Software

WPS v2.5X

User Manual

User Manual

Series: WPS

Language: English

Document number: 10001381480 / R11

Publication date: 03/2019

Contents

1 WPS v2.5X

Technical Support: Contact a local branch or representative.

Contact us: http://www.weg.net

Publication date: 03/2019

Start

Click the Welcome button to start.

Welcome

2 Introduction

2.1 Welcome

Welcome to the WPS!

WPS v2.5X is an innovative integrated tool to access several families of WEG products, adding the concepts of:

- Multi-Products, aiming at meeting wide range of WEG products;
- Multi-Use, in order to enable:
  - Equipment configuration,
  - Equipment programming in Ladder language,
  - Monitoring of equipment, and
  - Assistance for creation and configuration of applications in the automation area.

The programming environment of the Ladder language meets the requirements of IEC 61131-3.

It is strongly recommended that the user is familiar with these guidelines and concepts that this standard presents in order to use the software optimally.

2.2 System Requirements

Item	Description
Plataform	Windows 7 or greater
Processor	Minimum: Core i3
	Recommended: Core i5
Memory	Minimum: 1 GB
	Recommended: 4 GB
Screen Resolution	1024x768 or greater
Disk Space	2 GB
Communication	USB, Ethernet TCP/IP and Serial Port

2.3 Supported Equipments

The following table shows the equipments and firmware versions supported by WPS v2.5X:

Equipment	Ladder Available		Trace	Trend	Parameters
	WLP	WPS			
PLC300	-	v1.76+	-	v1.76+	v1.76+
SCA06	<v2.00	v2.00+	v1.10+	v1.10+	v1.10+
CFW300	-	v1.00+	-	v1.00+	v1.00+
CVW500	v1.00+	-	-	v1.00+	v1.00+
PSRW	-	-	-	v1.00+	v1.00+
SSW900	-	v1.10+	-	v1.00+	v1.00+
CFW100	<v3.00	v3.00+	-	v3.00+	v3.00+
CFW-11	v1.00+	-	v1.00+	v1.00+	v1.00+
CFW500	v1.10+	-	-	v1.10+	v1.10+
CFW501	v1.30+	-	-	v1.30+	v1.30+
MW500	v1.50+	-	-	v1.50+	v1.50+
SSW06	v1.40+	-	-	v1.30+	v1.30+
SSW07	-	-	-	v1.20+	v1.20+
SSW08	-	-	-	v1.30+	v1.30+

- WLP: WEG Ladder Programming

2.4 Copyright Notice

2.5 Safety Notice

Using this software may alter the drive's operation and performance. The user is responsible for adopting the necessary precautions in order to guarantee the safety of the equipment and persons involved. Before applying this Software, carefully read the Online Help Instructions. Non compliance with these instructions may cause serious damage to the device and result in severe bodily injury.

3 What's New

3.1 What's New - This Version

WPS v2.5X

New Functions:

- Included CFW500 V1.10+;
- Included CFW501 V1.30+;
- Included MW500 V1.50+;
- Included CFW-11 V1.00+;
- Included SSW06 V1.30;
- Included SSW07 V1.20;
- Included SSW08 V1.30.

Enhancements:

- Corrections in the manual (help).

3.2 What's New - Previous Versions

WPS V2.40

Functionalities:

- Configuration Wizard for CFW300, PSRW, SSW900;
- Main signal monitoring window's - CFW300, SSW900;
- CALL block for PLC300 since V3.41 and V4.11 (according to hardware model);
- USB driver enabling communication with PSRW;
- Enhanced search system (includes search by blocks, indexed variables, texts).

Improvements:

- Corrections in the manual (help);
- WComm (communication manager supporting FTP protocol);
- WEG USB driver supports Windows 10 (x86 and x64) - including build 1607 or newer;
- New markers addresses for setting up HMI LEDs (PLC300 V4.1X).

WPS V2.30

Functionalities:

- Included CFW300 V1.0X and V1.2X;
- Included PLC300 V3.3X;
- Windows 10 support (x86 and x64);
- Automatic installation of the WEG USB driver;
- Automatic installation of the FTDI driver;
- Welcome window with quick access to some functions;
- Improved communication configuration.

Improvements:

- Improved performance and trend graph interface;
- Improved organization of menus.

WPS V2.20

Functionalities:

- Function to remove unused variables
- Function to search and replace variables in the resource
- Variables filter option inside the variables table

Improvements:

- JRE (Java) embedded into WPS installation
- Improvements in trend performance
- Improvements in design, icons, windows ...
- Main resource set was simplified
- Menu actions restructured and actions sensitive to the context
- Configuration import with validation of .bkp file
- Alphabetic ordering of configurations and resources
- Support to bitfields in parameters table
- Parameter monitoring inside ladder

WPS V2.10

Functionalities:

- PLC300 v2.30 with support to Hot Download and Watchdog
- Importer/Conversor from WLP Projects to WPS to the equipment SCA06
- Full impression of configurations and resources
- Variables Monitoring with functionality to backup and restore variables values
- Created window with information about the device connected
- Create window with additional information such author, description and client about the resource

Improvements:

- WPS Help rewriten to all ladder blocks
- Window of variable selection changed to a table format with filter options
- Move variables from one block to another in the ladder editor
- Search of variables with filters
- Support the movement of multiple variables simultaniously in structures and also record of last type added for future additions
- USB Driver with support to Windows 8.1

WPS V2.00

Functionalities:

- Support to SCA06 starting at firmware version v2.00, agregatting the functios of SuperDrive G2 and WLP
- PLC300 v2.10 with support to Strings

WPS V1.80

Functionalities:

- PLC300 V2.00 Support (Hardware Version 2)
- P_RAMP Block (Pulse Train)

WPS V1.70

Functionalities:

- Upload and download variable data, this function allows to preserve variables values during the program download. This function is compatible with firmware v1.70 or above.
- Export and download binaries
- Creation of "text input" component on screen editor

Improvements:

- Project management:
- Creation of multiple CANopen configurations
- Possibility of association of a ladder to a task on the last wizard step during the new ladder wizard
- Ladder editor:
- Optimization in ladder diagram editor, reducing the use of CPU and memory
- Dragging variable from variables list and dropping it into the block argument
- RUNG title length increased
- Number of elements per RUNG increased
- Automatic tag break
- Screen name display during configuration of F buttons in screen editor
- Optimization in monitoring to reduce the response time
- Optimization in compiler to reduce compile-time
- Clock synchronization between the computer clock and project setup

Corrections:

- Correction during removal of POU from task, where changes are not saved when another action is not made in sequence
- Correction in the compiler when the same POU was linked to more than one task
- Correction of lock during the load of variables table when large structure type is used
- Correction for performance increase during search of structure fields
- When the component "numeric input" was configured as INT and contains decimal digits the limit values were not being compiled correctly
- Unset of main selection during resource version change
- Function keys were not working when used with variables of type BOOL with size bigger than 0 (zero), the compiler was not generating the correct memory address for this variables
- SD Card password was being stored even with the related option disabled, the compiler was not checking the signal of SD Card password
- During compiling errors where the variable does not exist anymore, the compiler was not displaying the variable name
- When two variable nodes of structures were expanded on variables table at the same time one of the structures displays the variables names as ???
- The variables sampled in event log were not displaying the correct values
- Correction of functional deviation in the compiler when any argument from the USERFB was a member of a

structure or an instance. In this case the number of bits that was being considered was the size of the whole structure, when the correct should be only the size of this member
- Correction of functional deviation in the compiler when distinct ladders were calling USERFB instances with the same name, occasionally, the compiler was inverting the call of USERFB's. The compiler was overwriting the arguments file from the USERFB, when the name of instance was the same
- Do not allow creation of folder and files related to USERFB with invalid names
- Block of "text" component editing when it reaches his limit size
- Alarm compilation was not displaying correct variables names in case of compiling errors
- Correction of validation on new task creation
- During compilation in case of duplicated variables the compiler was not displaying the duplicated variable name
- Correction into resource comparison, in specific situations in Ladder files, some small differences are not being detected by the function

WPS V1.60

Functionalities:

- USERFB User Function Block
- Comparison of the resource in the application to a resource of the device
- Download of the device setup and CANopen configuration
- New faster and easy-to-use screen editor
- Options to enable and disable alarms on the editor

Improvements:

- Project management:
- "Save as..." in the resource configuration
- "Search" for the variables in all the resource
- Properties of the resource folder
- Edition of structure members
- Indication in the files open on the editor if they belong to the main selection
- Highlight on the programs not used in any tasks
- Link on the output window of the compiler
- Alarm keys:
- Automatic resizing of the Message component
- Shortcut keys in the alarm edition
- Printing of the screens with their properties
- Ladder editor:
- Allow to change the contact or coil type already inserted in the ladder
- Change of the output variables of the BYTE counters for BOOL to use in contacts
- Variables:
- Possibility to erase multiple variables from the variable window
- Disable the variable window when closing or changing the resource selection
- Selection of the variable address by means of the Modbus address

Corrections:

- Increase of the number of retries before displaying the error messages during the online monitoring
- Correction of fault when saving the renamed variables. After renaming the variables, the save button was not being enabled and the changes were not being saved.
- Change of the default .csv variable importation file from "," to ";".
- Locking of the renaming action in the GLOBAL_IO group.

- Change of the maximum memory area size of the screens
- Compiler presented error when compiling structure with invalid element. Change to accept repeated variables in GLOBAL_IO group.
- 32-bit data types (DWORD, DINT, UDINT, REAL) enabled for the network variables
- Fault in the compilation of the watchdog variables of the start, stop and main tasks
- Fault in the compilation of String variables using the STRING# syntax in the ReadRecipe and WriteRecipe blocks. Correction to generate error in this situation
- Fault when using the STORE block of a REAL variable for a member of a REAL structure
- Correction of a fault when saving the monitoring variables. The new variables created were not being saved.
- Correction of the ordering of the numerical fields in the variable table. .When ordering the table, the numerical fields were not correctly ordered
- Locking of the edition of system variables. It was possible to edit the variable and duplicate it.
- Correction of the fault in the resource exportation. Not all the internal files were being copied.
- Correction of the fault when importing resource with the override option enabled. The configuration was not being overridden correctly.
- Correction to remove the selection edge of the component after copying components on different screens.
- Correction to reduce the memory consumption when loading the screens.
- Problem in the alignment of the structures containing the BOOL data type. It was observed that the monitoring of the BOOL array was presenting incorrect values.
- Correction of the edition of the array of the SDO_Write block.

WPS V1.50

Functionalities:

- Program upload
- Force I/O
- Language configuration in the PLC300 setup
- Option to display variables on the alarm screens
- Option of filling in the fields "Numeric Input" and "Numeric Output" with zeros on the user's screens
- Download option:
- Initialize output and volatile variables
- Stop/Start the execution of the program automatically
- Online command menu for the password-protected operations of recording and loading program, setup and firmware on the SD card.

Improvements:

- Printing of the ladder diagrams

Corrections:

- Increase the number of retries before displaying the error messages during the online monitoring
- Correction of fault when recording the renamed variables. After renaming the variables, the save button was not being enabled and the changes were not being recorded.
- Change of the default .csv variable import file to ";" instead of ",".
- Locking of the renaming action in the GLOBAL_IO group.
- Change of the maximum memory area size of the screens
- Compiler presented error when compiling the structure with invalid element. Change to accept repeated variables in GLOBAL_IO group.
- 32-bit data types (DWORD, DINT, UDINT, REAL) enabled for the network variables
- Fault in the compilation of the watchdog variables of the start, stop and main tasks
- Fault in the compilation of String variables using the STRING# syntax in the ReadRecipe and WriteRecipe

blocks. It was corrected to generate error in this situation
- Fault when using the STORE block of a REAL variable for a member of a REAL structure
- Correction of a fault when recording the monitoring variables. The new variables created were not being recorded
- Correction of the ordering of the numerical fields in the variable table. When ordering the table, the numerical fields were not correctly ordered
- Locking of the system variables edition. It was possible to edit the variable and duplicate it.
- Correction of the fault in the resource export. Not all the internal files were being copied.
- Correction of the fault when importing resource with the override option enabled. The configuration was not being overridden correctly.
- Correction to remove the component selection edge after copying component on different screens.
- Correction to reduce the memory consumption when loading the screens.
- Problem in the alignment of the structures containing the BOOL data type. It was observed that the monitoring of the BOOL array was presenting incorrect values.
- Correction in the edition of the SDO_Write block array.

WPS V1.40

Functionalities:

- WPS translated to english and spanish
- Update center

Corrections:

- Move variables between groups by the ladder editor
- Open modbus file manager

WPS V1.30

Functionalities:

- Variables Editor Update
- Ethernet configuration at Setup
- Structure Configuration
- Recipe Configuration
- Event Log Configuration
- Modbus File Manager

Ladder Editor:

- Changes in the Desktop
- Use of Literals
- Use of Arrays
- Use of Instances and Structures
- Function Blocks Optional Arguments
- Copy/Paste New Features
- Online Monitoring New Features

Corrections:

- Optimization of the ladder editor so as to guarantee ladder files with up to 200 rungs and moderate memory and CPU consumption. In such situation, the 1.20 version of WPS presented slow performance and

crashes.

PLC300:

- New Features and Corrections of PLC300.

WPS V1.20

- Communication with the PLC300 V1.2X equipment,
- Task programming,
- Change in tag addresses for compatibility with Modbus,
- Tool to import/export projects,
- Checking the PLC300 firmware version during download,
- Stop/run programs as global actions,
- Environment persistence when exiting,
- Improvements in communication,
- Download options:
  - Initialize retentive variables,
  - Clear alarm history,
  - Font code download.

4 Installation/Uninstallation

4.1 Before Installing

Check the following items before installing the WPS v2.5X:

- If the PC meets the minimum requirements;
- If the version of the WPS v2.5X is compatible with your equipment; and
- If your current user account has administrator privileges to install the software.

4.2 Installing

When installing a newer version of WPS it is recommended that the previous version be removed first and the system be rebooted, to make sure that the new installation is performed correctly.

Close all running programs to avoid interference with the installation process and follow the steps below.

1. Double-click the installer icon to start the installer. The installation program will start and a welcome window will appear.
2. In the installation wizard, respond to the License Agreement, then press Next button.
3. Specify an empty folder inside which the WPS v2.5X will be installed. Make sure the installation location is correct and that there is sufficient disk space for installation.
4. Confirm to create an empty folder.
5. Select the type of installation you want.
6. Select the program group to create the shortcuts for the WPS v2.5X.
7. Select shortcuts for additional programs: create a shortcut on the desktop and create a shortcut for quick start.
8. Click the Install button to start installation.
9. Click the Finish button to complete the setup.

4.3 Uninstalling

If necessary, remove the WPS v2.5X by using the following procedures.

NOTE!
Always use the Add or Remove Programs application to remove WPS. Do not delete files and folders manually.

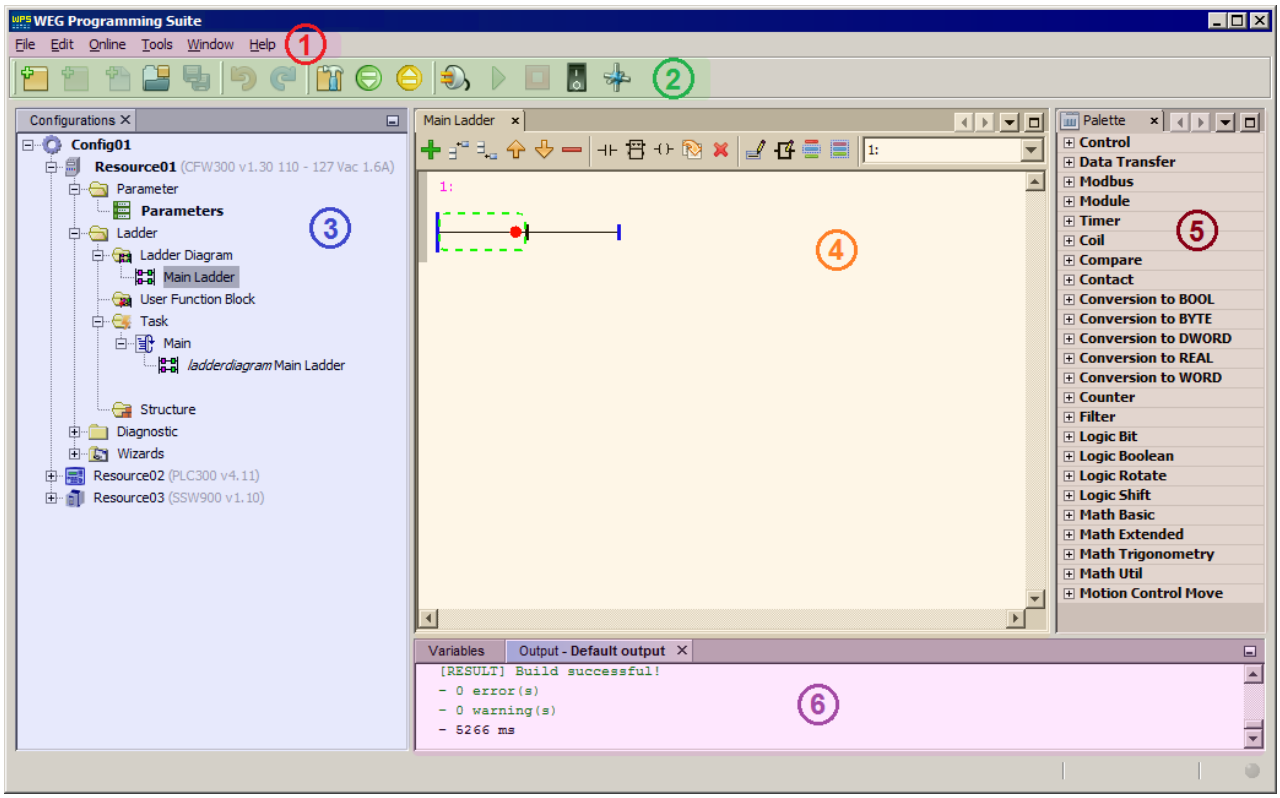
1. From the Start menu, select Control Panel.
2. Double click on the Add or Remove Programs button.
3. Select WPS v2.5X from the list, and click on the Remove button.
4. Follow the instructions to remove the software.

5 Getting to Know the Environment

5.1 Environment

The main window of WPS v2.5X may be divided in 6 main parts:

1. **Menus**: show the several options for editing, visualization, communication, and help in the development environment;
2. **Toolbar**: displays the main commands of the software;
3. **Configurations Window**: shows the tree including the open Configuration structure;
4. **Editor Window**: main part of the development environment where the edition of the components that will form the source file takes place;
5. **Palette Window**: includes the components to be inserted into the editor through drag'n drop;
6. **Output Window**: shows compilation and download messages.



5.2 Configuration Structure

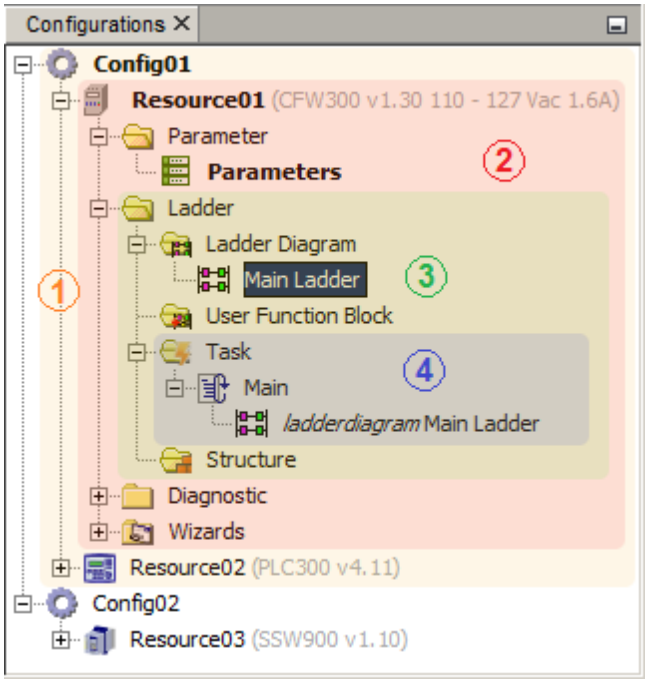
The WPS software follows the resolutions of the standard IEC 61131-3.

Therefore, every application developed on it has a configuration with the following hierarchical structure:

- **Configuration**: is at the highest level of the hierarchy, defining all the elements contained in a software application that interact to perform the control functions.
  - **Resource**: second level of the hierarchy, represents any element with processing capacity to implement programs. In the WPS, resources are independent from each other, and each one has a linked product to it. Global variables in an application are in this scope.

- **POU (Program Organization Unit)**: describe the instructions to be executed, in order to be implemented and how they interact with each other. Function blocks, functions and programs are contained in this category. In the WPS, the POU that stands out is the Ladder Diagram. Local variables of an application are in this scope, each confined to its own POU.
- **Tasks**: processes that control the order and the execution time of the POUs.

The following figure shows the layout of this hierarchy in the WPS.



Lettering:

1. Configuration;
2. Resource;
3. POUs;
4. Tasks;

6 Quick Start

6.1 Welcome Window

This chapter contains some basic information so that you get familiar with the software.

We will start by the welcome window.

The welcome window appears when the program is started for the first time.

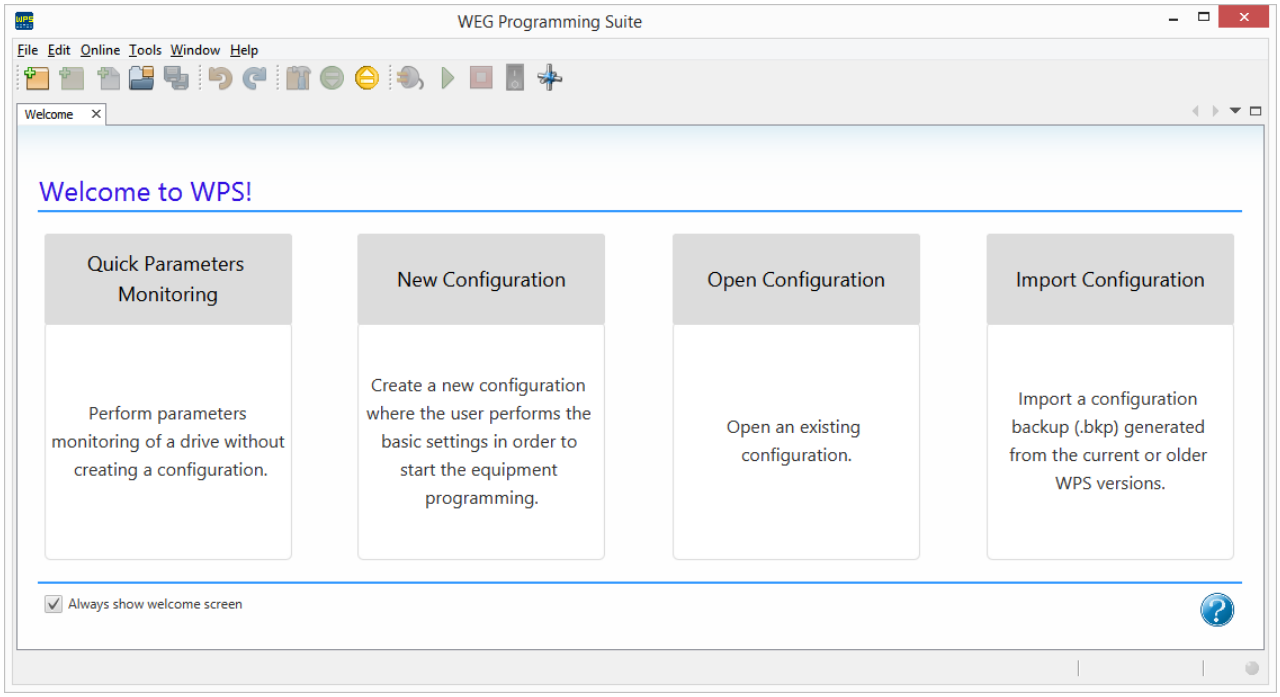
The following options are available in this window:

- Quick Monitoring of Parameters
- New Configuration
- Open Configuration
- Import Configuration

There is also one more option, **Always display welcome window**, which will be explained below.

The user can also close this window without selecting any of the options and use his/her own working methodology.

The following figure shows the WPS welcome window.

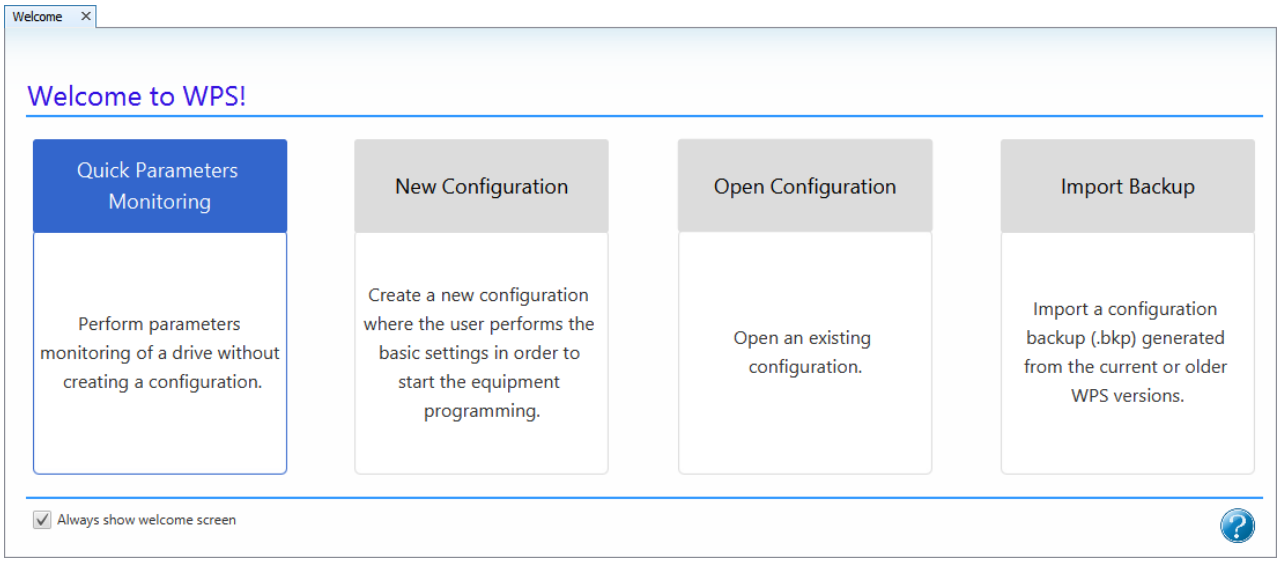


Quick Monitoring of Parameters

The quick monitoring of parameters allows viewing and changing the parameters online, but it is possible neither to create a new configuration, nor to save the parameters on the computer.

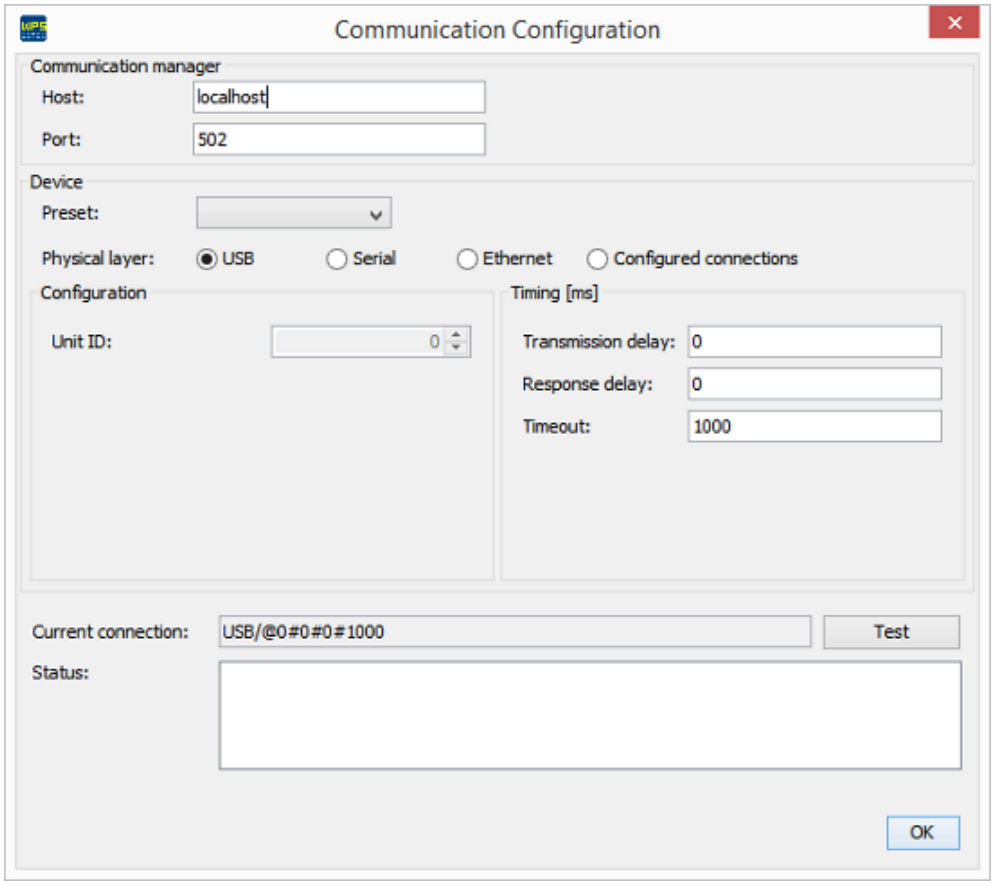
1) Function Selection

Click the option Quick Monitoring of Parameters.



2) Communication Configuration

The next window defines the options for communication with the equipment.

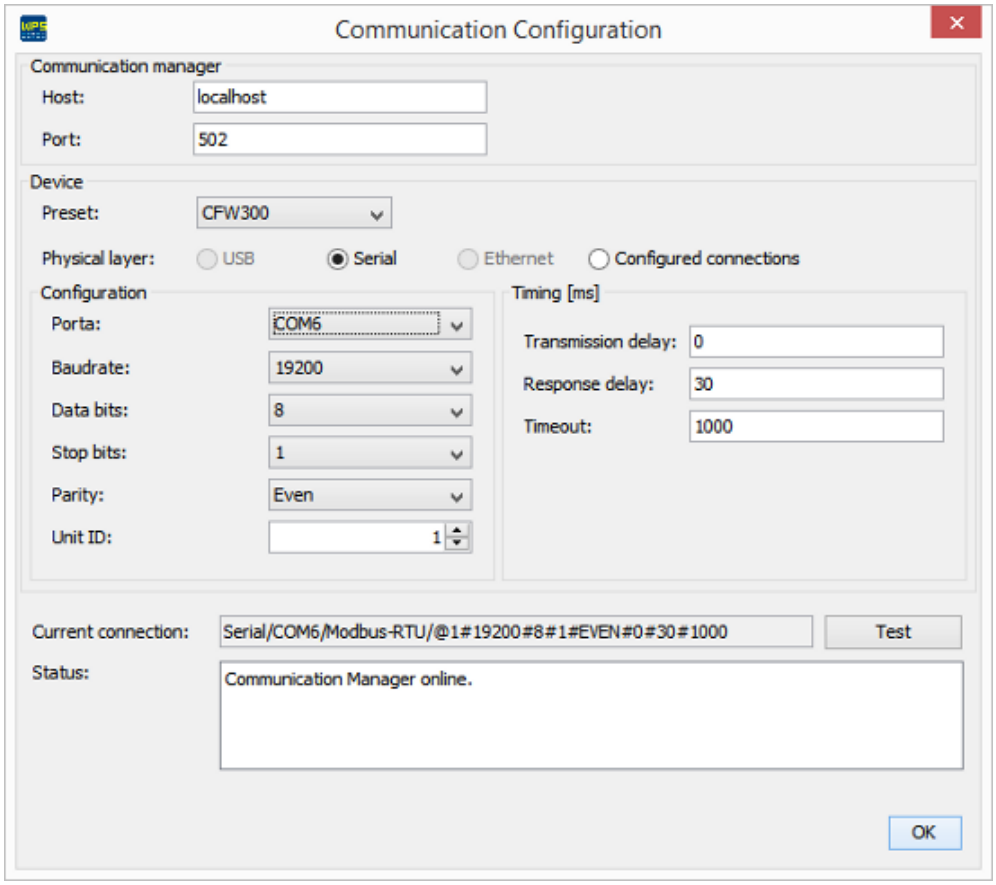


3) Select the Communication Options

Choose the correct communication options.

If you select one equipment, the default communication configuration of the equipment will be loaded to the window.

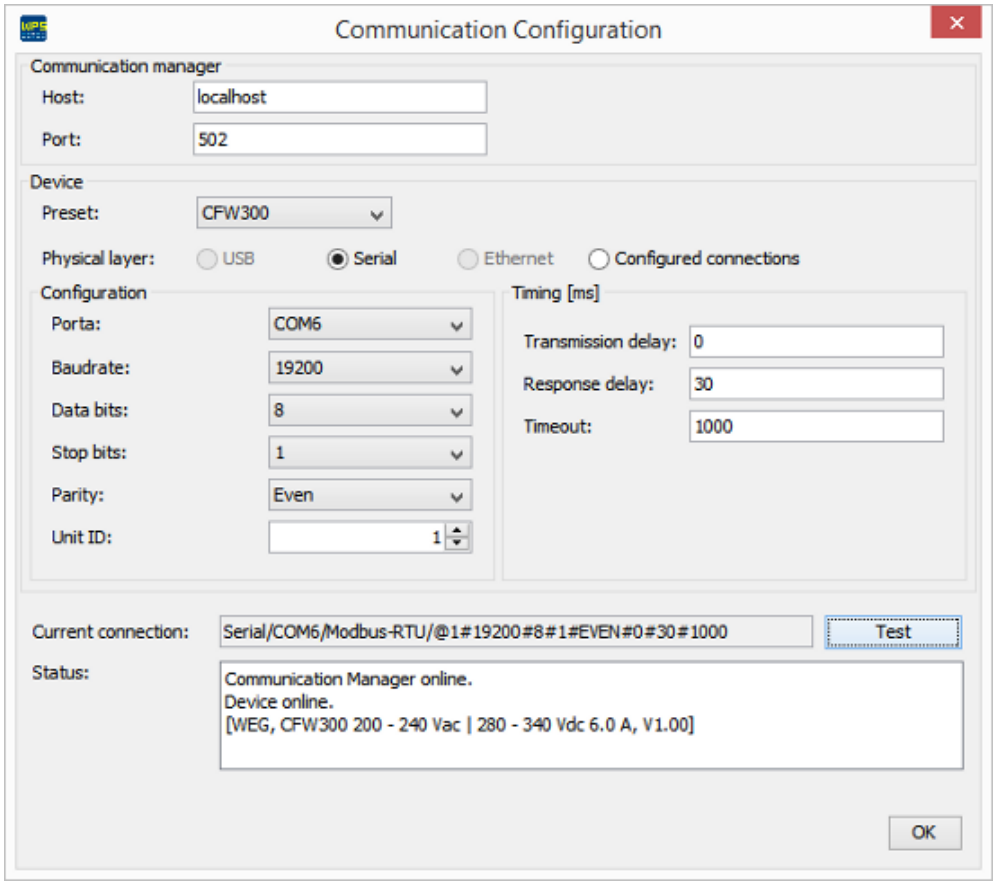
However, check that the window configuration is the same as of the equipment. If not, update this window according to the equipment configuration.



4) Testing the Communication

You can check if the communication is correct by clicking the Test button in this window.

The field is refreshed. The connected equipment appears in the status field.

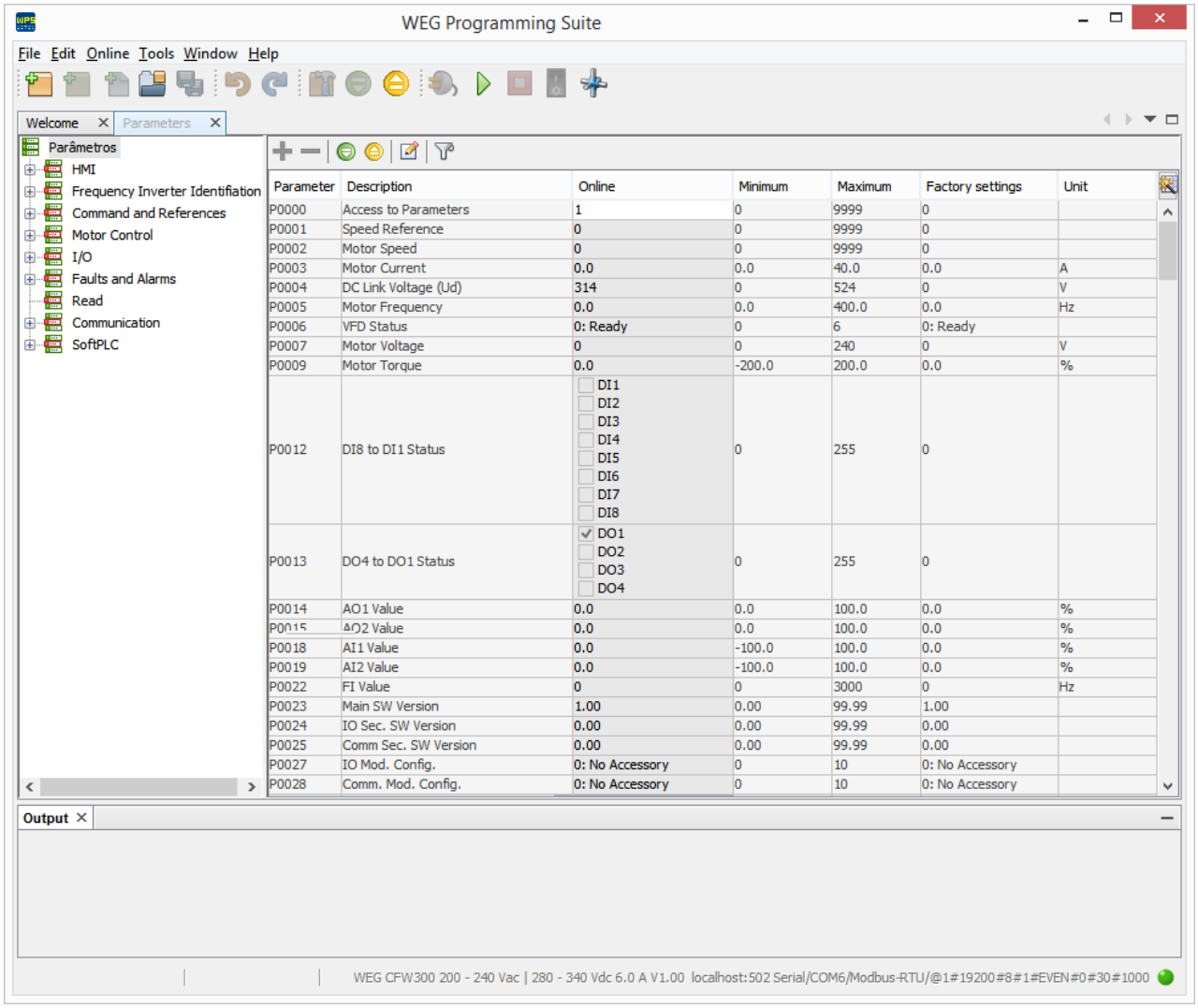


Then click OK.

5) Monitoring in Progress

A window containing all the parameters of the equipment is shown.

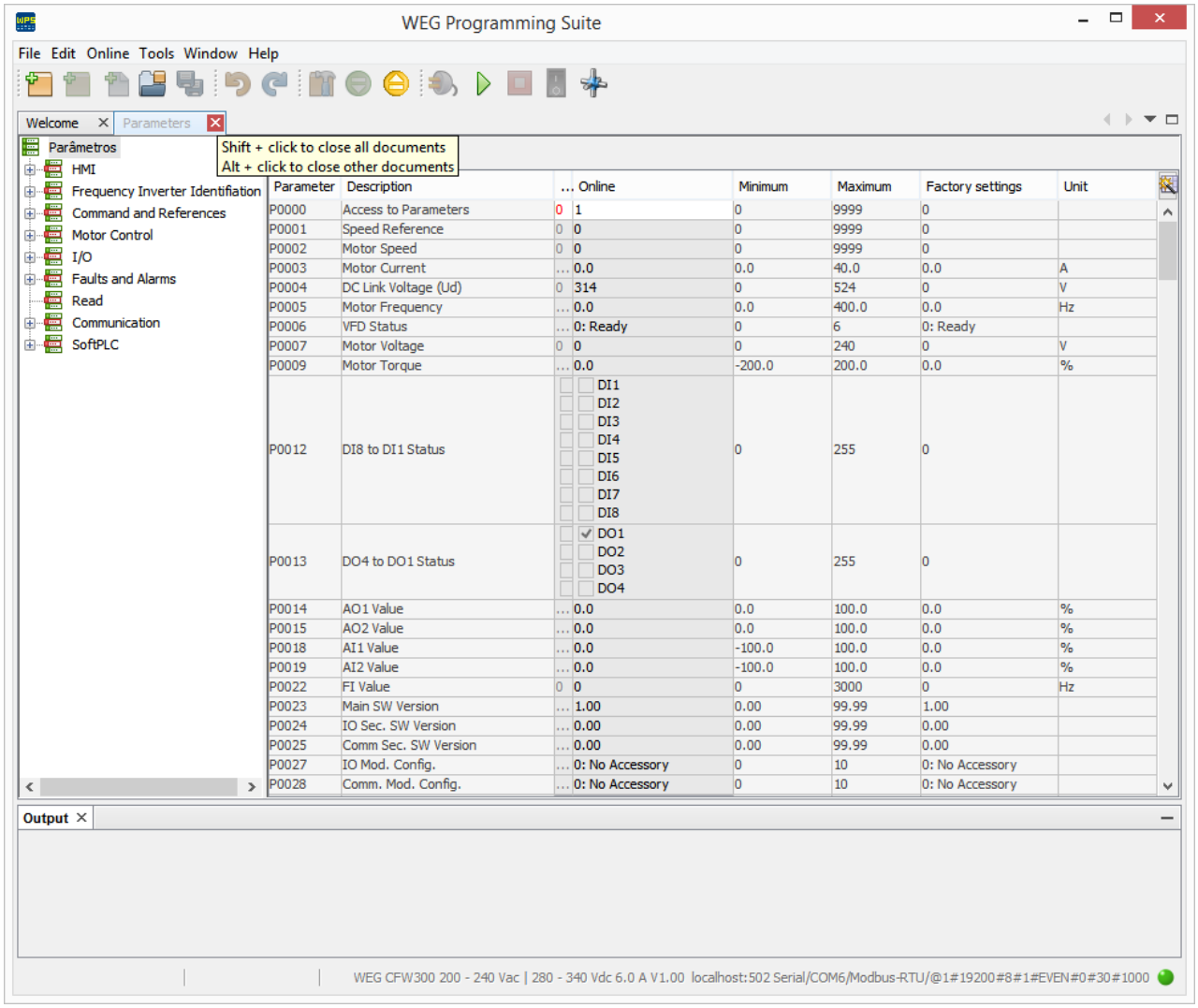
The Online field displays the real value of the parameters on the equipment, and most parameters allow modification. Reading parameters cannot be changed.



Parameter	Description	Online	Minimum	Maximum	Factory settings	Unit
P0000	Access to Parameters	1	0	9999	0	
P0001	Speed Reference	0	0	9999	0	
P0002	Motor Speed	0	0	9999	0	
P0003	Motor Current	0.0	0.0	40.0	0.0	A
P0004	DC Link Voltage (Ud)	314	0	524	0	V
P0005	Motor Frequency	0.0	0.0	400.0	0.0	Hz
P0006	VFD Status	0: Ready	0	6	0: Ready	
P0007	Motor Voltage	0	0	240	0	V
P0009	Motor Torque	0.0	-200.0	200.0	0.0	%
P0012	DI8 to DI1 Status	☐ DI1 ☐ DI2 ☐ DI3 ☐ DI4 ☐ DI5 ☐ DI6 ☐ DI7 ☐ DI8	0	255	0	
P0013	DO4 to DO1 Status	☑ DO1 ☐ DO2 ☐ DO3 ☐ DO4	0	255	0	
P0014	AO1 Value	0.0	0.0	100.0	0.0	%
P0015	AO2 Value	0.0	0.0	100.0	0.0	%
P0018	AI1 Value	0.0	-100.0	100.0	0.0	%
P0019	AI2 Value	0.0	-100.0	100.0	0.0	%
P0022	FI Value	0	0	3000	0	Hz
P0023	Main SW Version	1.00	0.00	99.99	1.00	
P0024	IO Sec. SW Version	0.00	0.00	99.99	0.00	
P0025	Comm Sec. SW Version	0.00	0.00	99.99	0.00	
P0027	IO Mod. Config.	0: No Accessory	0	10	0: No Accessory	
P0028	Comm. Mod. Config.	0: No Accessory	0	10	0: No Accessory	

6) Close the Window

In order to end the monitoring, click the Close button in the window.

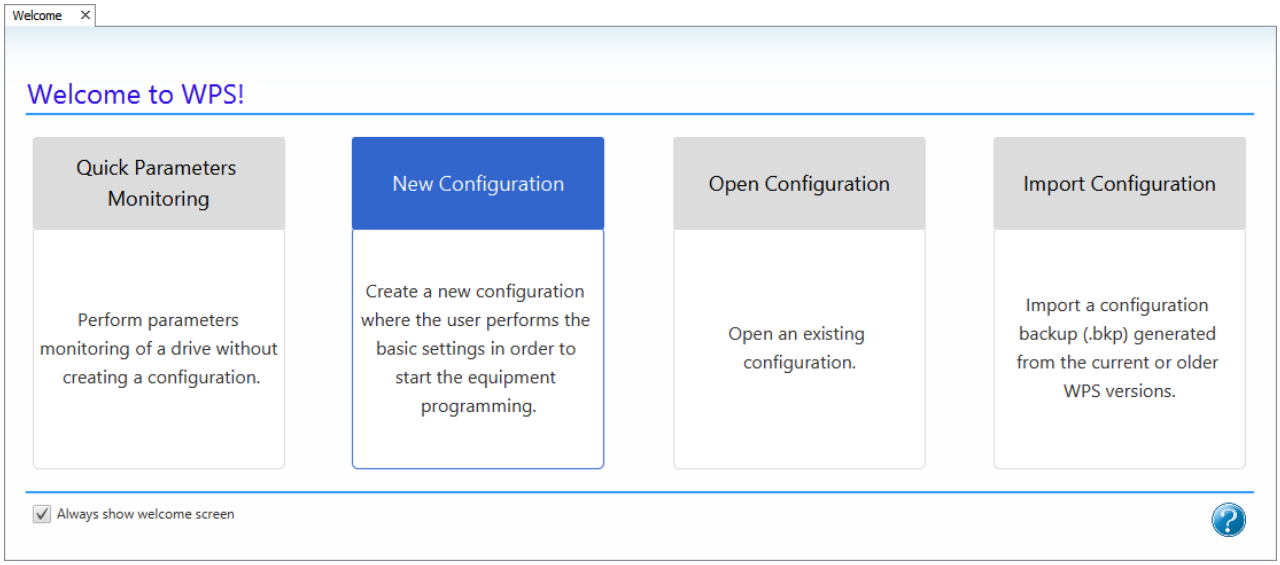


New Configuration

This creates a new configuration.

