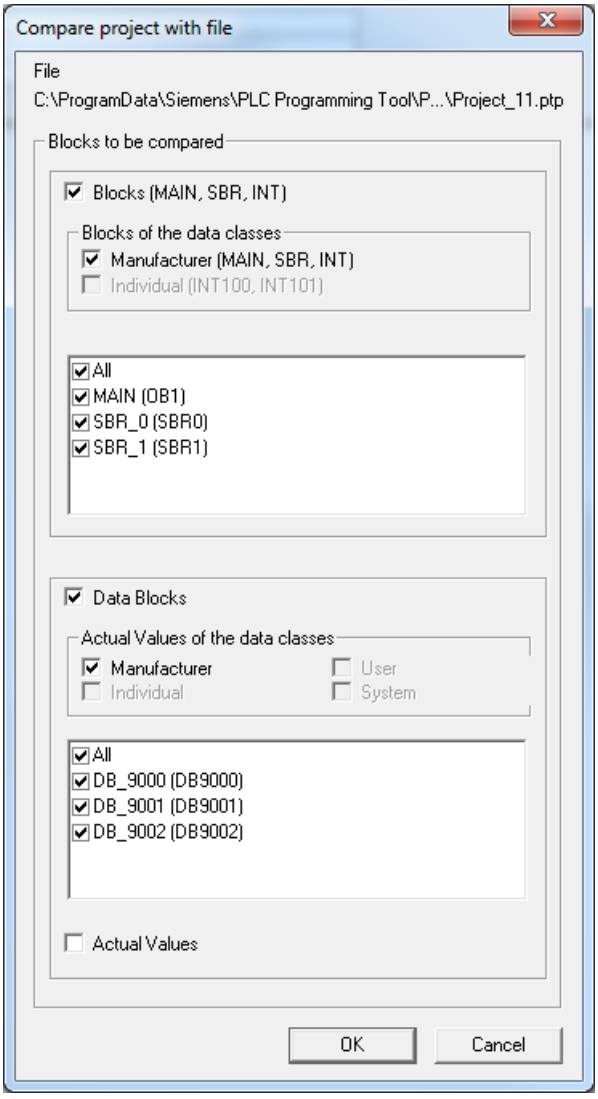



Figure 11-17 "File > Compare" dialog

5. In this comparison dialog, you can select the blocks that are to be compared.
   Program and data blocks are distinguished. A pre-selection can be made via the selection of data classes. By default, all blocks are selected.

6. Click on the **"OK"** button to start the comparison.
   The result of the comparison is displayed in the output window. All blocks compared are listed, even if no differences are present. Differences are displayed below the respective block name.

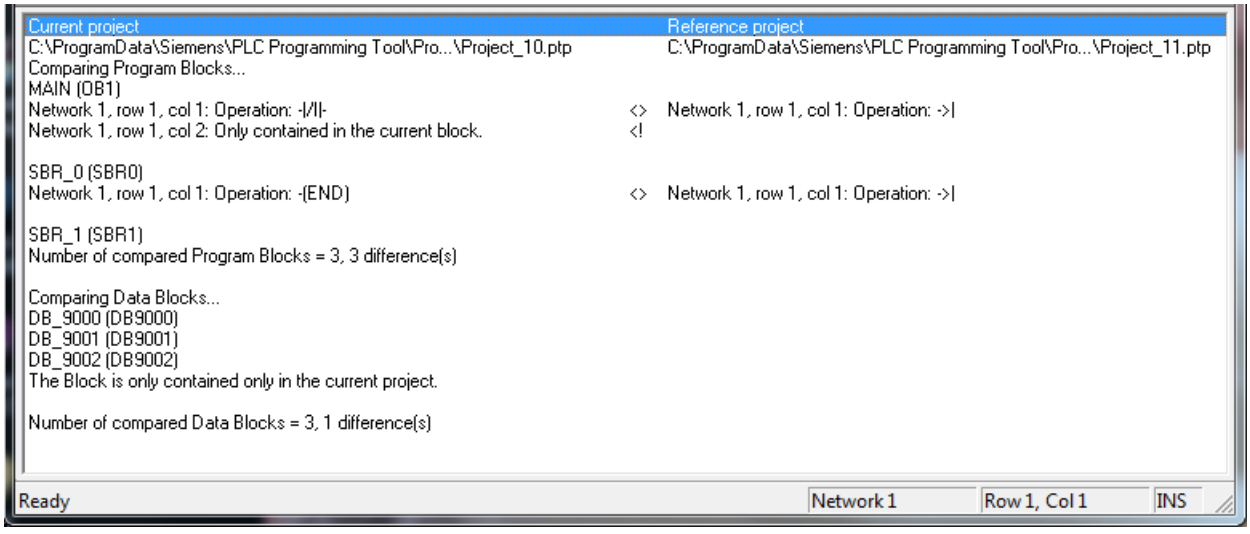



Figure 11-18 Comparison result in the output window

7. Double-click on a row with a difference in the output window to open the relevant row in the editor of the current project.

## Saving the comparison results

The comparison results can be saved and thus also be printed.

1. Right-click in the output window.

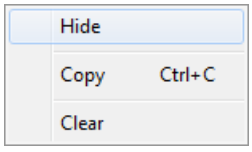



Figure 11-19 Shortcut menu of the output window

2. Select the "Copy" option.
   Alternatively, you can also click on the output window with the left mouse button and press "Ctrl + C".
   In this way, the entire content of the output window is copied as text to the clipboard.
3. Now insert the text in any text editor, text editing program, or in Microsoft Excel.
   The results can then be printed out from these programs.

## See also:

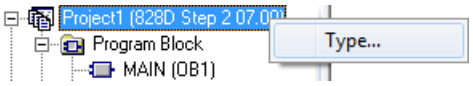
Information on using the software (Page 375)

First steps: Contents (Page 17)

Output window (View menu) (Page 451)

### 11.2.4.6 CPU type

In order to call up the "CPU Type" dialog, select the menu command **"PLC > Type"**, or open the shortcut menu of the project name:




When specifying the CPU type, the PLC Programming Tool can check instructions and parameters, which helps prevent errors as you create your program. If you have specified a CPU type for your project, the instruction tree shows any instructions that are not valid for your PLC with a red x: .

If you open a new project and begin editing a program without having specified the CPU type, you can program instructions, addresses and functions for the PLC system in the PLC Programming Tool that are not supported by all CPUs. If you use instructions, addresses or functions in your project that are not supported by your target CPU, when you try to download the project, it will be rejected by the PLC.

#### Specifying a CPU type selection

You can select a PLC by picking a model from the drop-down list box in the dialog box "CPU Type". If you are working, for example, with a "828D Step 2", the list box displays the following:


In addition to the CPU type, the firmware version is displayed in the list box.

When you specify the CPU type, the range and function restrictions for the latest firmware release of the "828D Step 2" are considered. However, you may have an early CPU firmware release that does not support functions available in the latest CPU firmware release. In order to ensure that both the CPU type and the version of the firmware are taken into consideration during the range checks, you can have the PLC Programming Tool read the CPU type information directly from the PLC.

**Note**

You do not need to configure communications when you are specifying the CPU type selection: You only need to have communications established if you want the PLC Programming Tool to read the remote CPU type for you.

#### Read-out of the removed CPU type by the PLC Programming Tool

In order to read the CPU type and the product version of the firmware, click the button in the "CPU Type" dialog.

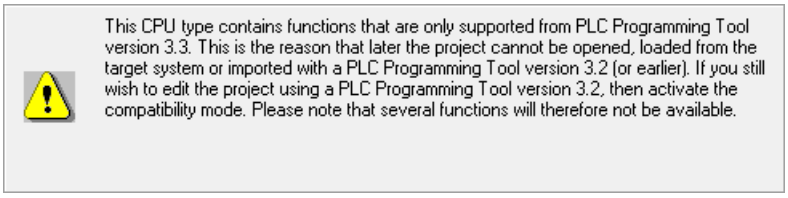
**Note**

You must already have established communications successfully in order to read the CPU type and firmware information. You can use the button Communications... in the dialog box "CPU Type" either to establish communication with a PLC or to select a target PLC when you have more than one PLC connected to a communication network.

### Compatibility

Some CPU types include new functions that are only supported with the PLC Programming Tool Version 3.3 or higher. Projects in which these CPU types are selected can only be opened with a PLC Programming Tool Version 3.3.