1) Function Selection

Click the New Configuration option.



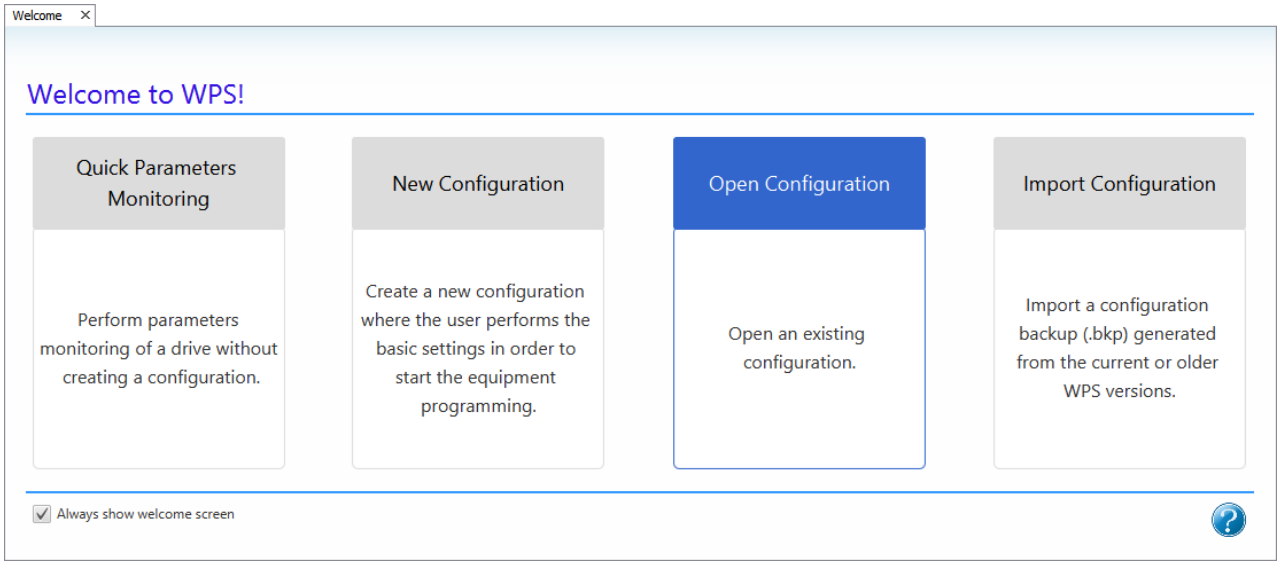
From this step on, see Creating New Configuration for further details.

Open Configuration

It allows opening a configuration created previously.

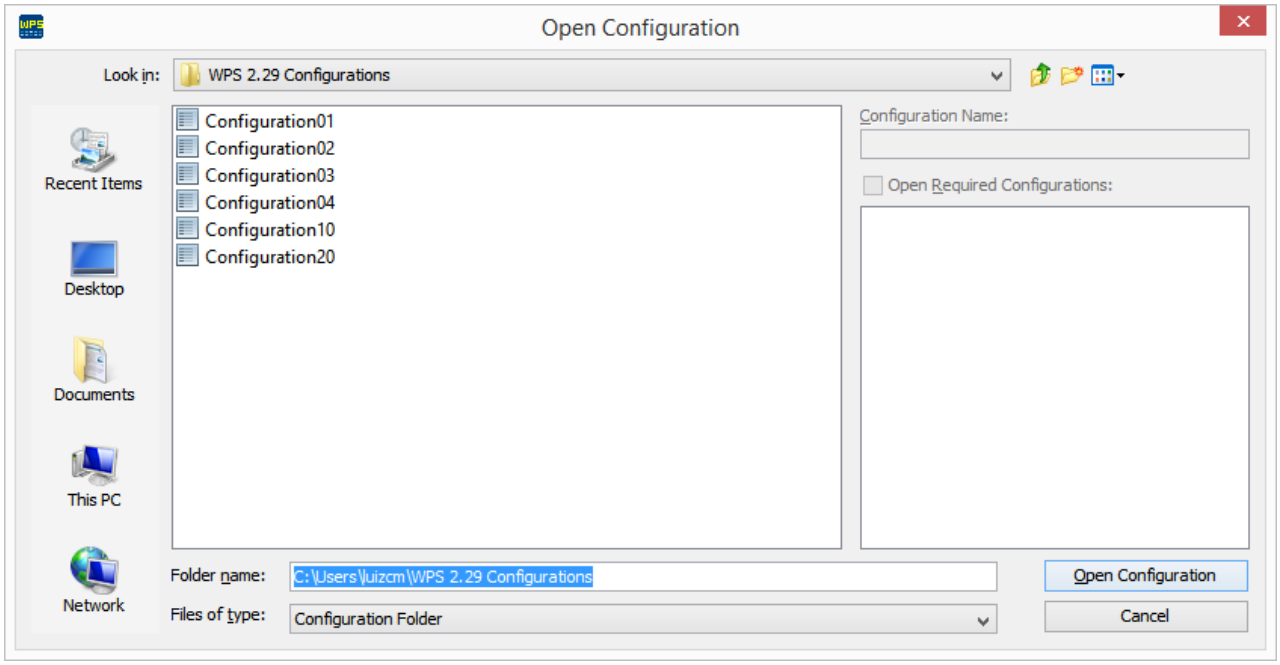
1) Function Selection

Click the Open Configuration option.



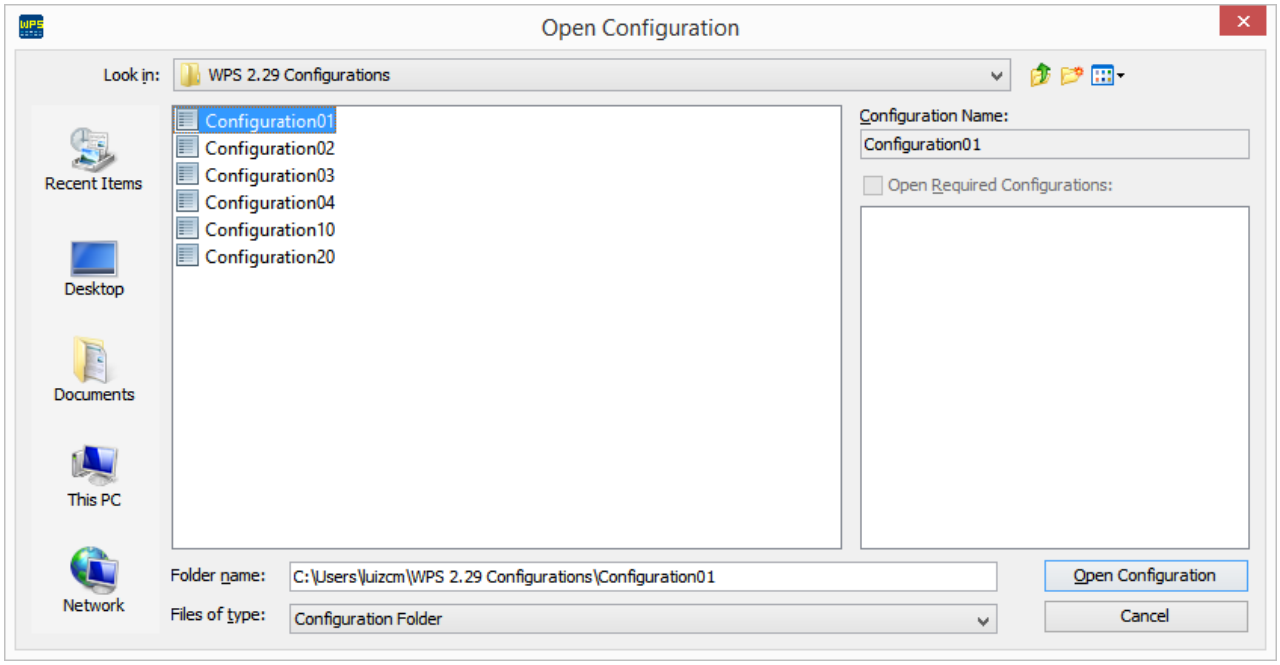
2) Open Configuration Window

All configurations are displayed in a window.



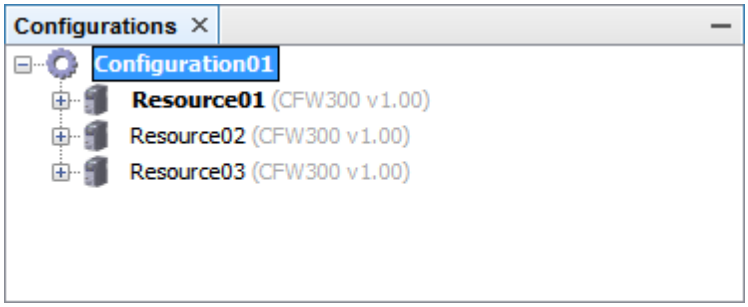
3) Configuration Selection

Select the desired configuration and click the Open Configuration button.



4) Configuration Window

At this moment, the configuration opens. In the configuration window, you can view the configuration and the resources that are part of the configuration.

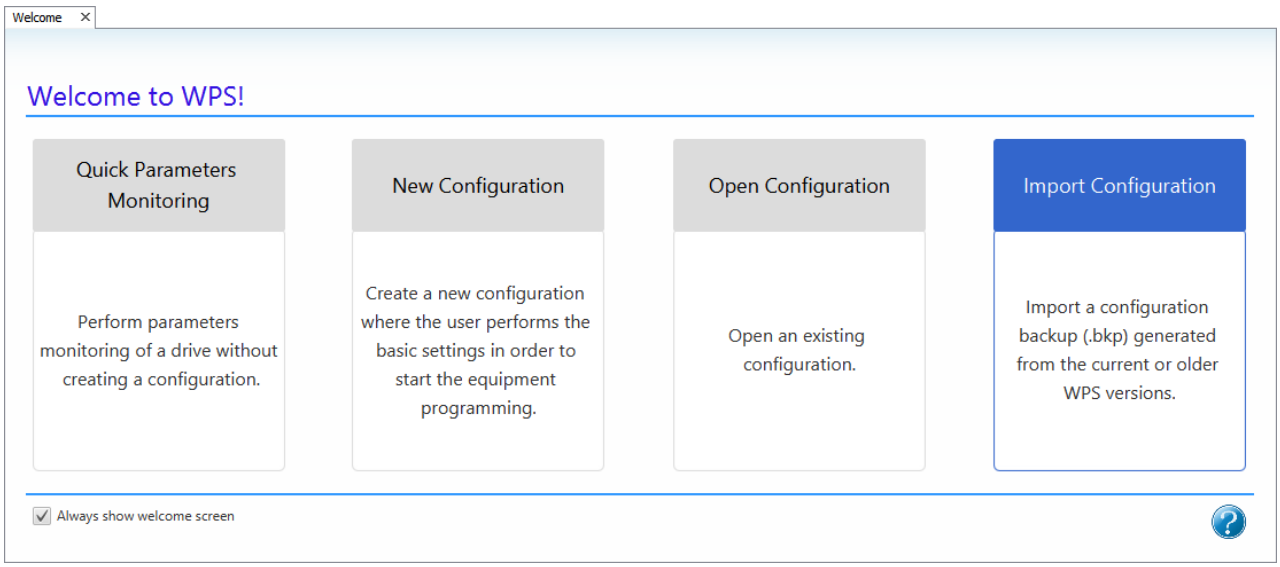


Import Configuration

It imports a configuration previously generated by the WPS.

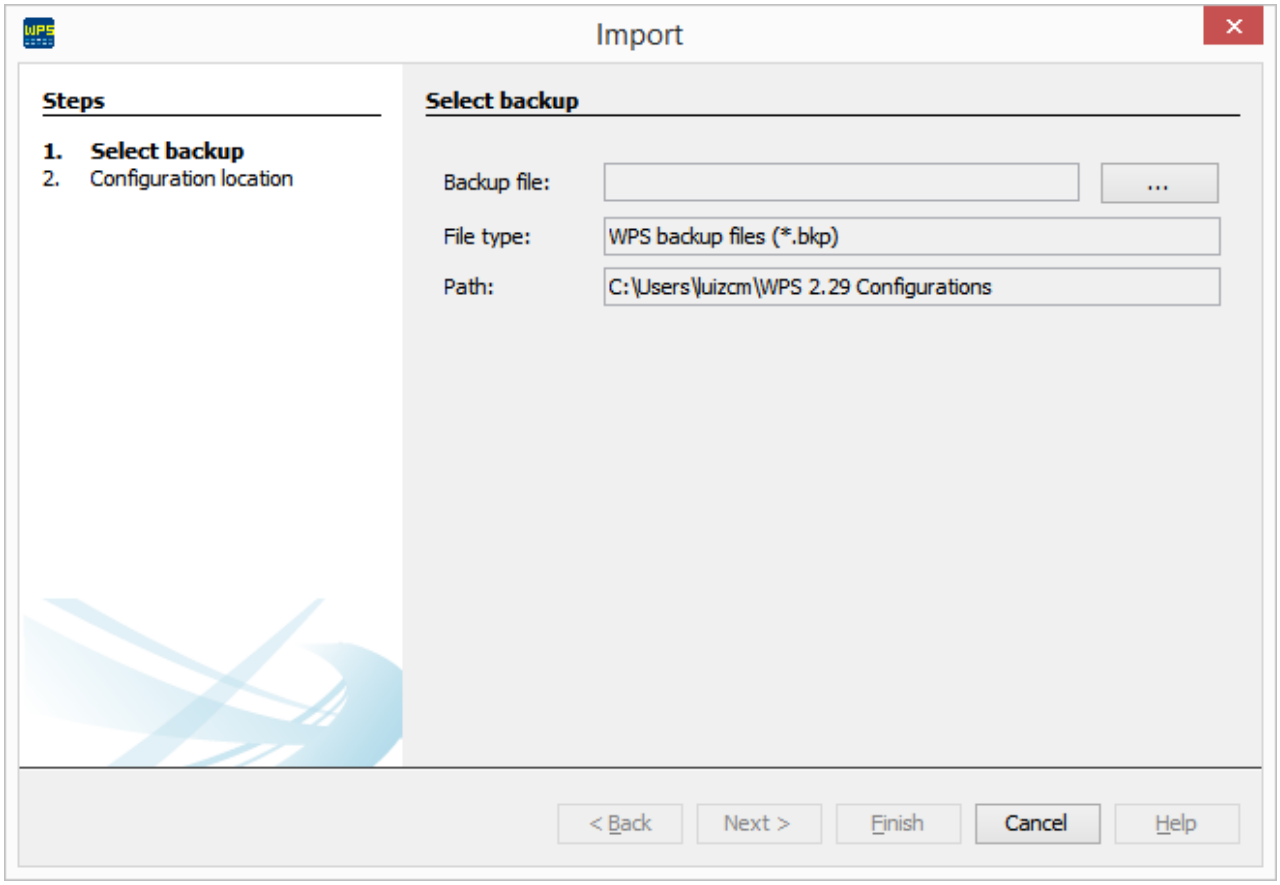
1) Function Selection

Click the Import Configuration option.



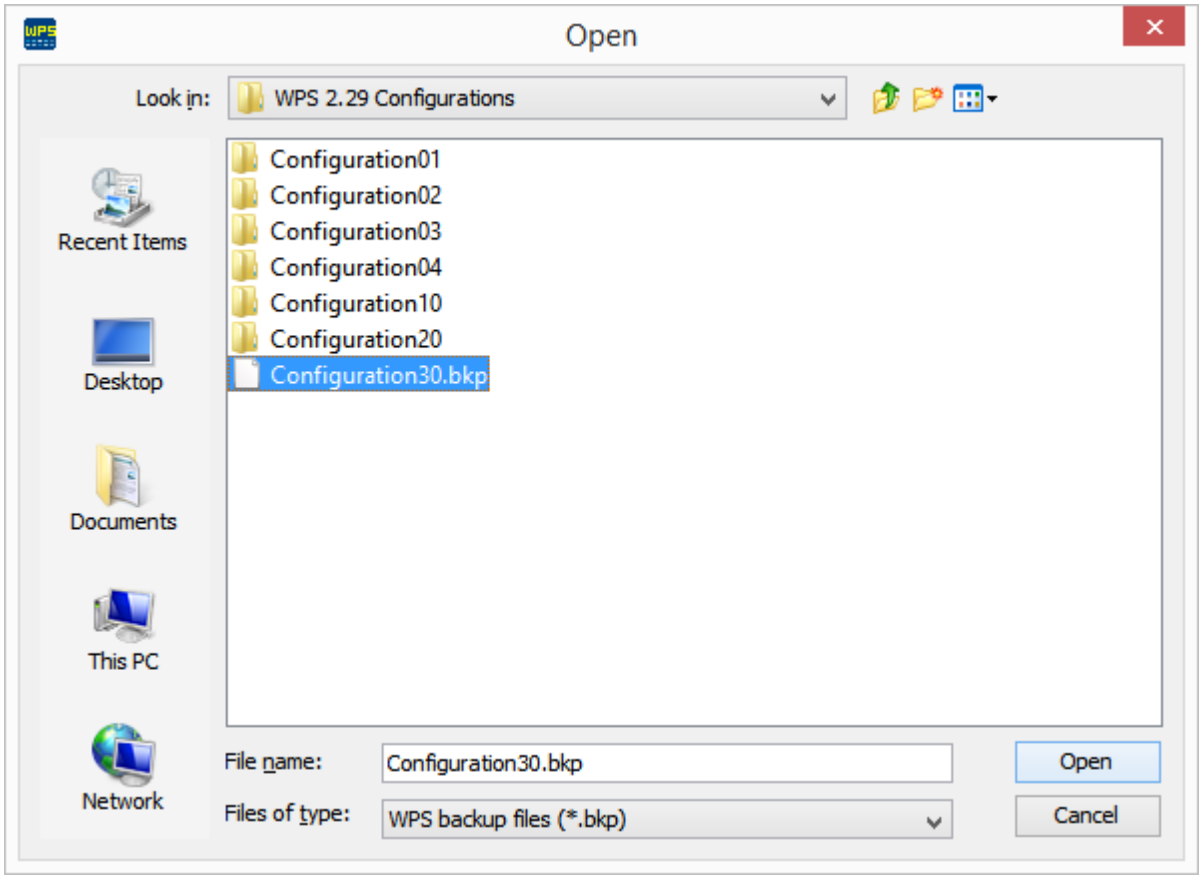
2) Import Window

Click the ... button in order to open the selection of the backup file containing the configuration that you wish to import.



3) Backup File Selection

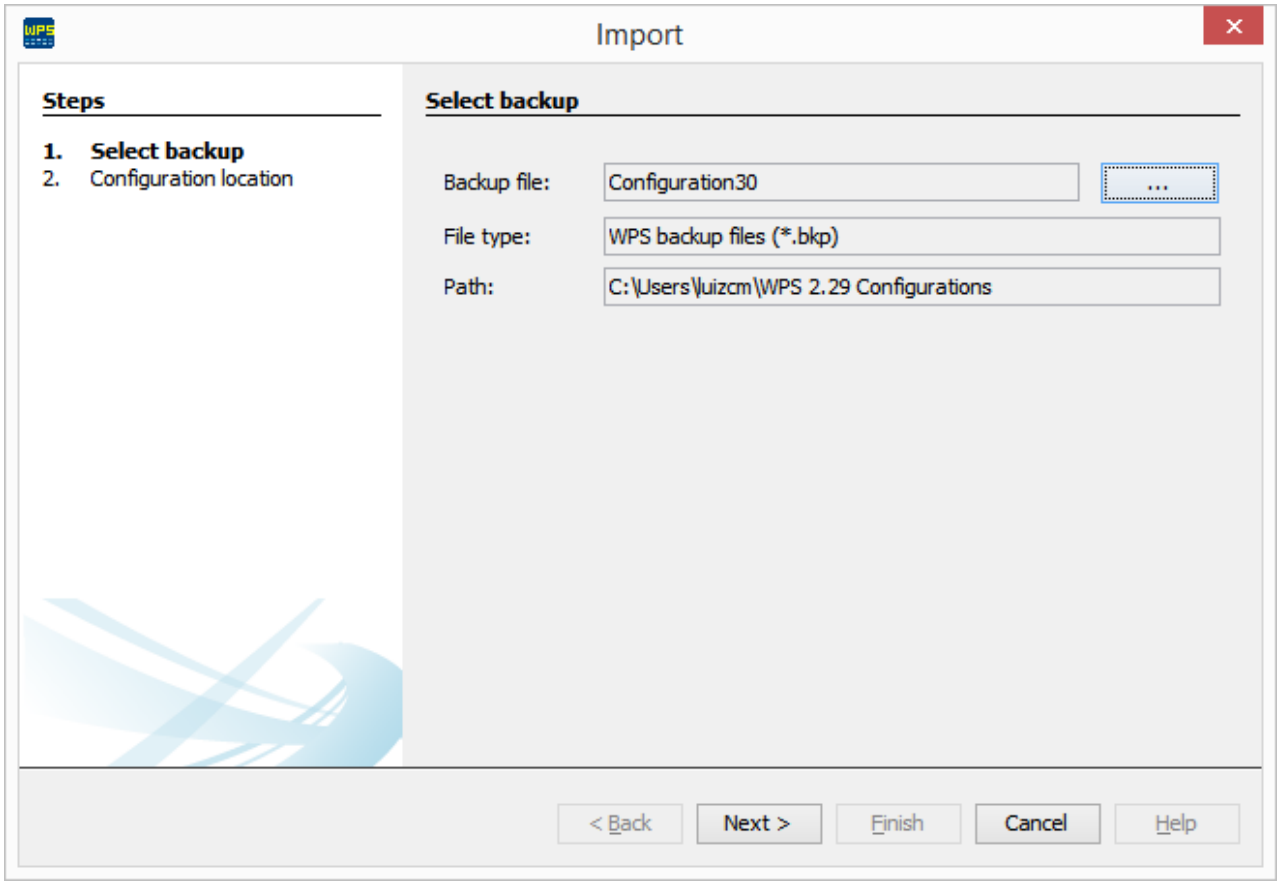
Select the file with the bkp extension that you wish to import and click Open.



4) BKP file to Be Imported

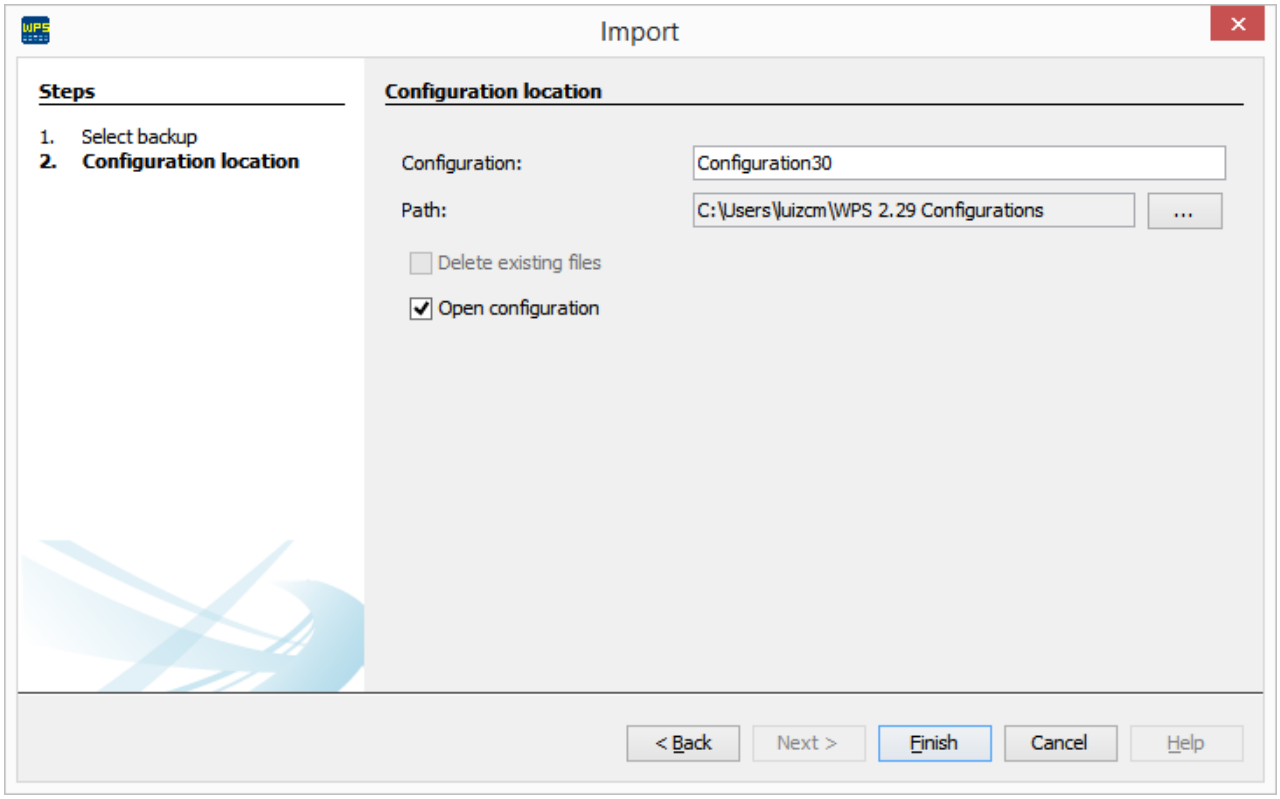
The Backup File field is filled out with the file selected in the previous step.

Click Next to continue.



5) Configuration Location

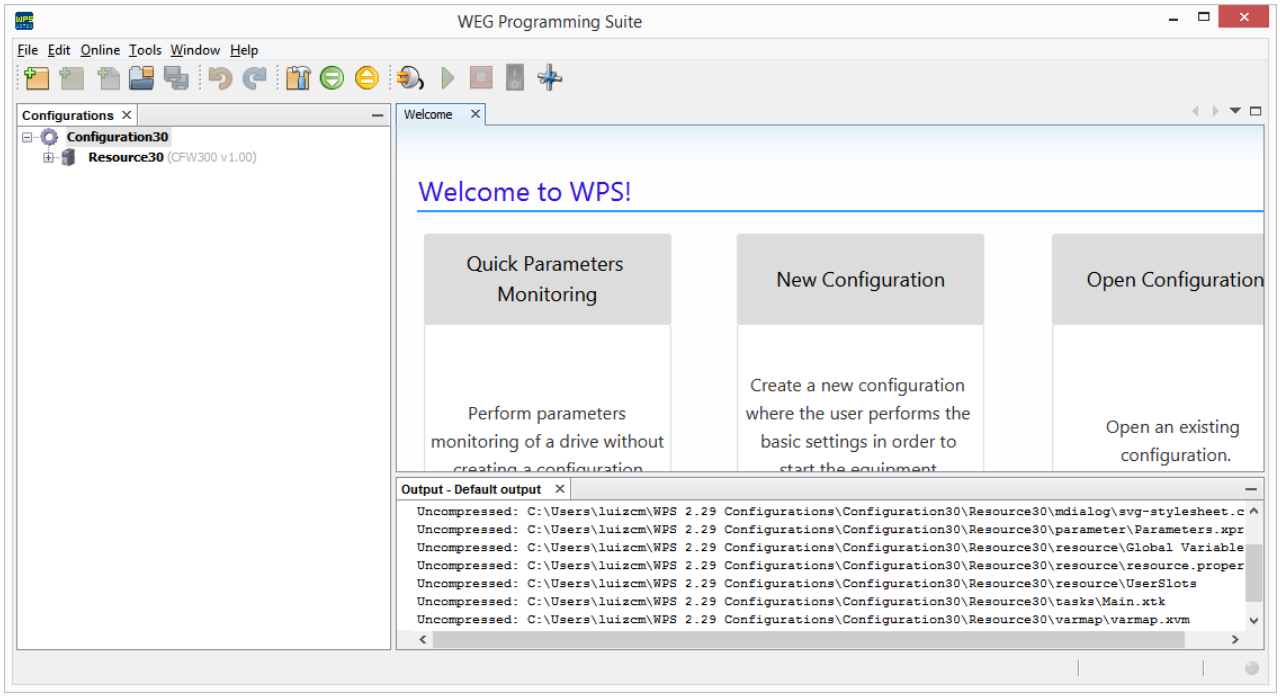
Click Finish to import and end the wizard.



6) Imported Configuration

The imported configuration can be viewed in the output window.

In the configuration window, you can view the imported configuration and its resources.



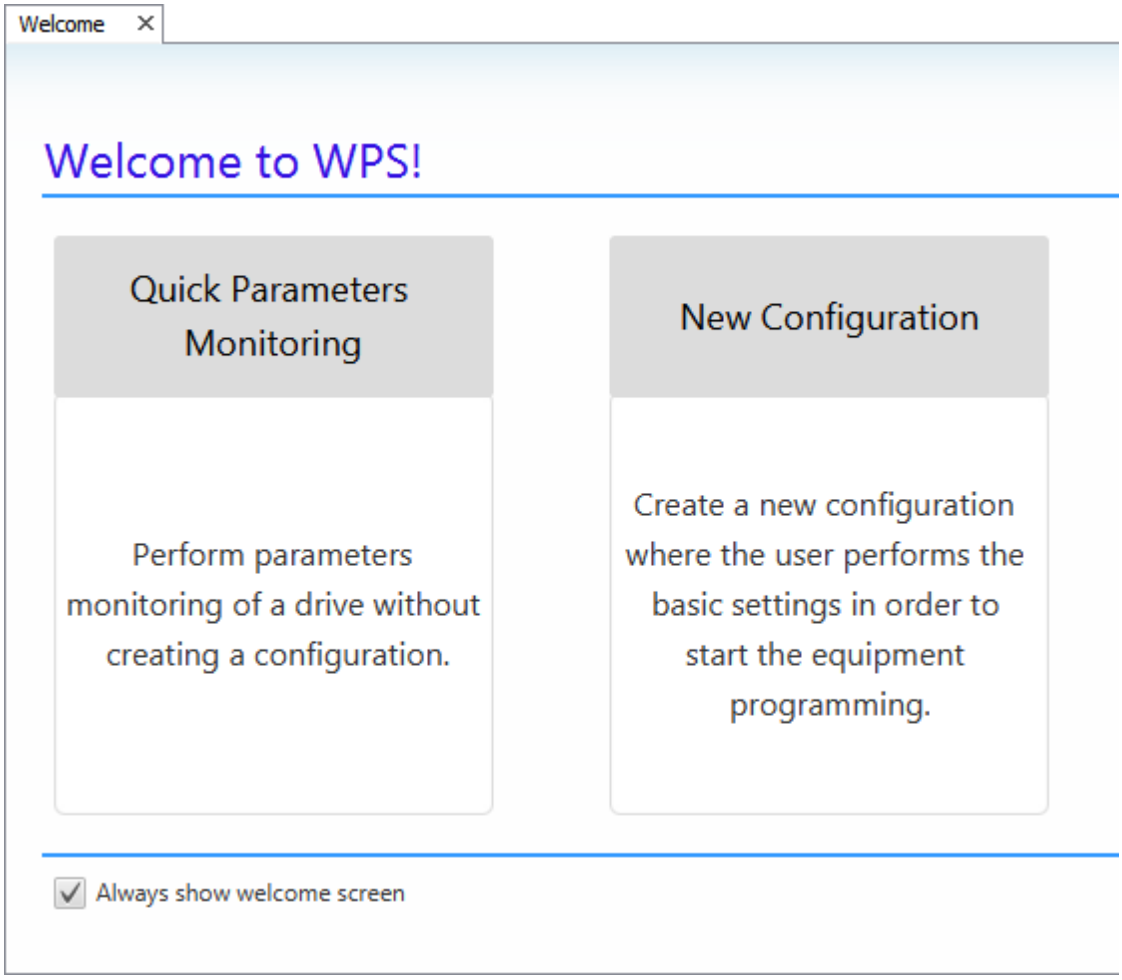
Always Show Welcome Window Option

The **Always show welcome window** option allows the window to be displayed in the initialization of the WPS.

This option is in the lower part of the Welcome Window.

Uncheck this option if you do not wish to see the welcome window in the next initialization.

You can enable this option later in **Help > Welcome**.



6.2 Creating New Configuration

There are two situations in which a new configuration may be created:

- Equipment is online (connected to the WPS)
- Equipment is offline.

Examples

Equipment is online:

- New Configuration - Online Equipment

Equipment Offline:

- New Configuration - Offline Equipment

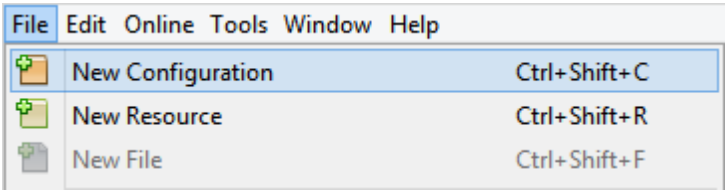
6.3 New Configuration - Online Equipment

In the following example, we will create a configuration with the CFW300. The windows may be different if other equipments are configured.

CFW300

1) New Configuration Menu Item/Button

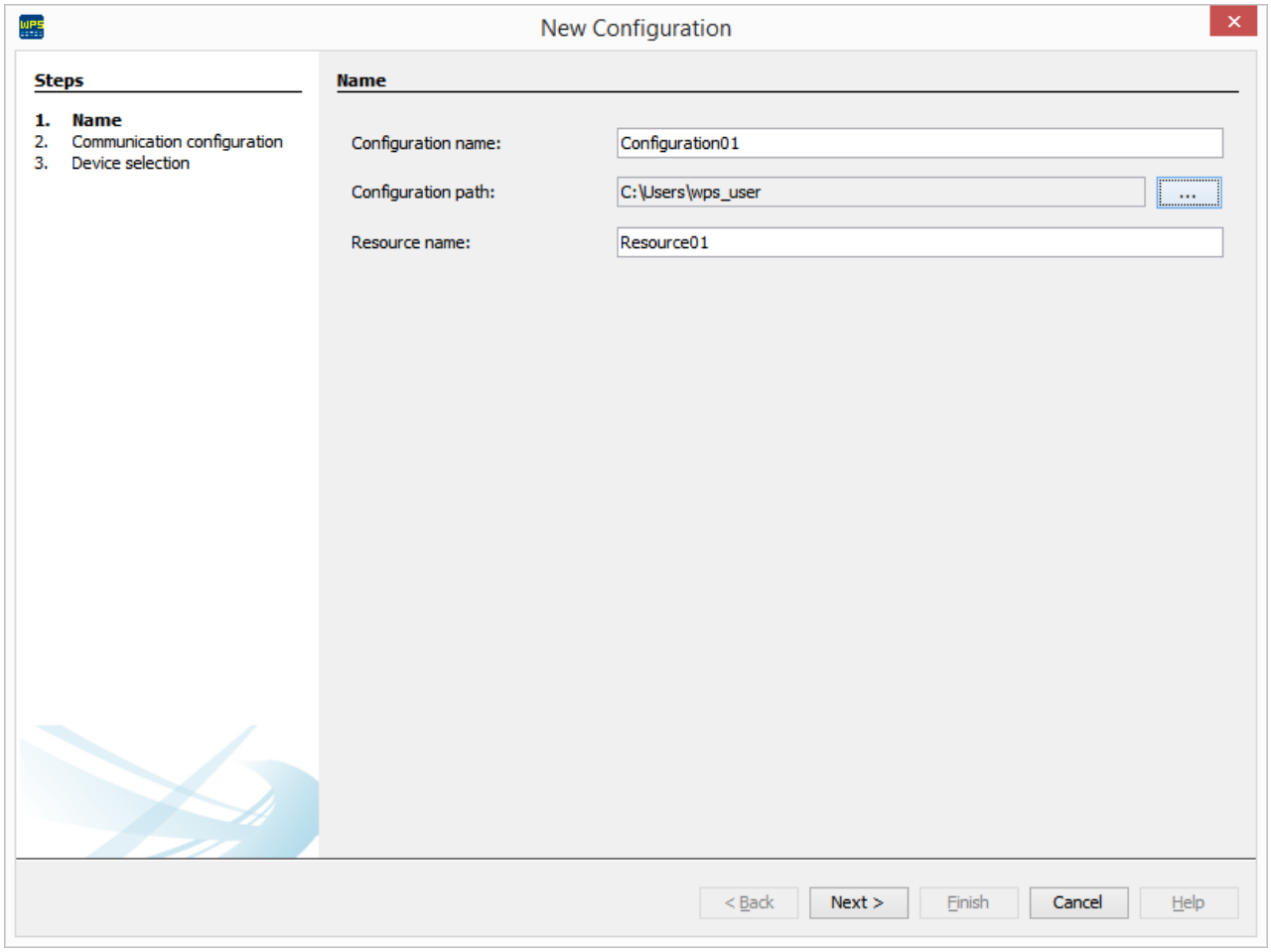
In the File menu, click New Configuration.



You can also use the keyboard shortcut (Ctrl+Shift+C) or the New Configuration button on the Toolbar: 

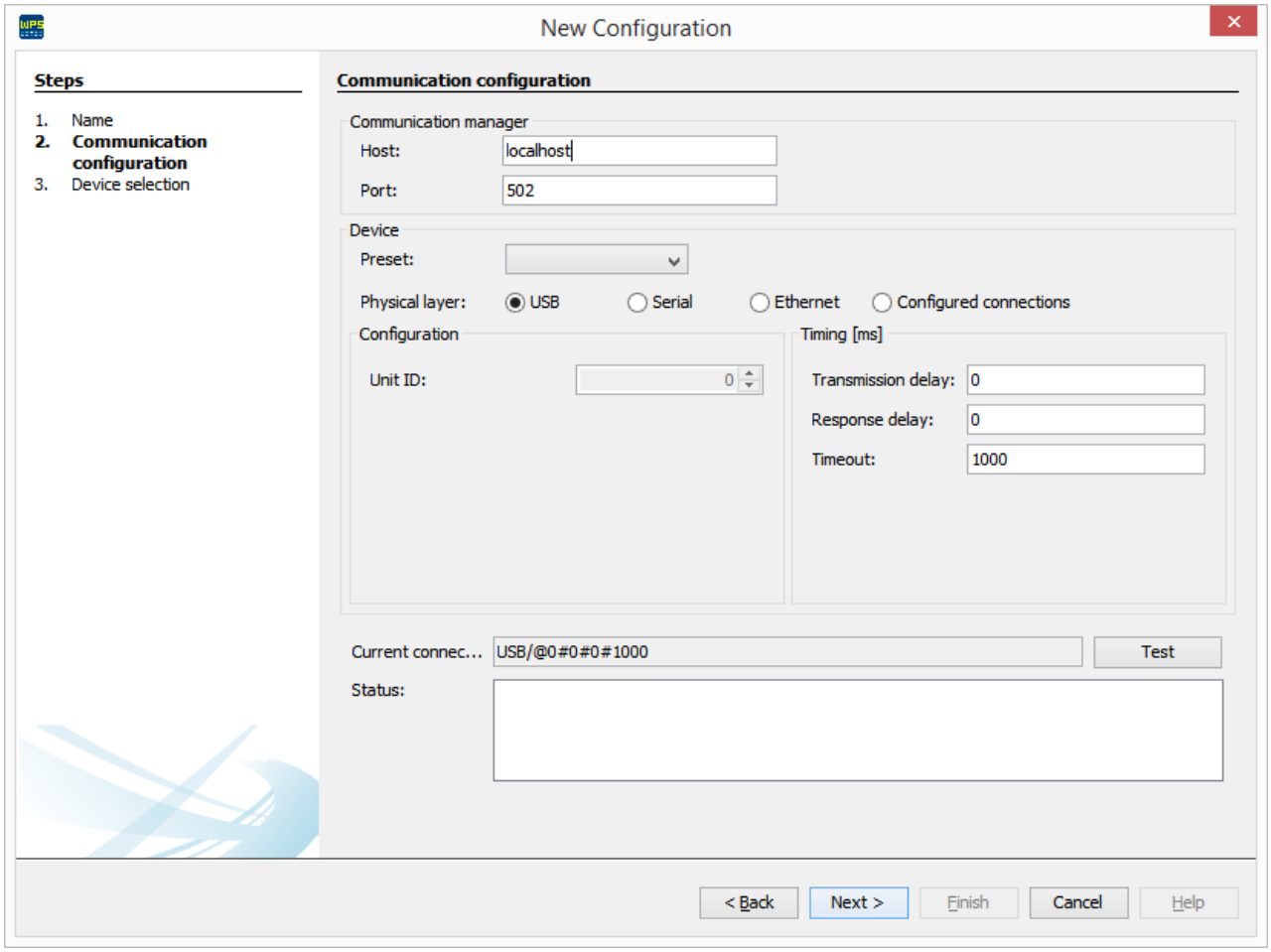
2) New Configuration Window

A window will pop up prompting you to enter the configuration name and the first resource to be created. Enter the names in the respective fields and click Next.



3) Communication Configuration

The next window defines the options for communication with the equipment.

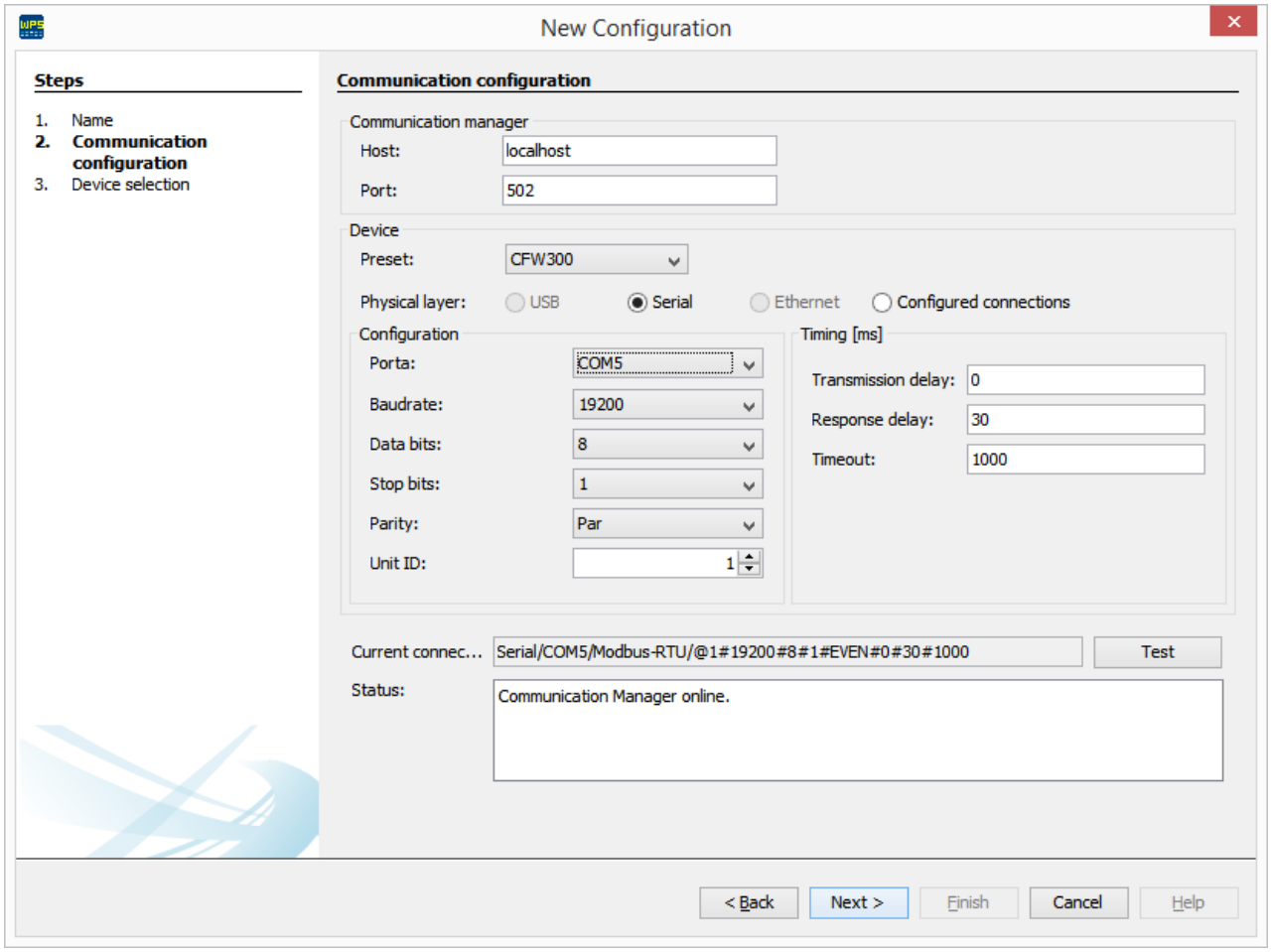


4) Select the Communication Options

Choose the correct communication options.

If you select one equipment, the default configuration of the communication of the equipment will be loaded to the window.

Still, check that the window configuration is the same as of the equipment. If not, update this window according to the equipment configuration.