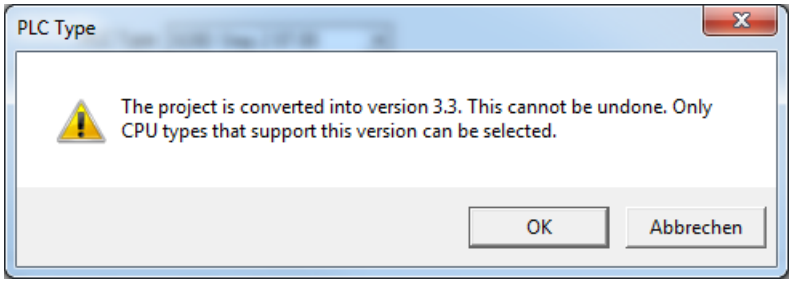
If you have selected such a CPU, a corresponding warning appears in the dialog:



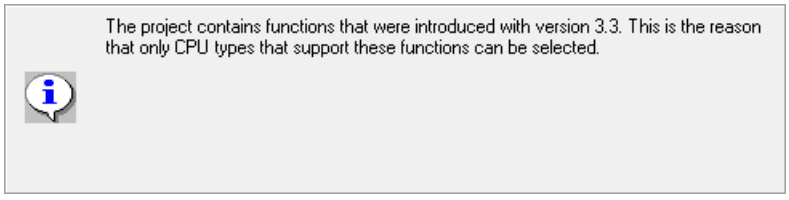

For these CPU types, the compatibility mode can be activated when the selection is made. The project can thus continue to be opened using the PLC Programming Tool Version 3.2 (or earlier).


Some functions are not supported in compatibility mode.

If you accept the CPU type without activating compatibility mode, the project will be converted into Version 3.3. This cannot be undone. A corresponding warning follows.




If you created a project with a CPU type that supports the new functions, but have not activated the compatibility mode, it will only be possible in this project to choose between CPU types that support these functions. In such a case, a corresponding note will be displayed in the dialog.




**See also:**

Communication (Page 50)

Compiling (Page 459)

Memory areas of the automation system (Page 262)

Supported instructions (Page 259)

## 11.2.5 Menu Debug

### 11.2.5.1 Program status

#### Program status in LAD

These help topics are explained in the following:

#### Enabling the program status

After you have successfully established communication between the Programming Tool and a PLC and have loaded a program into the PLC, you can operate and test the networks in your program with the function "Program Status".

Set up the program editor in such a way that the section and the networks of the program that you want to test are displayed.

**Note**

If you load a program into the PLC and are using a CPU that does not support loading in RUN mode, you are prompted to bring the PLC into STOP mode. If you want to be able to view continuous updates of the program status, you must make sure that you switch the PLC back into the RUN mode.

#### CPUs that support download in the RUN mode

| | |
|---|---|
| 828D | 808D-PPU14x |
| 828D Step 2 | 808D-PPU15x |
| | 808D-PPU16x |

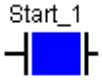
## Procedure

To enable the program status, proceed in one of the following ways:

- Select the menu command "Debug > Program Status".
- To test, click the "Program status" button in the function bar.

The status data is recorded in the mode previously selected on the "LAD Status" tab (Tools > Options) (Page 492). See description below.

The program status is displayed in the program editor.




**Note**

If you enable the program status, many other functions in the PLC Programming Tool are deactivated. For instance, you cannot change your program unless you disable the program status again. Other functions, e.g. switching the display from one program editor to another, mean that the program status is automatically disabled. If you wish to display the status again, you must reselect the command "Program Status".

### Troubleshooting

If you have problems when enabling the program status, please consider the following prerequisites:

- You must have set up communication (so that you can load your program into the PLC).
- You must have selected the correct CPU version so that you can load your program into the PLC.
- Your program must be able to be compiled error-free.
- Your program must be able to be loaded error-free into the PLC.
- Your PLC must be in the RUN mode in order to be able to display continuous updates of the status. Otherwise, only changes at the inputs and outputs (if they are available) are displayed. As the program is not executed in the PLC, changes at the inputs and outputs do not have the same effects as you would expect on the program logic in the displayed program status.
- If you display another program area, which is not executed (e.g. an interrupt program or subprogram or an area which was skipped due to a jump instruction), the status is not displayed as the code is not scanned.

**Pause Program Status**

- Select the menu command **Debug > Pause Program Status**.
- To test, click the "Pause program status" button in the function bar.

This can be necessary if values change too quickly for you to read them. In the execution status this can also make a single execution of a program section, e.g. a subprogram, visible. To do this, set the networks which are not being processed in the program editor and pause the status display with the execution status activated. The next time the networks are executed in the program editor, the status display is updated. The status update then stops.

## Displaying the program status

The program status can be executed for CPUs in the "execution status" mode and for all CPUs in the "end-of-scan status" mode. The selection is made on the "LAD Status" tab (Tools > Options) (Page 492). If the execution status is activated for a CPU that does not support this status, the end-of-scan status is automatically executed.

**Note**

If you display the program status in LAD, the Boolean operations (contacts, coils) are displayed as colored blocks if the value of the address is 1 (i.e. the bit is enabled). The actual data value from other addresses is displayed next to the address, or instead of the address. The display is updated when changes are read from the PLC. The value of non-Boolean addresses is displayed and updated as quickly as the communication permits it.

## CPUs that support the execution status

828D (as of PLC 05.04)
828D Step 2

808D-PPU14x
808D-PPU15x
808D-PPU16x

## Execution status

**Execution status (execution status activated)**

In the execution status in LAD, the value of operands and the signal flow for execution of the individual instructions during the execution of the program in the scan cycle are displayed. The execution status shows the values of the operands at the time of execution of the instruction; this may be overwritten again by subsequent program instructions. All of the displayed data values of the PLC are recorded in a single program scan cycle. The statuses of the local data memory and the accumulators are displayed. The status values are only updated and displayed if the PLC is in the RUN mode.