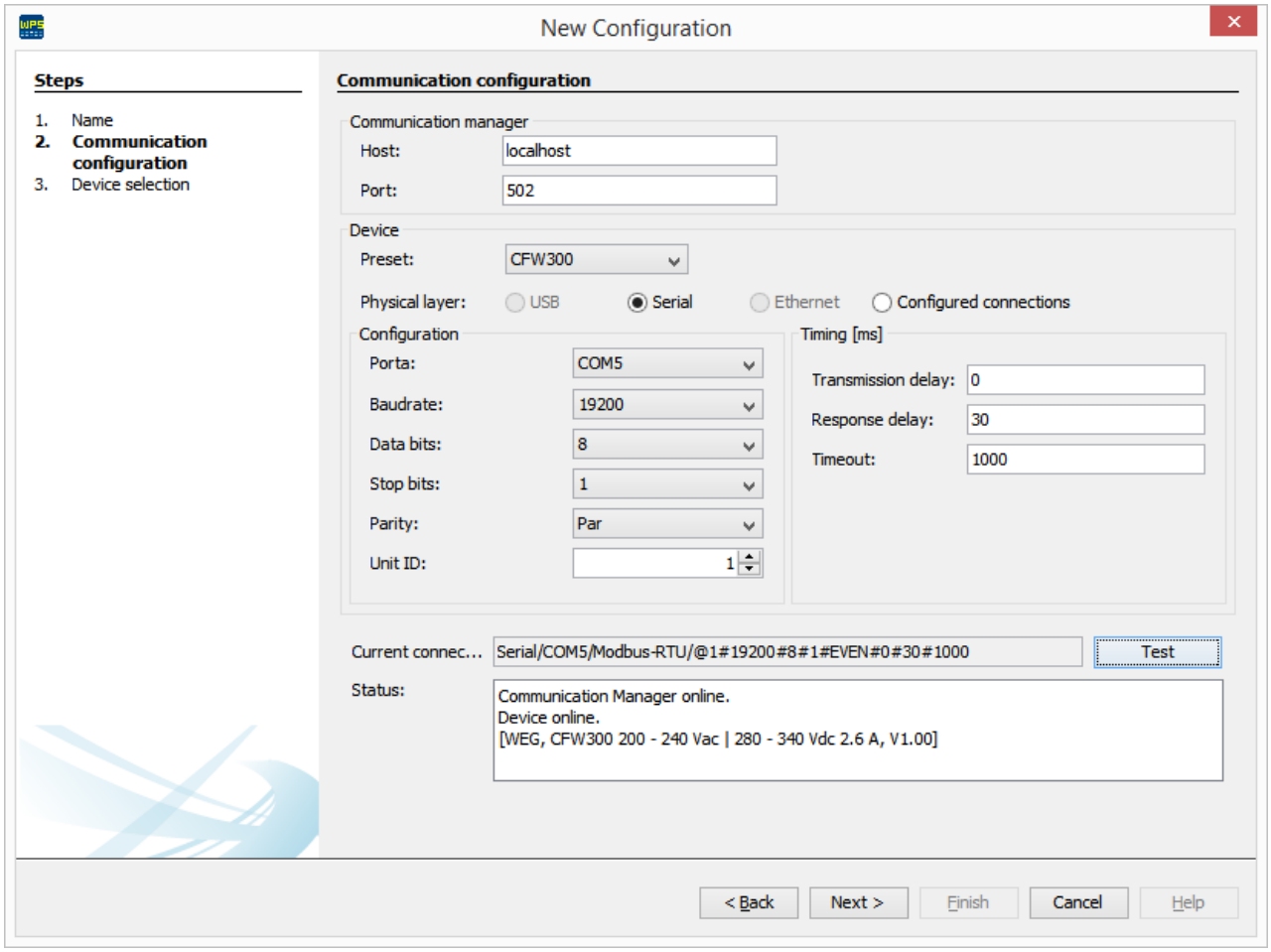
NOTE!

- In this example, the CFW300-CUSB accessory is being used, which appears in Windows as a USB serial port (virtual COM port);
- When the cable is connected to the personal computer/equipment, the Windows device manager informs the name of the USB serial port which was created.

5) Testing the Communication

You can check if the communication is correct by clicking the Test button in this window.

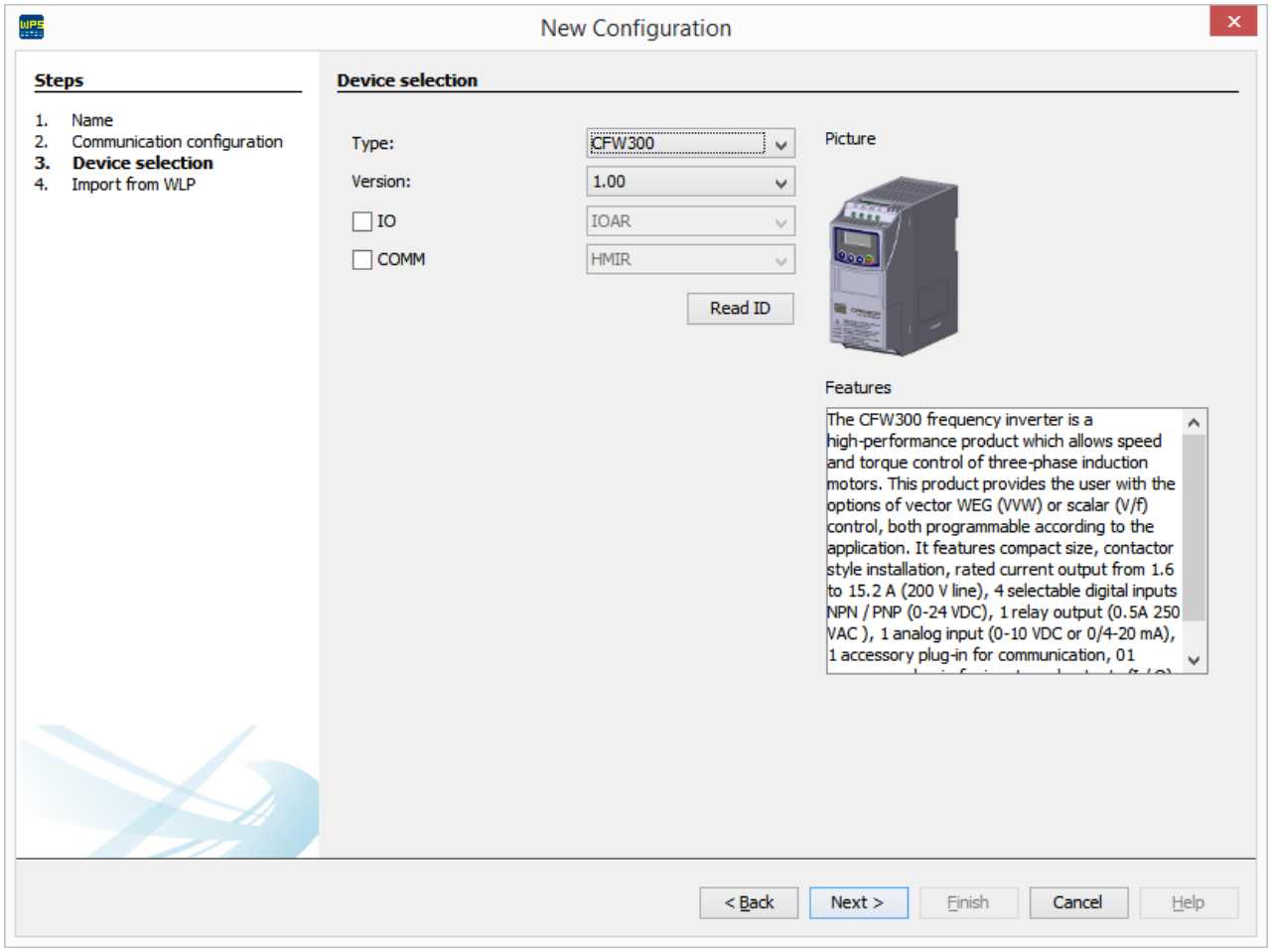
The status field is refreshed. The connected equipment appears in the status field.



Then click Next.

6) Equipment Selection

The window shows the equipment that is connected to the resource that was created.



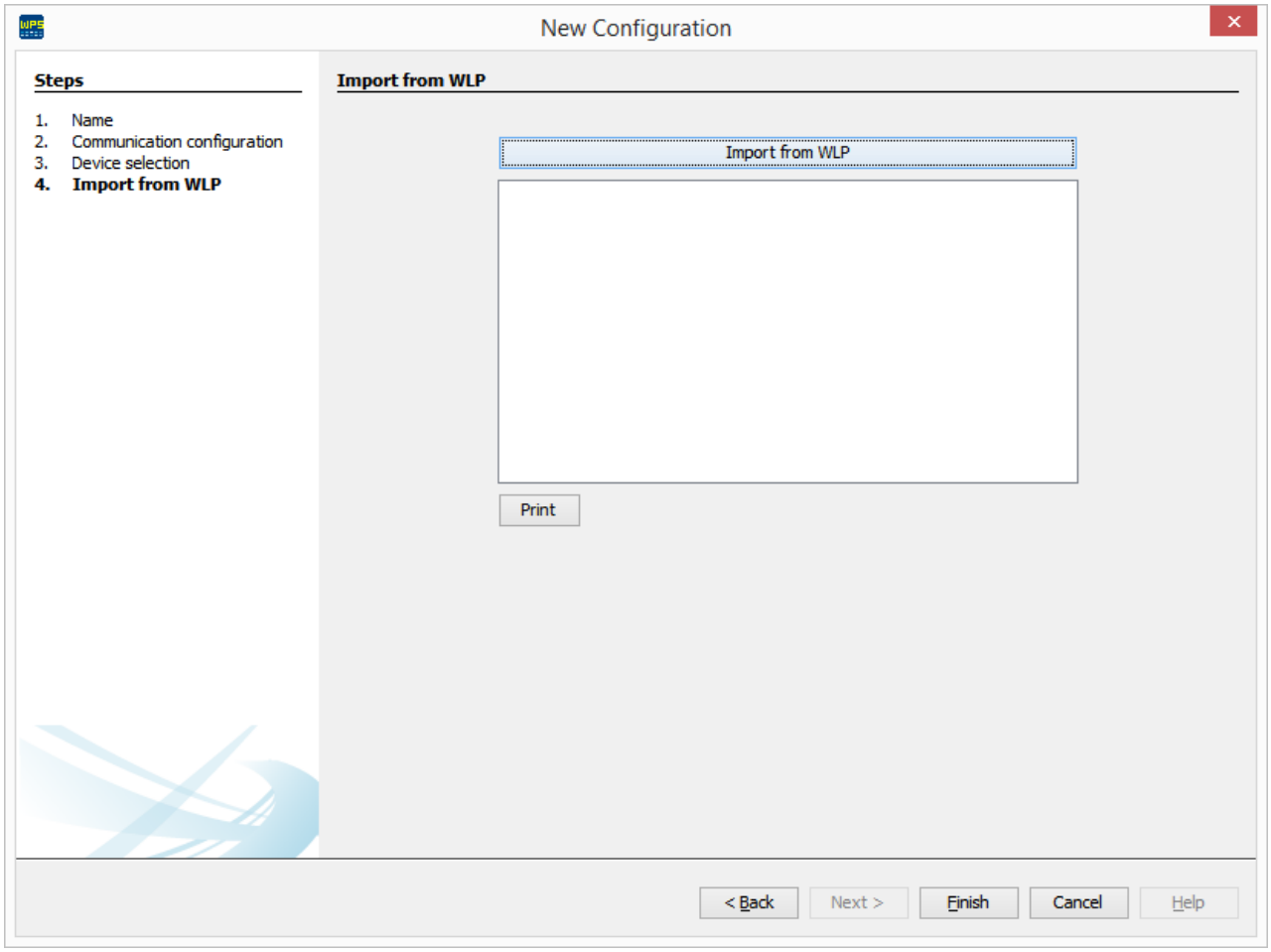
7) Read ID

If you click the Read ID button, a window containing the information on the equipment will open. Click OK to close it.

In order to go to the next step, click Next.

8) Import from WLP

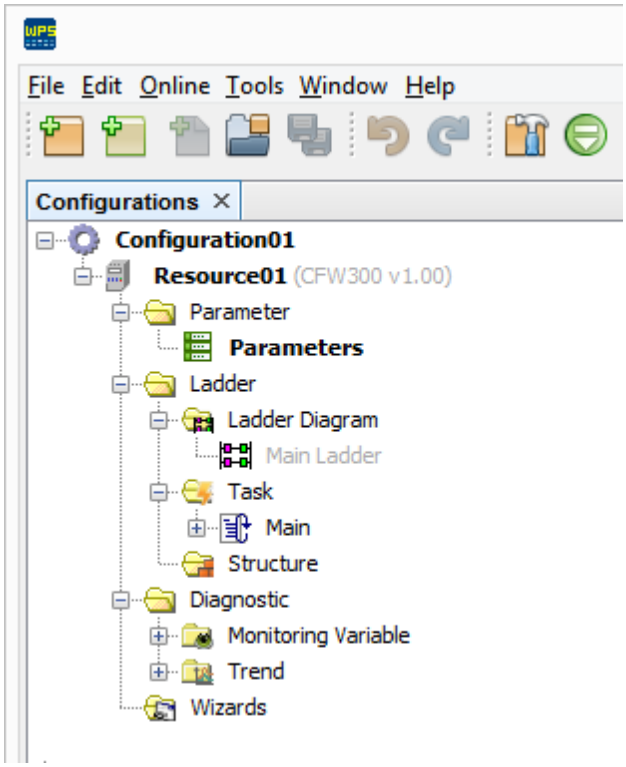
A new window pops up enabling to import from the CFW100 a Ladder project developed on the WLP (WEG Ladder Programmer).



Click Finish to close the wizard.

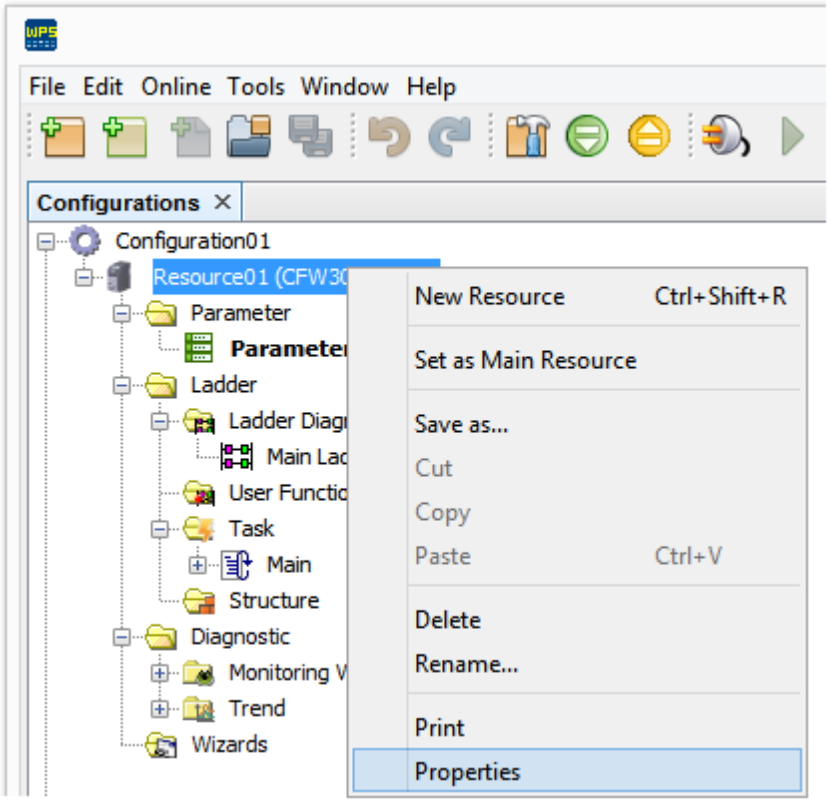
9) Configuration Tree

After those steps, the tree with the configurations should look like the following image.



10) Resource Properties

The settings made in the setup can be changed later in resource properties.



NOTE!

- Depending on the equipment, new windows may appear from this step on, with initial settings and oriented start-ups.

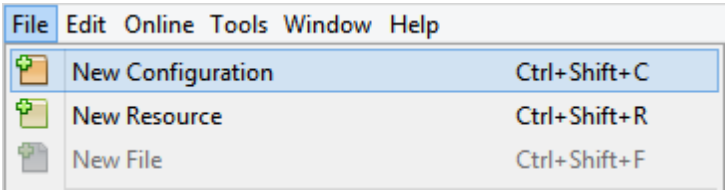
6.4 New Configuration - Offline Equipment

In the following example, we will create a configuration with the CFW300. The windows may be different if other equipments are configured.

CFW300

1) New Configuration Menu Item/Button

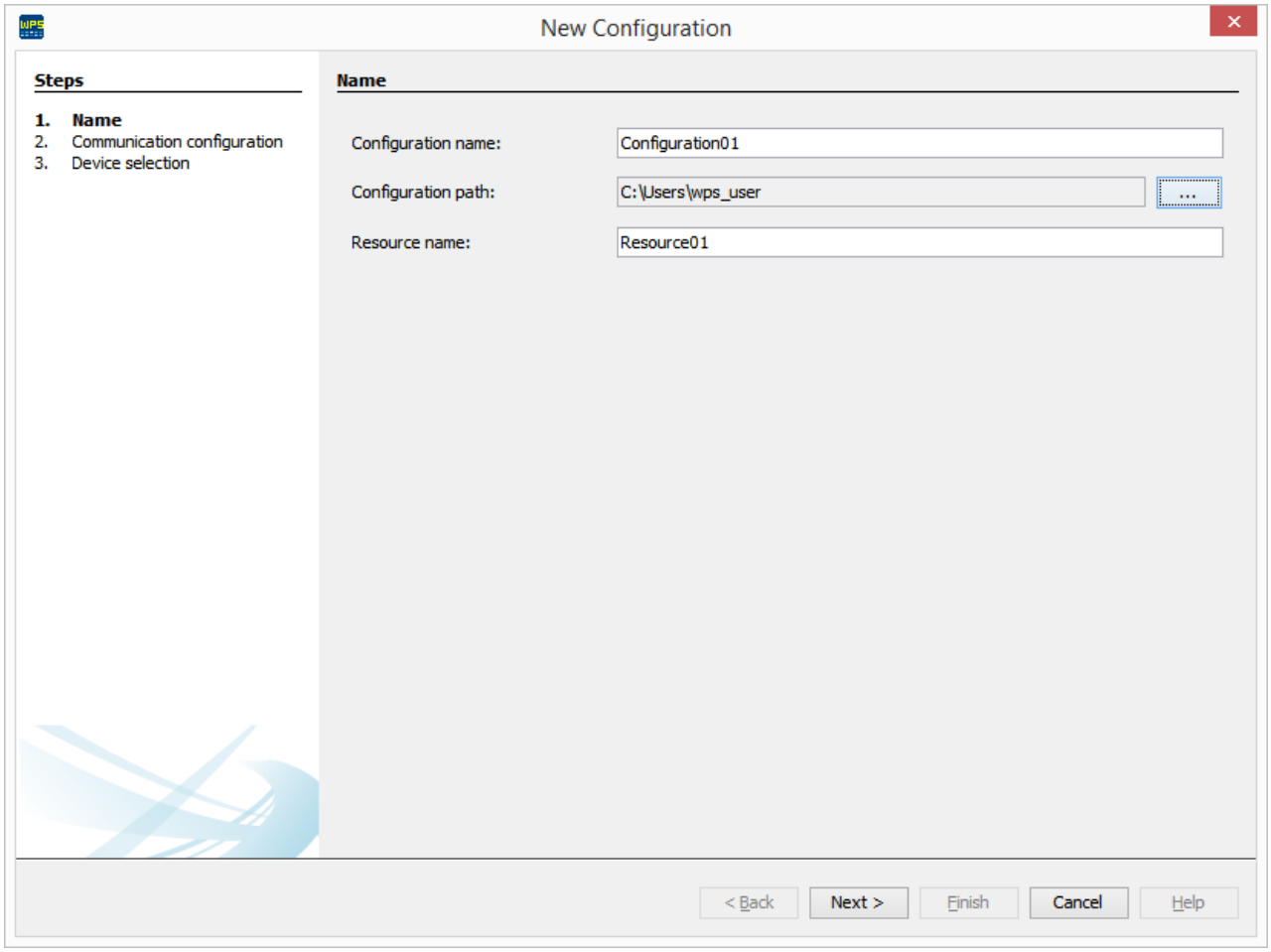
In the File menu, click New Configuration.



You can also use the keyboard shortcut (Ctrl+Shift+C) or the New Configuration button on the Toolbar: 

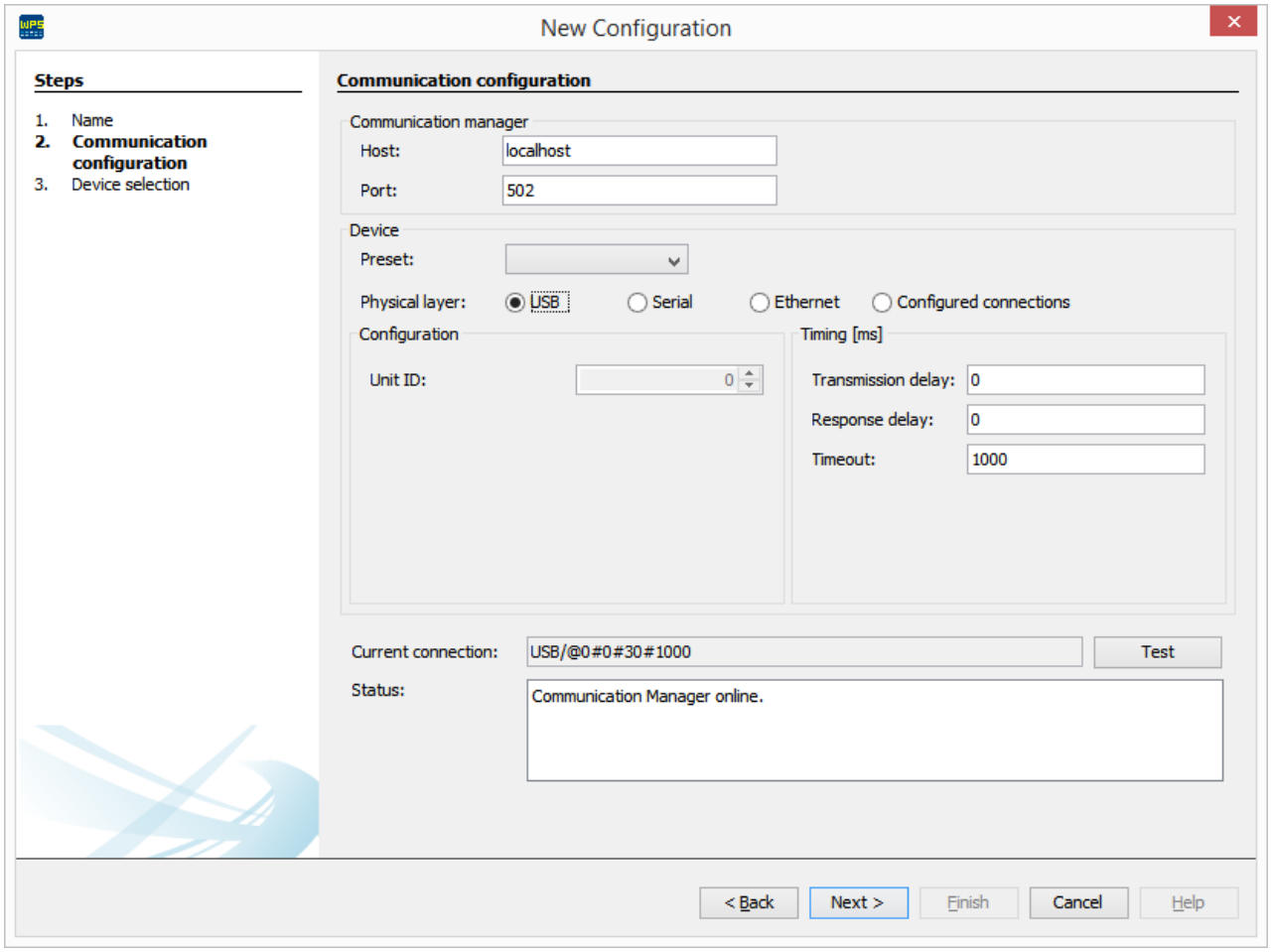
2) New Configuration Window

A window will pop up prompting you to enter the configuration name and the first resource to be created. Enter the names in the respective fields and click Next.



3) Communication Configuration

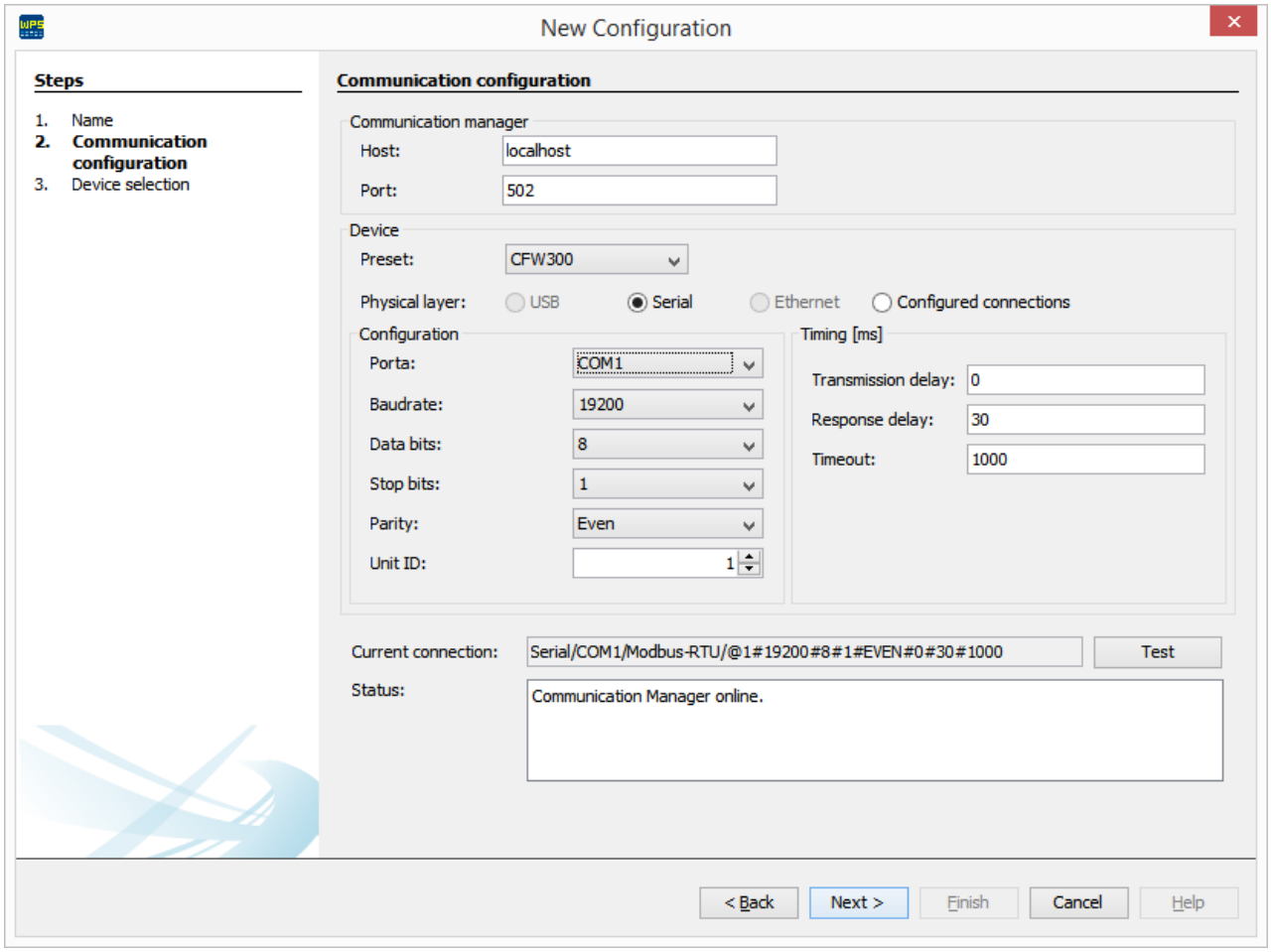
The next window defines the options for communication with the equipment.



4) Select the Communication Options

Choose the correct communication options.

If you select one equipment, the default communication configuration of the equipment will be loaded.



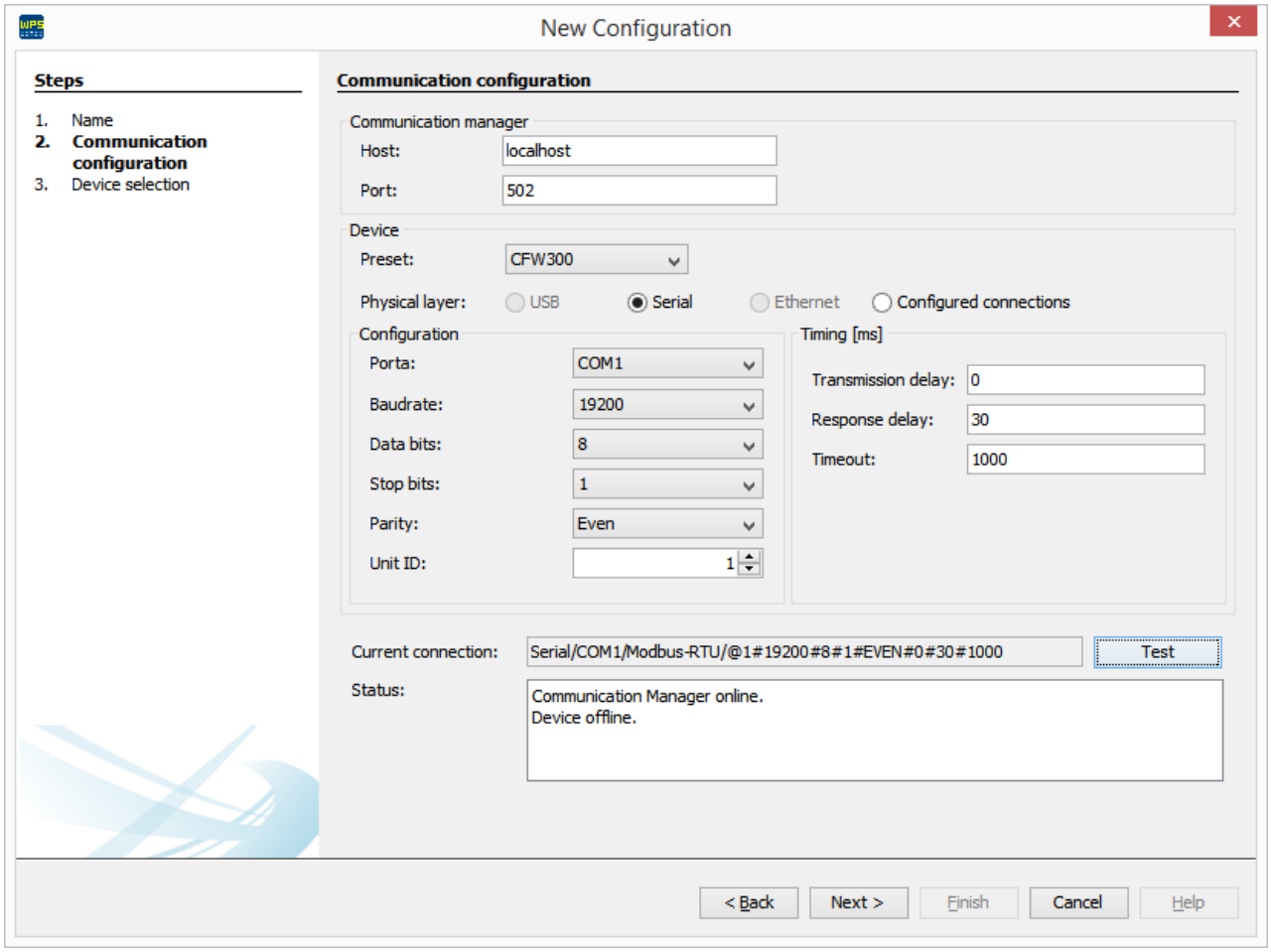
Depending on the physical layer, it will be necessary to review this configuration when the equipment is connected to the WPS.

For example, if the CFW300-CUSB accessory is used on the CFW300, Windows creates a USB serial port which is only known at the moment of the connection of the equipment and personal computer.

5) Testing the Communication

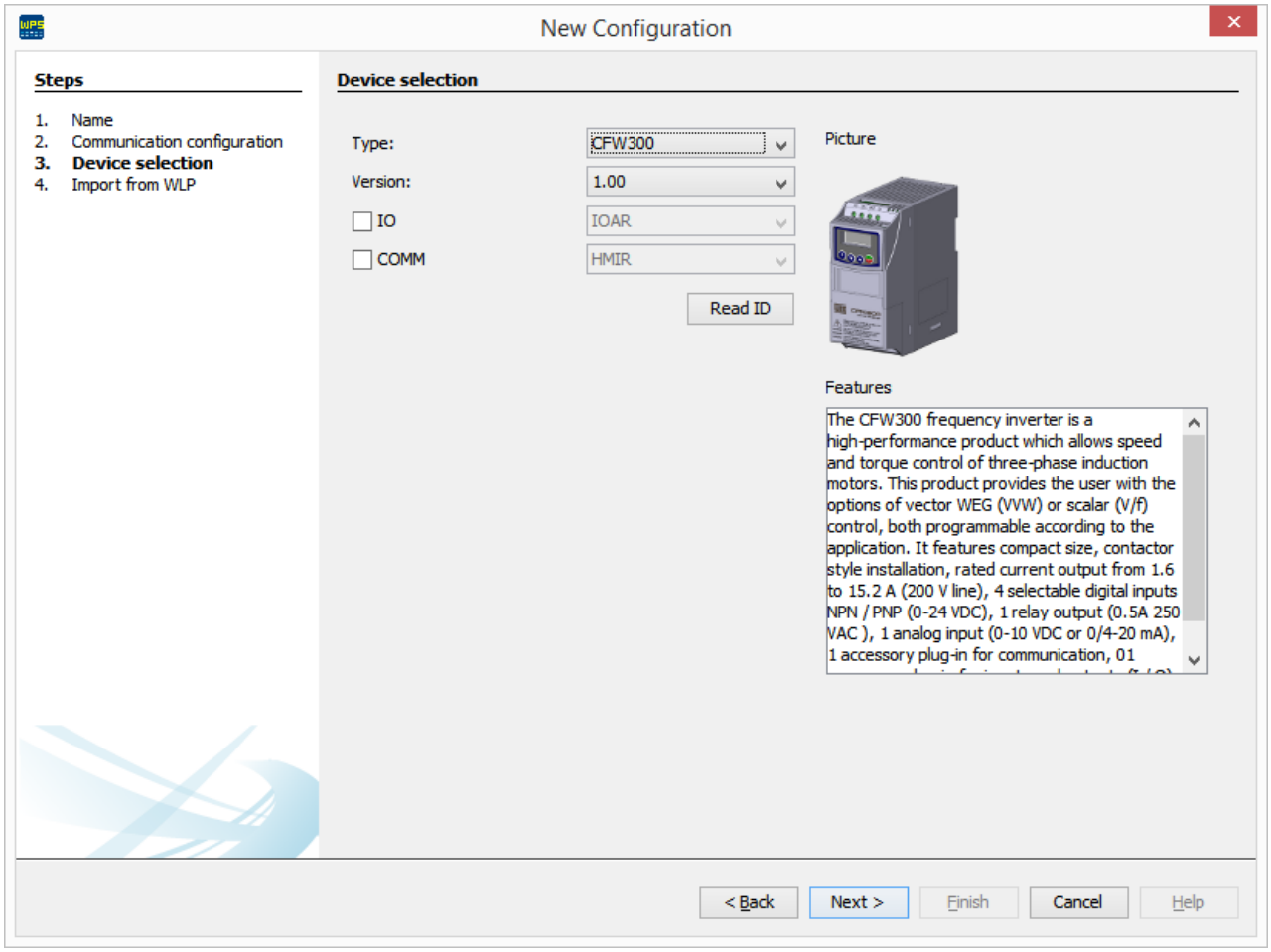
At this moment, it is not possible to test the communication (equipment is not connected).

Thus, when you click Test, the status indicates Equipment offline.



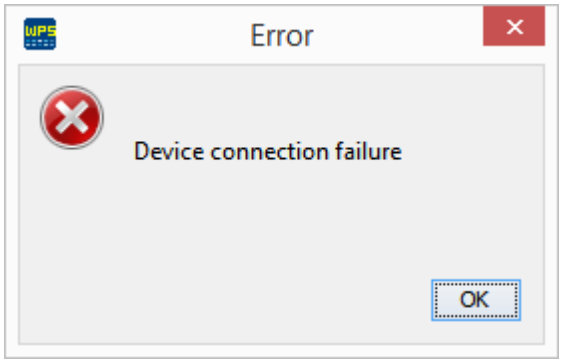
6) Equipment Selection

The window shows the equipment that is connected to the resource that was created.



7) Read ID

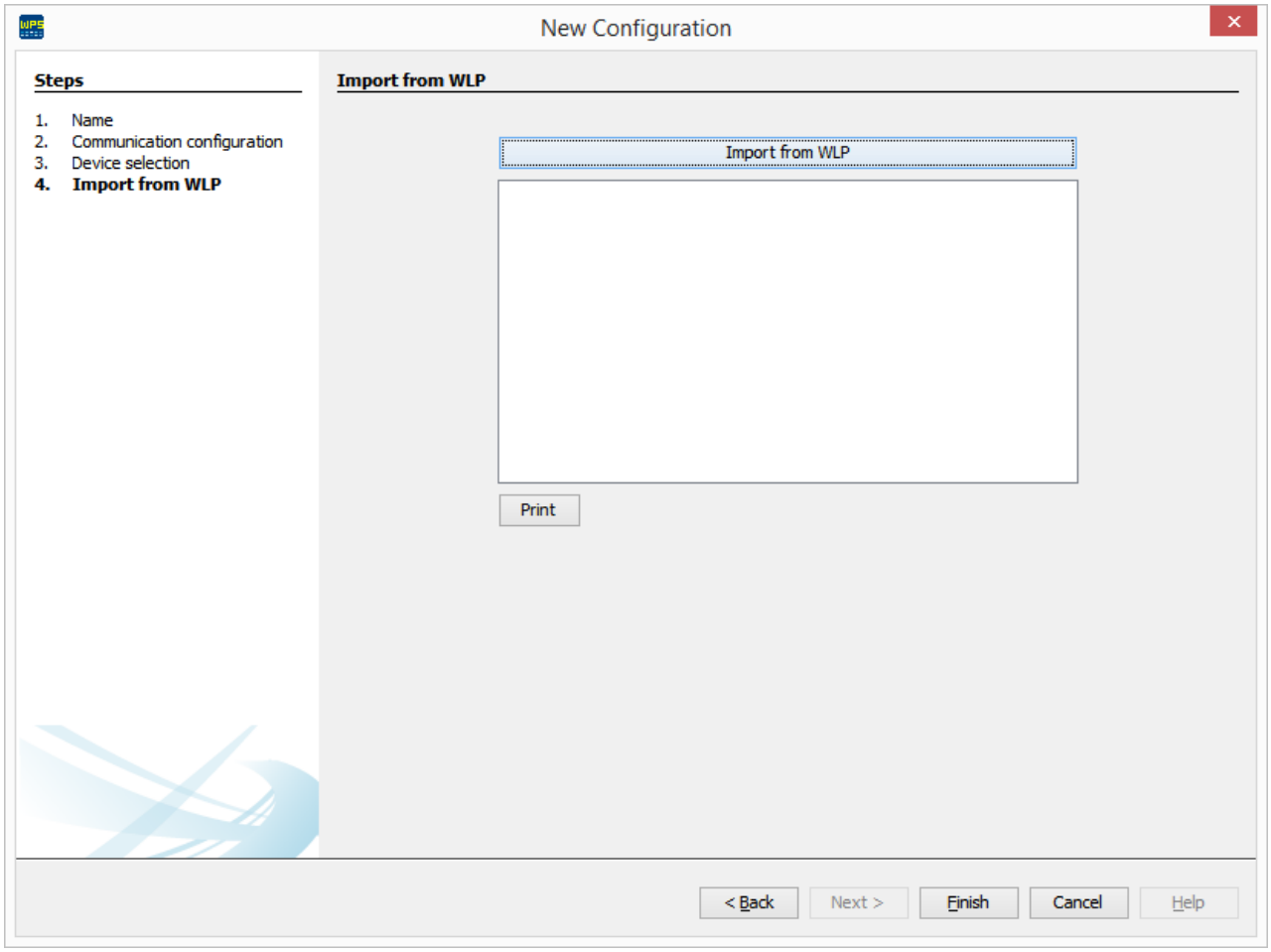
If you click the Read ID button, as there is no equipment connected, a fault window pops up (Device connection Failure). Click OK to close it.



In order to go to the next step, click Next.

8) Import from WLP

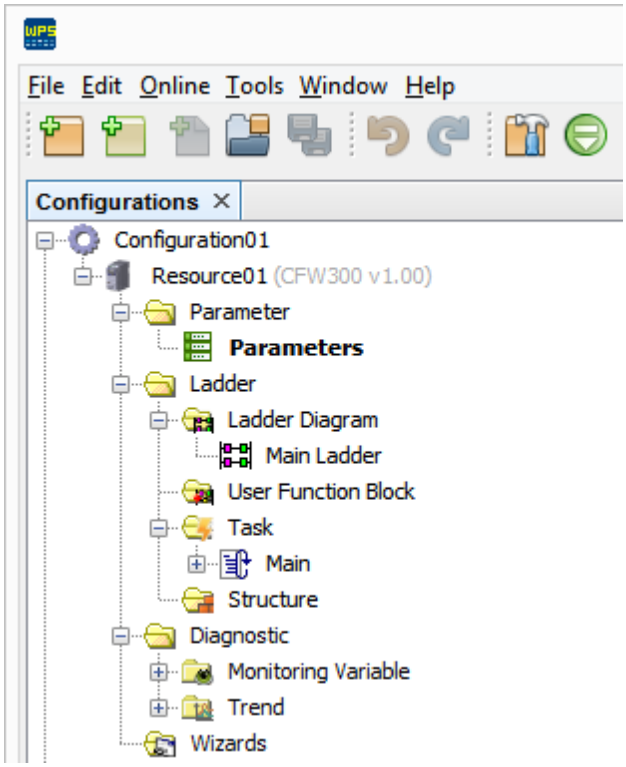
A new window pops up enabling to import from the CFW100 a Ladder project developed on the WLP (WEG Ladder Programmer).



Click Finish to close the wizard.

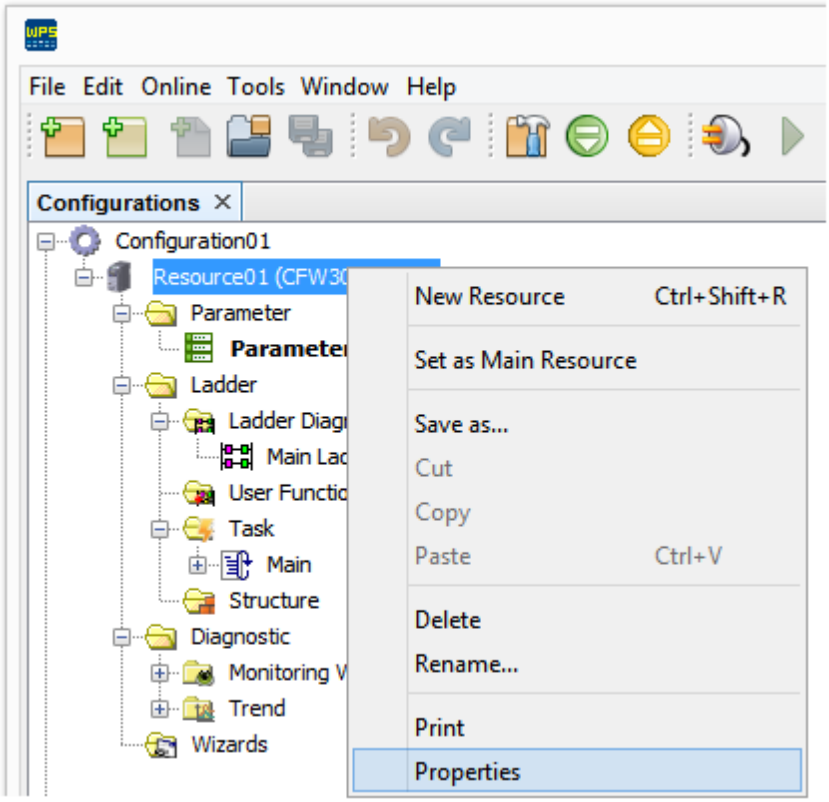
9) Configuration Tree

After those steps, the tree with the configurations should look like the following image.



10) Resource Properties

The settings made in the setup can be changed later in resource properties.



6.5 Creating New Resource

A new resource may be created within a configuration.

There are 2 situations:

- Equipment is online (connected to the WPS)
- Equipment is offline.

Example

Equipment is online:

- New Resource - Online Equipment

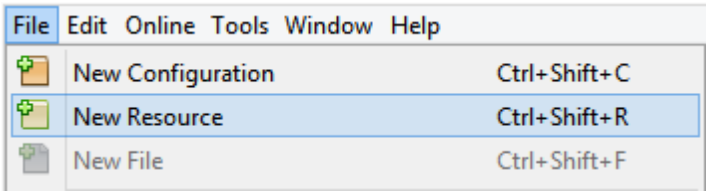
6.6 New Resource - Online Equipment

In the following example, we will create a new resource with the CFW300. The windows may be different if other equipments are configured.

CFW300

1) New Resource Menu Item/Button

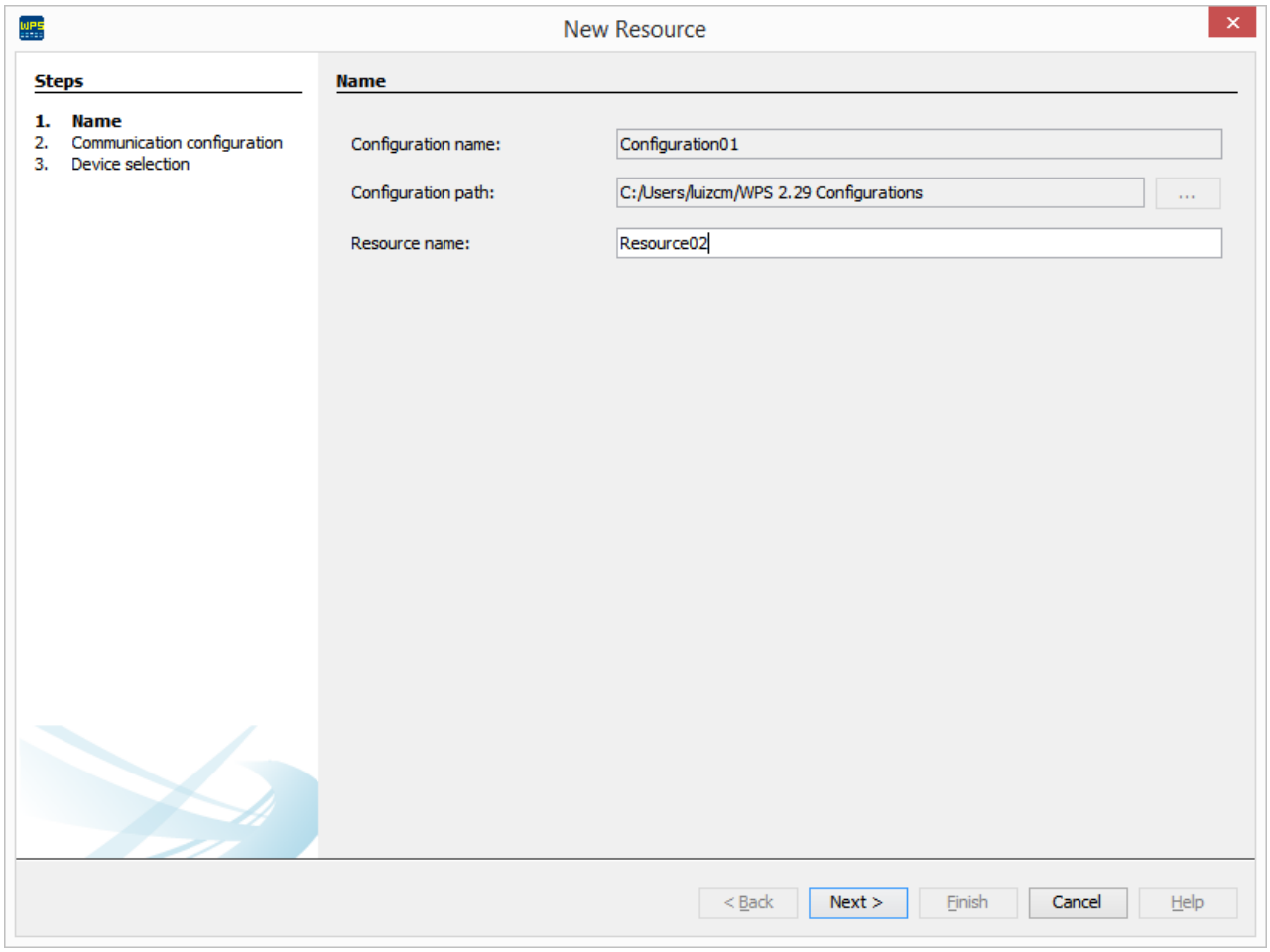
In the File menu, click New Resource.



You can also use the keyboard shortcut (Ctrl+Shift+R) or the New Resource button on the Toolbar: 

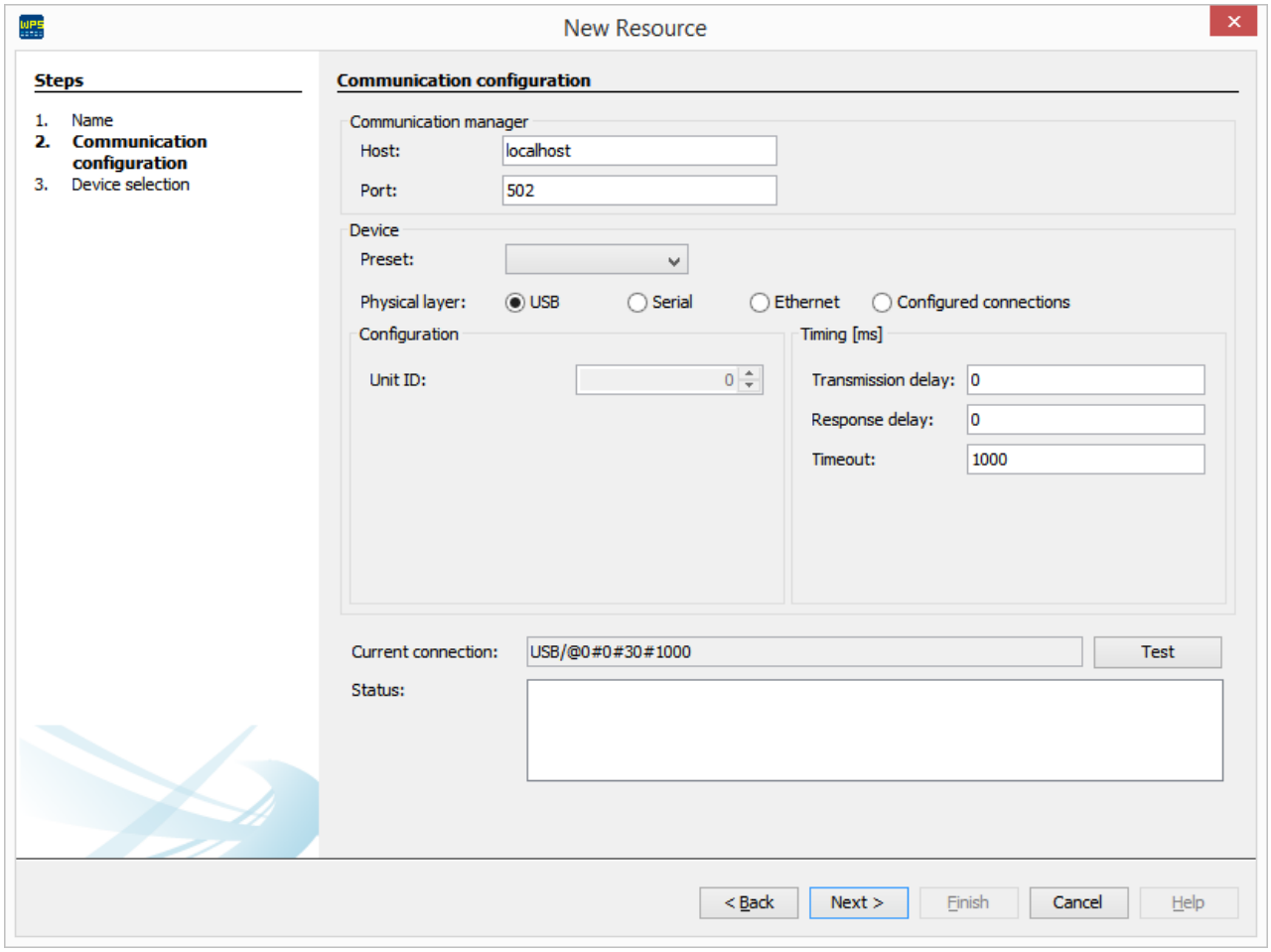
2) New Configuration Window

A window will pop up prompting you to enter the configuration name and the first resource to be created. Enter the names in the respective fields and click Next.



3) Communication Configuration

The next window defines the options for communication with the equipment.

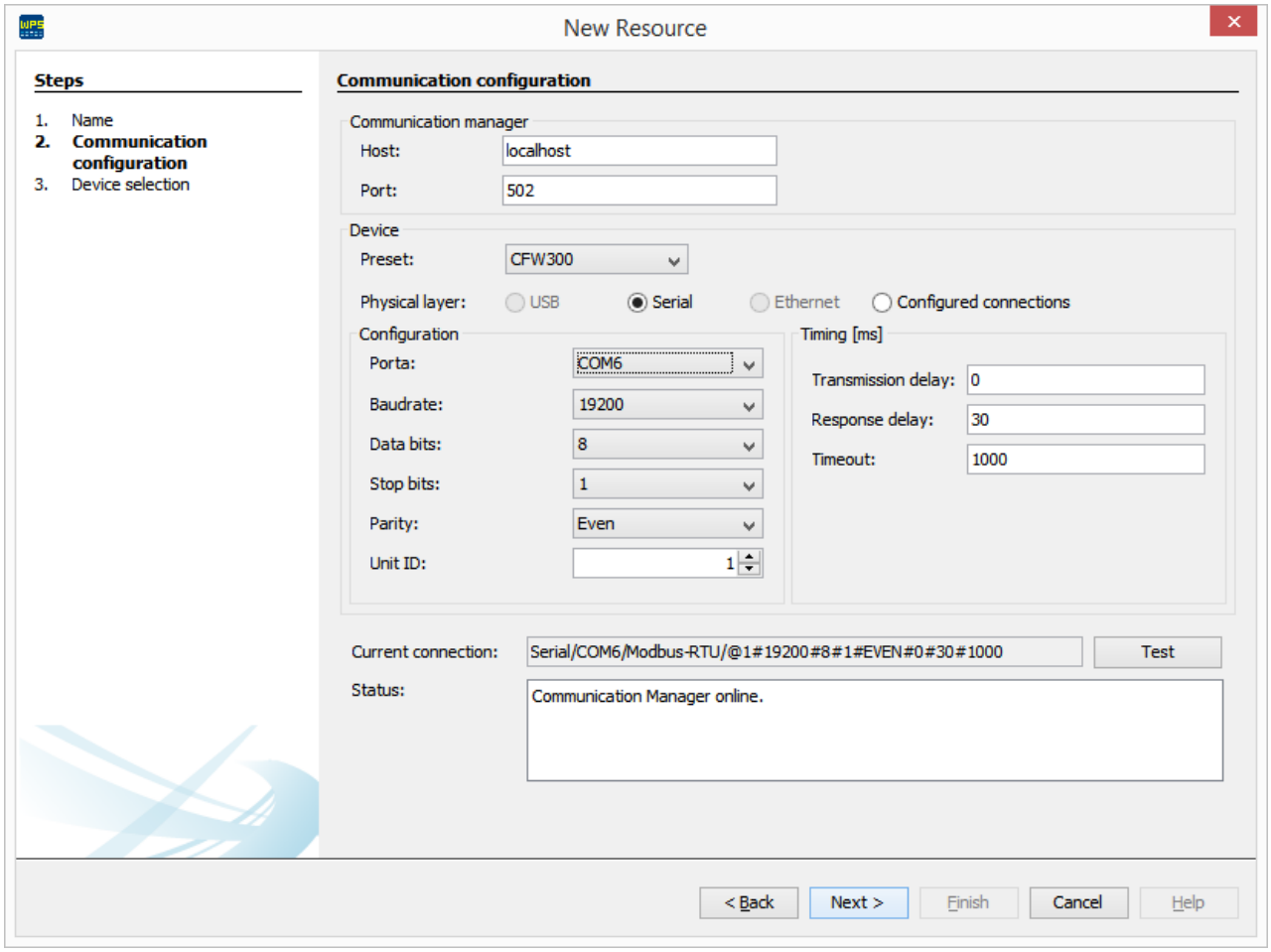


4) Select the Communication Options

Choose the correct communication options.

If you select one equipment, the default configuration of the communication of the equipment will be loaded to the window.

Still, check that the window configuration is the same as of the equipment. If not, update this window according to the equipment configuration.



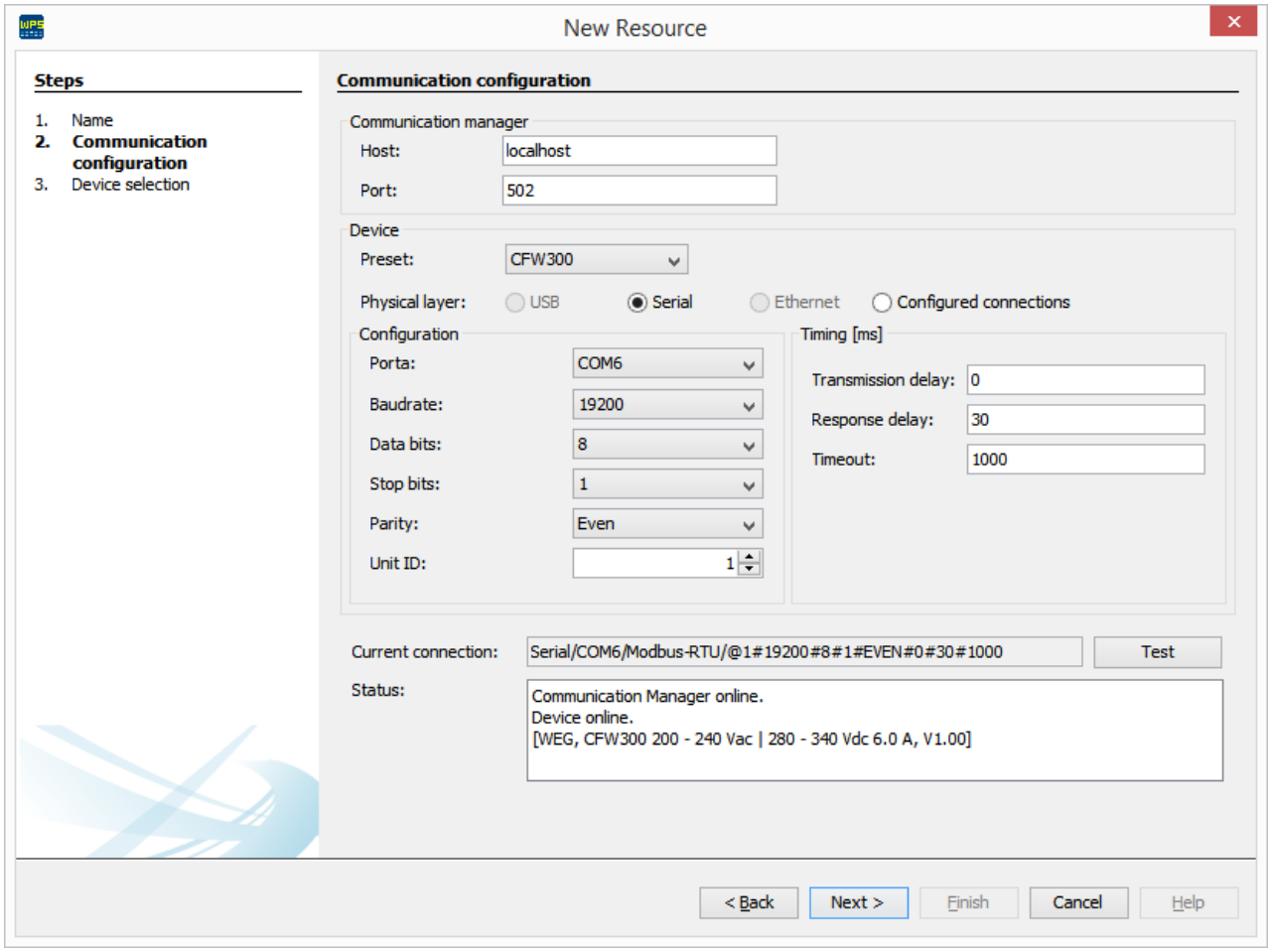
NOTE!

- In this example, the CFW300-CUSB accessory is being used, which appears in Windows as a USB serial port (virtual COM port);
- When the cable is connected to the personal computer/equipment, the Windows device manager informs the name of the USB serial port which was created.

5) Testing the Communication

You can check if the communication is correct by clicking the Test button in this window.

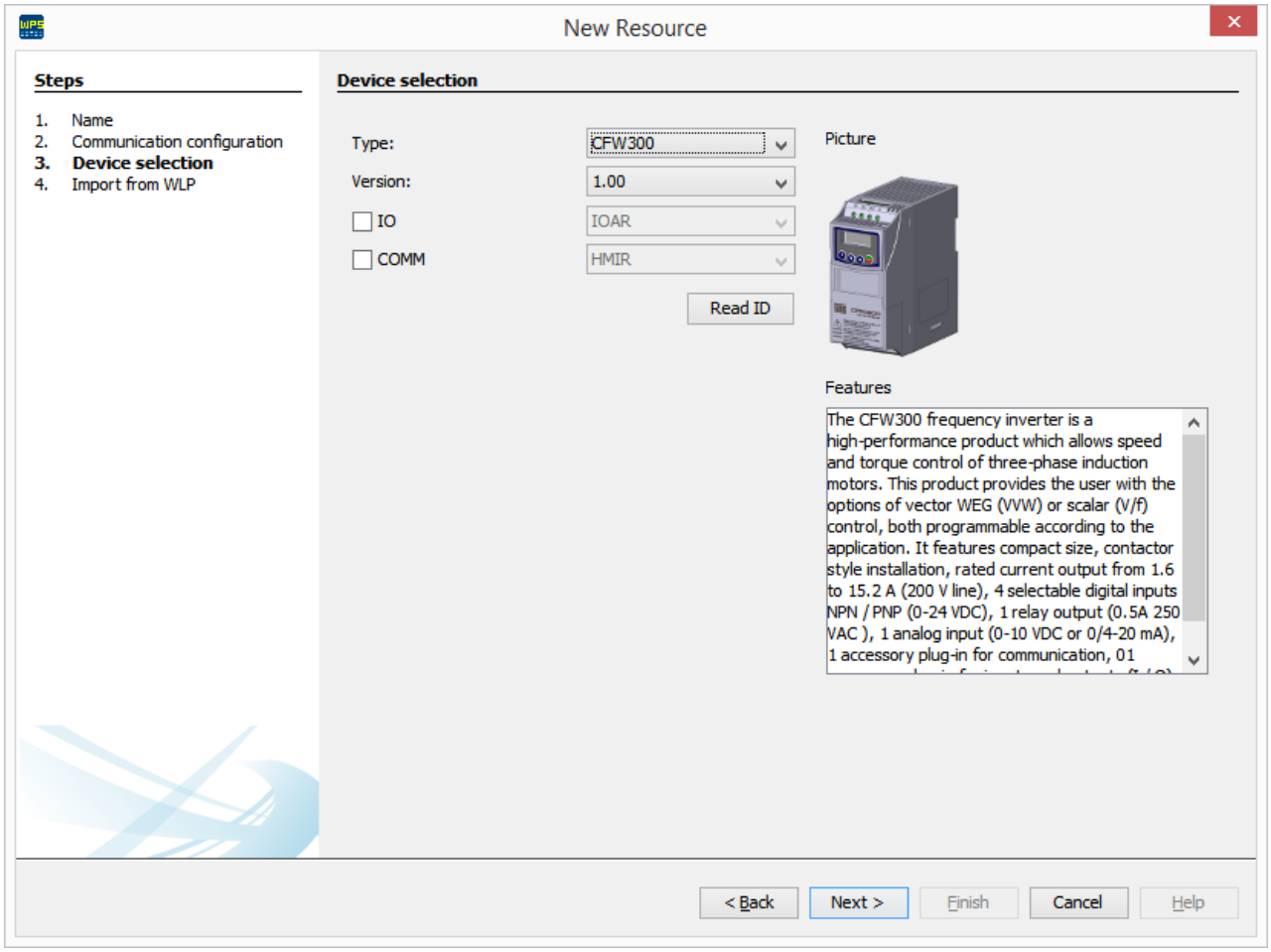
The status field is refreshed. The connected equipment appears in the status field.



Then click Next.

6) Equipment Selection

The window shows the equipment that is connected to the resource that was created.



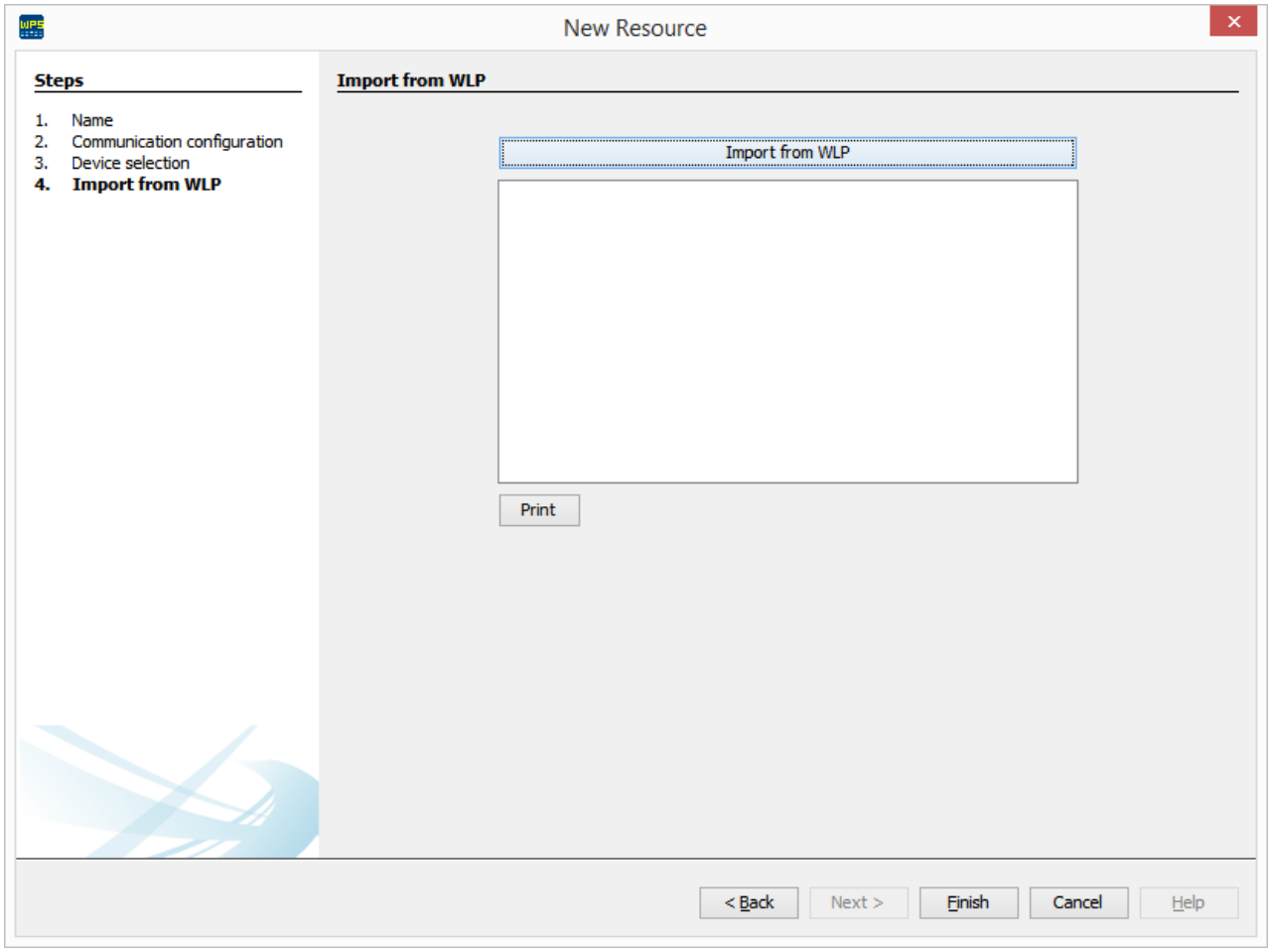
7) Read ID

If you click the Read ID button, a window containing the information on the equipment will open. Click OK to close it.

In order to go to the next step, click Next.

8) Import from WLP

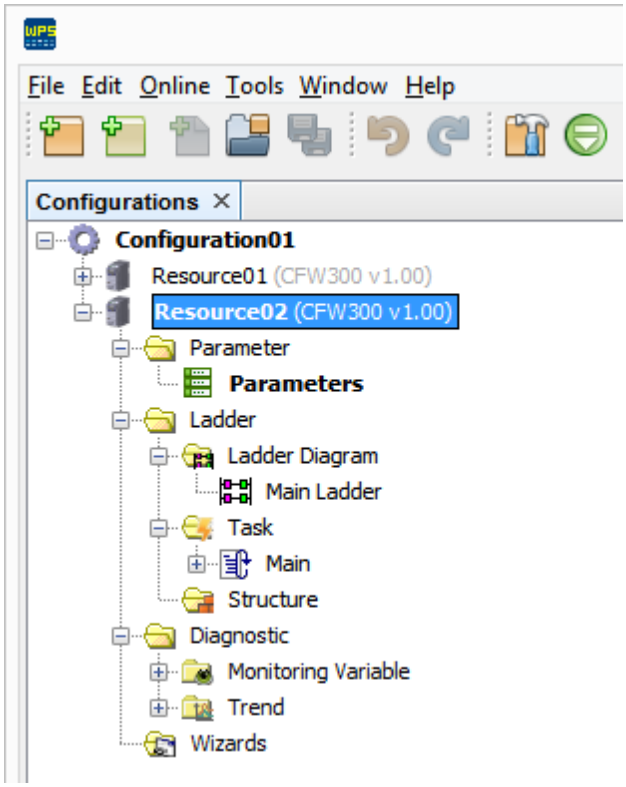
A new window pops up enabling to import from the CFW100 a Ladder project developed on the WLP (WEG Ladder Programmer).



Click Finish to close the wizard.

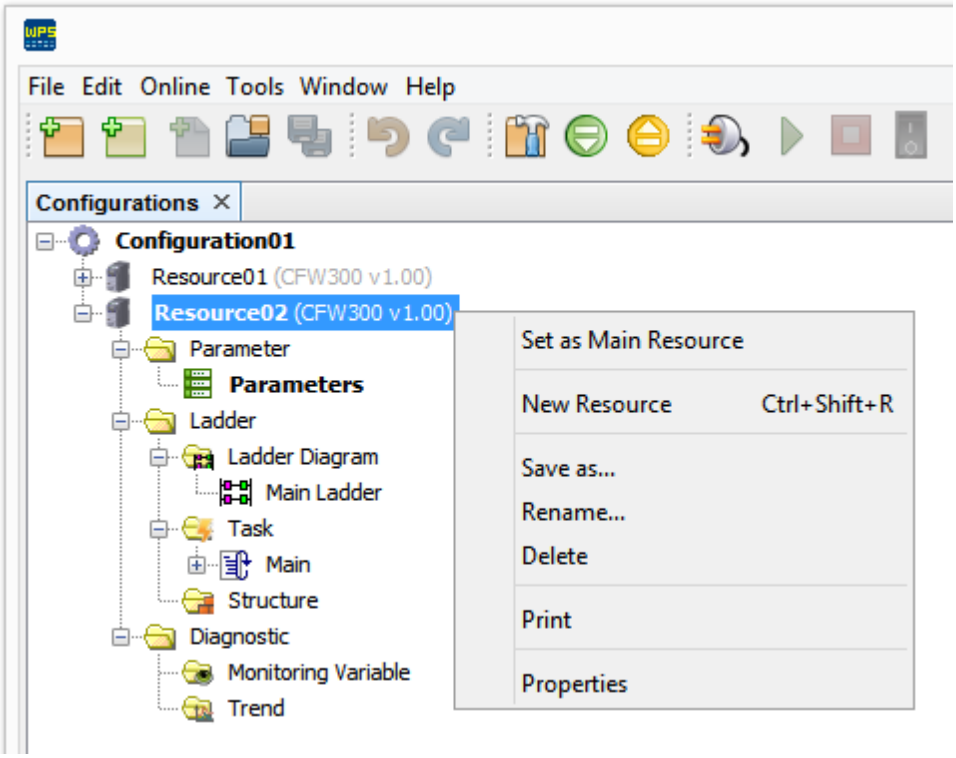
9) Configuration Tree

After those steps, the tree with the configurations should look like the following image.



10) Resource Properties

The settings made in the setup can be changed later in resource properties.



NOTE!

- Depending on the equipment, new windows may appear from this step on, with initial settings and oriented start-ups.

6.7 Pop-up Menu

The popup menu is known as context menu.

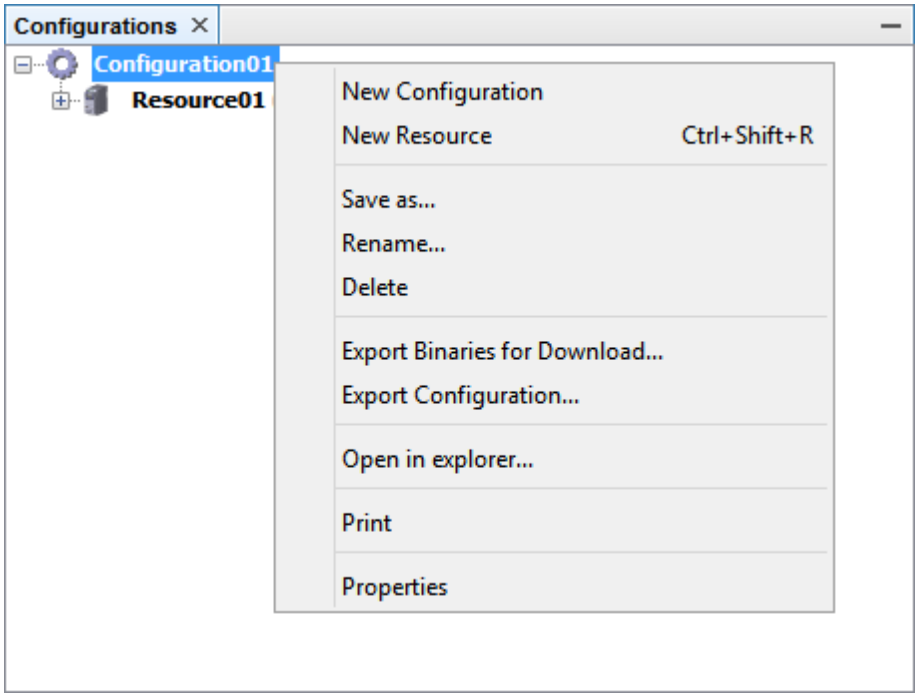
The pop-up menu is a floating menu accessed with the second or third button of the mouse, which enables quick access to the options related to the object selected by the cursor.

On the WPS, a pop-up menu adds options to make some operations simpler:

- Pop-up Menu - Configuration
- Pop-up Menu - Resource

6.8 Pop-up Menu - Configuration

The configuration pop-up menu provides some functions described below.



Click the link for quick access:

New Configuration

See the section Creating New Configuration in order to create a configuration.

New Resource

See the section Creating New Resource in order to create a resource.

Save As...

In order to duplicate a configuration, click Save As in the pop-up menu.

In the Name field, enter the configuration new name.

- Click OK to accept or
- Click Cancel in order to cancel the operation.

Rename...

In order to rename a configuration, click Rename in the popup menu.

Enter the new name.

- Click OK to accept or
- Click Cancel in order to cancel the operation.

Delete

In order to delete a configuration, click Delete in the pop-up menu.

A message requests confirmation to delete a configuration.

- Click OK to delete or
- Click Cancel in order to cancel the operation.

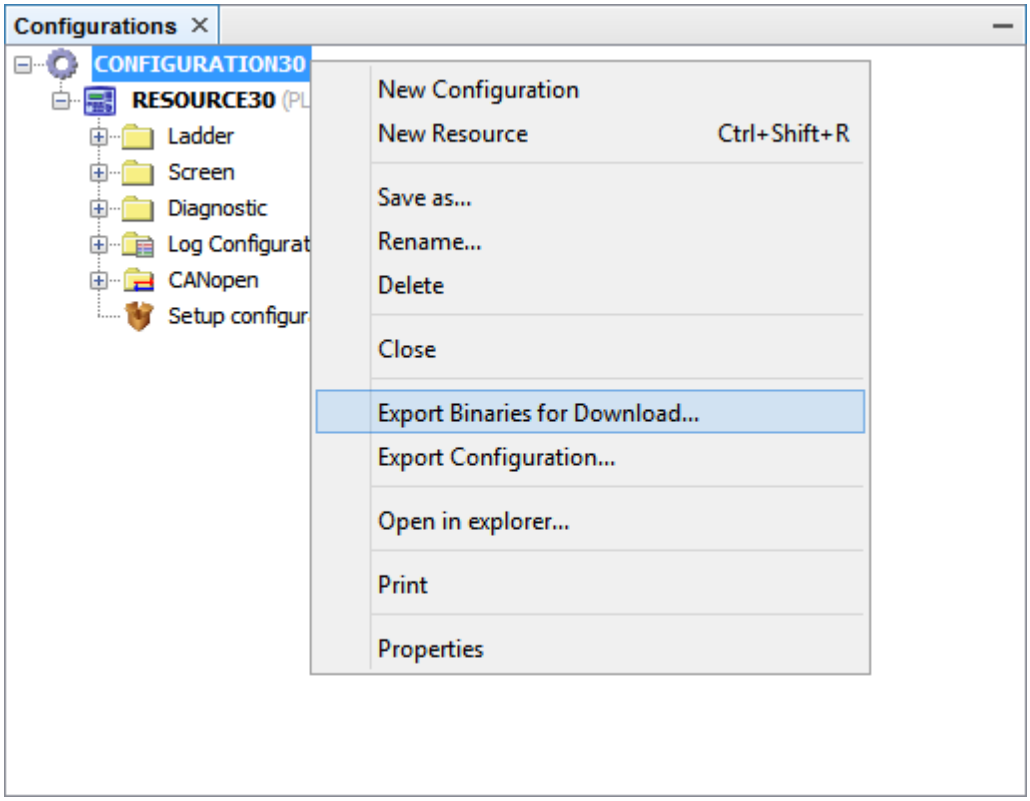
Export Binaries for Download

The WPS v2.5X compiler generates binary files from the source files in the resource. When these files are transferred to the machine, they are interpreted and executed.

The export binaries for download functionality encapsulates the binaries files into a .bbp file. This .bbp file can be redistributed, but the original source file can not be viewed or changed.

1) Export from the Configuration (Pop-up Menu)

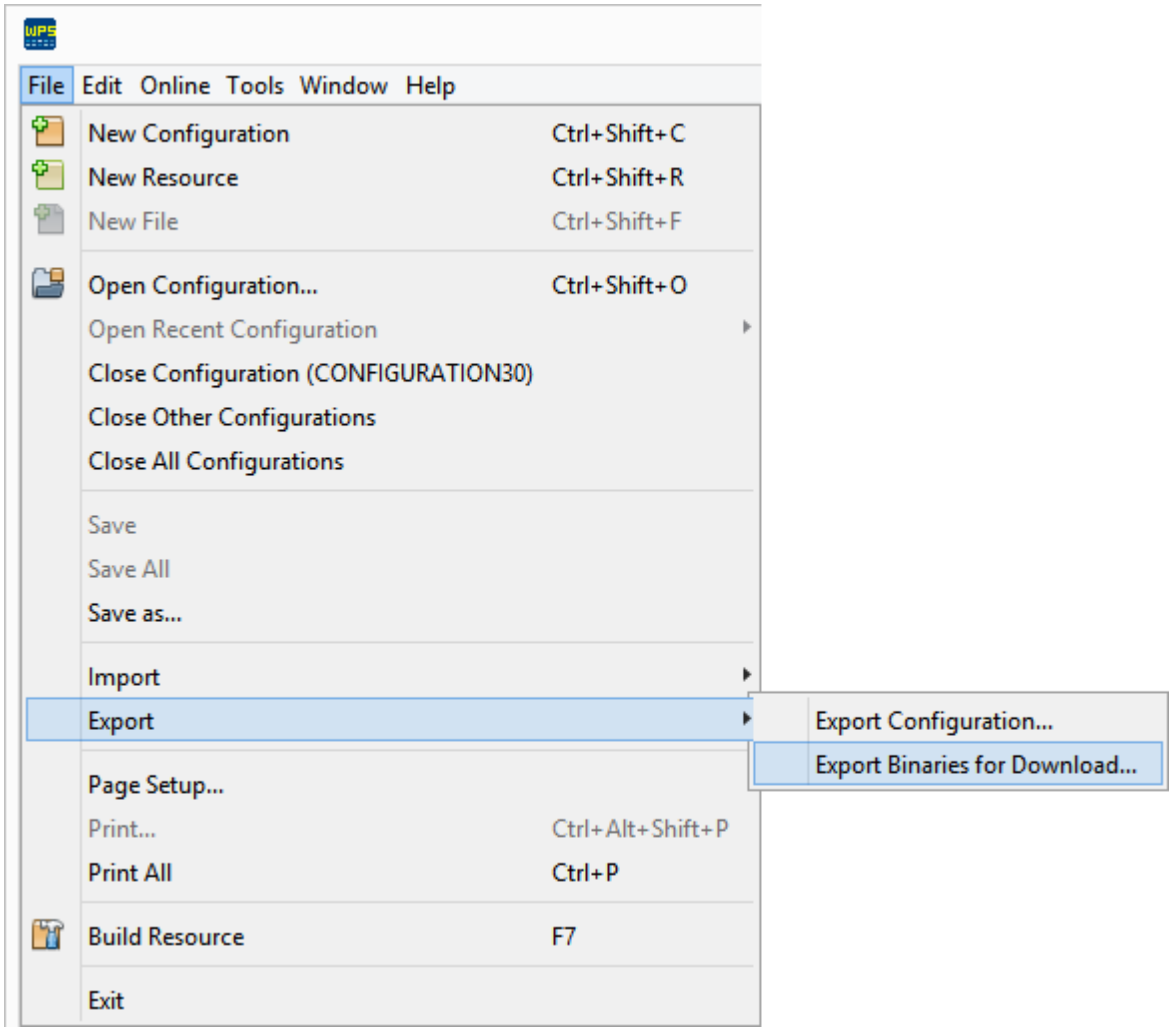
Right-click the configuration that contains the resource to be exported, and then, in the popup menu, click **Export Binaries for Download**.



2) Export from the Menu Item

The Export Binaries for Download menu item can also be used to export.

In the File menu, click Export, and then Export Binaries for Download.

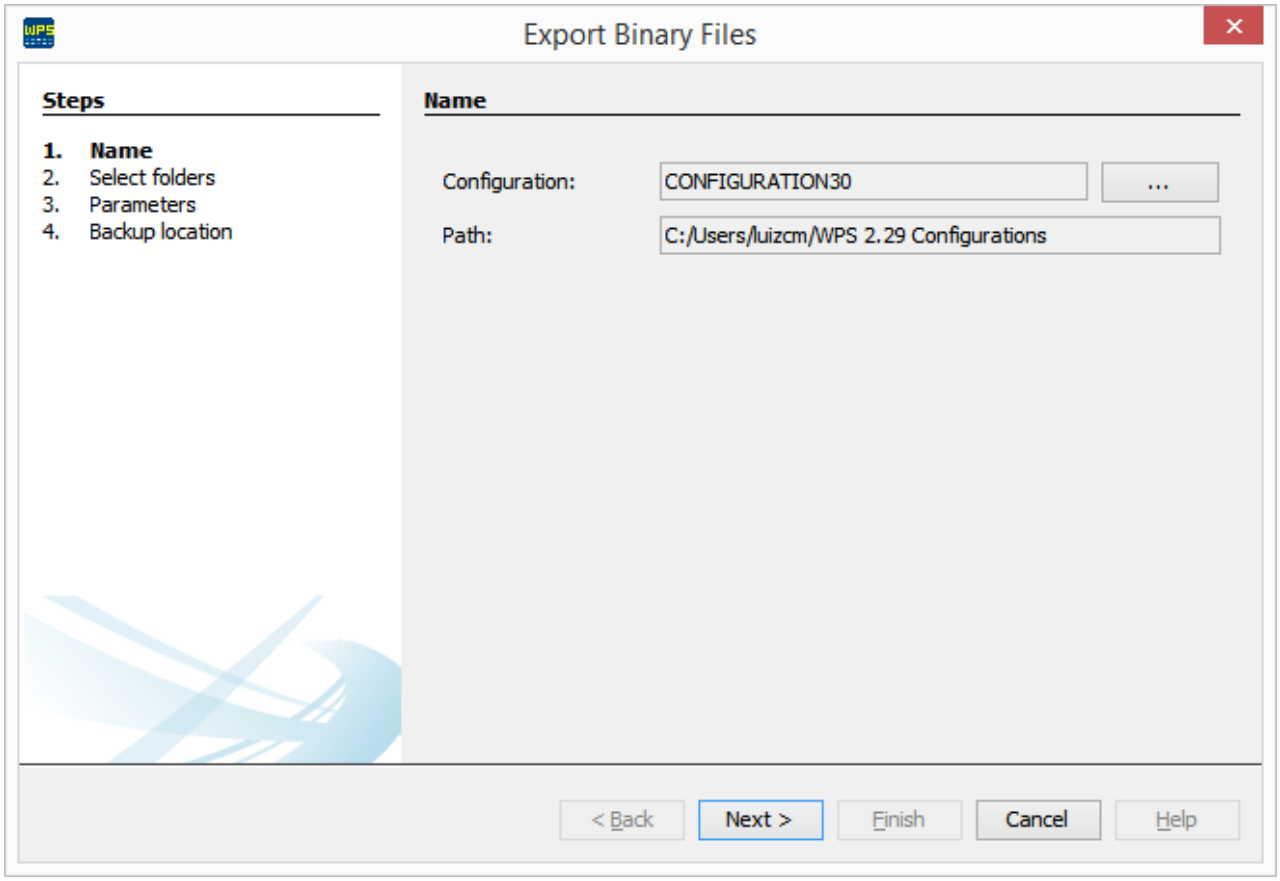


3) Configuration Selection

A window of the export wizard will open, displaying the name and path of the configuration containing the resource to be exported.

Click the ... button in order to modify those data.

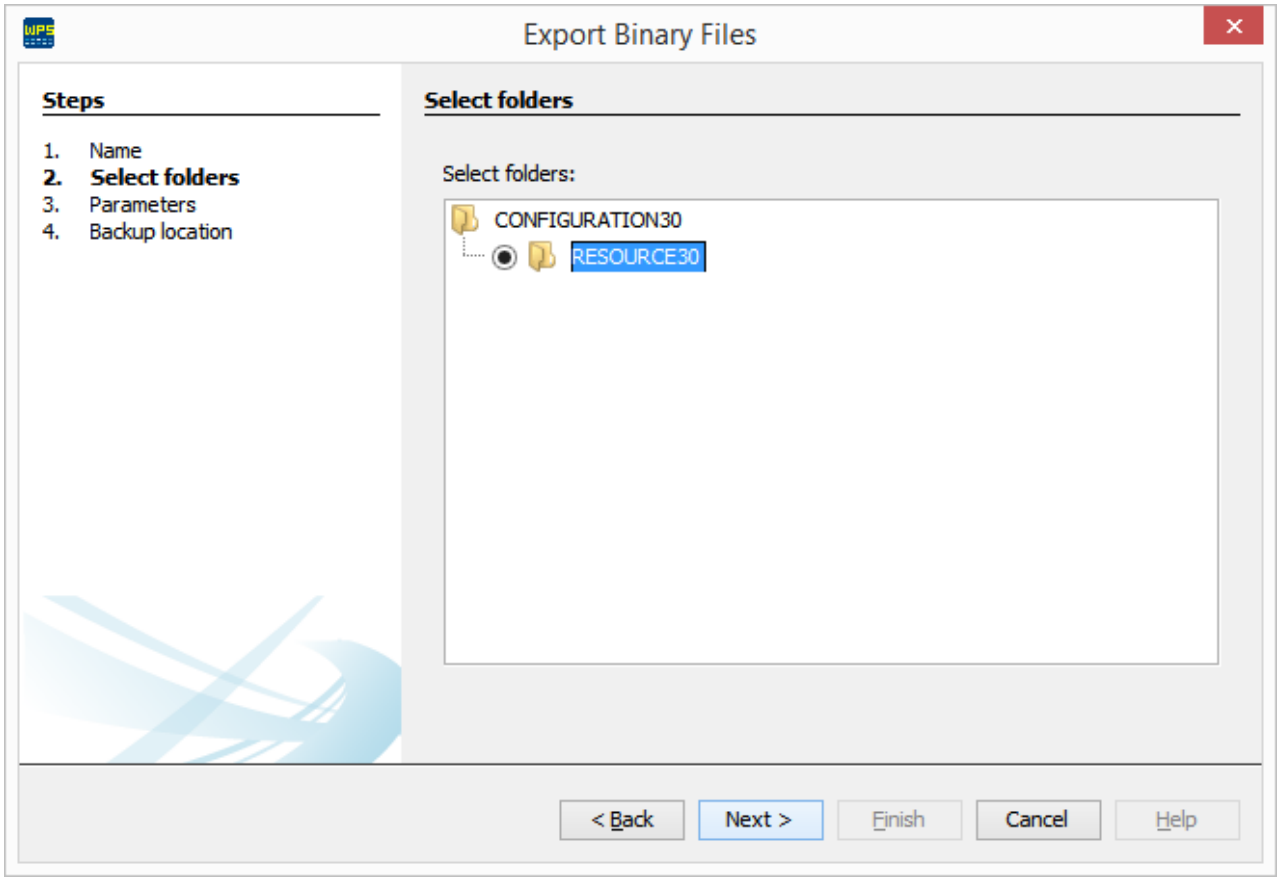
Then click **Next**.