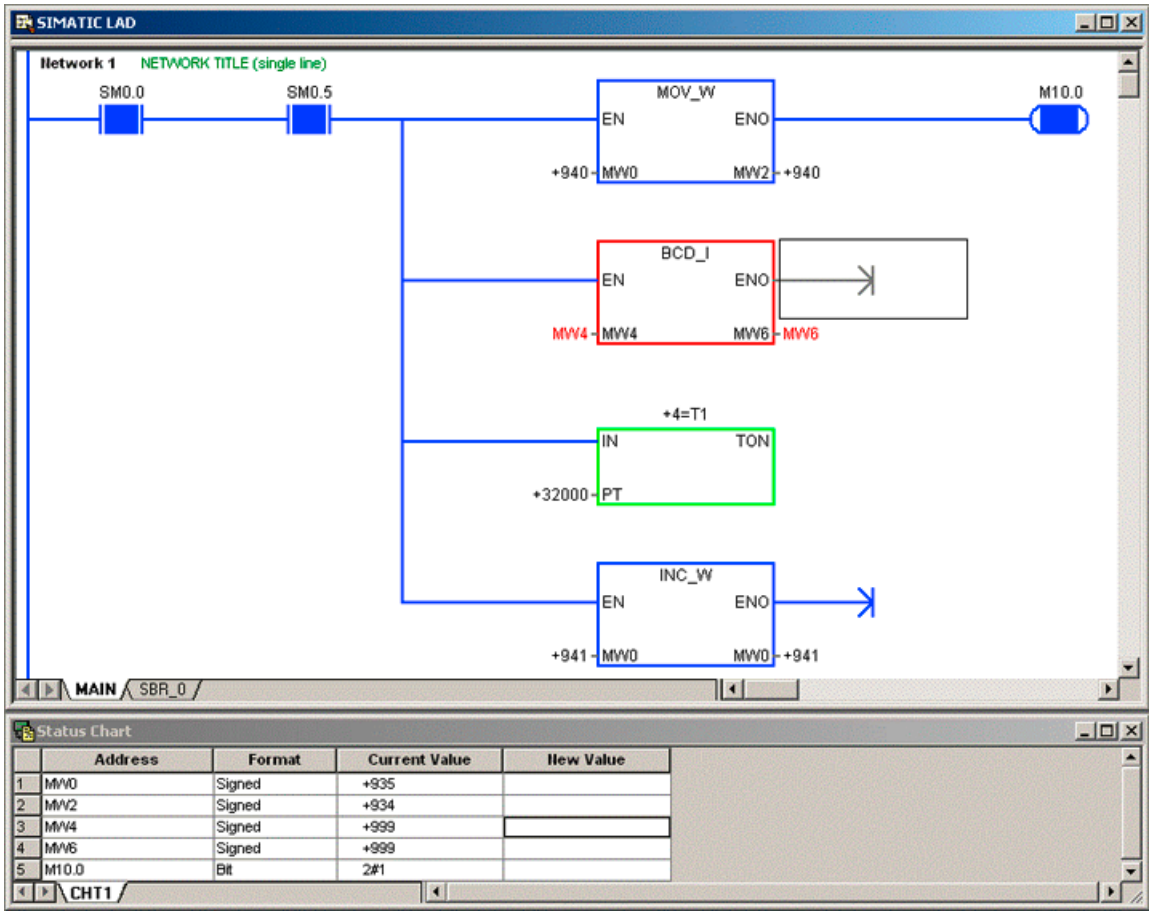
**Colors for execution status:**

- The busbar is marked with a color if the program is being scanned, default setting: blue.
- The signal flow in the diagrams is marked with a color, default setting: blue.
- Contacts: The instruction is marked with a color if the contact is switched on, default setting: blue.
- Coils: The instruction is marked with a color if the output is switched on, default setting: blue.
- Boxes and subprograms: Boxes and subprograms are marked in color if the instruction is enabled and is being executed without error, default setting: blue.
- Timers and counters: Timers and counters are marked in color if they have valid data, default setting: green.
- For jump and jump mark instructions the signal flow is marked in color if it is active. If it is not active, this is color-coded differently, default setting: gray.
- Red (red is the default setting) indicates that an instruction was not executed error-free.
- Gray (gray is the default setting) indicates that no signal flow is available, that the instruction is not being scanned (it is skipped or not called) or that the PLC is in the STOP mode.

You can also set your own colors on the "LAD Status" tab (Tools > Options) (Page 492).

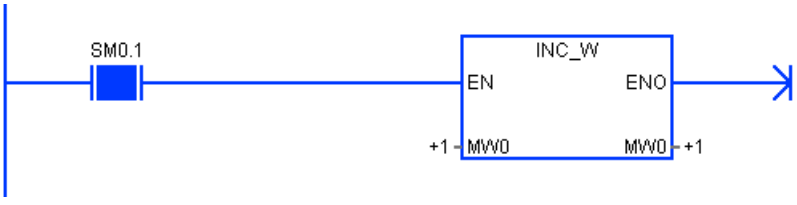
**Example for the execution status**

The following example uses the execution status in the program editor in LAD. The status charts only use the end-of-scan status data acquisition. Some differences in the displayed values of rapidly changing data can be seen if the LAD diagram and the corresponding status chart are compared. These differences are created by the various types of acquisition of the status data.




**Note**

If the same operand is used as the input and output parameter in an instruction, the value of this operand can only be determined by the execution status at one time. In this case, it is the value after execution of the instruction. The value after execution of the instruction is also displayed for the input parameter and not the actual input value.




**End-of-scan status (execution status deactivated or CPU does not support execution status)**

The end-of-scan status shows the status results that are read at the end of the program scan cycle. These results may not indicate all changes to the values of the addresses of data in the PLC, because subsequent program instructions write a value and overwrite it again before the end of the program scan cycle is reached. The end-of-scan status shows the data values that are scanned at the end of multiple scan cycles. This results from the different speeds

between the fast scan cycle of the PLC and the relatively slow transmission of the PC status data. The statuses of the local data memory and the accumulators are not displayed. If you switch the PLC into the STOP mode, you receive a status update too.

**Colors for end-of-scan status:**

- Contacts and coils: Contacts and coils are marked in color if they are live or logically true, default setting: blue (as signal flow).

You can also set your own colors on the "LAD Status" tab (Tools > Options) (Page 492).

---

**Note**

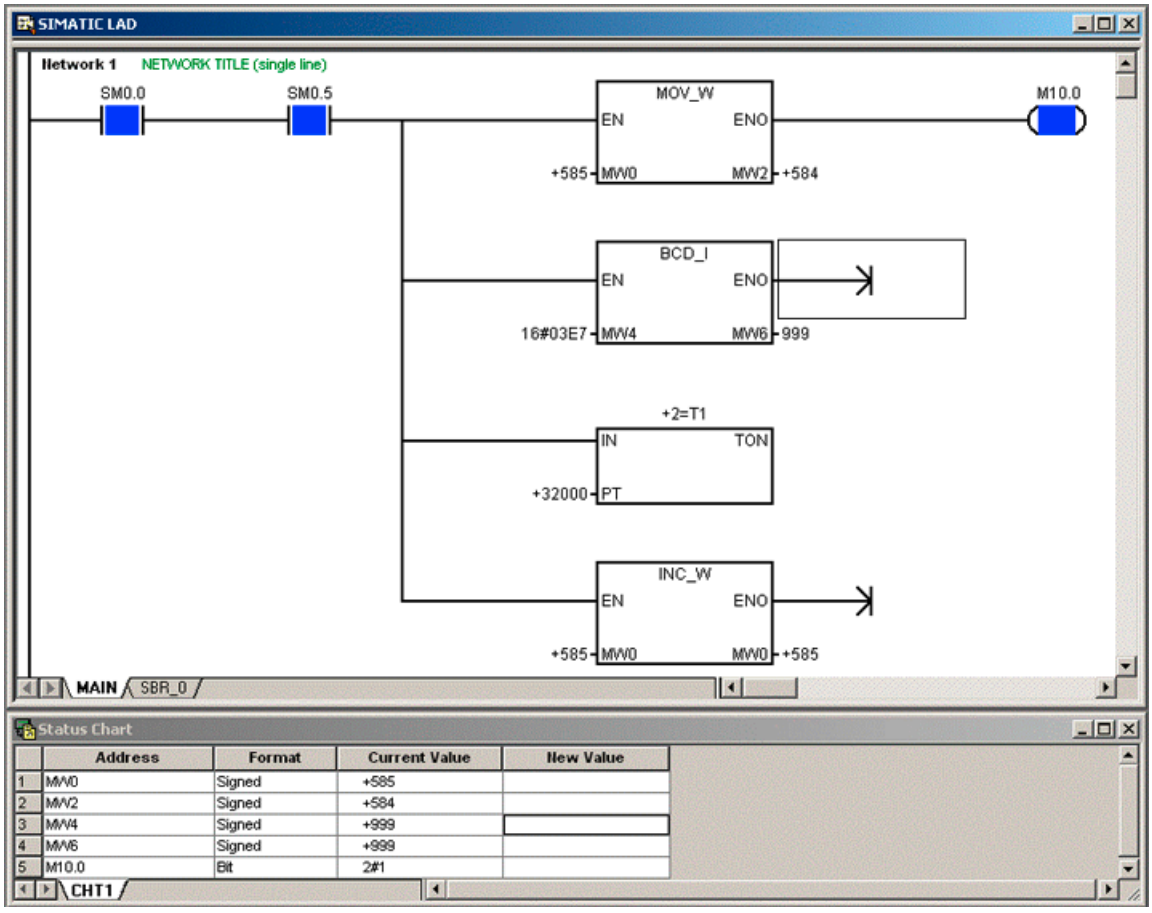
The color marking for "signal flow" does not mean that there is always a signal flow. Since only the values from the end of the scan cycle are displayed, it can sometimes be difficult to evaluate just what the "signal flow" representation really means. Boolean contacts and coils are marked in color in the program status in LAD corresponding to the value of the bit operand. If the bit value is 1 (bit is enabled), then the instruction is marked in color. However, this does not necessarily mean that the instruction was actually executed. There are multiple conditions that can result in an ambiguous signal flow display:

- If, when evaluating the status, the PLC is in the STOP mode, contacts can be activated, however, coils and boxes are not switched on because the program is not being executed.
- If the program contains a jump instruction, then it is possible that the networks that you are investigating do not display the expected results because the PLC skipped these instructions while executing the program.
- It is a similar situation if you consider a subprogram. The Boolean operands can be activated, however, the subprogram logic can only be executed if the subprogram is activated. If the subprogram was not called from the main program, then no logic of the networks was executed regardless of what the bit values of the instructions display.

---

**Example for the end-of-scan status**

The following example uses the end-of-scan status in the program editor in LAD. Because both windows (Program Status and Chart Status) use the same data of the end-of-scan status, the two windows show the same data values of the PLC. This can be misleading because a program can assign the same address many values before the status is recorded at the end of the scan cycle. Temporary values from intermediate calculations are not displayed. In this example, the box BCD_I was not executed error-free, because the input does not contain a valid BCD value. In contrast to the example for the execution status, in which the box is displayed in red, you must read the status values in order to determine that the BCD value is invalid.




## Restrictions regarding the program status

The advantage of displaying the status in the program editor is due to the fact that you obtain a graphic display of the events in your program. However, not all of the same tools are available as in the status chart (e.g. the functions "Single Read", "Write All").

It is important that you know the restrictions of the program status when you are testing and monitoring your program.

## Adjusting the display of the program status

Select the menu command **Tools > Options** and open the "LAD Status" tab in order to edit the following settings:

| | |
|---|---|
| Zoom factor | To edit the scaling<br>**Shortcut key:**<br>You can use the shortcut key to quickly set the zoom factor in the program status. Press the Ctrl key and the plus key in the numerical keypad of the keyboard to increase the display size. Press the Ctrl key and the minus key in the numerical keypad of the keyboard to reduce the display size. |
| Field width and height | To edit the grid settings<br>You can increase the field width so that information can be displayed which otherwise would be cut off.<br>You can reduce the field height so that there is space to show more networks on the screen. |
| Colors | You can change the colors for the various status categories. For the end-of-scan status, only the color for the signal flow is relevant. |
| Operand display | You can display the operands either within or outside the instructions. You can also display the status value without changing the name or the address of the operand. |
| Execution status | You can activate the execution status. |

**Note**

If the execution status is activated for a CPU that does not support this status, the end-of-scan status is automatically executed.

In addition, you can rearrange other windows in the Programming Tool in order to create more space for the window of the program editor (or to display it simultaneously with another window such as the status chart, the symbol table or the cross references). With the commands from the menus "View" and "Window", you can enlarge or reduce the windows with your mouse and move them to different positions until all windows are arranged according to your needs.

## See also

Overview of test and monitoring functions (GS 5.1) (Page 55)

Displaying the status in the status chart (GS 5.3) (Page 68)

Executing a certain number of scan cycles (GS 5.4) (Page 73)

Displaying the data block status (GS 5.5) (Page 74)

Downloading a program to the CPU (GS 4.3) (Page 52)

Cross-references and elements used (Page 439) (Ensure that program changes do not produce duplicate assignment.)

RUN/STOP modes of the PLC (Page 457)

### 11.2.5.2 First cycle / several cycles

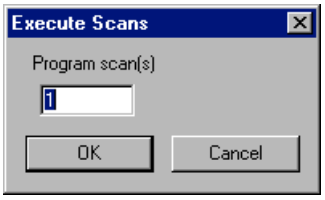
You can specify that the PLC should process a certain number of scan cycles of your program (from one scan cycle up to 65,535 scan cycles). If you specify that the PLC should execute a certain number of scan cycles, you can monitor the processing of the process variables.
In the first scan cycle, the value of SM0.1 = 1 (ON).

**Executing a single scan cycle:**

1. The PLC must be in the STOP mode. If it is not already in STOP, switch the PLC into the STOP mode (Page 101).
2. Select the menu command **Debug > First Scan**.

**Executing multiple scan cycles:**

1. The PLC must be in the STOP mode. If it is not already in STOP, switch the PLC into the STOP mode (Page 101).
2. To execute multiple scan cycles, select the menu command **Debug > Multiple Scans....** This opens the dialog box "Execute Scans".




3. Specify how many scan cycles should be executed and confirm with "OK".

**Note**

Make sure that you switch the PLC back into the RUN mode (this requires that you click the RUN button in the toolbar or select the menu command **PLC > RUN**) if you wish to return to normal program processing.

**See also:**

Overview of test and monitoring functions (GS 5.1) (Page 55)

Displaying the status in the editors (GS 5.2) (Page 60)

Displaying the status in the status chart (GS 5.3) (Page 68)

Displaying the data block status (GS 5.5) (Page 74)

Downloading a program to the CPU (GS 4.3) (Page 52)

Cross-references and elements used (Page 439) (Ensure that program changes do not produce duplicate assignment.)

RUN/STOP modes of the PLC (Page 457)

### 11.2.5.3 Chart status

#### Precondition

After you have loaded your program into the PLC, you can generate one or multiple status charts to monitor and test program execution.

The program is continuously executed if the PLC is in the RUN mode. You can enable the chart status so that the status values in the chart are continuously updated (not interrupted). As an alternative, using the function "Single Read", you can generate a "snapshot" of the status values in the chart without having to enable the status chart.

While you are viewing a status chart, you can also switch the PLC into the STOP mode and only execute the first or a certain number of scan cycles, in which you monitor program execution.

**Note**

Please note that you cannot change your chart if the chart status is enabled.

Disable the chart status if you wish to edit the chart.

These help topics are explained in the following:

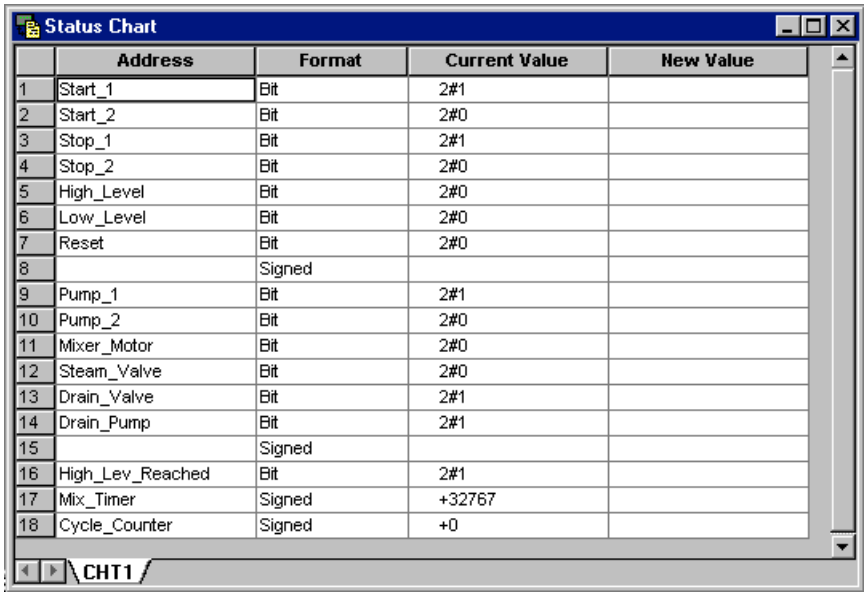
#### Example for the status chart

Opening a status chart is not the same as enabling a status chart. You can open and evaluate or change a status chart: However, if you do not execute the command "Single Read" (in the menu "Debug" or in the toolbar) or enable the chart status (in the menu "Debug" or in the toolbar), no status information will be displayed in the column "Actual Value".

If you use the function "Single Read" (this is only available when the chart status is disabled) to evaluate a status chart, the actual values of the PLC are accepted and displayed in the column "Actual Value". However, the values are not updated while the PLC is executing the program.

If you enable the chart status (in the menu "Debug" or in the toolbar), the actual values of the PLC are regularly updated. If changes are received from the PLC, then the column "Actual Value" is updated.

You can assign (write) certain values in the PLC using the column "New Value".




## Opening or enabling a status chart

Open a status chart to evaluate or change the contents of the chart. Enable the status chart so that the status information can be updated.

To open a status chart, proceed in one of the following ways:

- Click on the button of the status chart in the "Navigation" toolbar (Page 500)
- Select the menu command **View > Status Chart**.
- Open the directory of the status chart in the project tree (Page 447) and double-click the symbol of a table.
- If your project includes more than one status chart, using the tab for the CHT1 CHT2 CHT3 status charts at the lower edge of the window, you can switch over between the individual charts.