4) Resource Selection

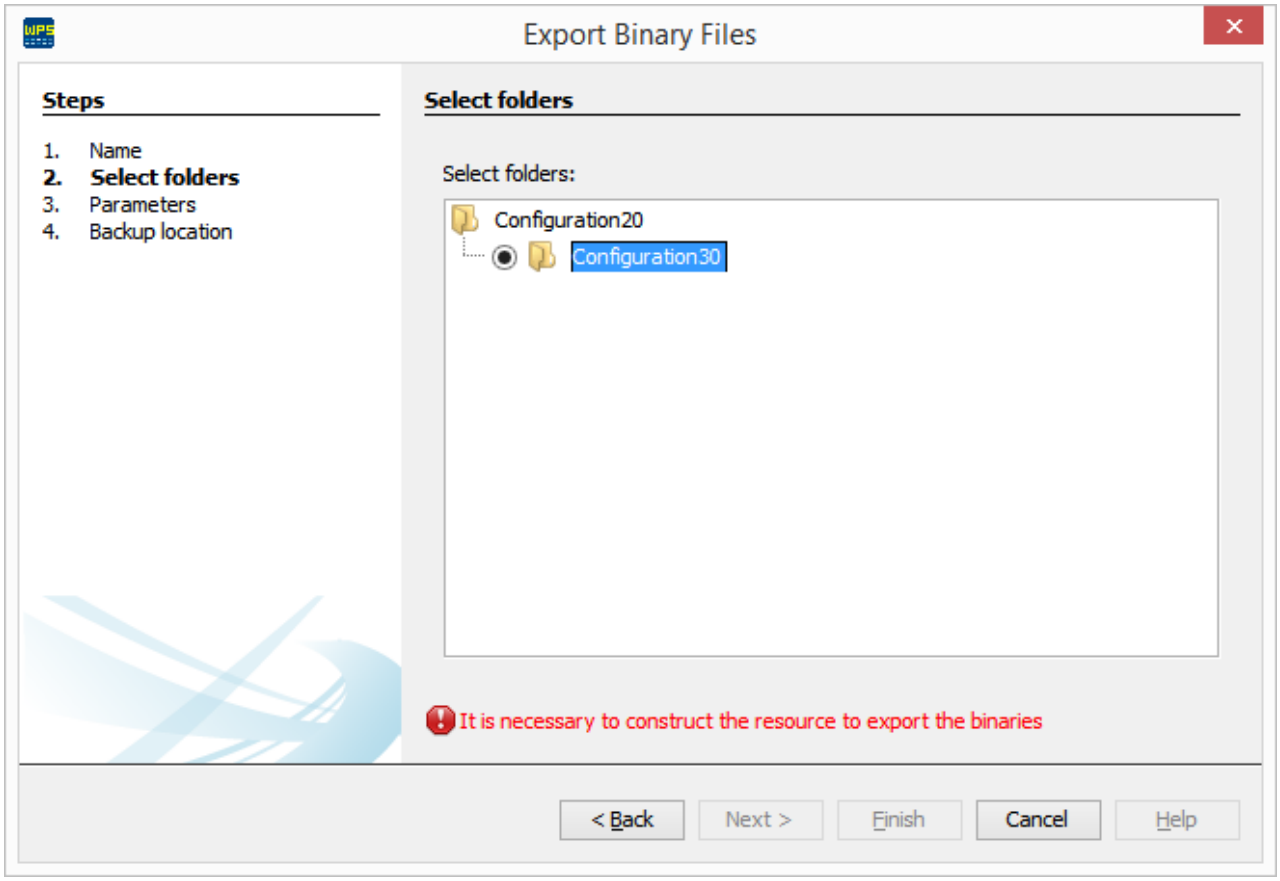
Then the resource selection screen will pop up, containing the binaries to be selected for exportation.

Select the desired resource and click **Next**.



5) Error Exporting

If the exported resource has not been previously compiled, an error message will pop up.



Click Cancel to exit.

Compile the resource and return to the first step of this tutorial.

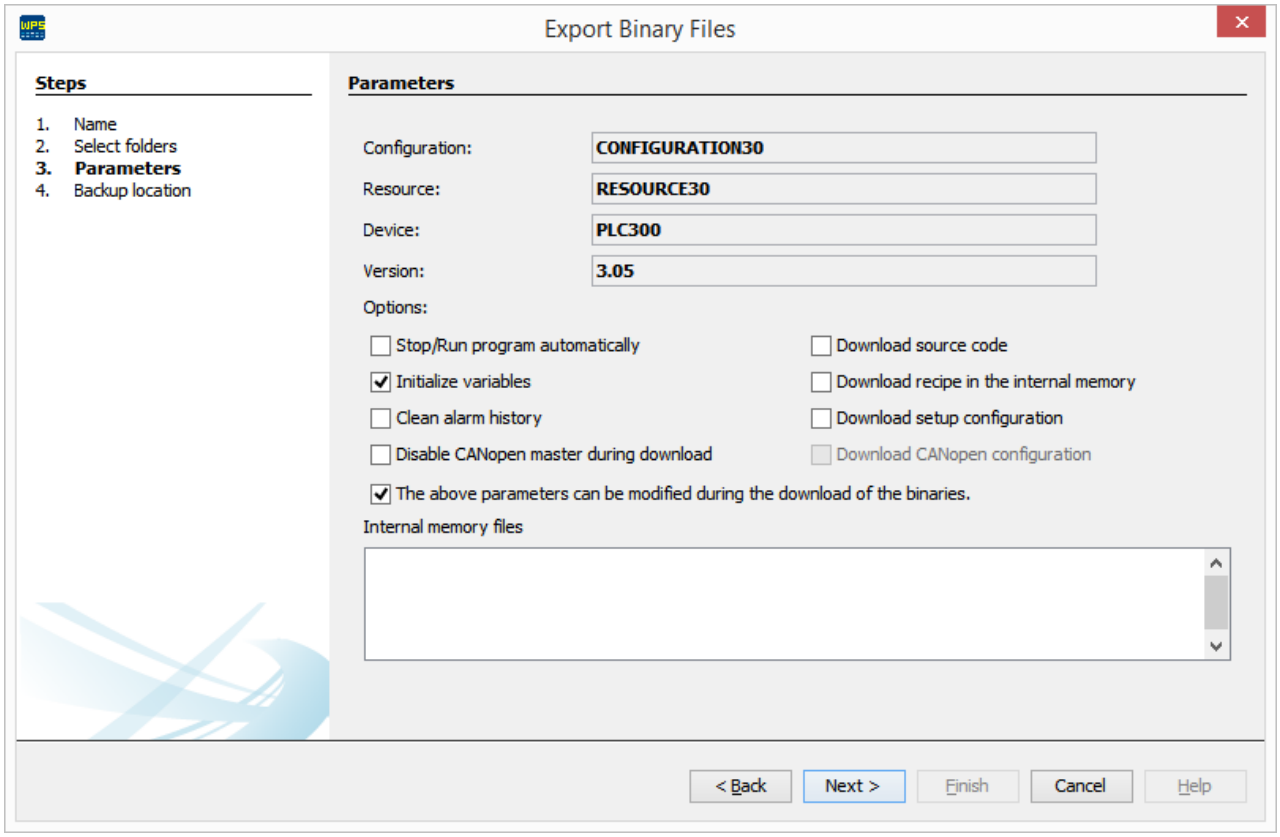
NOTE!
Always compile the desired resource before exporting its binary files.

6) Select Options

The next screen is similar to the resource download screen, enabling the configuration of several download options.

Refer to the section Download for further information.

The last checkbox defines whether those settings can be changed at the moment of the download of the binaries.



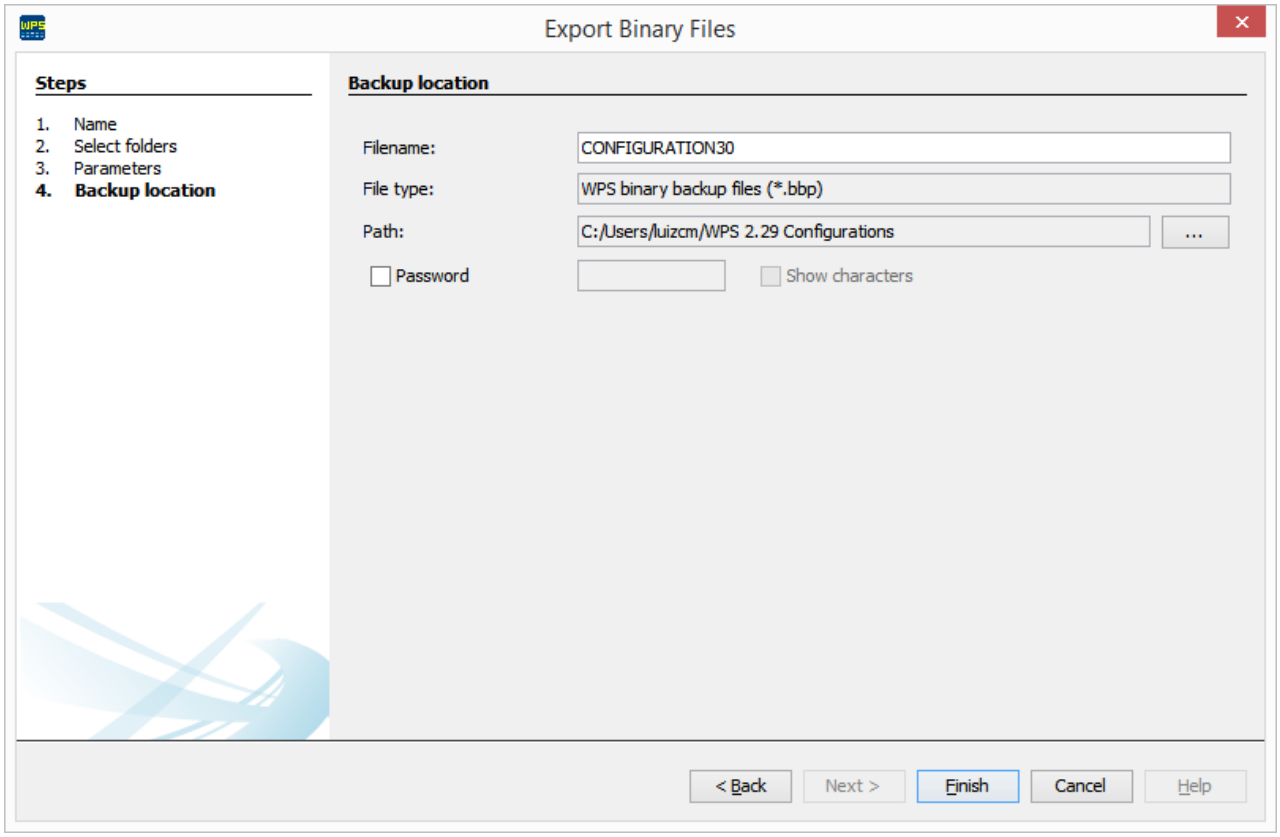
NOTE!
Download options change depending of the resource selected.

7) Location of the Exported File

Finally, one last screen allows defining the name of the file and its location. Click the ... button in order to change those data.

This screen also allows entering a password to protect the file. If desired, check the checkbox and enter the password.

Then click **Finish**.



Export Configuration

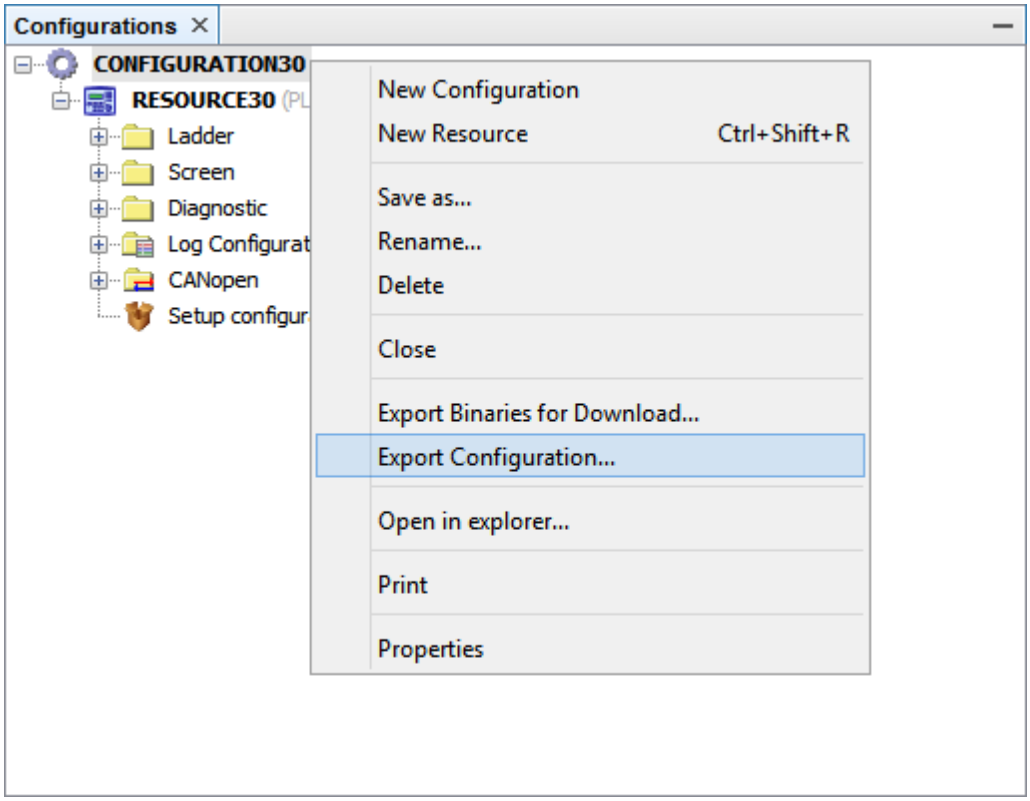
The WPS v2.5X allows the configurations to be exported to a backup file in the .bkp format.

That simplifies sending a configuration to another machine.

You should always use this procedure to prevent file corruption.

1) Export from the Configuration (Pop-up Menu)

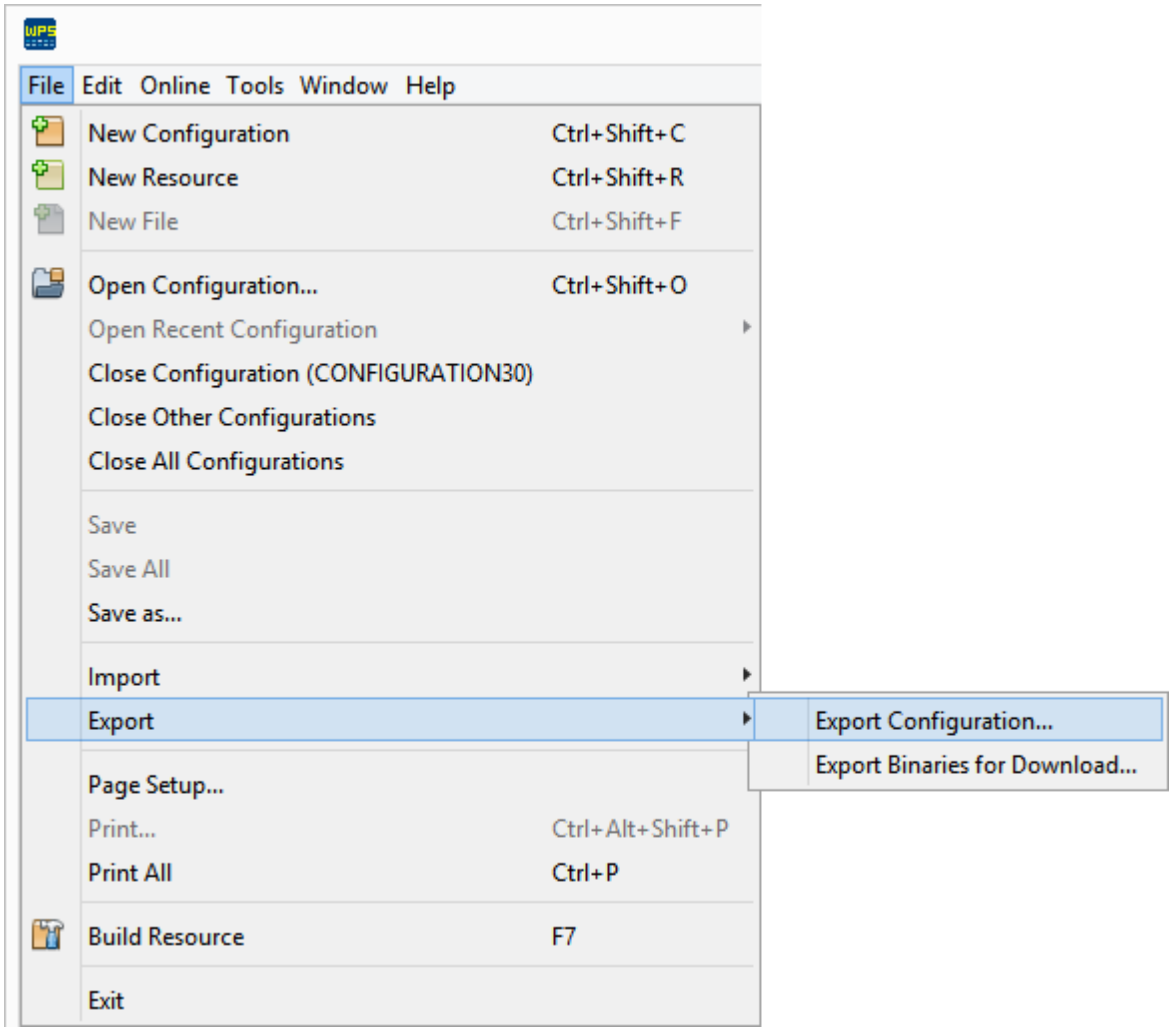
Right-click the configuration to be exported, and then, in the pop-up menu, click Export Configuration.



2) Export from the Menu Item

The Export Configuration menu item can also be used to export.

In the File menu, click Export, and then Export Configuration.

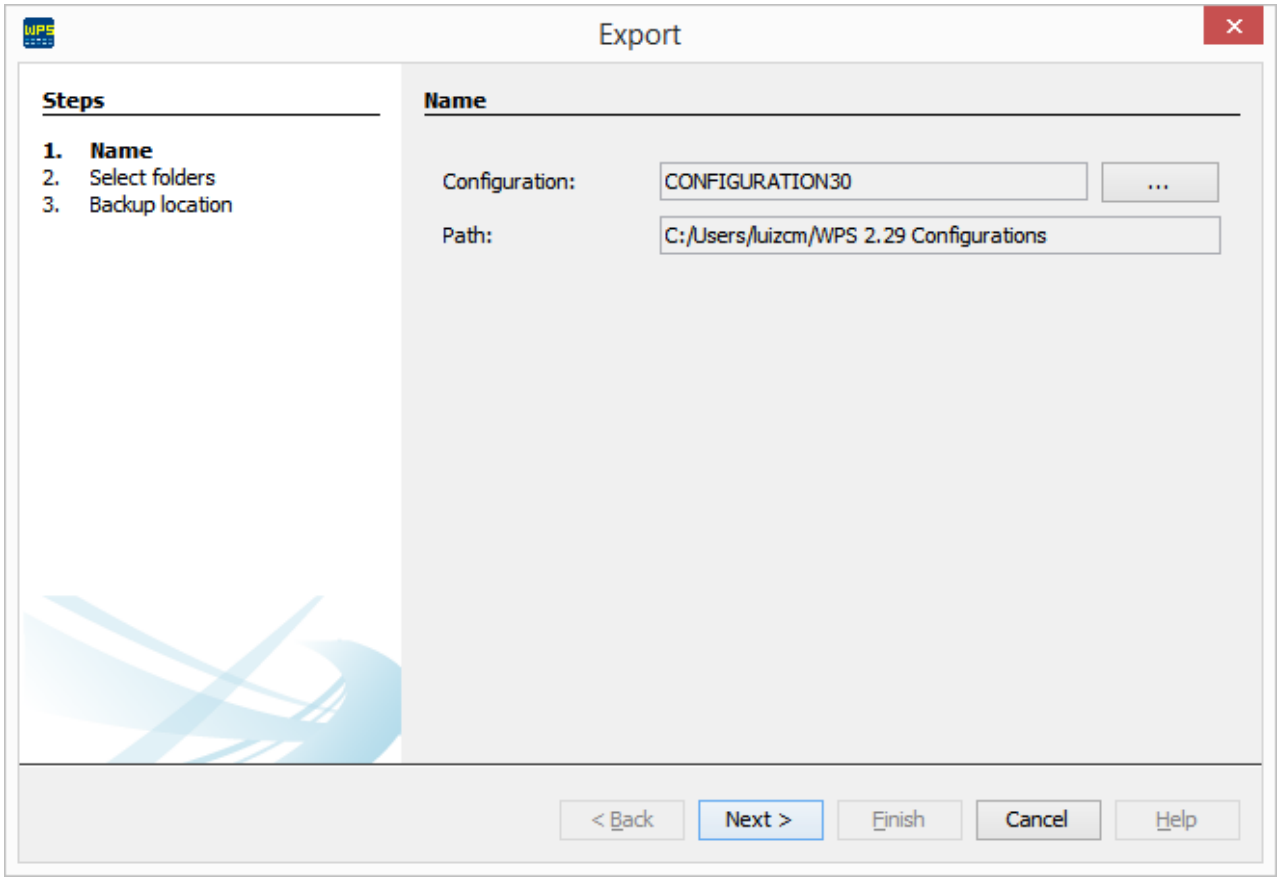


3) Configuration Selection

The export wizard will open, displaying the name and path of the configuration to be exported.

Click the ... button to modify those data.

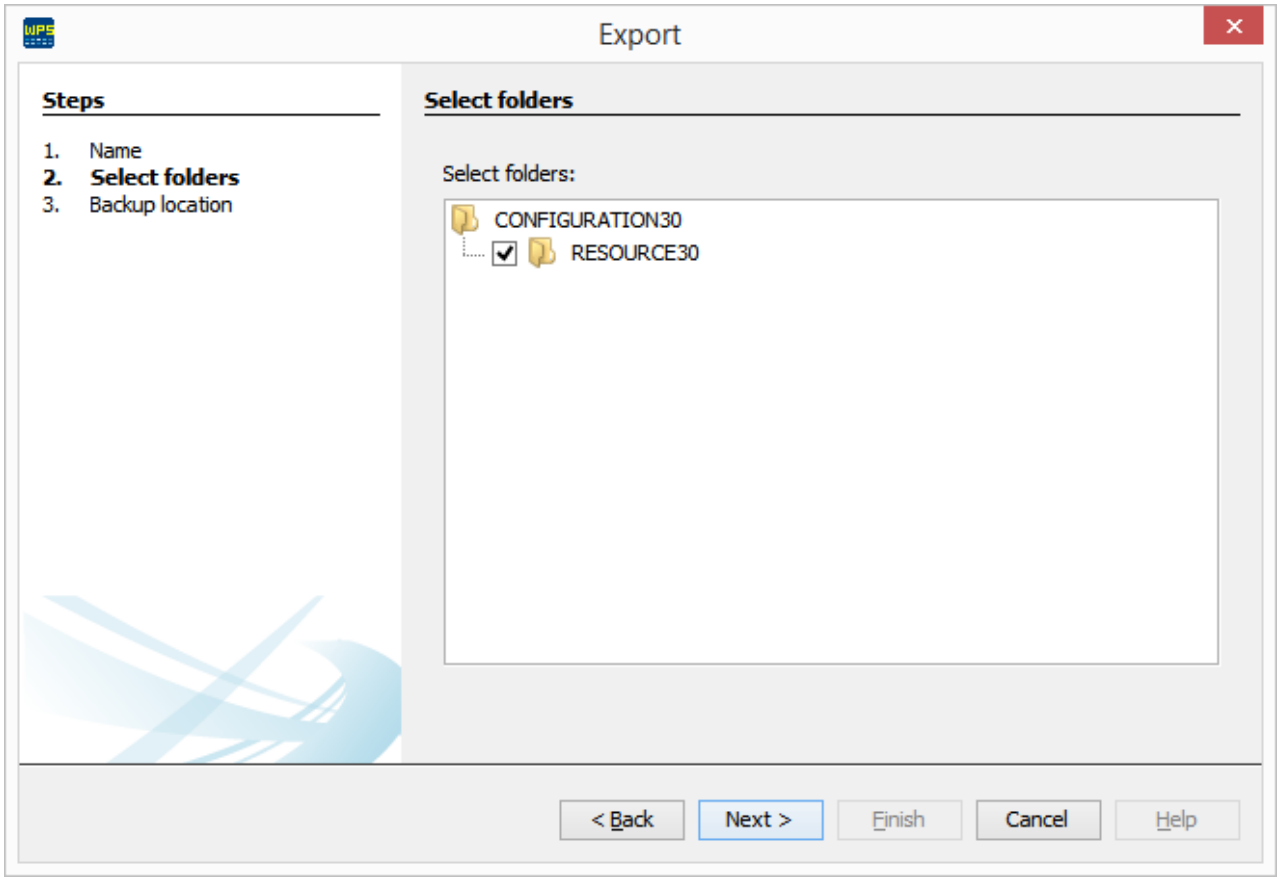
Then click Next.



4) Resource Selection

Then a screen to select the resources to be exported to the file will pop up.

Select the resources you wish to export and click **Next**.



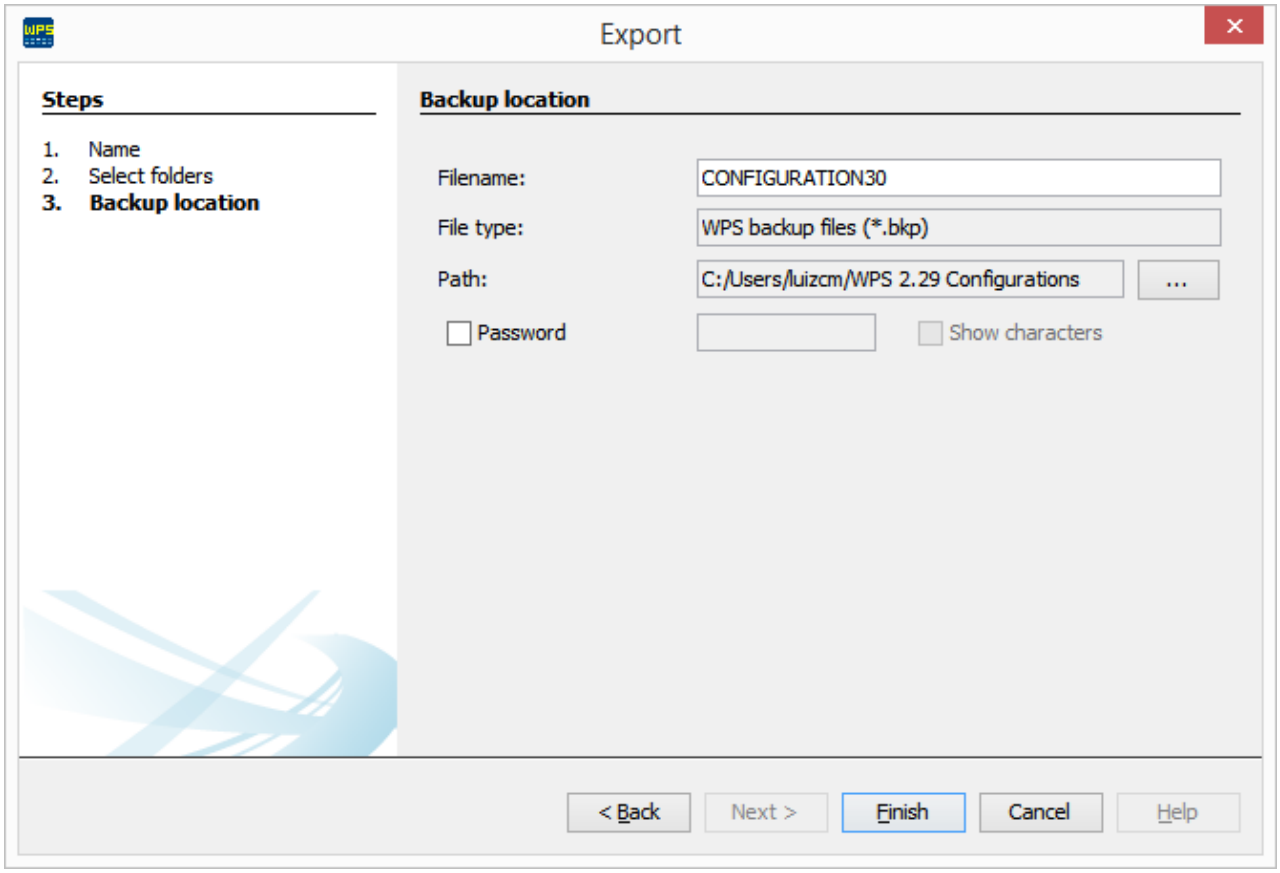
5) Select Options

Finally, one last screen allows defining the name of the backup file and its location.

Click the ... button in order to modify those data.

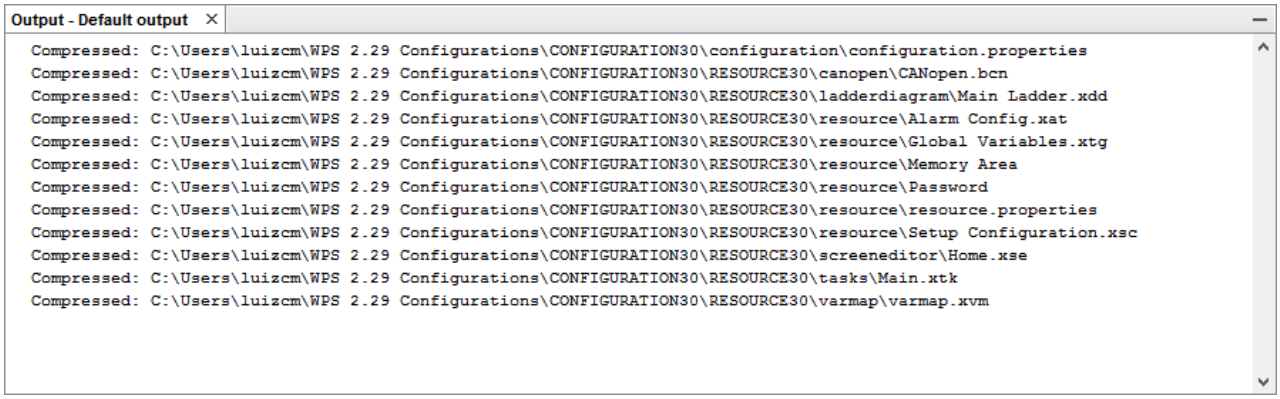
This screen also allows entering a password to protect the file. If desired, check the checkbox and enter the password.

Then click **Finish**.



6) Wait until Finish

In the exit window, you can view messages informing generation of the exported configuration.



Open in Explorer

Open Windows Explorer in the configuration folder.

The user can view the content of the folder in Windows Explorer.

Print

Select the option that you wish to print.

- Click OK in order to print or
- Click Cancel in order to cancel the operation.

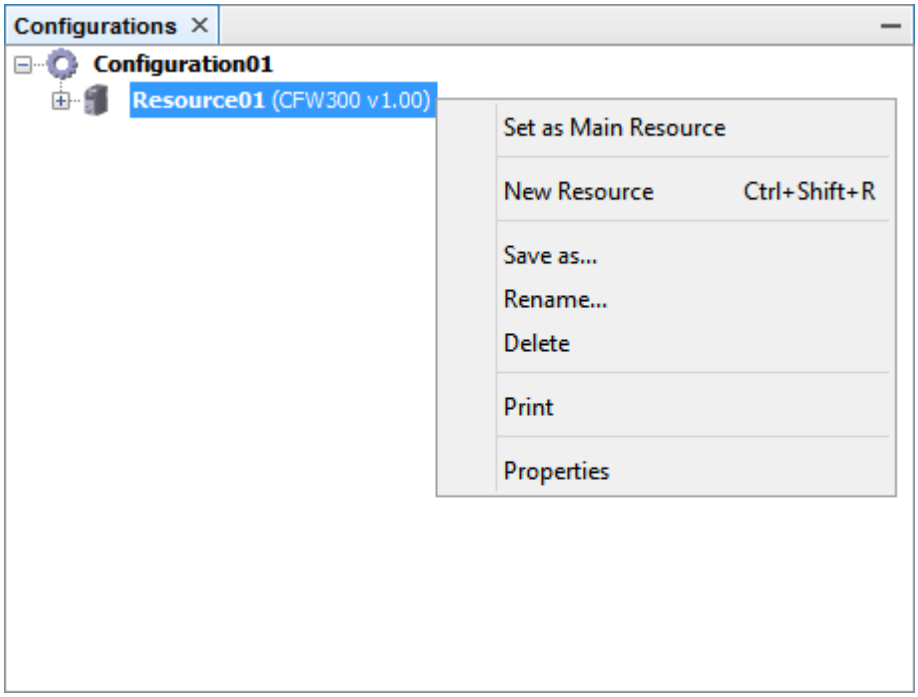
Properties

Click the Properties item, and a window will pop up with the following items:

- Name: configuration name;
- Path: configuration path;
- Size: configuration size.

6.9 Pop-up Menu - Resource

The pop-up menu of the resource provides some functions described below.



Click the link for quick access:

Define as Main Resource
New Resource
- - -
Save As...

Rename...
Delete
- - -
Print
- - -
Properties

Define as Main Resource

When a configuration has several resources, the resource in use must be defined as main resource.

In order to define a resource as main resource, click the **Define as Main Resource** item in the pop-up menu.

After clicking the menu item, the resource will show in bold.

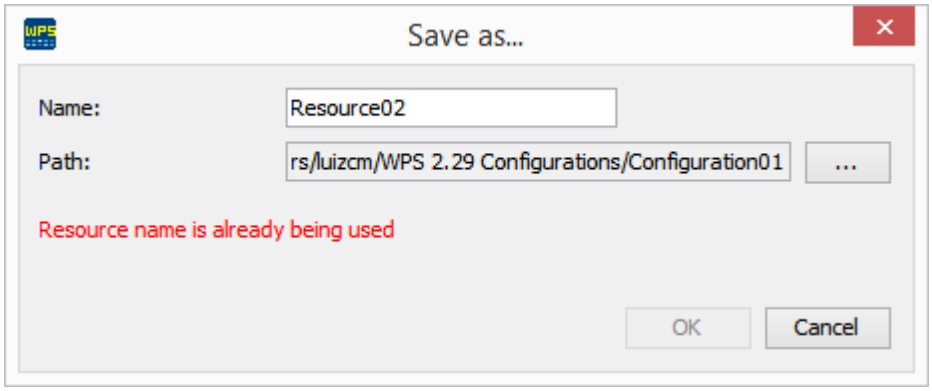
> **NOTE!**
> Remember to always set a resource you wish to work as main resource, preventing download and editing errors.

New Resource

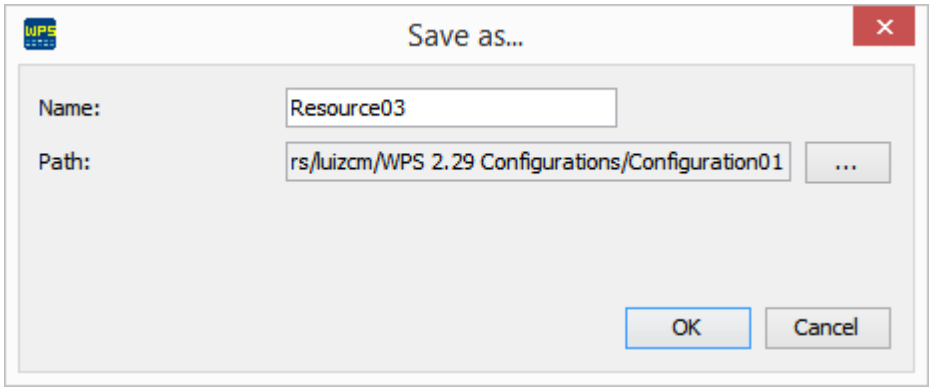
See the section Creating New Resource for details on creating a new resource.

Save As...

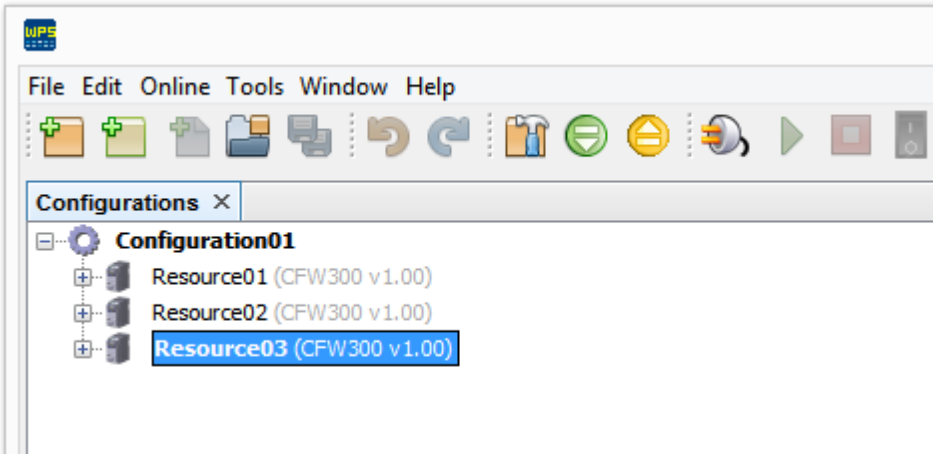
In order to duplicate a resource, click Save As in the pop-up menu.



In the Name field, enter the new name of the resource and click OK.



The new resource can be viewed in Configurations.



Rename

In order to rename a resource, click Rename in the pop-up menu.

Enter the new name.

- Click OK to accept or
- Click Cancel in order to cancel the operation.

Delete

In order to delete a resource, click Delete in the pop-up menu.

A message requests confirmation to delete the resource.

- Click OK to delete or
- Click Cancel in order to cancel the operation.

Print

Select the option that you wish to print.

- Click OK in order to print or
- Click Cancel in order to cancel the operation.

Properties

In order to view/edit the resource properties, click the Properties menu item in the pop-up menu.

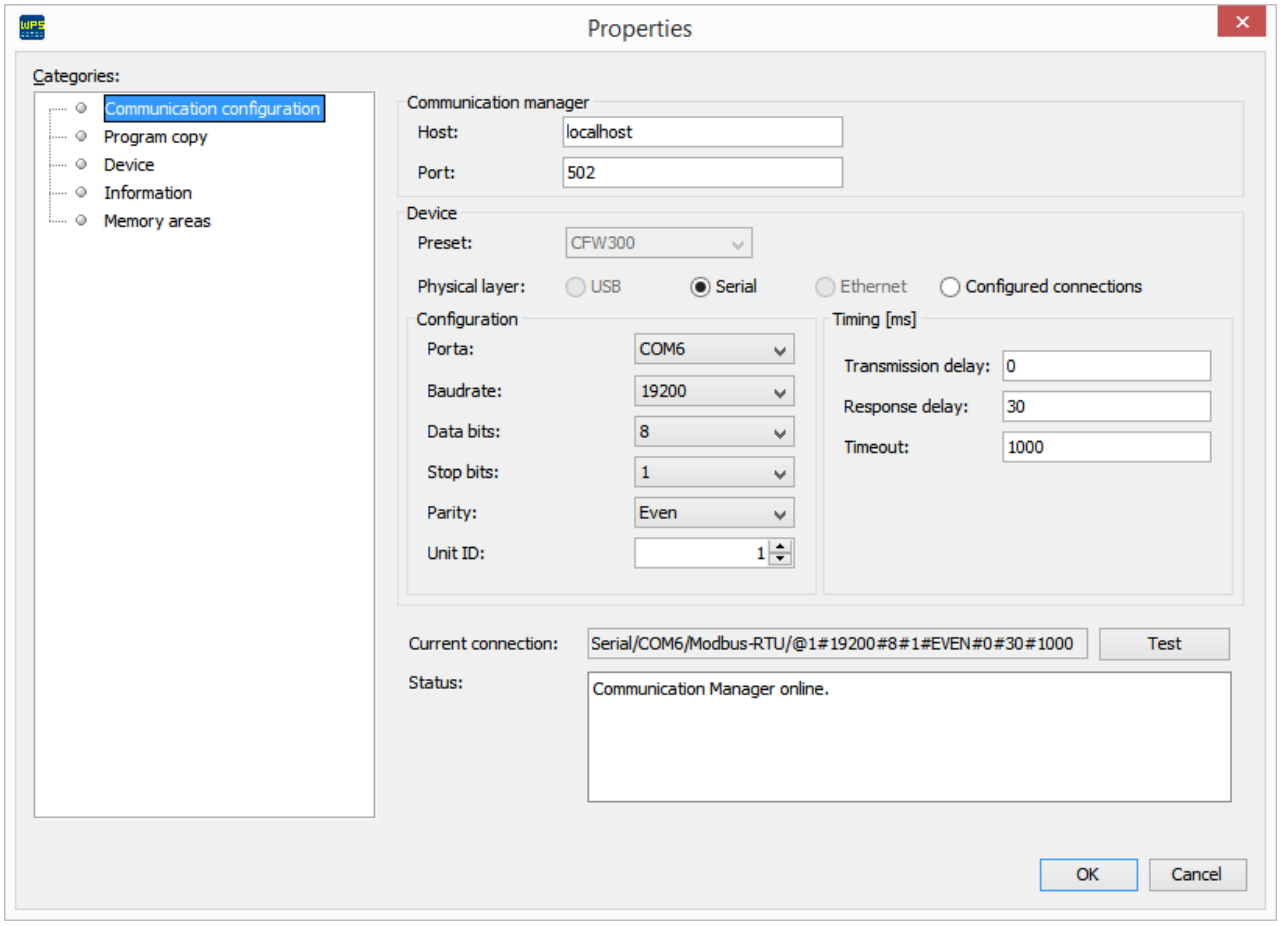
The properties window shows some categories:

- Communication Configuration;
- Program Copy;
- Equipment;
- Information;
- Memory Areas.

1) Communication Configuration

In Communication Configuration, you can change different communication variables:

- Name of the host and port of the communication manager;
- Device (Equipment);
- Physical layer;
- Port settings;
- Times;
- Communication test (current connection);
- Communication status.

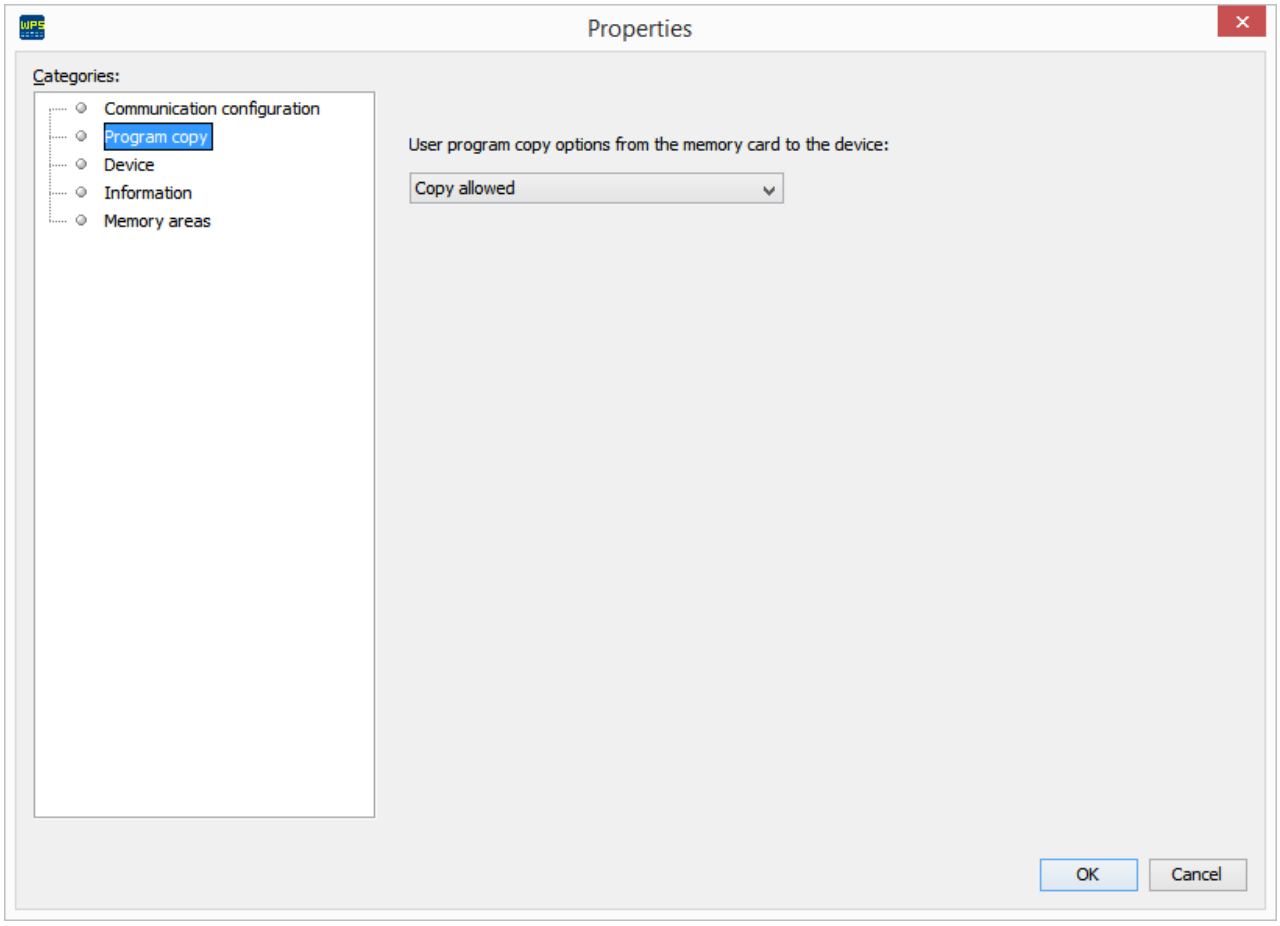


2) Program Copy

NOTE!
This option is available only for CFW300.