To enable a status chart, proceed in one of the following ways:

- If you wish to continually update the status information in the status chart, enable the chart status: Select the menu command **Debug > Chart Status** or click the appropriate button in the toolbar.
- If you only require a "snapshot" of the values, execute the function "Single Read": Select the menu command **Debug > Single Read** or click the appropriate button in the toolbar (if the chart status is enabled, then the function "Single Read" is deactivated).

**Tips:**

If you open a status chart, then the status is still not displayed. You must enable the status chart so that the status information is updated.

Enabling the status chart will have no effect if the chart is still empty: You must first create your status chart by entering program values (operands) in the column "Address" and entering a data type in the column "Format" for each address.

### Working with multiple status charts

You can create additional status charts in several ways:

- In the instruction tree, right-click the "Status Chart" folder and, in the shortcut menu, select the command **Insert Status Chart**.
- Open the window "Status Chart" and call the menu "Edit" or right-click and select the command **Insert Contents > Chart**.

**Note**

- After you have inserted a new status chart, a new tab 

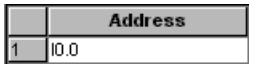
is displayed at the lower edge of the "Status Chart" window. If you wish to switch between the status charts, simply click the tab of the required status chart.
- Sometimes, a tab is hidden by the buttons on the right-hand side that are used to scroll. If a tab is not visible, drag the demarcation line between the tab area and the scroll buttons to display additional tabs.

### Creating a status chart

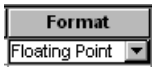
You can enter addresses in a status chart in order to monitor and control values from your program. Values of timers and counters can be displayed as bits or words. If you wish to display the value of a timer or a counter as a bit, then the status of the output is displayed (on or off). If you wish to display the value of a timer or a counter as a word, then the actual value is used.

To create a status chart, proceed as follows:

1. In the address field you enter the addresses of the desired values.
   Most of the memory types listed in the memory areas of the automation system (Page 262) are valid, with the exception of data constants and accumulators.
   To edit an address field, select the desired field using the cursor keys or the mouse. If you enter data, existing data are deleted and the new characters are entered. The field is selected if you double click with the mouse or press key "F2". You can then move the cursor using the cursor keys to the position that you wish to edit.


2. If the element involves a bit (e.g. I, Q or M), then the bit format is displayed in the second column. If the element involves a byte, word or double word, select the field in the column "Format" and double-click or press the space bar or ENTER in order to scroll through the valid formats until the correct format is displayed.


To insert a new row, open the menu "Edit" or right-click a field in the status chart so that the shortcut menu is displayed and select the command **Insert Contents > Row**. A new row is then inserted in the status chart above the cursor position. You can also move the cursor to a field in the last row and press the down arrow key in order to insert a row at the bottom of the status chart.

**Tips:**

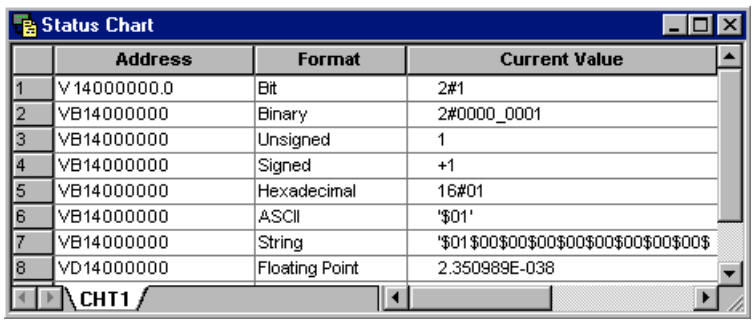
You can select addresses in the symbol table and copy these into the status chart in order to generate your chart faster.
You can display the status a multiple number of times. You can classify the elements in logical groups to display each group in an individual chart. In this way, you avoid having to scroll through extremely long lists.

## Shortcut key combinations for editing

- To insert a new row with the following address and the same data format, select an address field and press the Enter key.
- To scroll through the possible data formats for a specific address, select the field "Data Format" and press the Enter key several times. To display all available data formats, open the drop-down list box.
- Use the TAB key to jump into the next field of the chart.
- To set the width of a column, position the mouse cursor on the edge of a column until the appearance of the cursor changes and then drag to increase or decrease the width of the column.
- To select a complete row (to cut or copy), click once the number of the row.
- To insert a new row, select a field or a row and right-click. In the shortcut menu select the command **Insert Contents > Row**. The row is inserted at the cursor position. The subsequent rows are shifted down by one row.
- To insert a row at the bottom of the status chart, move the cursor to a field in the last row and press the down arrow key.
- To delete a field or a row, select the field or the row and right-click. Select the menu command **Delete > Selection**. If you delete a row, the subsequent rows are shifted up by one row.
- To select the entire status chart, click once the upper left corner above the row numbers.

## Data formats

The data format that you assign to a value defines how the value is represented in the status chart.
In the example below, VD14000000 (and therefore VB14000000 and V14000000.0) contains the value 1.



**Tips:**

Bit and binary values are both introduced by the number 2 and the # symbol. Hexadecimal values are introduced by the number 16 and the # symbol. Bit values have one digit. Binary values have eight digits.
Signed and unsigned values use the basis 10 (decimal).

## Single read or continuous chart status

- If you only require a "snapshot" of the values, execute the function "Single Read": Select the menu command Debug > Single Read or click the appropriate button in the toolbar  (if the chart status is enabled, then the function "Single Read" is deactivated).
- If you wish to continually update the status information in the status chart, enable the chart status: Select the menu command Debug > Chart Status or click the appropriate button in the toolbar  .

## Working with test functions in the status chart

You access the test functions (Single Read, Write All) using the menu Debug or using the toolbar with the test functions.

| | |
|---|---|
|  | **Single Read**<br>Use Single Read if you require a "snapshot", i.e. a single update of the program status of all values. By default, the chart status continually scans the PLC for status updates. If you click a status chart and the chart status is disabled, then the button for Single Read is activated. |
|  | **Write All**<br>After you have entered the values in the column "New Value" in the status chart, write the required changes to the PLC using the command "Write All". |

## See also

Overview of test and monitoring functions (GS 5.1) (Page 55)

Displaying the status in the editors (GS 5.2) (Page 60)

Executing a certain number of scan cycles (GS 5.4) (Page 73)

Displaying the data block status (GS 5.5) (Page 74)

Downloading a program to the CPU (GS 4.3) (Page 52)

Cross-references and elements used (Page 439) (Ensure that program changes do not produce duplicate assignment.)

RUN/STOP modes of the PLC (Page 457)

### 11.2.5.4 Data Block Status

#### Precondition

If the CPU that you are using supports data blocks, you can, after you have loaded your program into the PLC, display the current values of the data blocks in the PLC.

#### CPUs that support data blocks


828D
828D Step 2
808D-PPU14x (only special data blocks)
808D-PPU15x
808D-PPU16x

#### Procedure

The program is continuously executed if the PLC is in the RUN mode. You can enable the data block status so that the actual values in the data block are continuously updated (not interrupted).

While you are viewing a data block chart, you can also switch the PLC into the STOP mode and only execute the first or a certain number of scan cycles, in which you monitor program execution. The actual values of the data block are then continuously updated.

**Note**

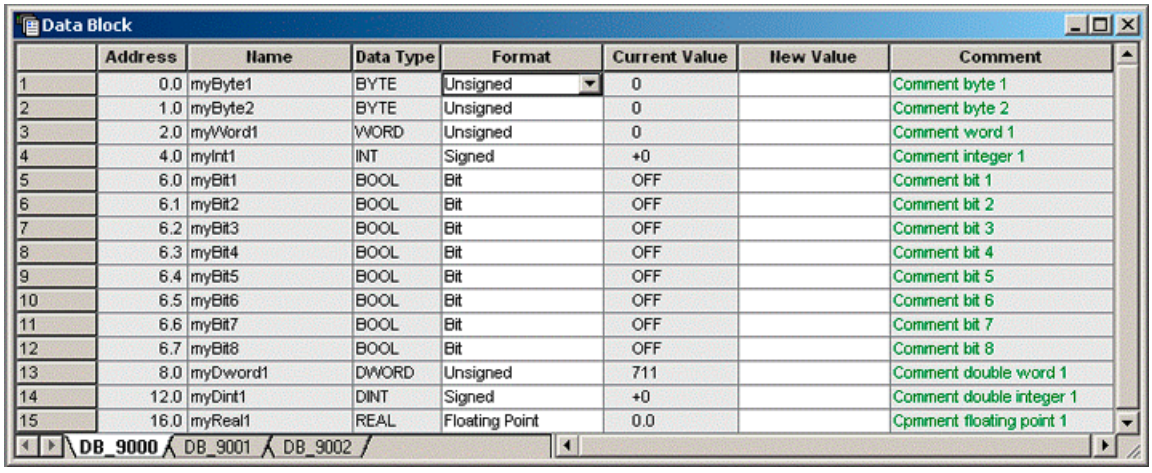
Please note that you cannot change your data block if the status is enabled.