In Program Copy, a checkbox allows selecting the copy type of the user program from the memory card to the equipment:

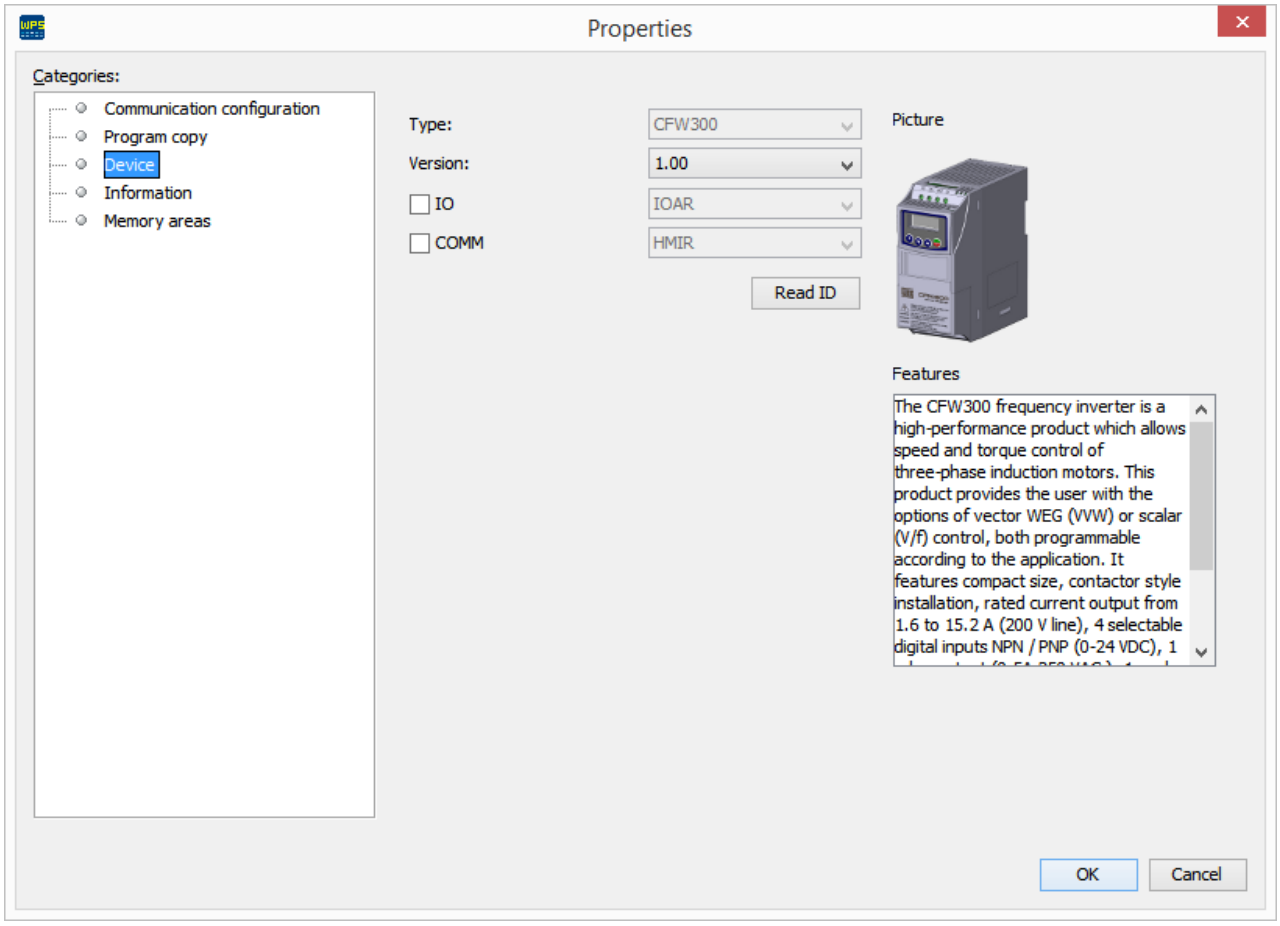
- Allow copying: allows the ladder program to be copied to a flash memory module (CFW300-MMF);
- Allow only one copy: allows the ladder program to be copied to a flash memory module (CFW300-MMF), which allows loading it to another equipment; however, it does not allow copying from that equipment to another flash memory module, that is, it does not allow copy of the copy;
- Do not allow copying: does not allow the ladder program to be copied to a flash memory module (CFW300-MMF);



3) Device

In Device, you can:

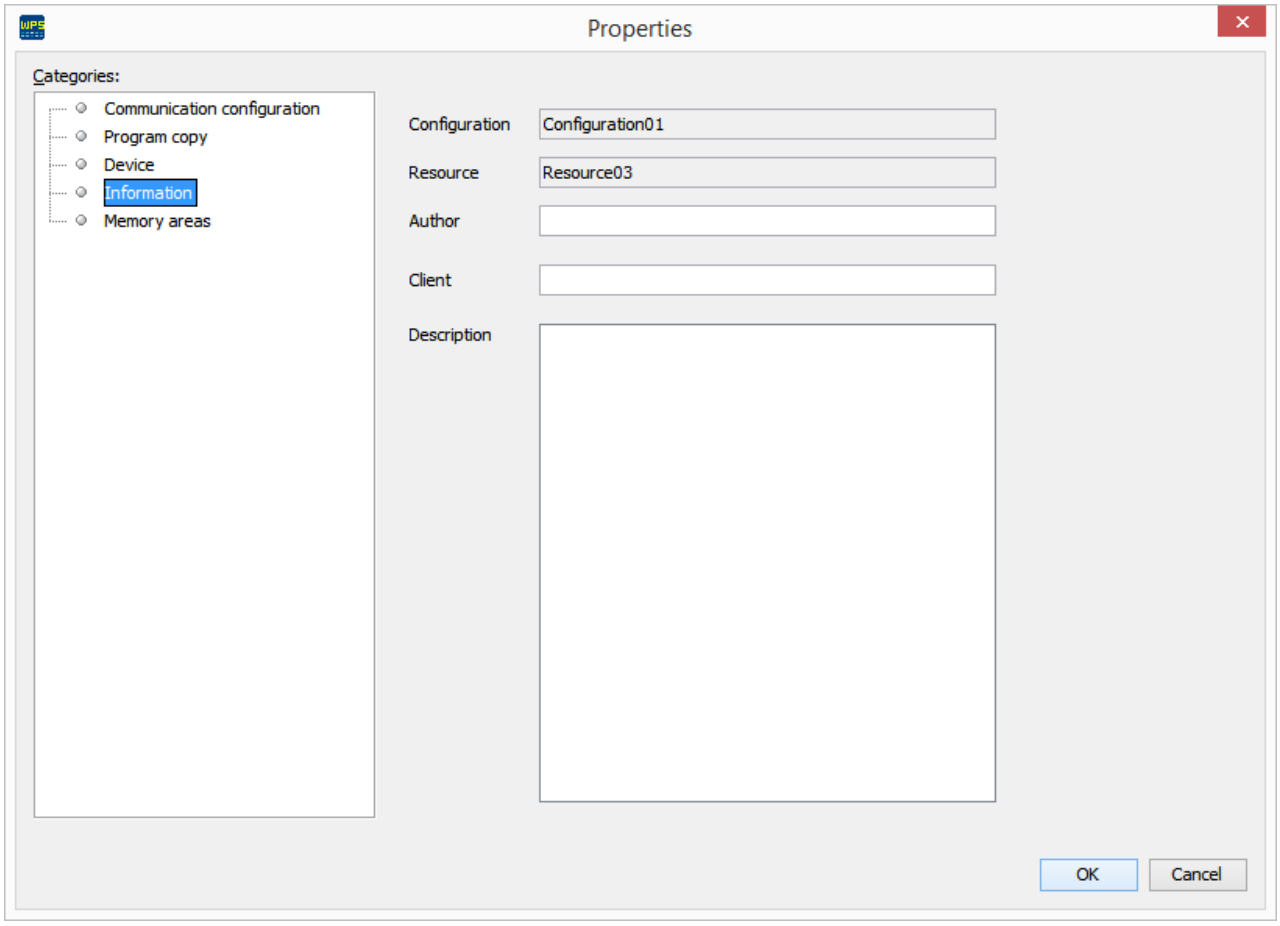
- Change the firmware version;
- Select/remove accessories;
- Read the equipment identification, viewing the model and firmware version.



4) Information

In Information, you can view/change configuration data:

- Author;
- Client;
- Description.



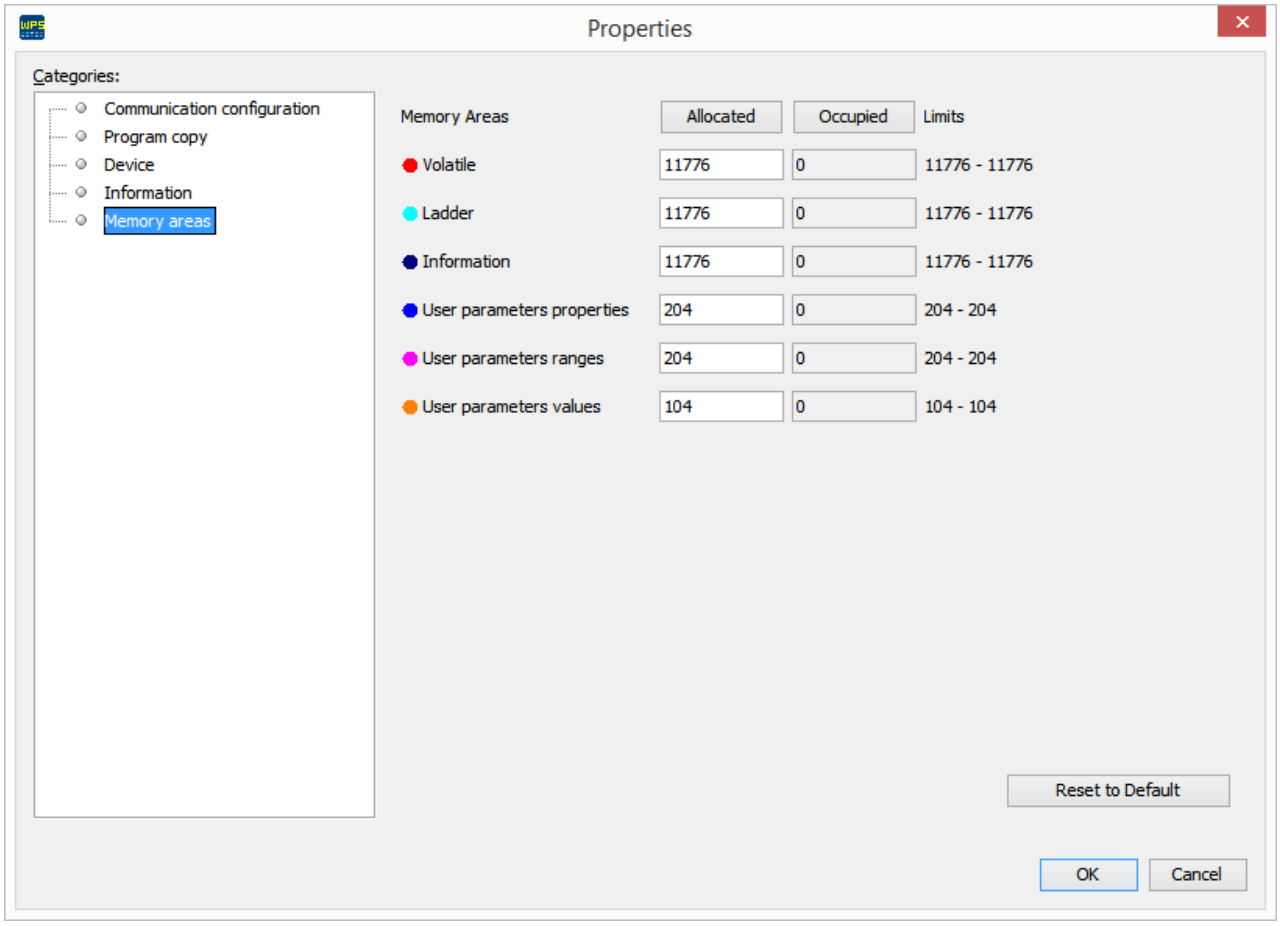
5) Memory Areas

In Memory Areas, it is possible to change the amount of memory (in bytes) allocated to each resource section.

- Allocated memory: the editable text boxes show information of allocated memory, enabling the user to change them;
- Used memory: the non-editable text boxes show information on the memory currently used by the resource loaded on the equipment;
- Limits: next to these boxes, it is displayed the range of acceptable values of allocated memory for each field.

One click on the Allocated button shows a chart of allocated memory.

One click on the Used button shows a chart of used memory.


Properties
Categories:
Communication configuration
Program copy
Device
Information
Memory areas
Memory Areas
Allocated
Occupied
Limits
Volatile
11776
0
11776 - 11776
Ladder
11776
0
11776 - 11776
Information
11776
0
11776 - 11776
User parameters properties
204
0
204 - 204
User parameters ranges
204
0
204 - 204
User parameters values
104
0
104 - 104
Reset to Default
OK
Cancel

7 Communication

7.1 Equipment Parameterization

The following parameters must be set so as to establish the communication with the equipments:

Equipment	Necessary Accessories	Connection Type	Driver	Parameter	Description	Recommended or Factory Setting
CFW300	CFW300 with CFW300-CRS232 module	RS232	- - - -	0308	Serial Address	1
				0310	Serial Baud Rate	1 = 19200 bits/s
				0311	Serial Byte Configuration	1 = 8 bits, even parity, 1 sb
				0312	Serial Protocol	2 = Modbus RTU
	CFW300 with CFW300-CRS485 module	RS485	USB drive supplied by the manufacturer of the Isolated USB to 485 Converter (external device)	0308	Serial Address	1
				0310	Serial Baud Rate	1 = 19200 bits/s
				0311	Serial Byte Configuration	1 = 8 bits, even parity, 1 sb
				0312	Serial Protocol	2 = Modbus RTU
	CFW300 with CFW300-CUSB module	USB Serial Port	FTDI	0308	Serial Address	1
				0310	Serial Baud Rate	1 = 19200 bits/s
				0311	Serial Byte Configuration	1 = 8 bits, even parity, 1 sb
				0312	Serial Protocol	2 = Modbus RTU
CVW500	- - - -	RS232	- - - -	0308	Serial Address	1
				0310	Serial Baud Rate	1 = 19200 bits/s
				- - - -	Serial Byte Configuration	8 data bits, even parity, 1 stop bit
				- - - -	Serial Protocol	Modbus RTU
SCA06	- - - -	USB	WEG USB Driver	- - - -	- - - - -	- - - - -
PLC300	- - - -	USB	WEG USB Driver	- - - -	- - - - -	- - - - -
SSW900	- - - -	USB	WEG USB Driver	- - - -	- - - - -	- - - - -

7.2 Establishing Communication - USB Serial Port

The USB serial port is known as Virtual Com Port.

The equipment must be closer than three meters to the computer.

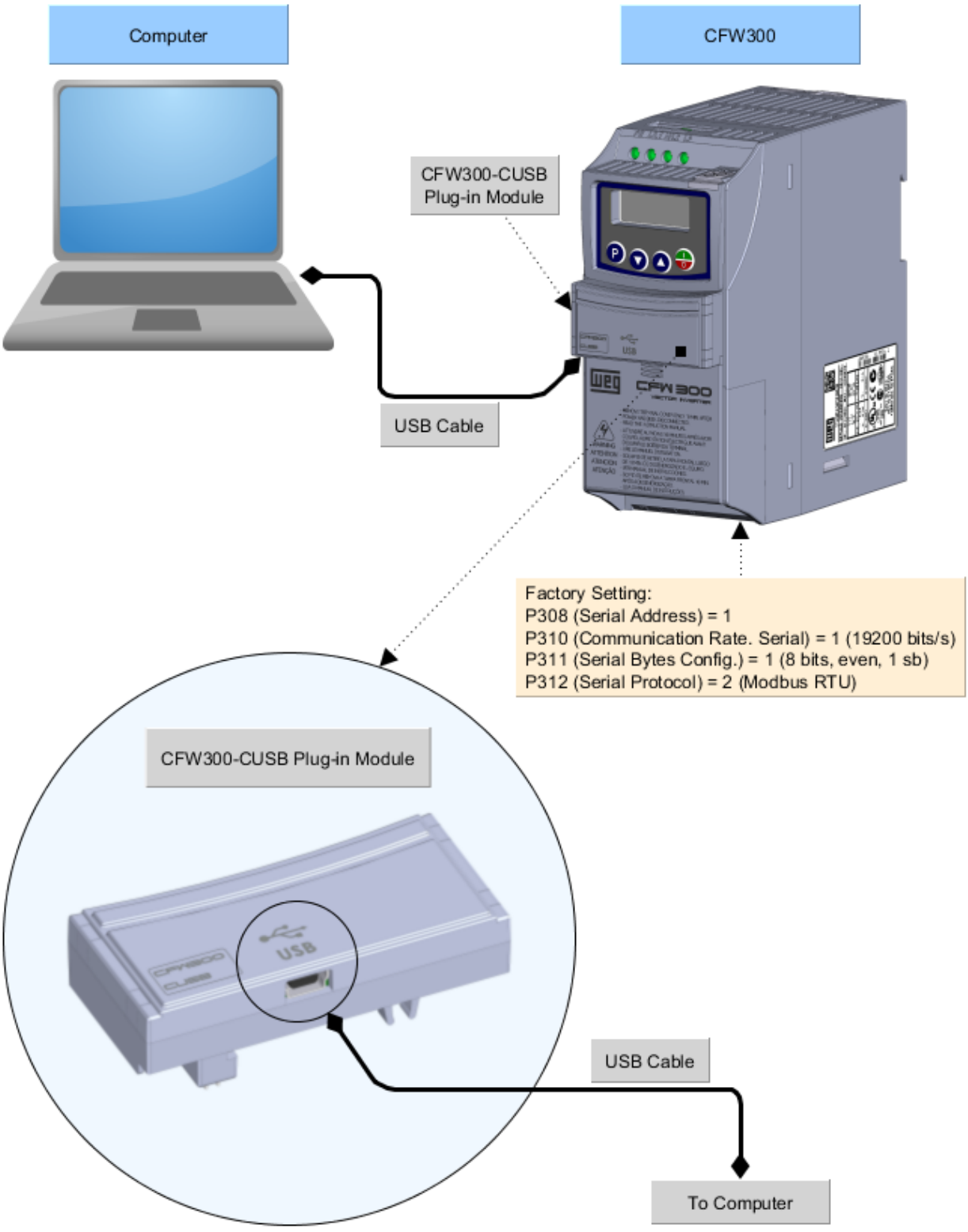
This connection is done by means of USB Cable.

USB Driver: it is necessary to install the USB FTDI drive available in the folder USB_Driver\FTDI.

CFW300

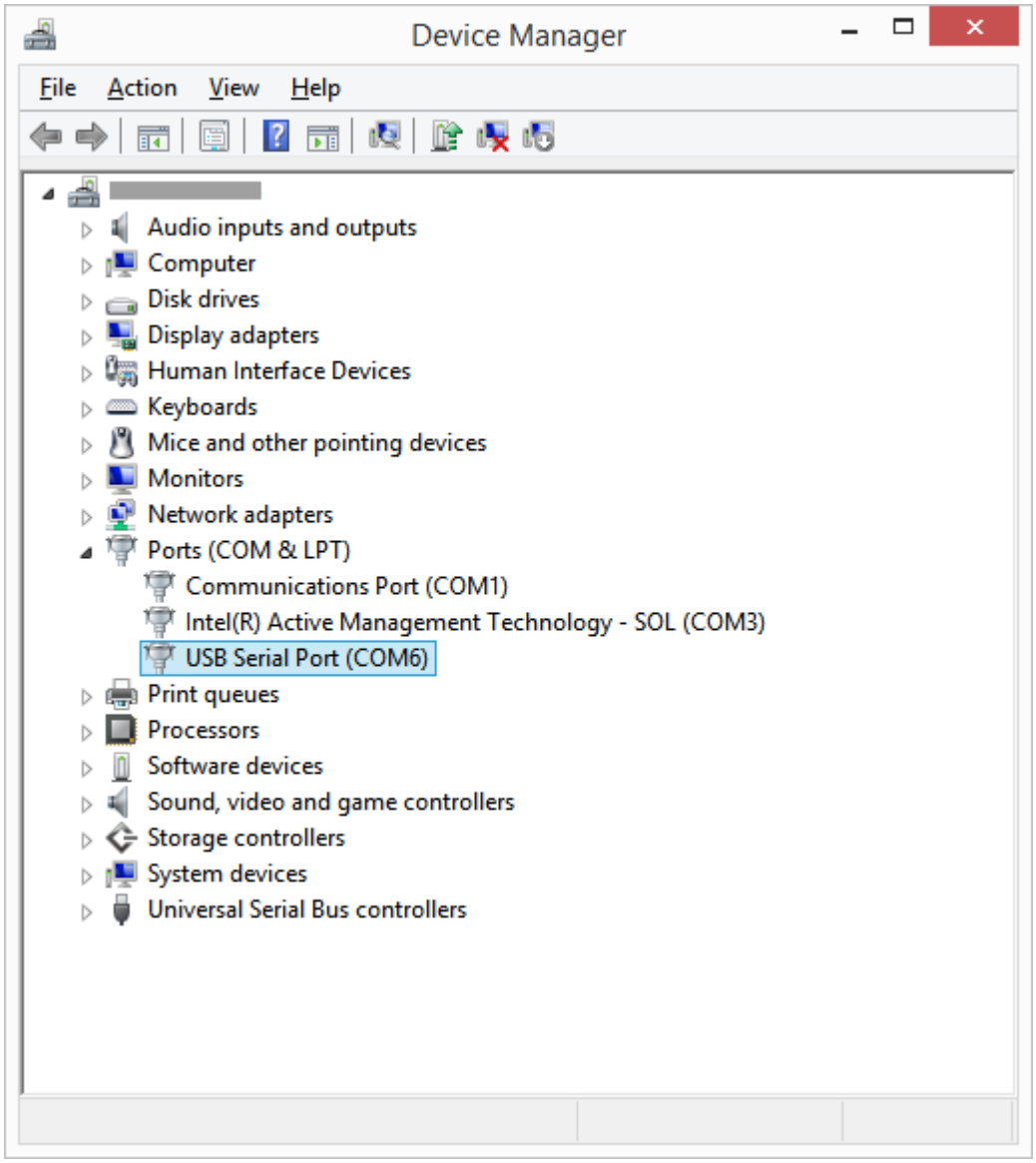
1) Connection of the Computer to the Equipment

The figure below shows how to connect a computer to the equipment via USB.



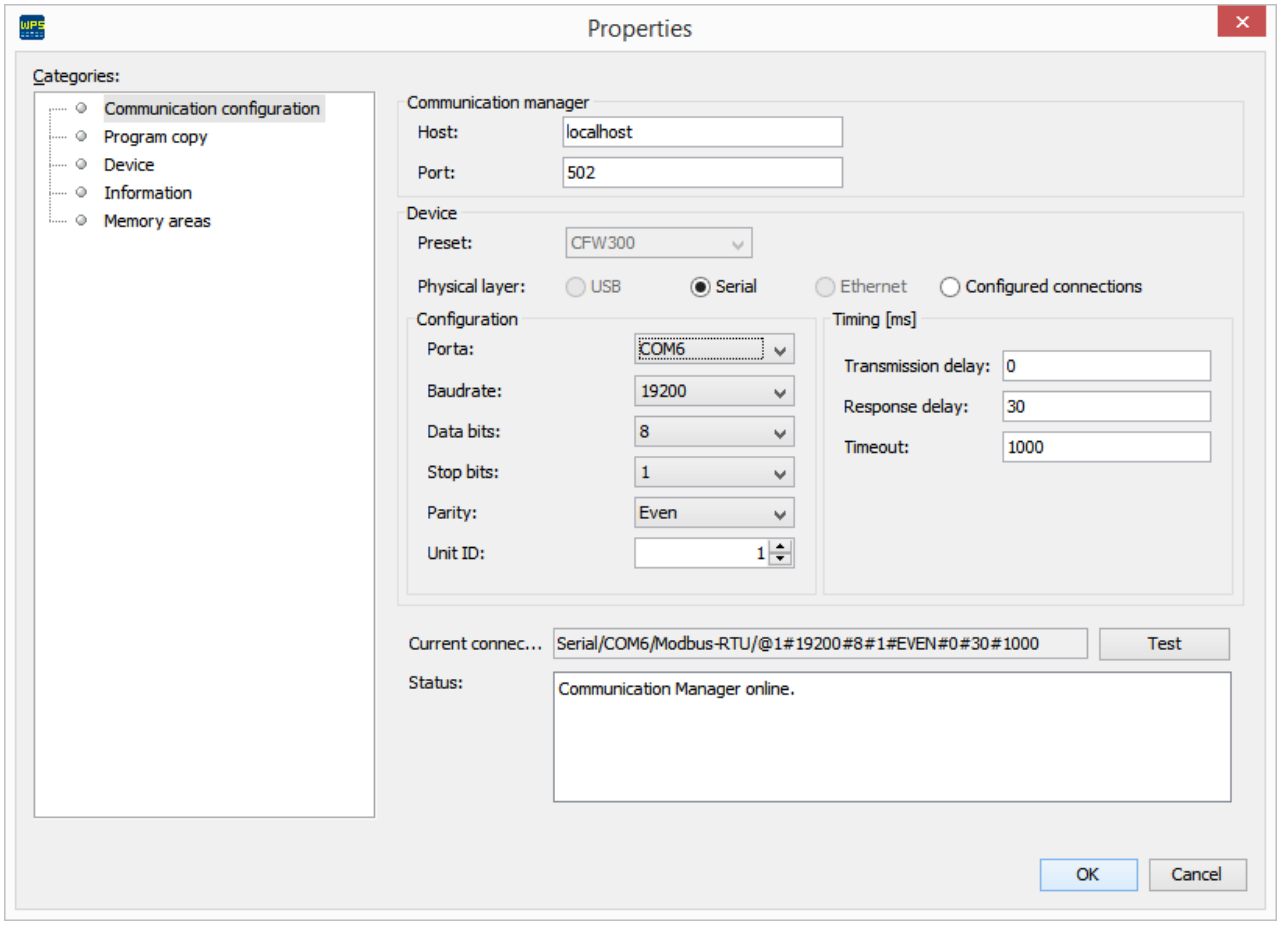
2) Windows Device Manager

The Windows device manager indicates the serial port connected to the equipment.
The computer name on the device manager is illegible on purpose.



3) Communication Configuration on the WPS

On the WPS, select the serial port correctly in the window Property > Communication Configuration as follows.



NOTE!
Turn off the equipment before making the connections.

4) Connecting the Equipment

1. Insert the mini-B connector of the USB cable into the USB connector of the equipment;
2. Insert the A-type connector of the USB cable into the computer USB port;
3. In the device manager, check which serial port is connected;
4. Make sure that the serial connection is selected in the Communication Configuration category of the resource Property window;
5. The serial port and its resource configuration on the WPS must be the same serial port that appears in the Windows device manager where the USB cable is connected;
6. Never change the values in parameters P308, P310, P311 and P312 during a connection. Changing those parameters causes the immediate loss of communication between the PC and the equipment.

7.3 Establishing Communication - RS232

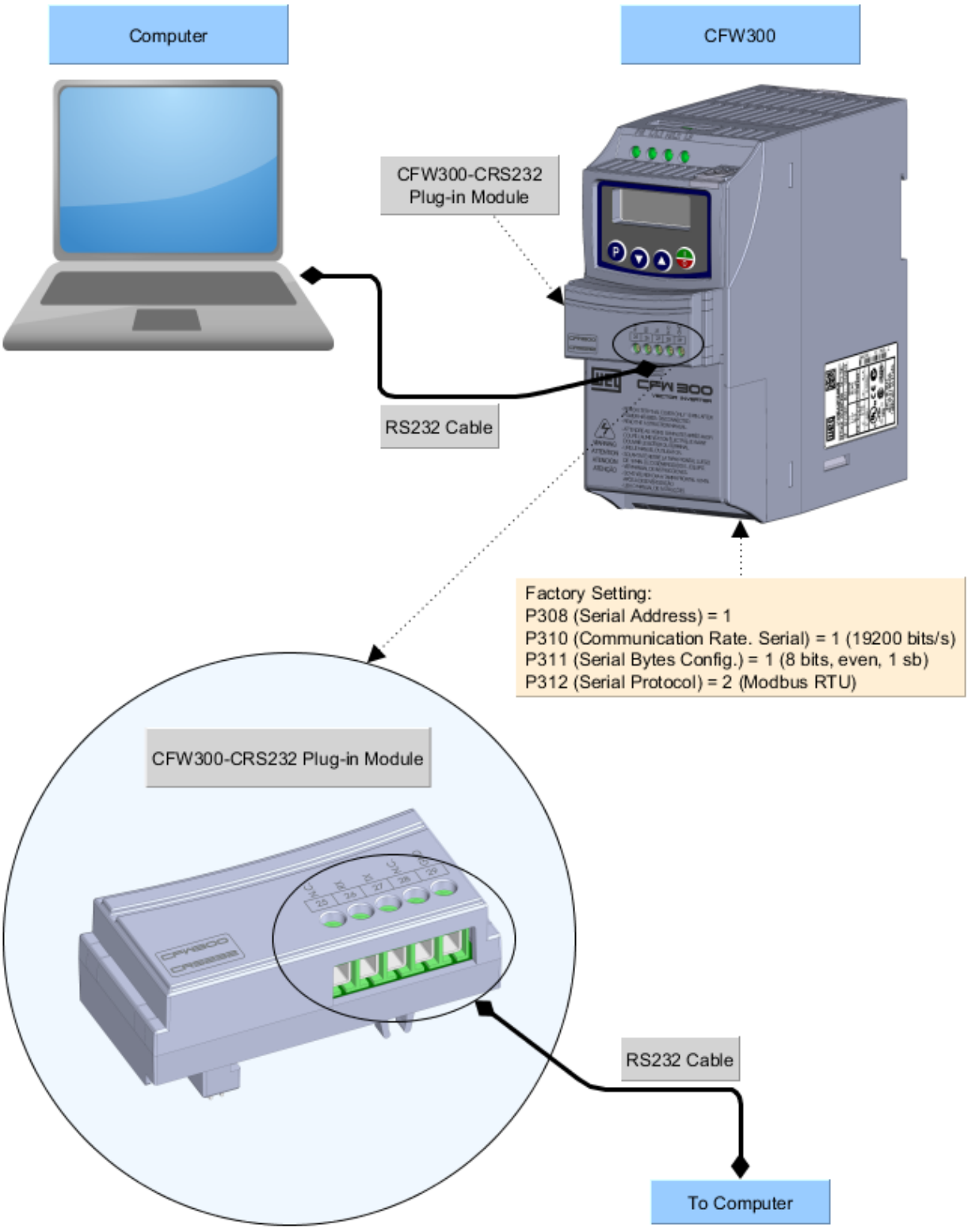
The equipment must be at a certain distance from the computer according to the table below.

Baud Rate (bps)	Maximum Cable Size (ft)	Maximum Cable Size (m)
9600	32.81	10
19200	24.93	7.6
38400	12.14	3.7

CFW300

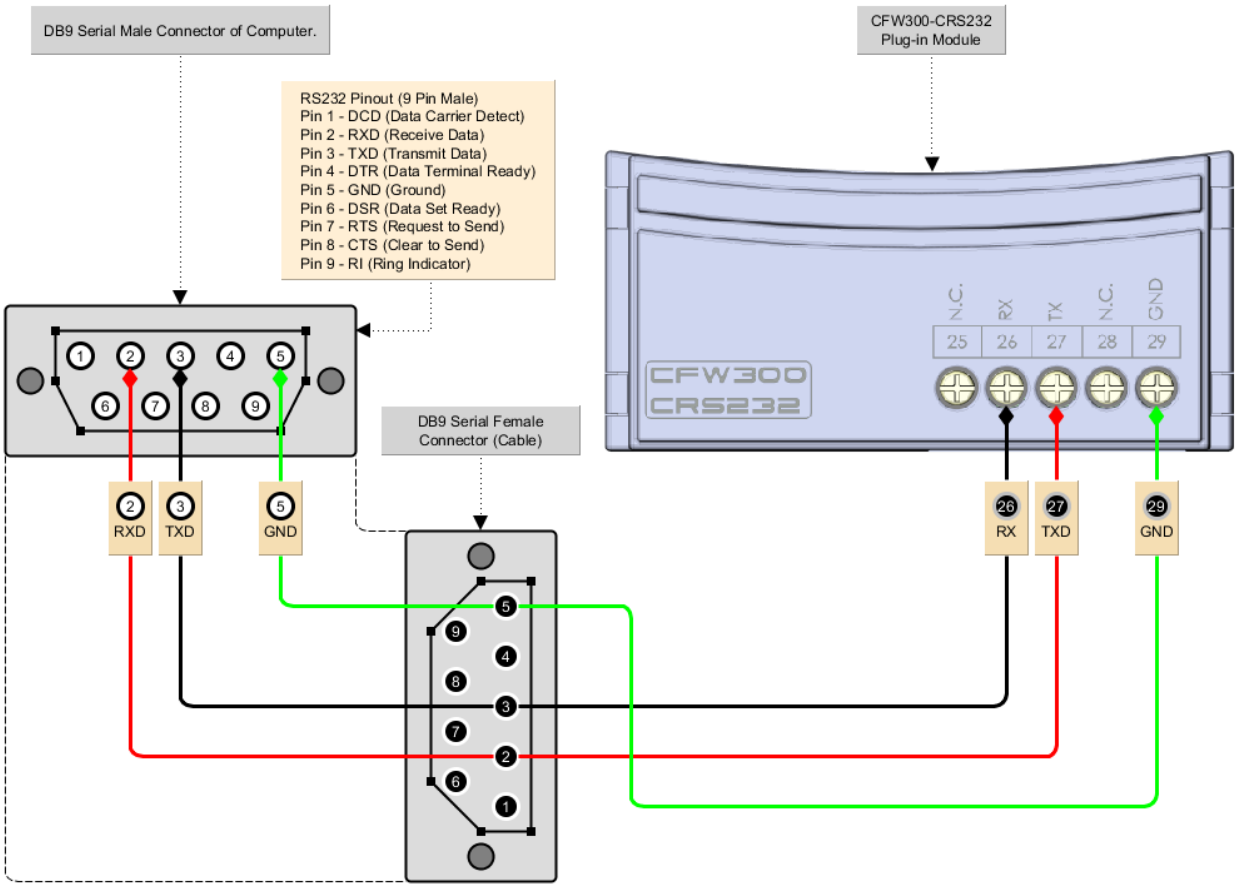
1) Connection of the Computer to the Equipment

The figure below shows how to connect a computer to the equipment via RS232.



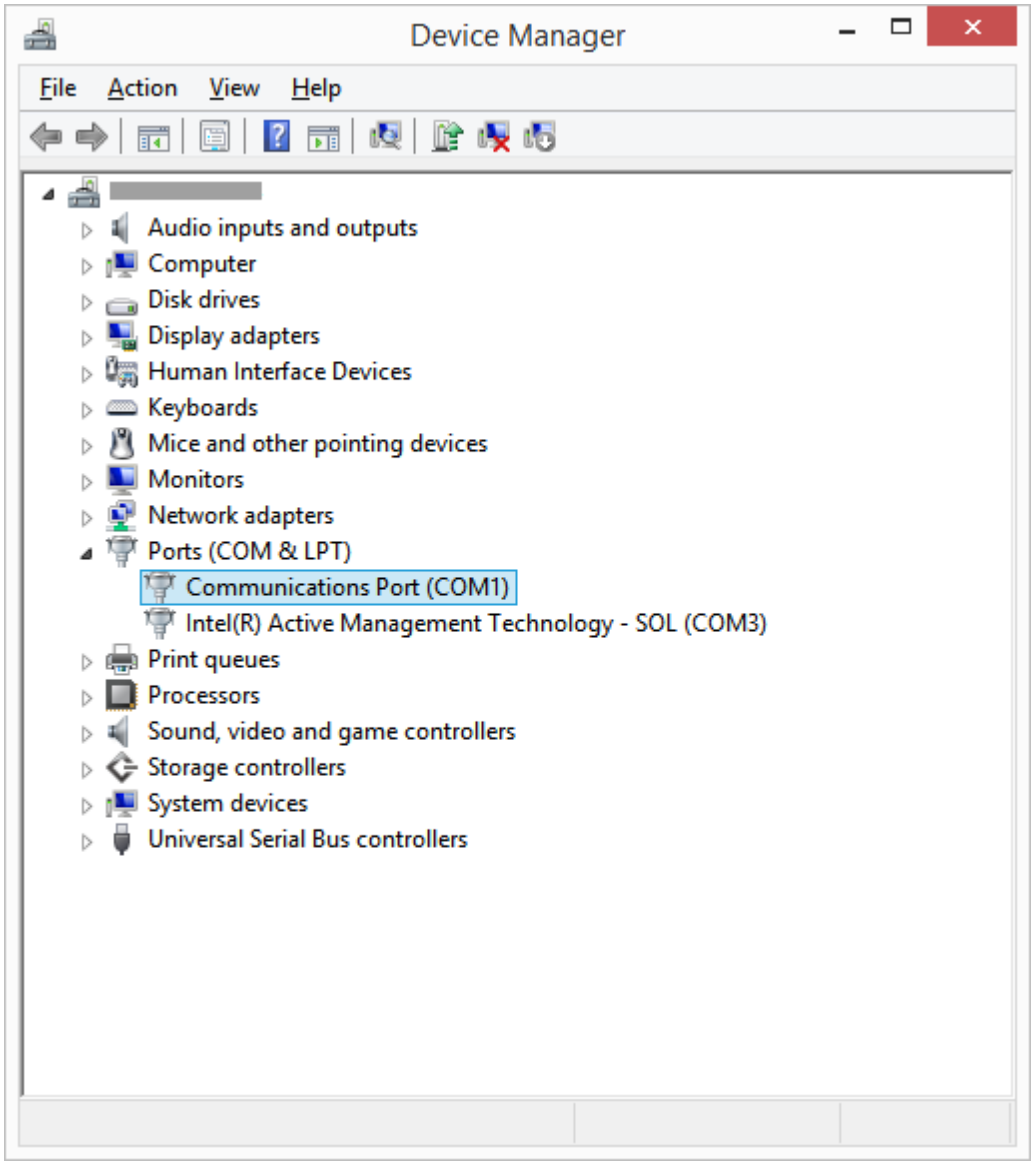
2) Computer - Plug-In Module Connection

The figure below shows details of the connection.



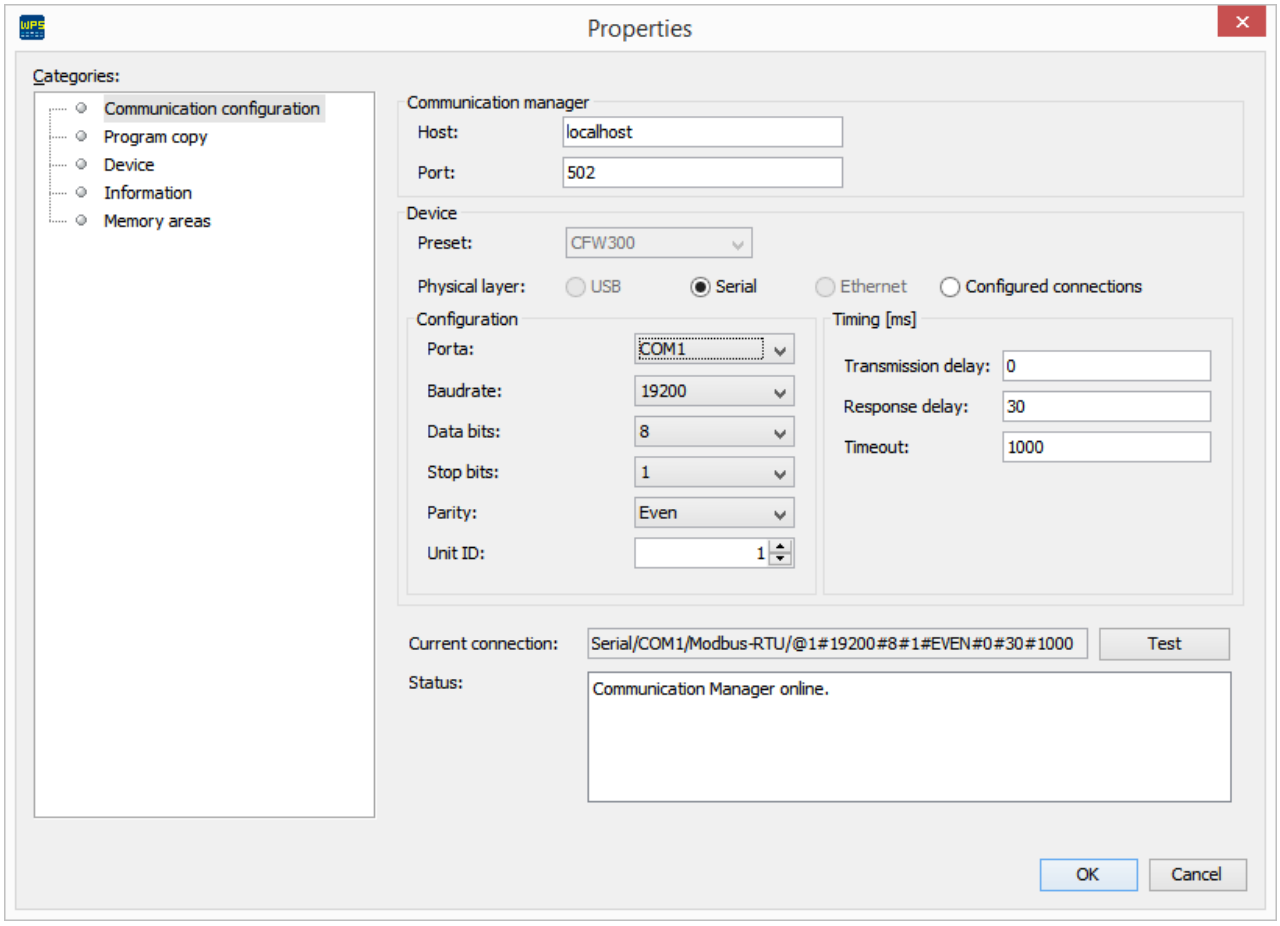
3) Windows Device Manager

The Windows device manager indicates the serial port connected to the equipment.
The computer name on the device manager is illegible on purpose.



4) Communication Configuration on the WPS

On the WPS, select the serial port correctly in the window Property > Communication Configuration as follows.



NOTE!
Turn off the equipment before making the connections.

5) Connecting the Equipment

1. Make the connections of the serial DB9 female connector of the RS232 cable to the CFW300-RS232 accessory as shown in the previous figures;
2. Insert the DB9 female connector of the RS232 cable into the DB9 male connector of the computer;
3. In the device manager, check which serial port is connected;
4. Make sure that the serial connection is selected in the Communication Configuration category of the resource Property window;
5. The serial port and its configuration of the resource in the WPS must be the same serial port that appears in the Windows device manager where the RS232 cable is connected;
6. Never change the values in parameters P308, P310, P311 and P312 during a connection. Changing those parameters causes the immediate loss of communication between the PC and the equipment.

7.4 Establishing Communication - RS485

The equipment must be at a certain distance from the computer according to the table below.

Baud Rate (bps)	Maximum Cable Size (ft)	Maximum Cable Size (m)
9600	3280.84	1000
19200	3280.84	1000
38400	3280.84	1000

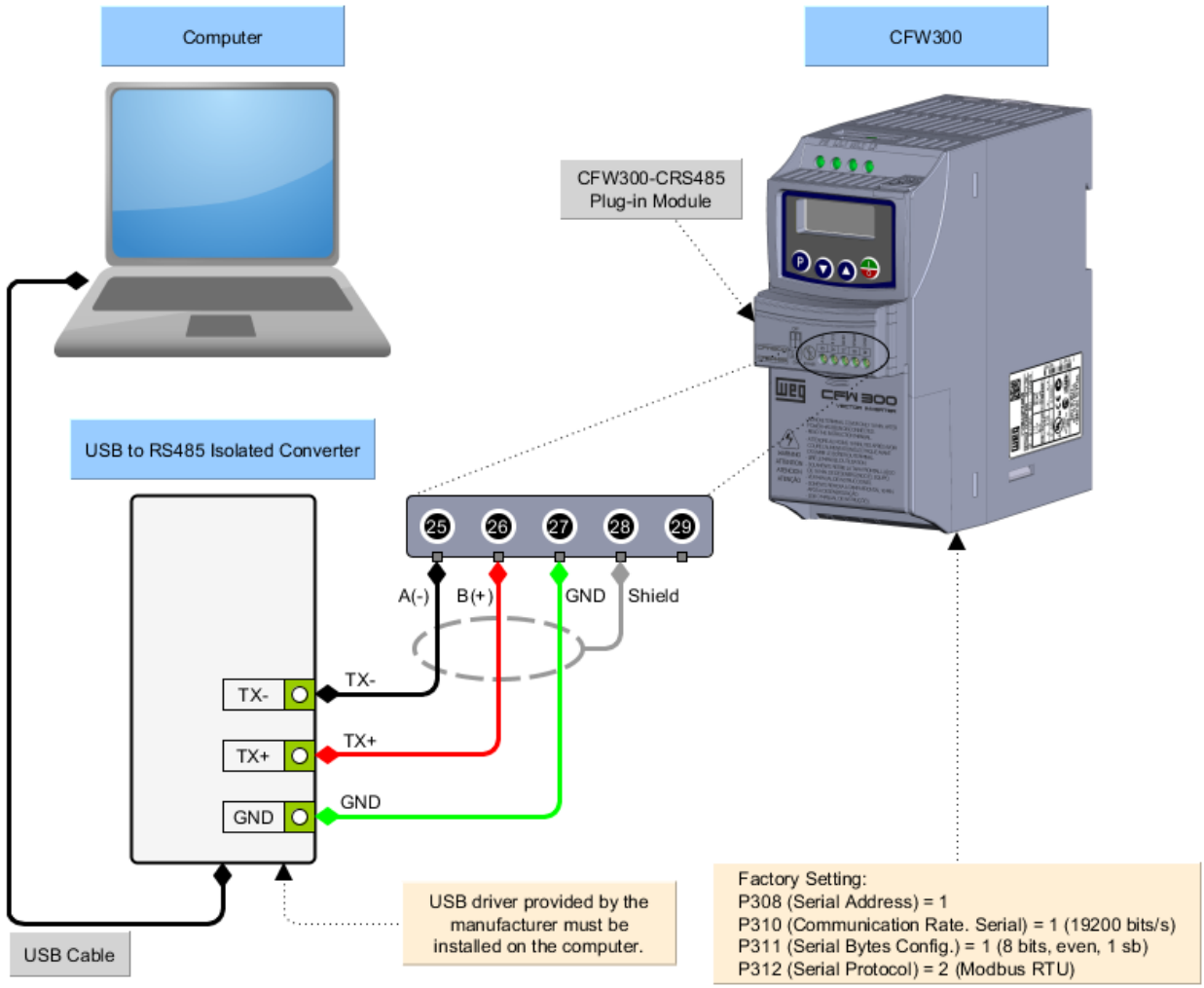
The connection of the computer to the isolated USB to RS485 converter is done by means of the USB Cable.

In this case, it is necessary to install the USB driver of the isolated converter, normally supplied by the manufacturer of the converter.