Disable the status if you wish to edit the data block.

#### Example for the data block status

If you enable the data block status (in the menu "Debug" or in the toolbar), the values in the column "Actual Value" are regularly updated with the actual values of the PLC.

You can assign (write) certain values in the PLC using the column "New Value".



Data Block

| | Address | Name | Data Type | Format | Current Value | New Value | Comment |
|---|---|---|---|---|---|---|---|
| 1 | 0.0 | myByte1 | BYTE | Unsigned | 0 | | Comment byte 1 |
| 2 | 1.0 | myByte2 | BYTE | Unsigned | 0 | | Comment byte 2 |
| 3 | 2.0 | myWord1 | WORD | Unsigned | 0 | | Comment word 1 |
| 4 | 4.0 | myInt1 | INT | Signed | +0 | | Comment integer 1 |
| 5 | 6.0 | myBit1 | BOOL | Bit | OFF | | Comment bit 1 |
| 6 | 6.1 | myBit2 | BOOL | Bit | OFF | | Comment bit 2 |
| 7 | 6.2 | myBit3 | BOOL | Bit | OFF | | Comment bit 3 |
| 8 | 6.3 | myBit4 | BOOL | Bit | OFF | | Comment bit 4 |
| 9 | 6.4 | myBit5 | BOOL | Bit | OFF | | Comment bit 5 |
| 10 | 6.5 | myBit6 | BOOL | Bit | OFF | | Comment bit 6 |
| 11 | 6.6 | myBit7 | BOOL | Bit | OFF | | Comment bit 7 |
| 12 | 6.7 | myBit8 | BOOL | Bit | OFF | | Comment bit 8 |
| 13 | 8.0 | myDword1 | DWORD | Unsigned | 711 | | Comment double word 1 |
| 14 | 12.0 | myDint1 | DINT | Signed | +0 | | Comment double integer 1 |
| 15 | 16.0 | myReal1 | REAL | Floating Point | 0.0 | | Comment floating point 1 |

DB_9000 DB_9001 DB_9002

## Enabling the data block status

Open a data block and enable the data block status so that the actual values from the PLC are displayed. If you enable the data block status without having opened a data block, the first data block from the data block list is opened automatically.

To enable the data block status, proceed in one of the following ways:

- Click the "Data block status" button in the toolbar.
- Select the menu command **Debug > Data Block Status**.

If your project includes more than one data block, if the data block status is active then you can switch between the individual data blocks. To do this, proceed in one of the following ways:

- Switch between the individual data blocks using the tab for the data blocks DB_9000 DB_9001 DB_9002 at the lower edge of the window.
- In the project tree, double-click the desired data block.

**Note**

- When the data block status is activated, the structure of the variables (name, data type, initial value, and comment) is write-protected. The format of the variables can be changed.
- The structure of the data block in the project must match the structure of the data block in the PLC. If you have modified the structure of the data block, then the complete project has to be loaded again.

## Changing the actual values in the data block status

If the data block is not write-protected, you can change the actual values in the PLC with the data block status active. To do so, enter the desired values in the column "New Value" and write the changes to the PLC with the command "Write All".

**See also**

Overview of test and monitoring functions (GS 5.1) (Page 55)

Displaying the status in the editors (GS 5.2) (Page 60)

Displaying the status in the status chart (GS 5.3) (Page 68)

Executing a certain number of scan cycles (GS 5.4) (Page 73)

Saving data in the variable memory with the help of a data block (Page 424)

Downloading a program to the CPU (GS 4.3) (Page 52)

Cross-references and elements used (Page 439) (Ensure that program changes do not produce duplicate assignment.)

RUN/STOP modes of the PLC (Page 457)

## 11.2.6 Menu Tools

### 11.2.6.1 Setup

**Setup**

Select the menu command **Tools > Customize** to configure the following settings:

- Tab "Commands" (Page 487) allows you to change the appearance and/or content of the Programming Tool toolbars.
- Tab "Add-On Tools" (Page 488) allows you to add frequently used tools to the menu Tools.

**Tab "Commands" (Tools > Customize)**

Rearrange the Programming Tool toolbars:

- Select the menu command **Tools > Customize**.

Click the "Commands" tab to change the appearance and/or content of the toolbars.

**Change the display:**

- Activate the "Display tooltips" checkbox. A brief description is displayed in the tooltip when the mouse cursor is positioned on a button.
- Activate the "Do not display button three-dimensional" checkbox. The buttons are then not represented plastic but rather flat.

**Move a button:**

- Select a toolbar from the field "Category" in order to display the buttons of that toolbar.
- To move a button from its default toolbar to another toolbar, select the name of the toolbar that currently contains the button from the field "Category". Drag the button out of the field "Buttons" and drop it onto the desired toolbar in the programing tool's working area to add it to that toolbar.
- To remove a button from a Programming Tool toolbar, drag the button off the Programming Tool toolbar and drop it in the field "Buttons" of the dialog box "Customize".

**See also:**

Information on using the software (Page 375)

First steps: Contents (Page 17)

### Tab "Add-On Tools" (Tools > Customize)

Add a tool to the menu Tools in the Programming Tool:

- Select the menu command **Tools > Customize**.

This function is intended to save the time spent browsing for and starting frequently used tools. To add a tool, click the "Add-On Tools" tab, click the Add button and complete the following fields:

| | |
|---|---|
| **Menu Text** | Choose a name to identify the tool in the menu. |
| **Command** | Supply the file name of the tools program or batch file. |
| **Arguments** | Supply the command line arguments used by *.exe files. |
| **Initial Directory** | Supply the initial directory for the tool. |

Use the browse ... buttons to find files and directories.

When you have made the settings for a tool, it appears in the Programming Tool as an item on the menu Tools.

**See also:**

Information on using the software (Page 375)

First steps: Contents (Page 17)

## 11.2.6.2 Options

### Options

Select the menu command **Tools > Options** to configure the following settings:

- Open the "General" (Page 489) tab to make the language and regional settings (measurement system, time format and date format).
- Open the "Colors" tab (Page 489) to assign fonts and a color scheme to the various windows of the Programming Tool working area.

"LAD Editing" tab (Page 490)

"LAD Program Status" tab (Page 492)

"NC variables" tab (Page 493)

**See also:**

Information on using the software (Page 375)

First steps: Contents (Page 17)

## Tab "General" (Tools > Options)

Set up Programming Tool programming options:

- Select the menu command **Tools > Options**.

Open the "General" tab and select the following options:

| | | |
|---|---|---|
| **Default Editor** | LAD | Predefined |
| **Mnemonics** | International | Predefined |
| **Programming** | SIMATIC | Predefined |
| **Address View** | Variable memory, data block | Selectable |
| **Language** | German, English, French, Spanish, Italian or Chinese | Selectable |
| **"Regional Settings" tab** | Measurement System (U.S. or metric), Time Format (12 or 24 hour), Date Format (mm/dd/yy or dd.mm.yy) | Selectable |

**See also:**

Information on using the software (Page 375)

First steps: Contents (Page 17)

## Tab "Colors" (Tools > Options)

Change the appearance of windows and window elements in the Programming Tool working area:

- Select the menu command **Tools > Options**.

Select the tab "Colors" of the dialog box "Options" to change selections for the colors and fonts of windows and window elements.

1. From the field "Windows", choose which window (all Windows or an individual window) you want to change.
2. To assign a new color to your window selection, make a selection from the field "Window Color".

3. To change the color or font of a window element, first select the element from the field "Category".

   **Note**

   Valid elements differ from window to window, and may include Default Text, Comments, Literals, Symbols, Instructions, Addresses, and Keywords.