CFW300

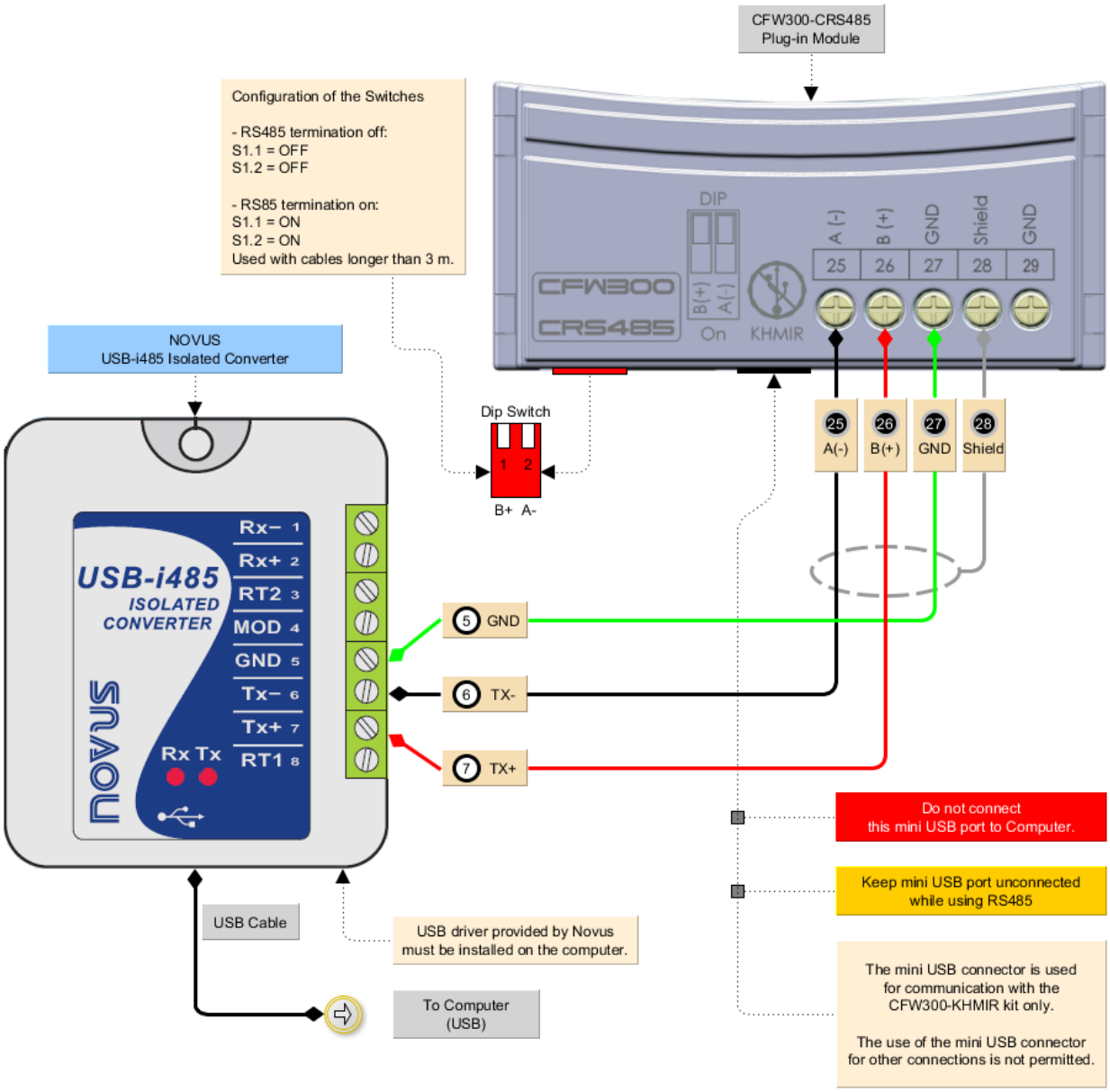
1) Connection of the Computer to the Equipment

The figure below shows how to connect a computer to the equipment via RS485.



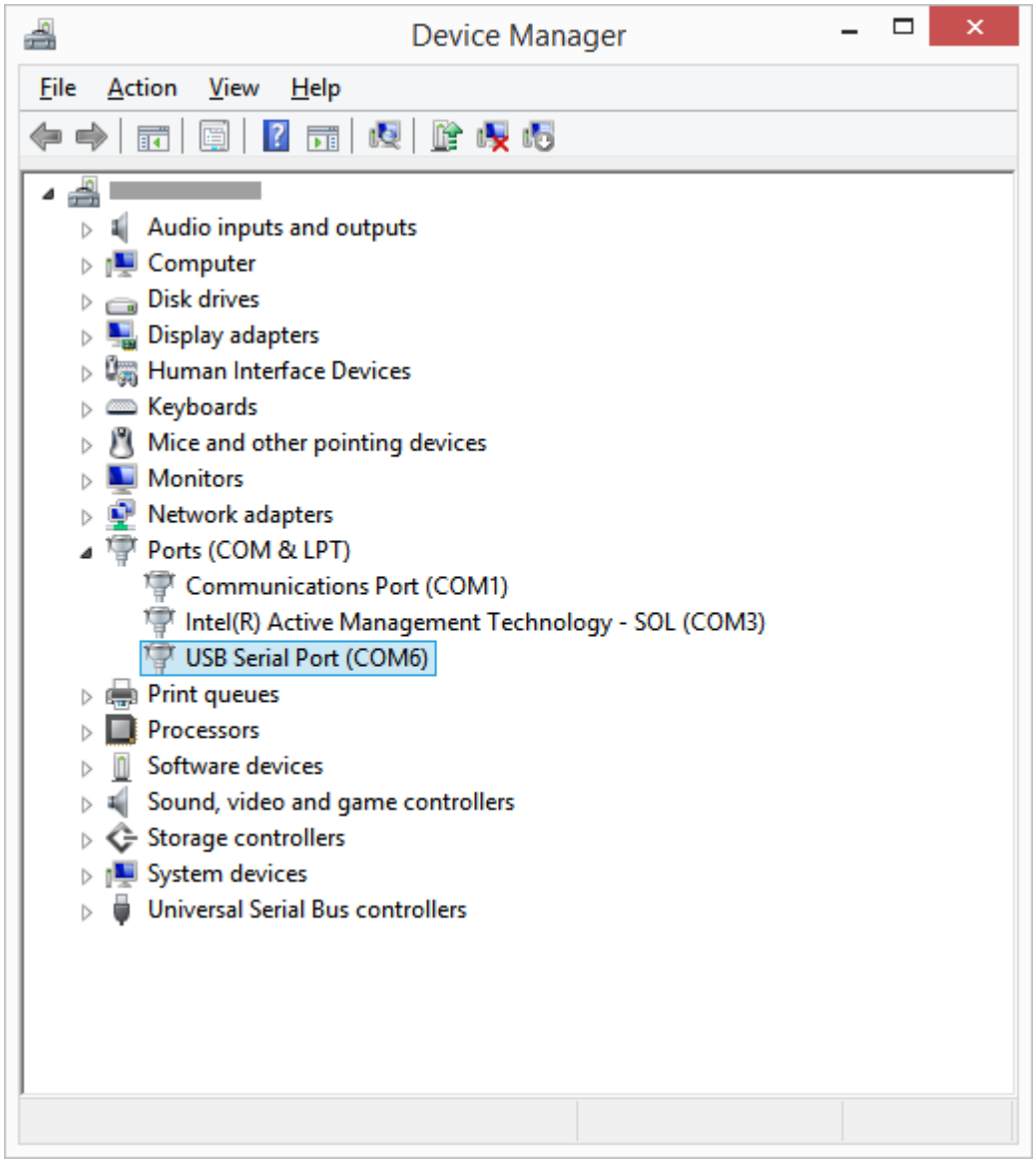
2) Computer - Plug-In Module Connection

The figure below shows an example using the Novus USB-i485 converter.



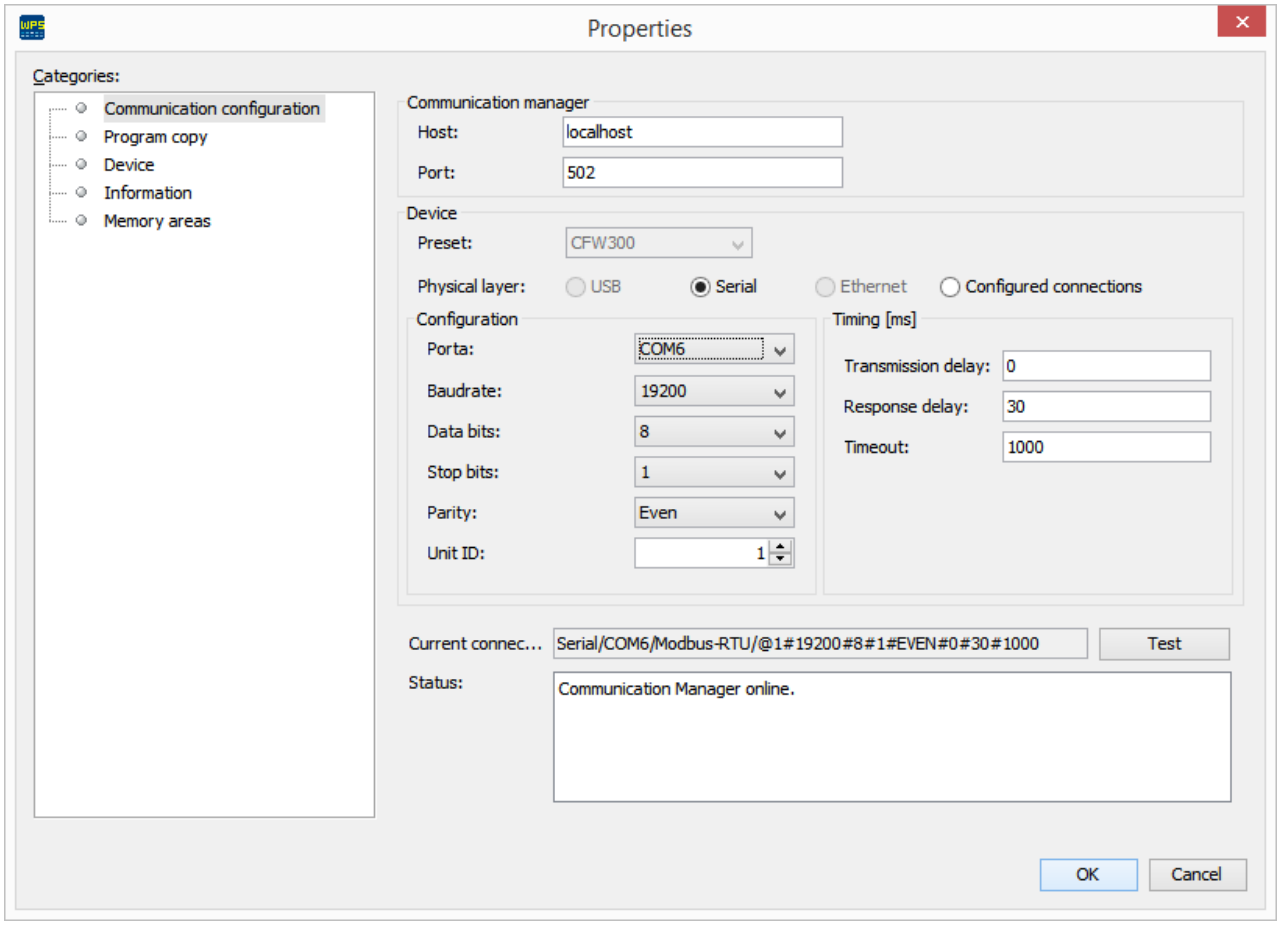
3) Windows Device Manager

The Windows device manager indicates the serial port connected to the device.
The computer name on the device manager is illegible on purpose.



4) Communication Configuration on the WPS

On the WPS, select the serial port correctly in the window Property > Communication Configuration as follows.



NOTE!
Turn off the equipment before making the connections.

5) Connecting the Equipment

1. Make the connections from the connector of the isolated USB RS485 converter to the CFW300-RS485 accessory as shown in the previous figures;
2. Insert the USB cable into the connector of the isolated USB RS85 converter and into the computer USB port;
3. In the device manager, check which serial port is connected;
4. Make sure that the serial connection is selected in the Communication Configuration category of the resource Property window;
5. The serial port and its resource configuration on the WPS must be the same serial port that appears in the Windows device manager where the USB cable is connected;
6. Never change the values in parameters P308, P310, P311 and P312 during a connection. Changing those parameters causes the immediate loss of communication between the PC and the equipment.

7.5 Cables

The figure below details the cable for the point-to-point USB connection.

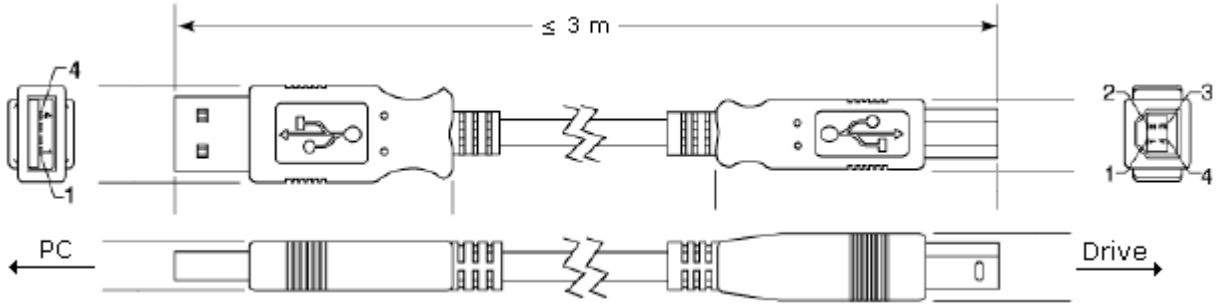


Figure 1: USB Cables

The figure below shows the connectors.

Figure 2: USB Connectors

A = Type A Connector
B = Type B Connector

Maximum cable size: 3 meters.

NOTE!
Always use a USB shielded interconnection cable, standard host/device shielded USB cable. Unshielded cables may generate communication errors.

NOTE!
The USB connection is galvanically isolated from the power grid and other elevated voltages internal to the drive. The USB connection, though, is not isolated from the protection earthing (PE). Use an isolated laptop to connect to the USB connector, or a desktop connected to the same protection earthing (PE) as the drive.

Purchasing Suggestions

Manufacturer:
- Samtec, Inc: http://www.samtec.com

If you wish to purchase a USB cable directly from Samtec, please see below:

Description	Item
High speed shielded USB cable Revision 2.0, 1 m, Samtec	USBC-AM-MB-B-B-S-1
High speed shielded USB cable Revision 2.0, 2 m, Samtec	USBC-AM-MB-B-B-S-2
High speed shielded USB cable Revision 2.0, 3 m, Samtec	USBC-AM-MB-B-B-S-3

At the time this manual has been written, the specification could be found at
http://www.samtec.com/ftppub/cpdf/USBC-AM-BM-B-B-S-X-MKT.pdf

7.6 USB/Serial Converter

USB/Serial Converters are the best solution for those who wish to connect a serial equipment (RS232) to USB ports. It is a low cost solution that solves the need to install new serial ports in microcomputers with all busy busbars, or in an equipment (laptop) that does not have RS232 ports.

The USB/SERIAL converter allows a plug & play connection with your microcomputer, leaving the existing serial port free.

Purchase Suggestions in Brazil

Manufacturer: Comm5 Tecnologia
Product: USB Converter for 1 saída serial RS232
Model: 1S-USB - USB converter for 1 serial
Web Site: http://www.comm5.com.br/1S-USB--CONVERSOR-USB-PARA-1-SERIAL
Refer to the user manual for more information about installation.

Manufacturer: Tripp Lite
Product: USB High Speed Serial Adapter
Model: USA-19HS
Web Site: http://www.tripplite.com/en/products/model.cfm?txtSeriesID=849&txtModelID=3914
Refer to the user manual for more information about installation.

8 Ladder

8.1 Concepts

8.1.1 Introduction

Ladder Programming is the graphical representation of boolean equations combining contacts (input arguments) with coils (output results).

Ladder programming enables testing and modifying data by standard graphical symbols. These symbols are positioned in the Ladder diagram in a way that is similar to a line of a logic diagram with relays. The Ladder diagram is delimited on the left and on the right by busbar lines.

Graphical Components

The basic graphical components of a Ladder diagram are shown below.

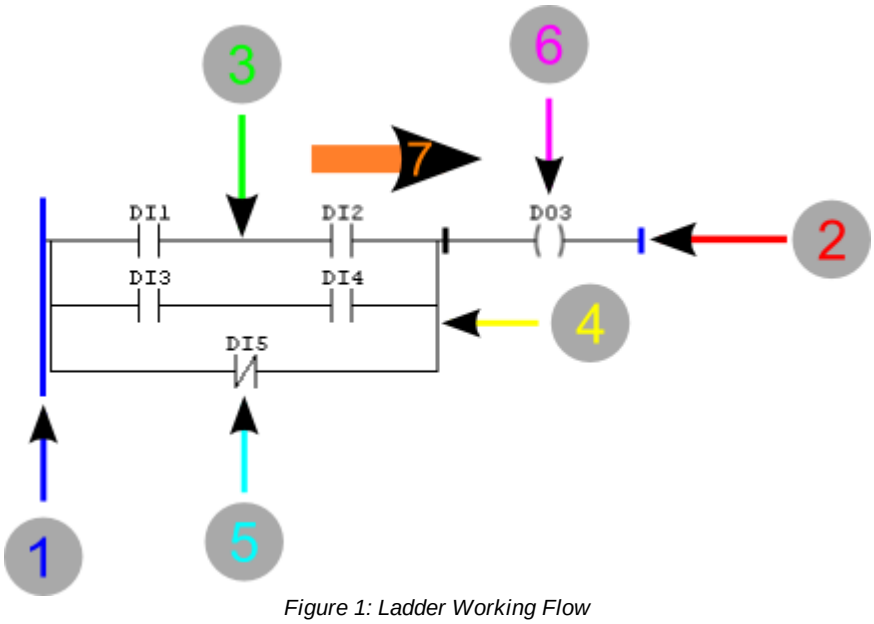


Figure 1: Ladder Working Flow

1. Left busbar
2. Right busbar
3. Horizontal connection
4. Vertical connection
5. Contact
6. Coil
7. Power flow

Busbars

The editor is delimited on the left by a vertical line known as left busbar, and on the right by a vertical line known as right busbar.

Connection Elements and States

The connection elements may be horizontal or vertical. The state of the connection elements may be denoted by 1 or 0, corresponding to the literal Boolean value 1 or 0, respectively. The term connection state must be synonymous with the term power flow.

The state of the left busbar may always be considered 1. No states are defined on the right busbar.

The horizontal connection element must be indicated by a horizontal line. A horizontal connection element transmits the state of the element immediately on the left to the element immediately on the right.

The vertical connection element must consist of vertical lines intersected by one or more horizontal connections on each side.

The state of the vertical connection must represent the logical OR of states 1 of horizontal connections on the left side, i.e. the state of the vertical connections must be:

- 0 if the state of all horizontal connections included on its left is 0,
- 1 if the state of one or more horizontal connections included on its left is 1.

The state of the vertical connections must be copied to all associated horizontal connections on its right.

The state of vertical connections must not be copied to associated horizontal connections on its left.

Execution Control

The following figure shows how the program in Ladder is executed. The card continually runs a Scanning cycle. The cycle begins when the I/O System in the hardware compiles the last values of all input signals and records those values in fixed areas of the memory.

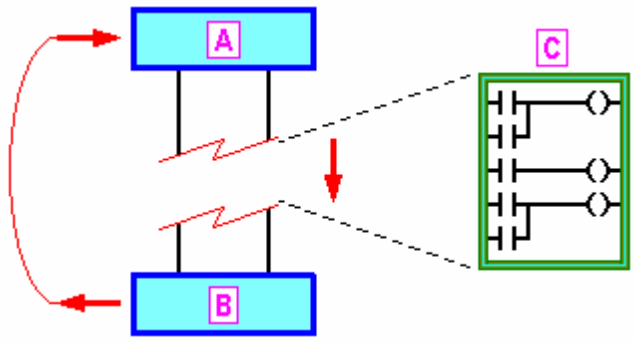


Figure 2: Execution Control

A - Inputs read to the memory
B - Memory written in the Outputs
C - Scanning of Ladder lines

The lines in the Ladder program are then run in a fixed order, starting with the first line. During program scanning, new values for physical outputs, as determined by the logic of several Ladder lines, are initially written to an area of the output memory. Finally, when the Ladder program has finished running, all output values retained in the memory are inscribed in the physical outputs by the hardware in a single operation.

8.1.2 Legend

AT: Direct representation of a variable.

FUNCTION BLOCK: It is a function that needs an instance.

CONFIGURATION: It is the organization of a software application in a higher level. It may contain Resources within a Configuration.

FUNCTION: It is a block responsible for executing a certain behavior or an action based on possible parameters (VAR_IN, VAR_IN_OUT, VAR_OUT).

INSTANCE: Memory area taken according to the Functional Block.

LD (Ladder Diagram): Graphical language based on the electric diagrams (interconnected contacts and coils), according to the power flow between the elements.

OVERFLOW: It occurs when the result of a mathematical calculation exceeds the limits permitted for the result data type.

POU: Program Organization Unit. It may be: Program, Functional Block or Function.

PROGRAM: It is the logical grouping of all programming elements and constructions necessary for processing the signals required to control a machine or process.

RESOURCE: They consist of any element with processing capacity responsible for executing the programs.

SCAN: Scan cycle of a program.

STACK: Stack of the Ladder program. It is the memory area used to perform the Program Logics.

TAG: Variable Name.

TASK: Responsible for the execution of programs, in a periodical or triggered way, with trigger by event.

DATATYPE: It informs the compiler the space taken by a variable and its respective format (to the Blocks).

VARIABLE: It consists of a memory position able to withhold and represent a value or expression. It may have scope:
- **Location**: whose automatic position is calculated by the compiler.
- **Global**: located in a determined memory area with digital inputs and outputs, it may be accessed at any point of the Configuration.

The variable may be:
- **Retentive**: it stores the value after powering down the device;
- **Volatile**: it begins with the value contained in the initial value field after the powering down of the device;
- **Constant**: it does not allow the modification of its content.

VAR: Variables for internal use of a User's Block (USERFB). Equivalent to a variable of Local scope.

VAR_IN: Input argument of a User's Block (USERFB). Variables configured in this field will be only read on the USERFB.

VAR_IN_OUT: Input and output argument of a User's Block (USERFB). It will not allow CONSTANT variables, PHYSICAL INPUTS (%I_), NETWORK INPUTS (%I_) or READING SYSTEM MARKERS (%S_), because they will be read at the beginning of the USERFB and will be updated with new values (written) at the end of the USERFB.

VAR_OUT: Output argument of a User's Block (USERFB). It will not allow CONSTANT variables, PHYSICAL INPUTS (%I_), NETWORK INPUTS (%I_) or READING SYSTEM MARKERS (%S_), because they will be updated with new values (written) at the end of the USERFB.

WATCHDOG: It is a way provided by the manufacturer to perform specific actions if the integrity of the system is violated.

8.1.3 Contact Logic

AND LOGIC – Contact in Series

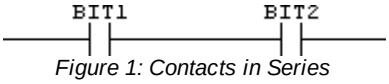


Figure 1: Contacts in Series

The figure above performs an AND Logic between the two last elements charged on the STACK, lowers a level of the STACK, and places the result on top of the STACK. That means that the following Boolean operation is performed: top of the STACK = BIT1.BIT2.

In IL (Instruction List) language it becomes like this:

```
LD      BIT1    (* loads the value of variable BIT1 to the STACK *)
LD      BIT2    (* loads the value of variable BIT2 to the STACK *)
AND             (* Performs AND Logic between BIT1 and BIT2 through the STACK *)
```

Truth Table

BIT1	BIT2	STACK
0	0	0
0	1	0
1	0	0
1	1	1

OR LOGIC – Contact in Parallel

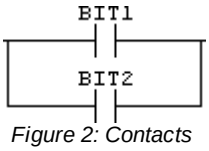


Figure 2: Contacts in Parallel

The figure above performs an OR Logic between the two last elements charged on the STACK, lowers a level of the STACK, and places the result on top of the STACK. That means that the following Boolean operation is performed: top of the STACK = BIT1 + BIT2.

In IL (Instruction List) language it becomes like this:

```
LD      BIT1    (* loads the value of variable BIT1 to the STACK *)
LD      BIT2    (* loads the value of variable BIT2 to the STACK *)
OR              (* Performs OR logic between BIT1 and BIT2 through the STACK *)
```

Truth Table

BIT1	BIT2	STACK
0	0	0
0	1	1
1	0	1
1	1	1

8.1.4 Data types

Data type	Size	Signal	Range
BOOL	1 bit		0 or 1
BYTE	8 bits (1 byte)		0 to 255
USINT	8 bits (1 byte)		0 to 255
SINT	8 bits (1 byte)	Yes	-128 to 127
WORD	16 bits (2 bytes)		0 to 65535
UINT	16 bits (2 bytes)		0 to 65535
INT	16 bits (2 bytes)	Yes	-32768 to 32767
DWORD	32 bits (4 bytes)		0 to (232 - 1)
UDINT	32 bits (4 bytes)		0 to (232 - 1)
DINT	32 bits (4 bytes)	Yes	- 231 to (231 - 1)
LWORD	64 bits (8 bytes)		0 to (264 - 1)
ULINT	64 bits (8 bytes)		0 to (264 - 1)
LINT	64 bits (8 bytes)	Yes	- 263 to (263 - 1)
REAL	32 bits (4 bytes) Floating point - IEEE 559 standard	Yes	±10±38; Precision = 2-23
LREAL	64 bits (8 bytes) Floating point - IEEE 559 standard	Yes	±10±308; Precision = 2-52
STRING	8 bits (1 byte) for each position + 8 bits (1 byte) for the null character		1 to 254 ASCII characters

NOTE!
The data type STRING considers only the number of characters in the size of the variable. The byte occupied by the null terminator is not added to the size.

8.1.5 Direct Representation

Used to define the memory position of a Global Variable.

Syntax: *%<Format><Size>*

First Letter:

At (Format)	Description
I	Input: It receives the values from analog and discrete variables, or input network variables.
Q	Output: It stores the values to be written in the analog and discrete outputs, or output network variables.
M	RAM memory internal marker.
S	System status marker
C	System command marker

Second Letter:

At (Format)	Description
X	Bit
B	Byte (8 bits)
W	Word (16 bits)
D	Double Word (32 bits)
L	Long Word (64 bits)

8.2 Editor

8.2.1 Desktop

Whenever a Ladder file is opened through its shortcut in the project tree, the variable/Ladder editor will show as per the figure below.

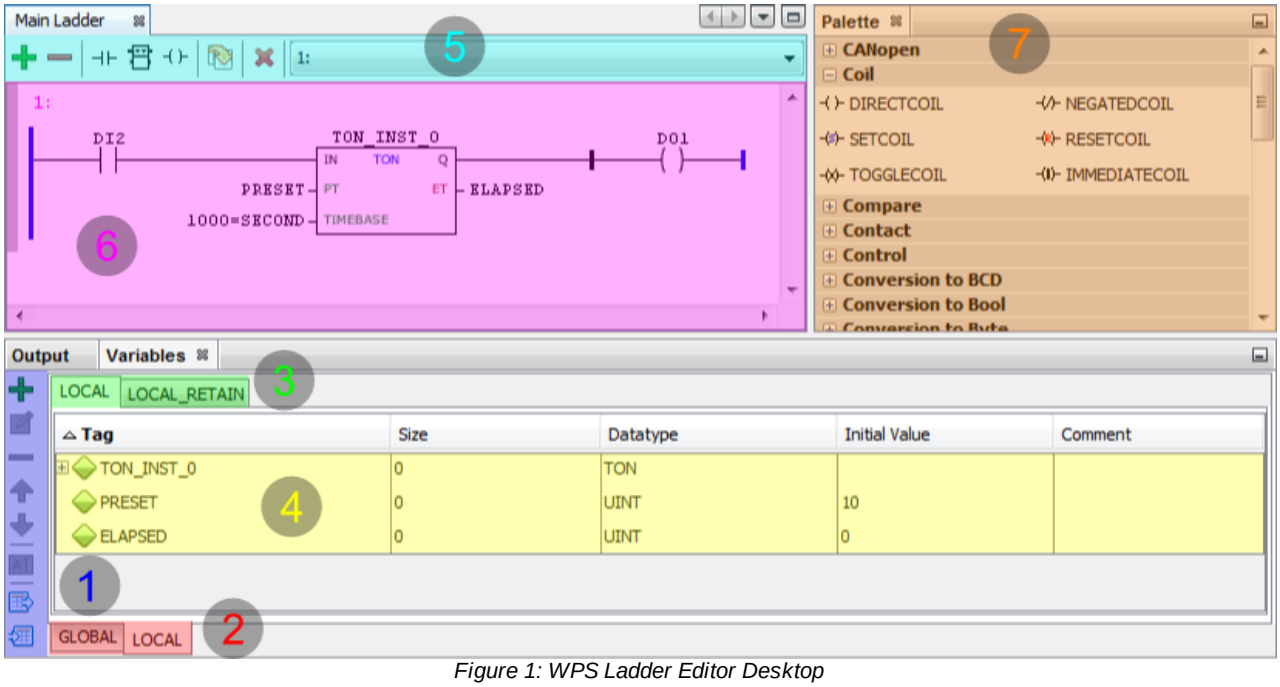


Figure 1: WPS Ladder Editor Desktop

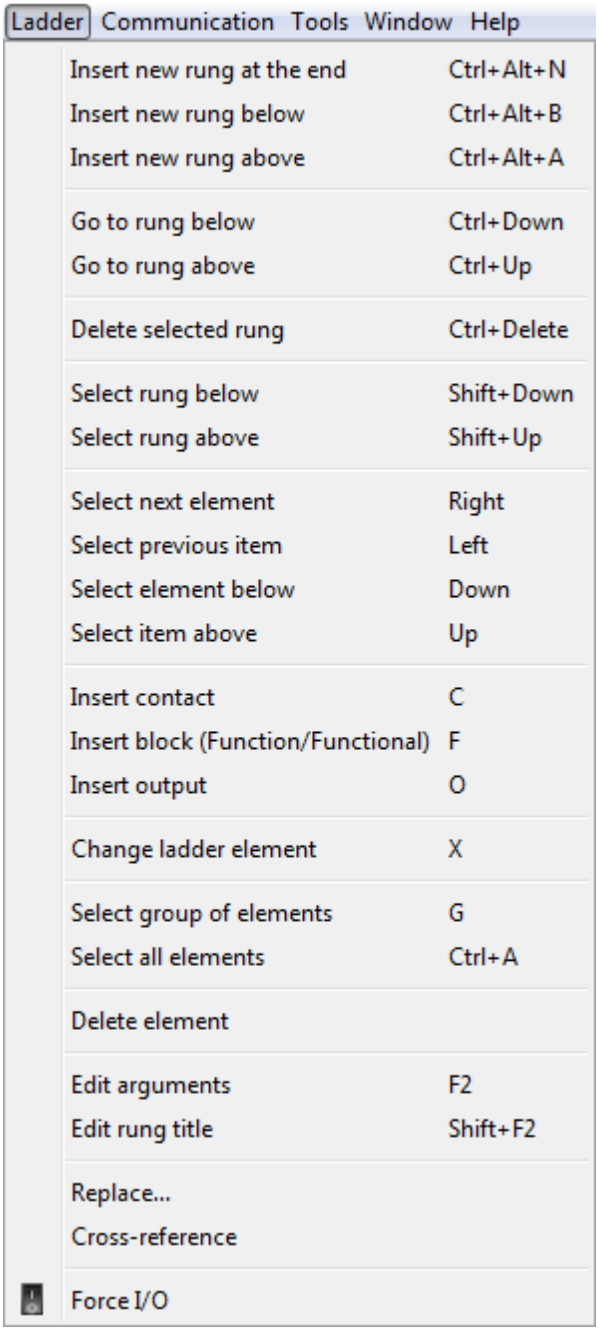
The variable/Ladder editor has the following components:

1. Toolbar to edit variables
2. Variable scope
3. Variable group
4. Variable list/editor
5. Program edition toolbar
6. Editor Ladder/rungs
7. Ladder component palette

NOTE!
If any docker is not visible, activate through the Palette item in Window menu.

8.2.2 Ladder Menu

When editing a Ladder file of your resource, the **Ladder** menu will be active as per the figure below.



Through this menu, it is possible to execute all the operations regarding the Ladder as well as to know the keyboard shortcuts for those operations.

8.2.3 Rungs

8.2.3.1 Overview

The Ladder program is edited through a graphical editor that organizes it through the rungs. Every rung in the Ladder program corresponds to an interlock that relates input elements to output elements. Said editor is shown in the following figure.

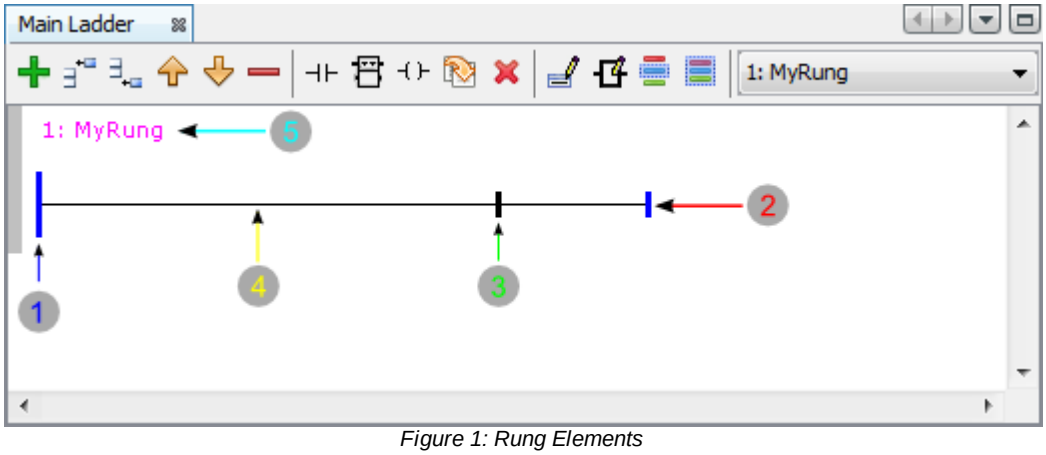


Figure 1: Rung Elements

The main elements of a rung are:
1. Left busbar
2. Right busbar
3. Output busbar
4. Connection between the elements
5. Rung title and comment

NOTE!
Outputs will always be connected to the right of the output busbar.

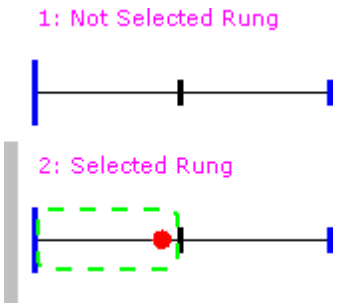
For further details on Ladder programming click here.

Through the Ladder/rungs editor tool bar, it is possible to perform the following operations:

- Add a new rung (end)
- Add a new rung (above)
- Add a new rung (below)
- Move *rung* up
- Move *rung* down
- Remove the selected rung
- Insert a contact in the rung
- Insert a functional block in the rung
- Insert an output coil in the rung
- Change Ladder element
- Delete an element in the rung (contact, functional block, or coil)
- Edit rung title
- Edit arguments
- Select element group
- Select all elements
- Select the rung for edition

8.2.3.2 Editing

In order to edit a rung, the rung must be selected. In order to select the rung, you may click on it with the mouse or use the rung selection control in the **Ladder Editor** toolbar. Whenever the rung is selected, there will be a grey bar on the left side of the rung as shown in the following figure.



8.2.3.3 Title and Comment

In order to edit the rung title and comment, just double click on the title and comment area with the mouse or press the **Shift**+**F2** keys. The following dialog will appear.

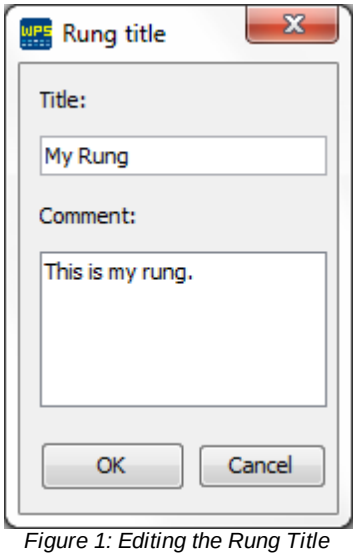


Figure 1: Editing the Rung Title

After editing the title and comment, the editor will become as follows.

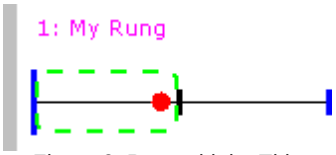


Figure 2: Rung with its Title

8.2.3.4 Inserting Elements

8.2.3.4.1 Overview

In order to insert an element in the rung, it is necessary that an already existing element is selected, as shown in the following figure.

The selected element will be involved by a green, striped rectangle. There will also be a red dot indicating where the new element will be inserted (insertion point).

The insertion of Ladder elements may be performed in three distinct ways.

1. With the keyboard through the following shortcuts

key: **C** - insert contact
key: **F** - insert functional block
key: **O** - insert output coil
key: **X** - change ladder element

2. With the Ladder/rungs editor tool bar

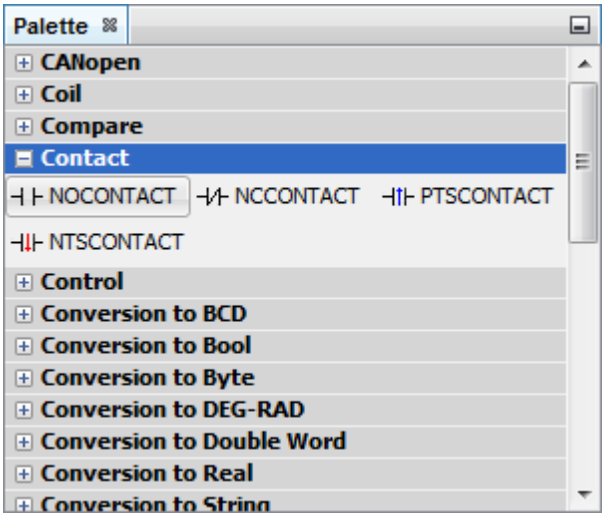
 - Insert a contact in the rung

- Insert a functional block in the rung

- Insert an output coil in the rung

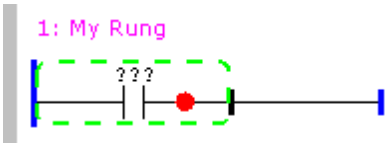
- Change ladder element

3. With the palette of Ladder elements



A mouse drag and drop operation must be used in order to insert elements using the palette. For that purpose, click on the palette element, keep the mouse pressed, move the mouse up to the insertion point in the Ladder, and release the mouse button.

After inserting an element, e.g. a contact, the rung will become as follows.



8.2.3.4.2 In Series

Through the insertion point of elements, it is possible to insert an element in series. For that purpose, use the following insertion points:

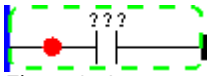


Figure 1 - Inserts an element in series before the selected element

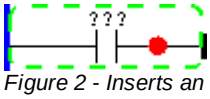


Figure 2 - Inserts an element in series after the selected element

8.2.3.4.3 In Parallel

Through the insertion point of elements, it is possible to insert an element in parallel. For that purpose, use the following insertion points:

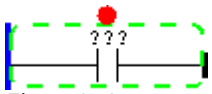


Figure 1 - Inserts an element in parallel above the selected element

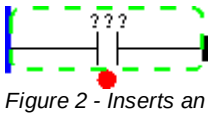


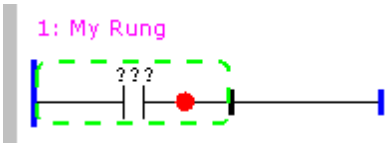
Figure 2 - Inserts an element in parallel under the selected element

8.2.3.5 Browsing

8.2.3.5.1 With the Keyboard

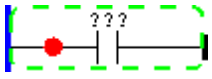
Browsing with the keyboard in the rung is done by the keys ←, →, ↑, ↓. Through these keys, it is possible to select several elements inside the rung, and also define the insertion point of new elements.

As shown in the item previous to insert an element of the contact type in the rung, it will become as follows.

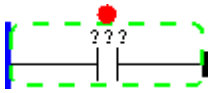


By pressing the browsing keys, we can modify the insertion point of a new element, as shown in the following figures:

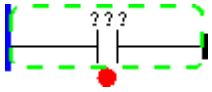
Key ← indicates that the new element will be inserted before the selected element



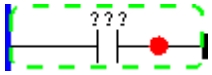
Key ↑ indicates that the new element will be inserted parallelly above the selected element



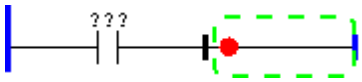
Key ↓ indicates that the new element will be inserted parallelly below the selected element



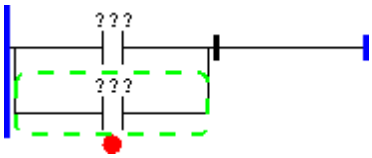
Key → indicates that the new element will be inserted after the selected element



As the insertion point has already been selected, and by pressing the same key as the corresponding direction, the selection will move on to the next element, as shown in the following figure.

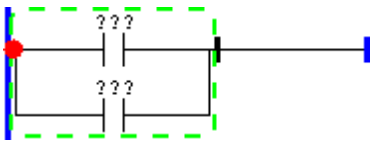


When inserting elements to the browser in parallel through the elements, we browse through each element individually, conforming to the following example.

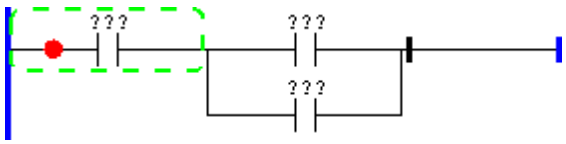


In this situation, when inserting an element, we are making this insertion related to this element. Should it be necessary to insert an element related to the parallel that is a group of elements, we must use the G key, which will select the group, and then we can define the insertion point, and make the insertion as shown in the following figures.

Pressing **G**:

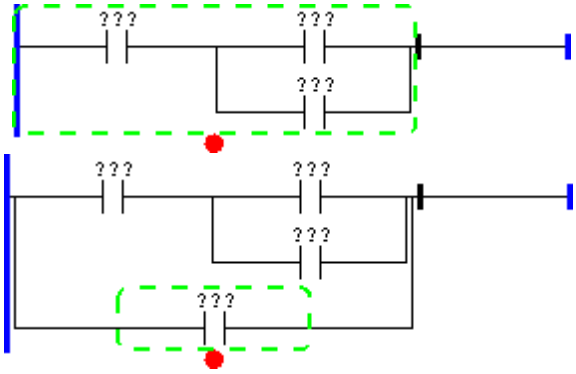


Inserting contact before the parallel:

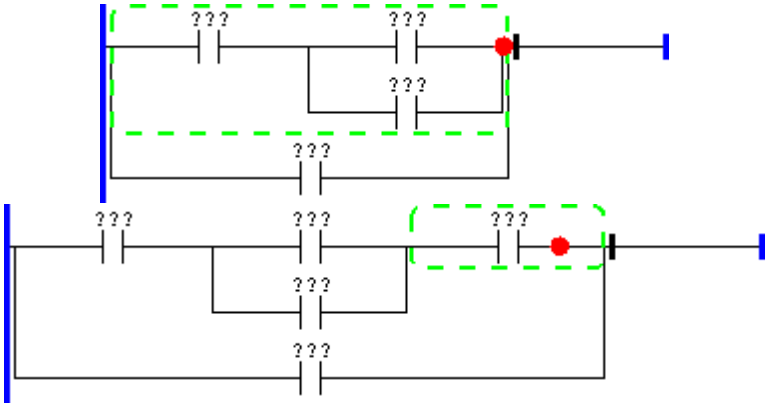


Every time the **G** key is pressed, we select the group immediately above the selected element/group, and in the end, we go back to the original element, so, in some situations, it is necessary to touch the **G** key more than once. Through the **G** key it is possible to make many insertion operations conforming to the following examples.

Insert an element in parallel to the group of elements in the rung

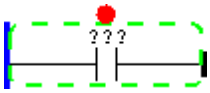
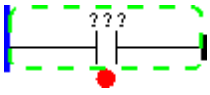
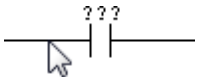
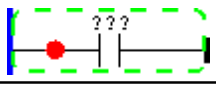
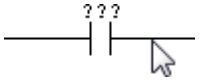
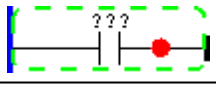


Insert an element in series to the group formed by elements that are internal to a parallel



8.2.3.5.2 With the Mouse

All browsing functionalities with the keyboard in the rung are also available in the mouse. It is possible to select an element and the respective insertion point directly with the click of the mouse.

Cursor	Action	Representation
	Selects element and upper insertion point	
	Selects element and low er insertion point	
	Selects element and anterior insertion point	
	Selects element and posterior insertion point	

During the element insertion operation with the mouse, by dragging and releasing the mouse in the palette, as we drag the elements over the rung, the selection and the insertion point follow the mouse cursor so as to determine the insertion point. The following figure exemplifies the insertion of elements via mouse.

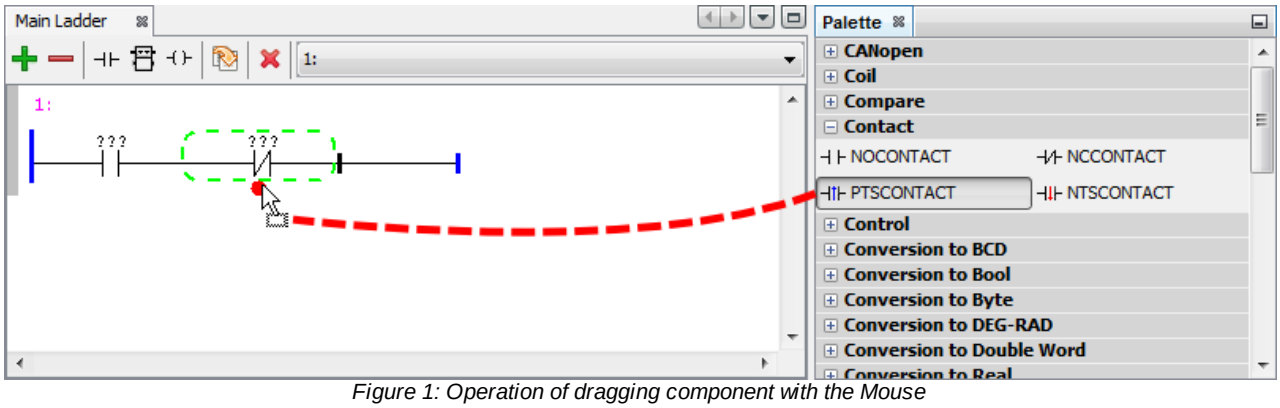


Figure 1: Operation of dragging component with the Mouse

Element insertion sequence via mouse:
1. Select the palette element you wish to insert by clicking on it with the mouse, and keeping the mouse button pressed
2. Drag this element over the rung
3. Select the insertion point and release the mouse button on it

The insertion result in this example will be as follows.