4. To change the color of a selected window element, choose a new color from the "Color" field.
5. To change the font of a selected window element, choose a new font from the "Font" field and/or a new font size from the "Size" field.

   **Note**

   The Font and Size options are not available for all window elements.

6. Click "OK" to confirm your changes or "Cancel" to exit the dialog box without saving your changes.

**See also:**

Information on using the software (Page 375)

First steps: Contents (Page 17)

## Tab "LAD Editing" (Tools > Options)

This tab allows you to change the way your Program Editor window appears.

**Note**

The settings that you define here only affect how the window looks when you are editing; they do not have to match your settings for Program Status (which are controlled by the "LAD Program Status" tab).

### Scale (zoom)

Allows you to zoom the Program Editor window to a larger (percentages greater than 100) or smaller (percentages less than 100) display.

**Note**

Hold down the `Ctrl` key and the `plus` (+) or `minus` (-) key from the numeric keypad to quickly increase or decrease the scale percentage (that is, the zoom factor).

### Grid (field width, field height)

Allows you to change the ratio of width to height for the fields of the grid.

The unit is one pixel.

**Note**

- Increase the width for long symbolic names that should not be truncated.
- To display more rows on your screen, reduce the height.

**Symbol Information Table ("Name", "Address", and "Comment" field width)**

Allows you to change the "Name", "Address", and "Comment" field widths of the Symbol Information Table.

The unit is one field.

**Example:** If your grid width is 85 pixels per field, then a name width of 1 is equal to 85 pixels per field, a name width of 2 is equal to 170 pixels, etc.

Increase this measurement if you do not want long symbolic names, addresses or comments to be truncated.

**Note**

The function "Symbol Information Table" is accessed from the menu "View". A check mark next to the command means that the function is turned on and a Symbol Information Table appears beneath every network in your program that contains any symbols.

**Symbolic addressing**

If you have symbolic addressing turned on, you can choose whether to view only symbols or to view both symbols and absolute addresses.

**Note**

The function "Symbolic Addressing" is accessed from the menu "View". A check mark next to the command means that the function is turned on. Otherwise, all addresses are displayed as absolute addresses only.

**See also:**
Tab "LAD Program Status" (Tools > Options) (Page 492) (On this tab you change the way your Program Editor window appears when you are viewing Program Status.)

## Tab "LAD Program Status" (Tools > Options)

### LAD Program Status

This tab allows you to change the way your Program Editor window appears.

**Note**

The settings that you define here only affect how the window looks when you are viewing Program Status; they do not have to match your settings for editing programs (which are controlled by the "LAD Editing" tab).

**Scale (zoom)**

Allows you to zoom the Program Editor window to a larger (percentages greater than 100) or smaller (percentages less than 100) display.

**Note**

Hold down the `Ctrl` key and the `plus` (+) or `minus` (-) key from the numeric keypad to quickly increase or decrease the scale percentage (that is, the zoom factor).

**Grid (field width, field height)**

Allows you to change the ratio of width to height for the fields of the grid.

The unit is one pixel.

**Note**

You might want to increase the width measurement in order to view data values or long symbolic names without truncation.
You might want to decrease the height measurement in order to view more rows on your screen.

**Signal flow**

If you have already used the tab "Colors" of the dialog box "Options" to customize the way that various elements are displayed in the Programming Tool, you may also want to select your own color to represent the program status. To do this, select a color from the list box.

Signal flow:
- The busbar is marked in color if the program is being scanned.
- The signal flow in the diagrams is marked in color.
- Contacts are marked in color if the contact is switched on.
- Coils are marked in color if the output is switched on.
- Boxes and subprograms are marked in color if the instruction is enabled and is being executed without error.
- For jump and jump mark instructions the signal flow is marked in color if it is active.

No signal flow: Shows that there is no signal flow and that the instruction is not being sampled (it is skipped or not called) or that the PLC is in the STOP mode.

| | |
|---|---|
| Not executed: | The operands are marked in color if the instruction is not being executed. |
| Always valid: | Timers and counters are marked in color if they have valid data. |
| Execution error: | Shows that an instruction was not executed without error. |

**Note**

These color settings are only relevant for the execution status. Only the "signal flow" color is used for the end-of-scan status. This highlights the contacts and coils when they are switched on.

**Display**

Operand Inside: for box instructions, operands are truncated if they exceed the field width (controlled by the Grid selection, above).

Operand Outside: operands may be truncated if instructions are so close in a network that names overlap.
Only Status Value: operands are not displayed.

**Execution status**

This activates the execution status. If the execution status is not activated, the end-of-scan status is performed. If the execution status is activated for a CPU that does not support the execution status, the end-of-scan status is performed automatically.

### CPUs that support the execution status

828D (as of PLC 05.04)
828D Step 2

808D-PPU14x
808D-PPU15x
808D-PPU16x

### See also:

Tab "LAD Editing" (Tools > Options) (Page 490) (On this tab you change the way your Program Editor window appears when you are editing programs.)

### "NC variables" tab (Tools > Options)

You can change the units system for NC variables in this tab.

The following can be selected:

- Active system of units, i.e. variables are displayed in the system of units active in the NC.
- Metric system of units, i.e. variables are displayed in the metric system of units independent of the system of units active in the NC.
- Inch system of units, i.e. variables are displayed in the inch system of units independent of the system of units active in the NC.

**Note**

The settings made here are valid for all selected NC variables.

#### 11.2.6.3 Additional tools

In this tab can use your own tools which you have set up before.

**See also:**

Setting up additional tools (Page 488)

### 11.2.7 Menu Window

#### 11.2.7.1 Cascade

Select the menu command **"Window > Cascade"** to arrange all open windows in an overlapping pattern with all title bars visible. Click any title bar to make that the active window.

**See also:**

Information on using the software (Page 375)

First steps: Contents (Page 17)

#### 11.2.7.2 Horizontal (menu Window)

Select the menu command **"Window > Horizontal"** to arrange all open Programming Tool windows so that all are visible and stacked vertically. You can maximize any window to see more of its contents.

**See also:**

Information on using the software (Page 375)

First steps: Contents (Page 17)

#### 11.2.7.3 Vertical (menu Window)

Select the menu command **"Window > Vertical Connection"** to arrange all open Programming Tool windows so that all are visible and stacked next to each other. You can maximize any window to see more of its contents.

**See also:**

Information on using the software (Page 375)

First steps: Contents (Page 17)

#### 11.2.7.4 Open Windows Listing (menu Window)

The menu Window contains a numbered list of currently open windows. Select the menu command **"Window > 1, 2, 3, 4 or 5"** to make a particular window the active window.

**See also:**

Information on using the software (Page 375)

First steps: Contents (Page 17)

### 11.2.8 Menu Help

#### 11.2.8.1 Contents and Index

The menu command **"Help > Content and Index"** opens the help. The help has the "Content", "Index" and "Find" tabs.

**See also:**

Information on using the software (Page 375)

First steps: Contents (Page 17)

#### 11.2.8.2 Direct Help

Obtain What's This? help for an item as follows:

- Choose the menu command **Help > What's This.**
- Press the **Shift+F1** shortcut key.

Move the mouse pointer over the item for which you want help, and then left-click.

If the What's This? help does not provide the specific help topic that you need, try clicking the item and pressing the F1 key. Not all elements of the software have related What's This? help topics: For some items, only F1 help is active.

**See also:**

Information on using the software (Page 375)

First steps: Contents (Page 17)

#### 11.2.8.3 828 in the Internet

If you have an Internet browser installed and have access to the Internet, you can get the latest product information directly from Siemens websites by using the menu command **"Help > 828 on the Web"**.

**See also:**

Information on using the software (Page 375)

First steps: Contents (Page 17)

### 11.2.8.4 Info

The menu command **Help > About** displays the version number and copyright of your Programming Tool software.

**See also:**

Information on using the software (Page 375)

First steps: Contents (Page 17)

## 11.3 Shortcut keys

You can use shortcut keys in the Programming Tool to make a variety of tasks more efficient.

**Note**

If multiple keys are listed, all keys must be pressed together in order for the shortcut to work.

These help topics are explained in the following:

### LAD shortcuts

**Navigation keys**

| Key sequence | Action |
|---|---|
| HOME | Moves cursor to the first column of the same row |
| END | Moves cursor to the last column of the same row |
| PAGE UP | Moves one screen up vertically |
| PAGE DOWN | Moves one screen down vertically |
| LEFT ARROW | Moves cursor one field to the left |
| RIGHT ARROW | Moves cursor one field to the right |
| UP ARROW | Moves cursor one field up |
| DOWN ARROW | Moves cursor one field down |
| CTRL + HOME | Moves cursor to the first field of the first network |
| CTRL + END | Moves cursor to the last field of the last network |
| F1 | Help on the current field instruction |
| CTRL + PAGE UP | Displays the next POU<br>Moves left through POU tabs |
| CTRL + PAGE DOWN | Displays the next POU<br>Moves right through POU tabs |

### Editing

| Key sequence | Action |
|---|---|
| CTRL + X | When series of networks is selected, cuts series. When cursor is on network title, cuts entire network. When cursor is on field, cuts content of field. |
| CTRL + C | When series of networks is selected, copies series. When cursor is on network title, copies entire network. When cursor is on field, copies content of field. |
| CTRL + V | When series of networks is selected, pastes series. When cursor is on network title, pastes network. When cursor is on field, pastes field. Pastes selection before/above current location of cursor. |
| F2 | Edits current field's mnemonic or operand |
| SPACEBAR | Edits current field's mnemonic or operand |
| ENTER | Edits current field's mnemonic or operand |
| CTRL + LEFT ARROW | Drops a horizontal line and moves the cursor one field to the left |
| CTRL + RIGHT ARROW | Drops a horizontal line and moves the cursor one field to the right |
| CTRL + UP ARROW | Drops a vertical line and moves the cursor one field up |
| CTRL + DOWN ARROW | Drops a vertical line and moves the cursor one field down |
| DELETE | Deletes current field, network, or selection of networks |
| BACKSPACE | Moves cursor one field to the left and deletes current field |
| SHIFT + LEFT ARROW | Selects current network and moves cursor over one field to left |
| SHIFT + RIGHT ARROW | Selects current network and moves cursor over one field to right |
| SHIFT + HOME | Selects current network and places cursor at first field of current row |
| SHIFT + END | Selects current network and places cursor at last field of current row |
| SHIFT + DOWN ARROW | Selects from current network downward |
| SHIFT + UP ARROW | Selects from current network upward |
| SHIFT + PAGE UP | Selects a page of adjacent networks from the current network upward |
| SHIFT + PAGE DOWN | Selects a page of adjacent networks from the current network downward |
| CTRL + SHIFT + HOME | Selects all adjacent networks from cursor to top of POU |
| CTRL + SHIFT + END | Selects all adjacent networks from cursor to bottom of POU |
| CTRL + A | Selects all networks in a POU |
| CTRL + Y | Toggles between symbolic and absolute addressing mode |
| CTRL + B | Toggles between data block and variable address view |
| CTRL + T | Displays symbol information tables for each network |

### Zooming

| Key sequence | Action |
|---|---|
| CTRL + PLUS SIGN | Zooms in (from numeric keypad only) |
| CTRL + MINUS SIGN | Zooms out (from numeric keypad only) |

### Dropping instructions

| Key sequence | Action |
|---|---|
| F4 | List box containing all contact mnemonic types |
| F6 | List box containing all coil mnemonic types |
| F9 | List box containing all box mnemonic types |

### Data block, symbol table, local variable table, status chart

| Key sequence | Action |
|---|---|
| CTRL + X | Cuts selected text |
| CTRL + C | Copies selected text |
| CTRL + V | Pastes selected text |
| DOWN ARROW | Moves cursor down (adds new rows to the end of the table) |
| UP ARROW | Moves cursor up |
| LEFT ARROW | Moves cursor left |
| RIGHT ARROW | Moves cursor right |
| DELETE | Deletes selected text |
| F1 | Provides help on current window |
| F2 | Lets you edit current field |
| CTRL + A | Selects all fields |
| TAB | Moves cursor from left to right and top to bottom |
| SHIFT + TAB | Moves cursor from right to left and bottom to top |
| ENTER | Confirms field and moves cursor to the right |
| HOME | Moves cursor to the first column |
| END | Moves cursor to the last column |
| CTRL + PAGE UP | Displays the next table/ chart. Moves left through the Symbol Table/ Status Chart tabs. |
| CTRL + PAGE DOWN | Displays the next table/ chart. Moves right through the Symbol Table/ Status Chart tabs. |
| SHIFT + ARROWS | Selects text |
| CTRL + I | Inserts new row above the current row |

### Changing between absolute and symbolic addressing mode

| Key sequence | Action |
|---|---|
| CTRL + Y | Toggles between symbolic and absolute addressing mode |
| CTRL + T | Displays symbol information tables for each network |
| CTRL + B | Toggles between data block and variable address view |

**Note**

You can choose between two representation forms for the symbolic addressing:

- Display only symbolic addresses
- Or display symbolic and absolute addresses

To do this, go to the **Tools > Options** menu and open the "LAD editor" tab.

# 11.4 Toolbars

## 11.4.1 Common features of toolbars

**Common features of toolbars**

**A toolbar button is colored when it is possible to use the tool.**

For example, the LAD toolbar is active if you have selected the menu command **View > Ladder**. If a toolbar button is disabled, it appears gray.

**Moving toolbars**

If you place the mouse pointer on the area that is inside a toolbar frame but outside of any buttons, you can click and drag the toolbars. The toolbars can be moved within the main window or out of the main window. When moved outside the main window, toolbars become a separate window.

**Toolbar context help links**

1. To see the name of a toolbar button, position the mouse pointer on the button. The button name appears.
2. For details about the function of a tool, press Shift+F1, position the mouse pointer on a button and click.

## 11.4.2 Standard toolbar

**Standard toolbar**



Open New Project (Page 378)


Open Existing Project (Page 379)


Save Current Project (Page 379)


Print (Page 394)


Print Preview (Page 393)


Cut Selection and Copy to Clipboard (Page 398)


Copy Selection to Clipboard (Page 399)


Paste Clipboard Contents at Cursor Location (Page 400)


Undo Last Entry (Page 398)

- Program Block (the block in the active window) (Page 459)
- Upload Project from the PLC to the Programming Tool (Page 382)
- Download Project from the Programming Tool to the PLC (Page 382)
- Sort Ascending: Sort Symbol Table Name Column A-Z (Page 446)
- Sort Descending: Sort Symbol Table Name Column Z-A (Page 446)
- Zoom: Set Magnification of LAD Views (Page 452)
- Constant Descriptor: Toggle ON/OFF Type Specifiers
- View Data Block Values (Page 424)

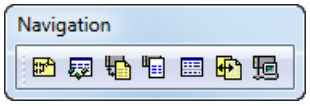
## 11.4.3 Toolbar "Debug"

**Toolbar "Debug"**


- Set PLC to RUN Mode (Page 457)
- Set PLC to STOP Mode (Page 457)
- Toggle Program Status ON/OFF (Page 60)
- Pause Program Status (Page 60)
- Toggle Chart Status ON/OFF (Page 68)
- Activate/Deactivate Data Block Status (Page 74)
- Status Chart: Single Read (Page 68)
- Status Chart: Write All (Page 68)

## 11.4.4 "Navigation" toolbar

**"Navigation" toolbar**



- Program block (Page 412)
- Symbol table (Page 413)
- Status chart (Page 420)
- Data block (Page 424)
- NC variables (Page 434)
- Cross-references (Page 439)
- Communication (Page 443)

## 11.4.5 LAD instruction toolbar

**LAD instruction toolbar**


- Insert Line Down (Page 36)
- Insert Line Up (Page 36)
- Insert Line Left (Page 36)
- Insert Line Right (Page 36)
- Insert Contact (Page 36)
- Insert Coil (Page 36)
- Insert Box (Page 36)
- Insert Network (Page 401)
- Delete Network (Page 404)
- View/Hide Symbol Information Tables for each Network (Page 445)

## 11.4.6 STL instruction toolbar

**STL instruction toolbar**



Toggle Bookmark: Set or Remove Bookmark (Page 258)


Next Bookmark: Scroll Program to Next Bookmark (Page 258)


Previous Bookmark: Scroll Program to Previous Bookmark (Page 258)


Clear All Bookmarks (Page 258)

# Index

## D



## E

## M



## N



## O

## P



## Q



## R



## S

## T

## U



## V



## W



## X



## Z