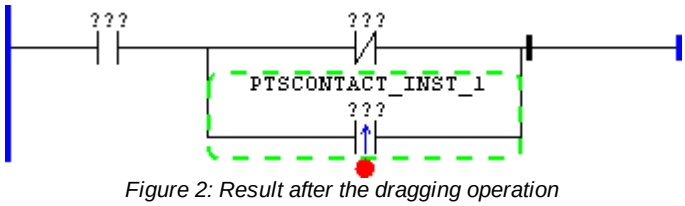


Figure 2: Result after the dragging operation

During the insertion via mouse, it is also possible to select the group of elements. For that purpose, as the previously selected element is dragged, press the **Ctrl** function key and keep it pressed; while the **Ctrl** key is being pressed, the mouse will select insertion points related to groups.

8.2.3.6 Copy/Paste

Overview

All the copy, cut and paste functions are available in the Ladder editor through the edit menu or through the corresponding keyboard shortcuts. In order to execute these operations it is necessary to have rung and/or element selected.

Copy/paste an element

Below is an example of copying and pasting an element.

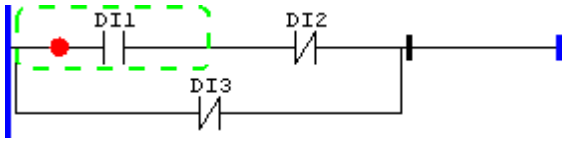


Figure 1: First, select the element and press Ctrl+C

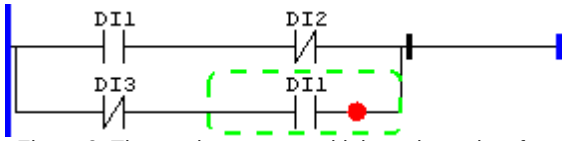


Figure 2: Then, select contact with insertion point after and press Ctrl+V

Copy/paste multiple elements

It is also possible to copy, cut and paste multiple elements selected either by the mouse or by the G key as previously mentioned. Below is an example of copying and pasting multiple elements.

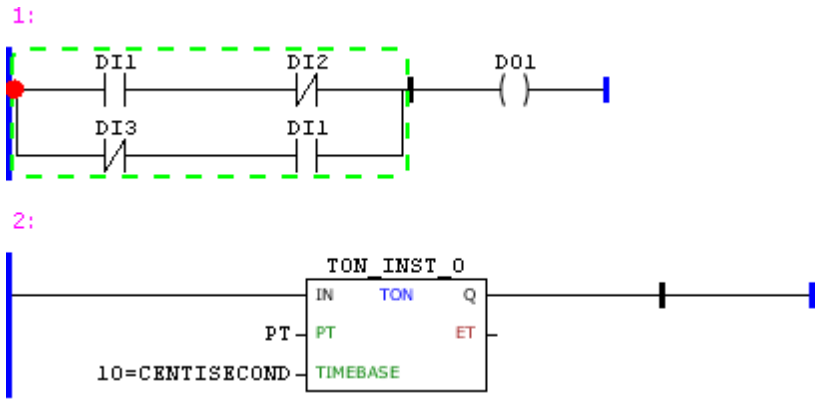


Figure 3: First, select the group and press Ctrl+C

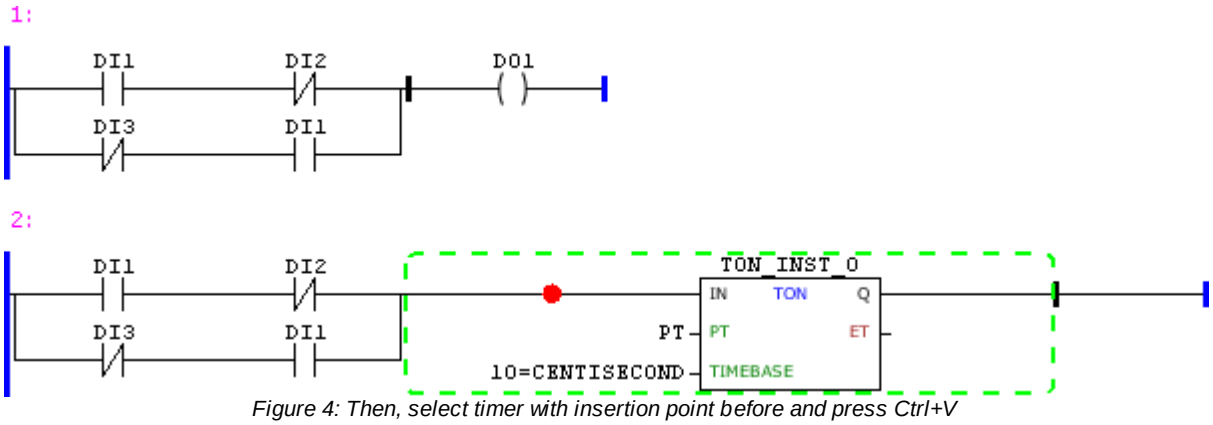


Figure 4: Then, select timer with insertion point before and press Ctrl+V

There might be cases in which it is not possible to paste and there will not be modification in the rung after the command.

Copy/paste rungs

The selection of rungs for the operations of copy, cut and paste is only performed through the mouse as shown in the figures below.

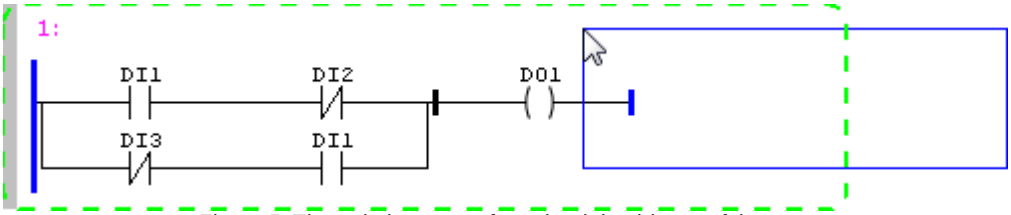


Figure 5: Through the mouse from the right side out of the rung

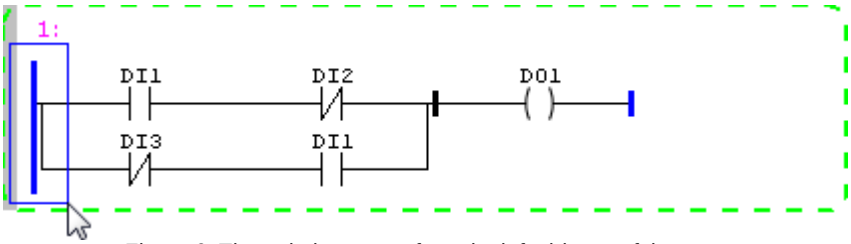


Figure 6: Through the mouse from the left side out of the rung

It is also possible to select multiple rungs for these operations.

Below is an example of copying / pasting a rung.

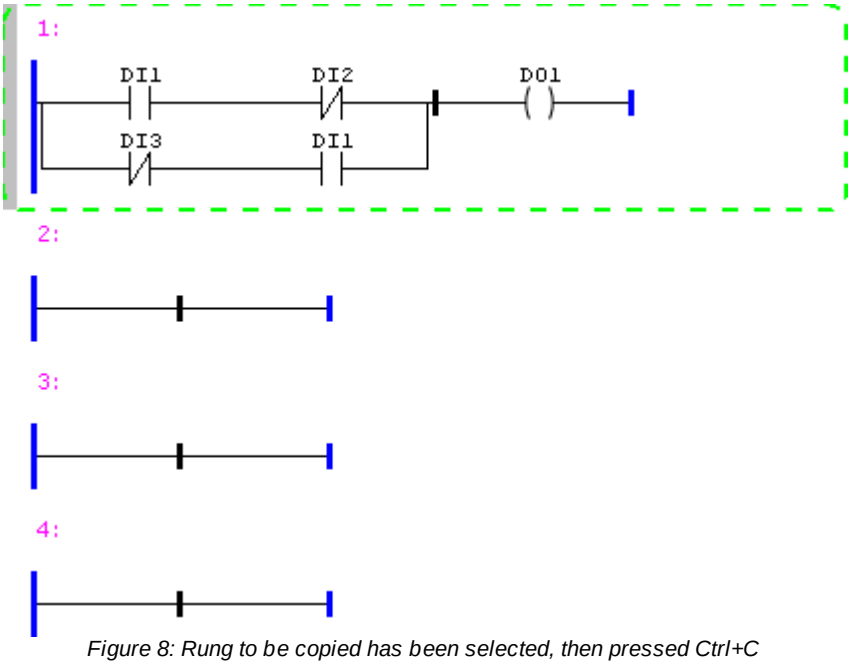


Figure 8: Rung to be copied has been selected, then pressed Ctrl+C

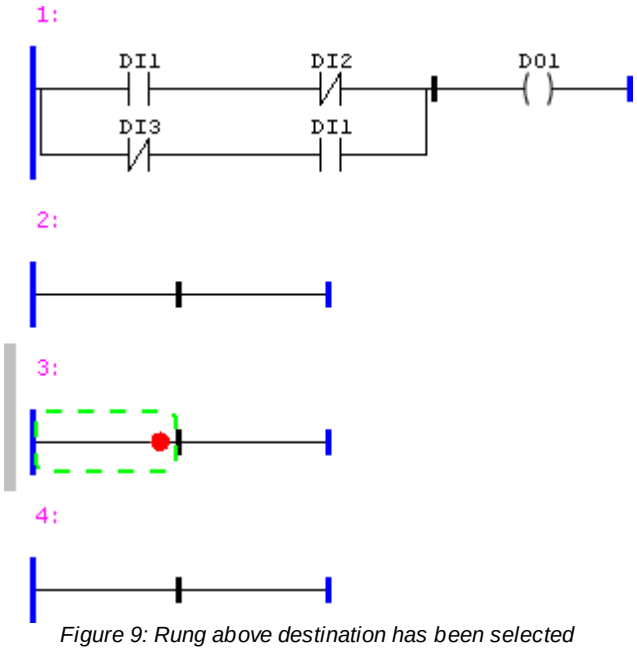


Figure 9: Rung above destination has been selected

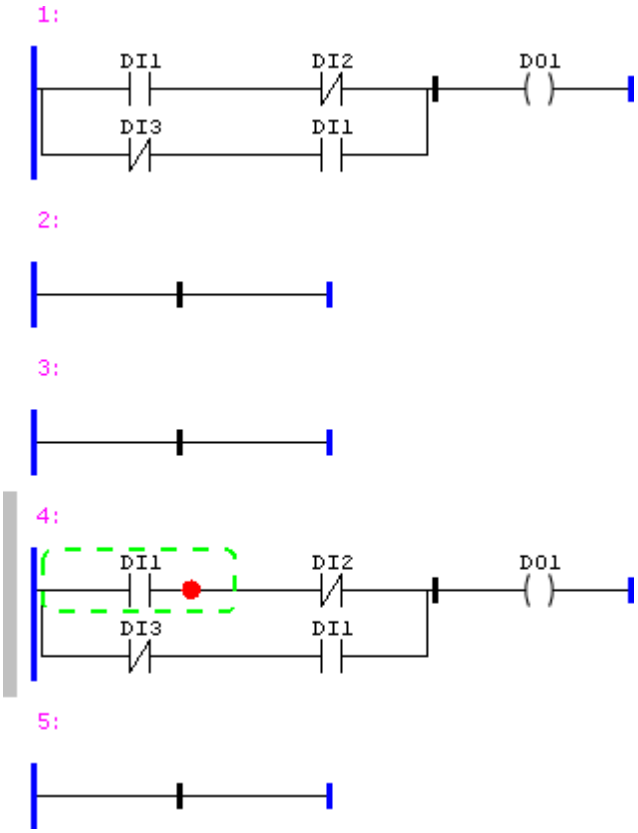


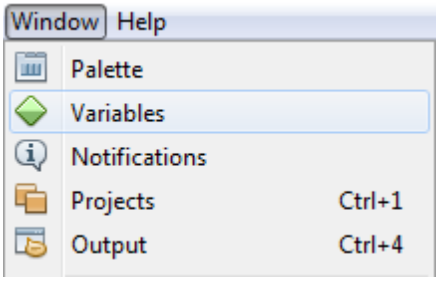
Figure 10: After pressing Ctrl+V

8.2.4 Variables

8.2.4.1 Overview

During the Ladder edition, it is necessary to define the variables used in the Ladder components. This definition can be done directly in the Ladder and/or in the editor/list of variables.

The table of variables must be activated through the menu **Window** > **Variables** as shown in the following figure.



In the variable edition window, the following commands are available:

- Add a new variable
- Edit the selected variable
- Remove the selected variable
- Move the selected variable one row up in the table
- Move the selected variable one row down in the table
- Rename the selected variable
- Export variables from this group to a file
- Import variables to this group from a file

The variables of the **GLOBAL** scope (1) present the groups (2) as shown in figure 1.

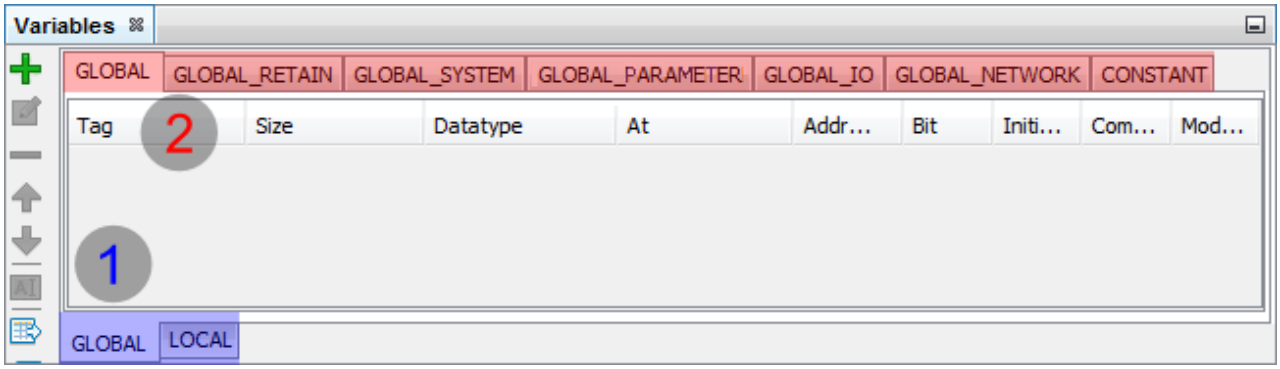


Figure 1: Variable Editor for the "GLOBAL" Scope

In this area of the global variables, the following groups are available:

- **Constant**: variables that store constant values
- **Global**: variables accessible by all the Ladder files
- **Global Retentive**: similar to the global group, but with retentive memory

- **System Global**: variables previously defined with system functions
- **Global Parameter**: variables from device accessible by Ladder and HMI
- **I/O Global**: physical inputs and outputs of the equipment
- **Network Global**: Variables previously defined for network communication

The variables of the **LOCAL** scope (1) present the groups (2) as per figure 2.

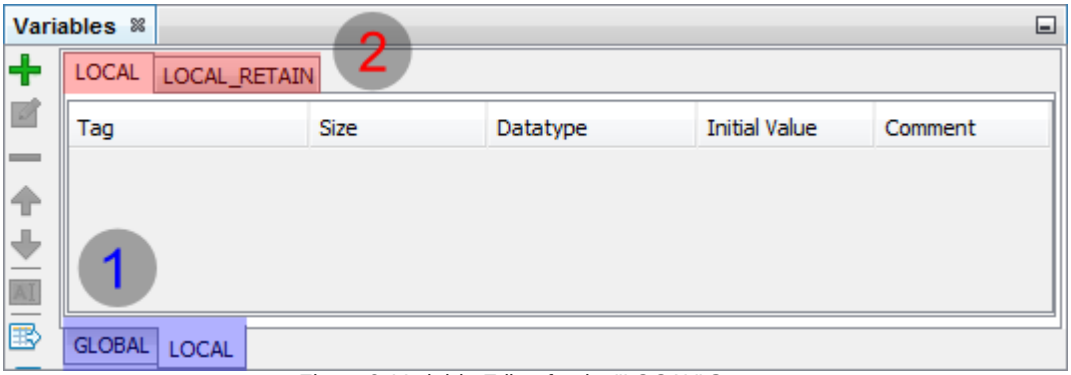


Figure 2: Variable Editor for the "LOCAL" Scope

In this area of local variables, the following groups are available:

- **Location**: variables only accessible through the Ladder that is being edited
- **Local Retentive**: similar to the local group, but with retentive memory

> **NOTE!**
> A retentive memory keeps its value even with the equipment off.

8.2.4.2 Fields

When defining a variable through the variable editor/list, some data must be defined for the variables. In the following items, those data will be presented according to the group the variable belongs to.

Local and Local Retentive:

Tag	Size	Datatype	Initial Value	Comment
LOCAL_VAR	0	BOOL	1	Comment

- **Tag**: variable identification
- **Size**: number of elements of the array* related to the variable
- **Data Type**: variable numeric type
- **Initial Value**: value that will be loaded for variable during the initialization of the equipment
- **Comment**: comment of the variable in the selected language

Constant:

Tag	Datatype	Initial Value	Comment
CONST	REAL	7.89	Comment

- **Tag**: variable identification
- **Data Type**: variable numeric type
- **Value**: constant value of the variable
- **Comment**: comment of the variable in the selected language

Global and Global Retentive:

Tag	Size	Datatype	At	Address	Bit	Initial Value	Comment	Modbus
GLOBAL_VAR	0	BYTE	%MB	100		12	Comment	8050

- **Tag**: variable identification
- **Size**: number of elements of the array* related to the variable
- **Data Type**: variable numeric type
- **At**: defines which global memory area the variable accesses
- **Address**: address related to the global memory area. If not configured (empty), the compiler will automatically define its address.
- **Bit**: for Boolean data type, it is necessary to define the bit it accesses (0...7)
- **Initial Value**: value that will be loaded for variable during the initialization of the equipment
- **Comment**: comment of the variable in the selected language
- **Modbus**: modbus address

System:

Tag	Size	Datatype	At	Address	Bit	Initial Value	Comment	Modbus
FIRMWARE	0	UINT	%SW	0	0	0	Firmware version	3000

- **Tag**: variable identification
- **Data Type**: variable numeric type
- **At**: defines which global memory area the variable accesses
- **Address**: address related to the global memory area.
- **Bit**: for Boolean data type, it is necessary to define the bit it accesses (0...7)
- **Comment**: comment of the variable in the selected language
- **Modbus**: modbus address

Global Parameter:

Tag	Datatype	At	Address	Initial Value	Comment	Modbus
PAR_023	UINT	%SW	32	100	Main SW Version	23

- **Tag**: variable identification
- **Data Type**: variable numeric type
- **At**: defines which global memory area the variable accesses
- **Address**: address related to the global memory area.
- **Initial Value**: initial standard value
- **Comment**: comment of the variable in the selected language
- **Modbus**: modbus address

I/O:

Tag	Size	Datatype	At	Address	Bit	Initial Value	Comment	Modbus
DI1	0	BOOL	%IB	0	0	0	Digital input 1	16000

- **Tag**: variable identification
- **Size**: number of elements of the array* related to the variable
- **Data Type**: variable numeric type
- **At**: defines which global memory area the variable accesses
- **Address**: address related to the global memory area.
- **Bit**: for Boolean data type, it is necessary to define the bit it accesses (0...7)
- **Comment**: comment of the variable in the selected language
- **Modbus**: modbus address

NOTE!
When size is greater than zero, the variables are accessed in the Ladder through their array index.

8.2.4.3 Editing in the Rung

The Ladder elements inserted in the rung require that variables be defined for each argument. See figure below.

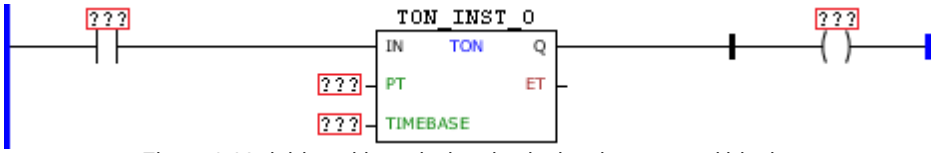


Figure 1: Variables without declaration in the elements and blocks

In order to define a variable for the argument, it is necessary to enter the argument edition mode, which is done in two ways.

ATTENTION!
As from the version 1.30 of the WPS the output arguments of the functional blocks can be optional.
The arguments in question will not be initialized with the declaration **???** and may be omitted if they do not have to be used in the Ladder logic.

1. By the mouse:
Double clicking the mouse directly on the argument

2. By the keyboard:
Pressing the **F2** key.
For elements with one argument, it directly enters the argument edition mode.
In the other elements with more than one element, it is necessary to select the argument through the arrow keys and then press the **F2** key again.

When entering the edition mode, the element will appear similar to the following figure.

Figure 2: Attributing the variables

At this moment, an edition box will be enabled for you to enter the variable name. When the **Edit** button is pressed, a box to create the new variable will be enabled.

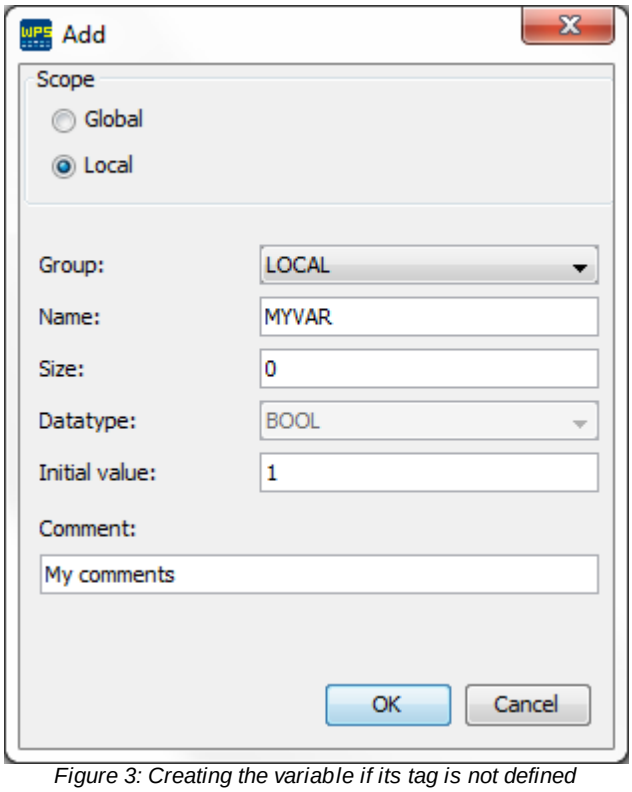


Figure 3: Creating the variable if its tag is not defined

In this box you should define the following options for the variable:

- **Scope**: if it belongs to the Global or Local group
- **Tag**: variable identification
- **Size**: number of elements of the array* related to the variable
- **Data Type**: variable numeric type
- **Group**: Group to which the variable belongs

If there are already variables defined for the type compatible with the Ladder element, a selection box with these variables will show together with the edition box. In order to select the desired variable, press the down arrow key and after having the variable selected, press enter. The figure below shows this function.

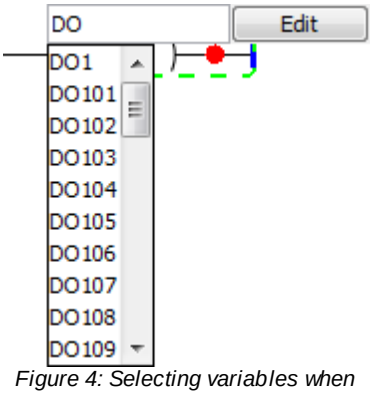


Figure 4: Selecting variables when typing

8.2.4.4 Literals in the Rung

In the functional blocks, it is also possible to enter literal values as shown in the figure that follows.

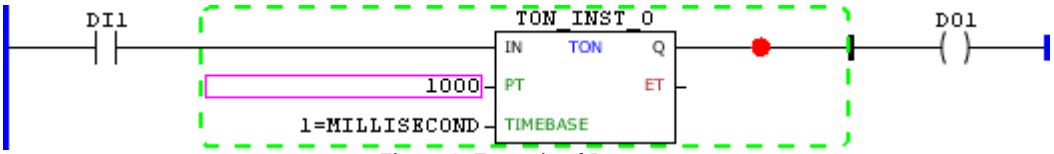


Figure 1: Example of Program

In this example, the PT input of the TON block was configured with the value 1000, which is a literal.

In order to enter literals, the following conventions must be used:

- Whole number has no decimal point.
  - E.g.: 12, 1000, 1555
- Real numbers with floating point must necessarily have point.
  - E.g.: 1.5, 2.25, 3.0
- Numbers represented in hexadecimal must necessarily define the data type.
  - E.g.: BYTE#16#7F, WORD#16#3CF0, DWORD#16#00FF0088
- Numbers represented in binary must necessarily define the data type.
  - E.g.: BYTE#2#1010_0000, WORD#2#0111_0000_0000_0001

ATTENTION!
In some blocks, for data verification and consistency reasons, it will be necessary to define the data type of the literal through specific notation that will contain the following options: BOOL#, BYTE#, INT#, UINT#, DINT#, UDINT#, WORD#, DWORD# and REAL#. Example: WORD#17321

8.2.4.5 Arrays in the Rung

In the contacts, coils and functional blocks, it is also possible the access of variables of the array type as shown in the following figures.

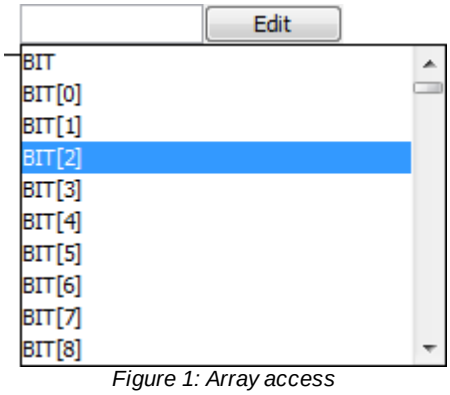


Figure 1: Array access

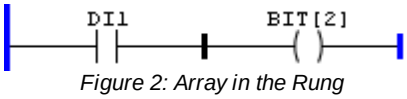


Figure 2: Array in the Rung

In order to view the indices of a variable of the array type, expand the variable in the variable window as shown in the following figure.

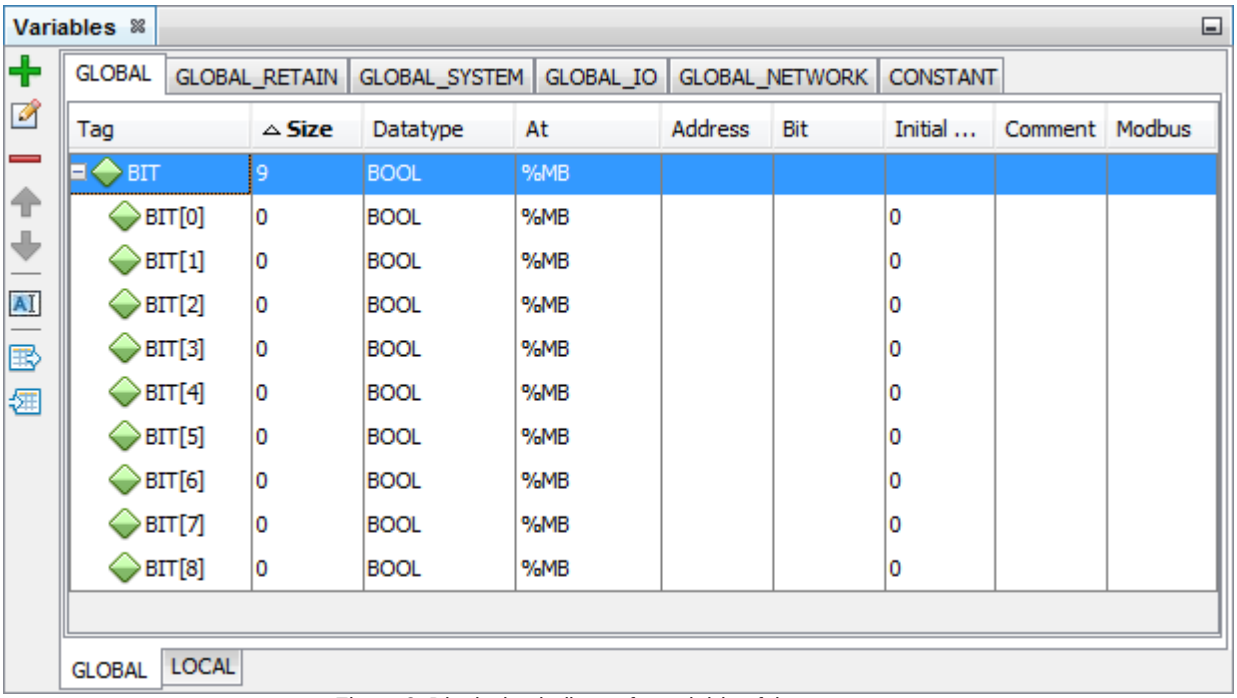


Figure 3: Displaying indices of a variable of the array type

8.2.4.6 Instances and Structures in the Rung

In the contacts, coils and functional blocks, it is also possible to access the internal variables of instances and structures as shown in the following figures.

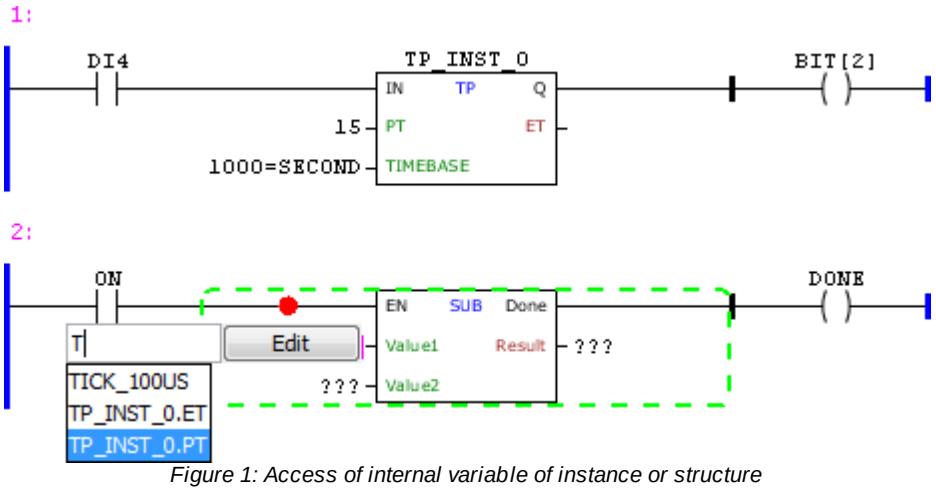


Figure 1: Access of internal variable of instance or structure

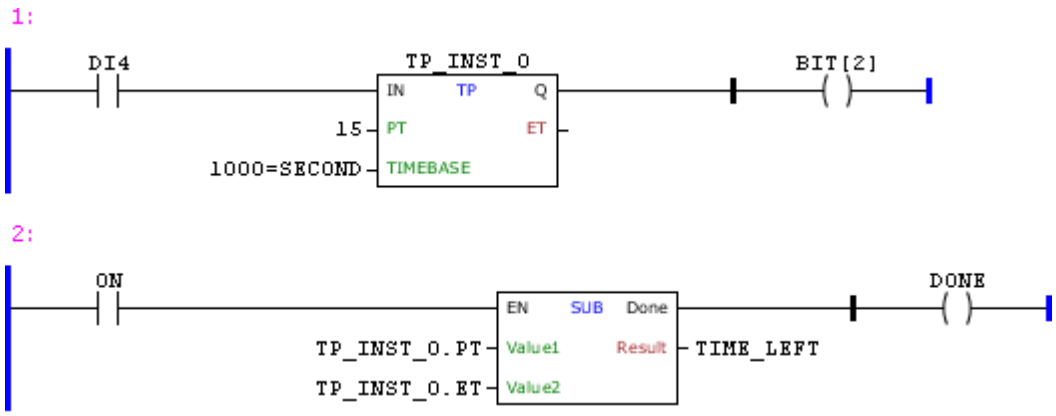


Figure 2: Internal variable of instance or structure in the Rung

In order to view the internal variables of the instances and structures, expand the variable in the variable window as shown in the following figure.

Tag	Size	Datatype	Initial Value	Comment
TP_INST_0	0	TP		
TP_INST_0.PT	0	DWORD		
TP_INST_0.ET	0	DWORD	0	
TP_INST_0.IN	0	BOOL	0	
TP_INST_0.Q	0	BOOL	0	

Figure 3: Displaying internal variables of instance or structure

8.2.4.7 Volatile and Retentive Instances

Function Blocks (FBs)

The FBs have internal variables that store their data during the consecutive execution cycles. According to the application requirement, these FBs may have their instances configured as retentive (**LOCAL_RETAIN** or **GLOBAL_RETAIN**) or volatile (**LOCAL_RETAIN** or **GLOBAL_RETAIN**). The input and output variable associated to the FB can also be configured as retentive or volatile. The retentive variables retain their values

after the device is shut down, whereas the volatile variables load their initial values after a reset.

When we want the FB to keep the values after the reset of the device, it is necessary that the FB instances and the variables associated to its inputs be configured as retentive. That will make the FB internal variables and the associated input variables keep the values previous to shutdown.
In the example below, we have the use of the TON block with retentive instances and variables:

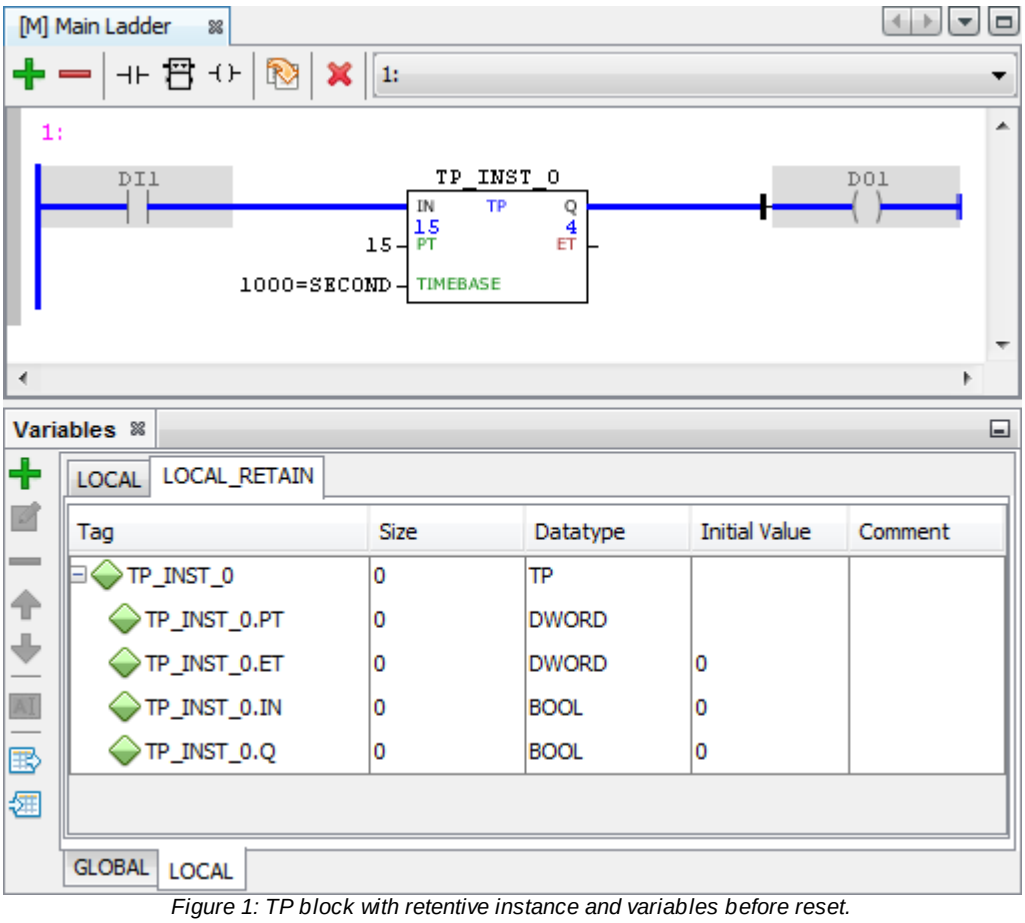


Figure 1: TP block with retentive instance and variables before reset.

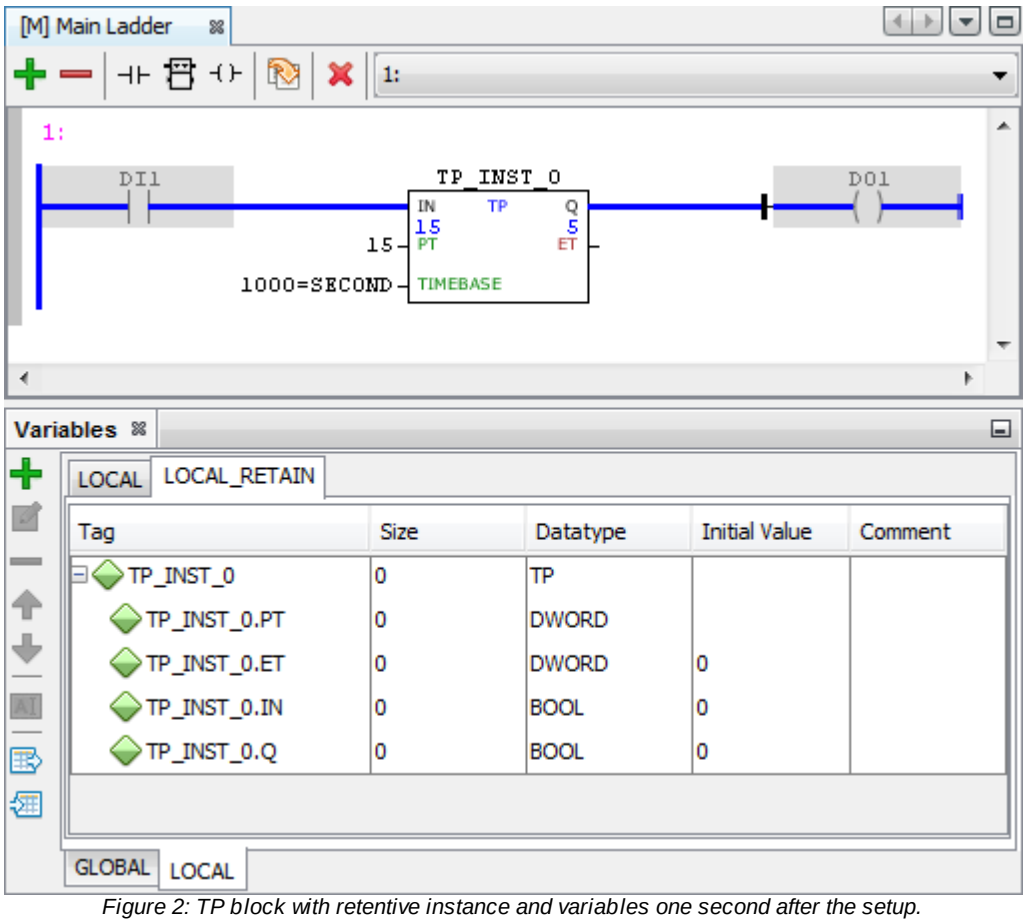


Figure 2: TP block with retentive instance and variables one second after the setup.

When we want the FB to reset its values after the shutdown of the device, it is necessary that the FB instance and the variables associated to its inputs be configured as volatile. That will make the FB internal variables and the associated input variables reset the values previous to shutdown.
Below is an example of use of the CTU block with retentive instance and variables:

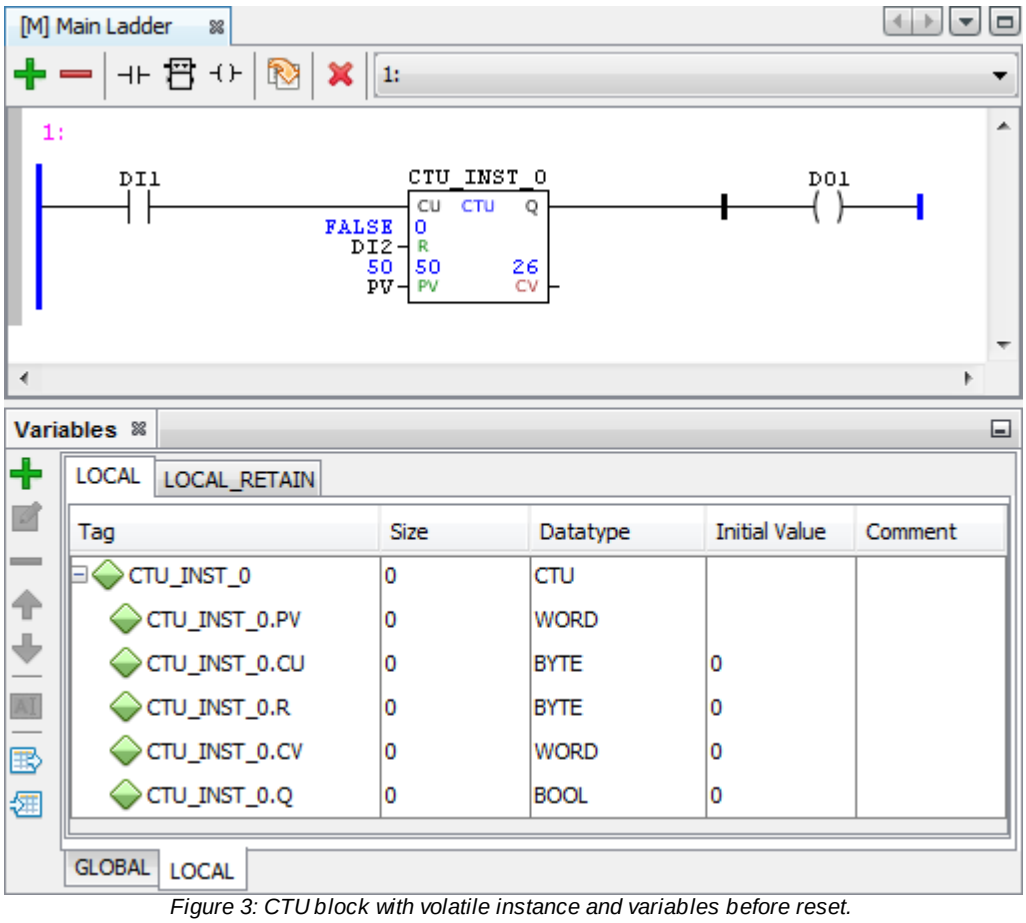


Figure 3: CTU block with volatile instance and variables before reset.

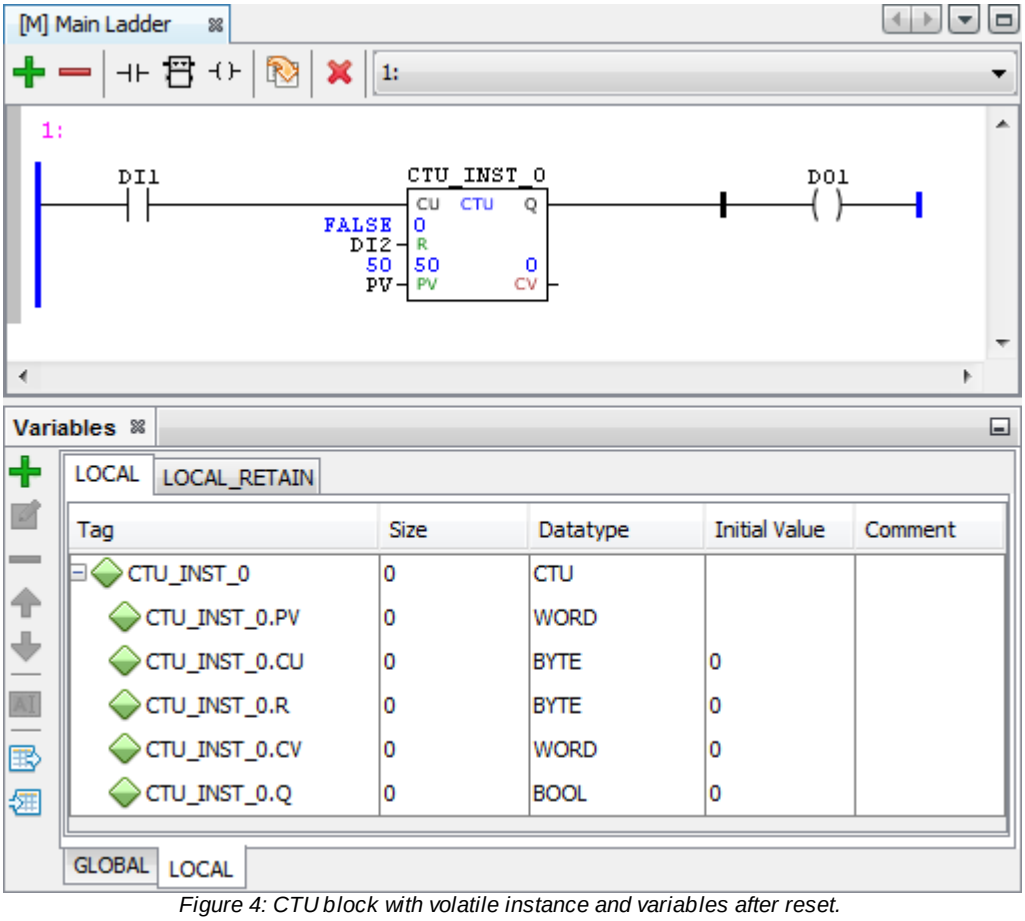


Figure 4: CTU block with volatile instance and variables after reset.

User's Block (USERFB)

In the use of the USERFB, it is possible to define variables of **LOCAL**, **LOCAL_RETAIN**, **VAR_IN**, **VAR_OUT** and **VAR_IN_OUT** type. The internal variables defined as **LOCAL** will always be volatile, and the **LOCAL_RETAIN** ones will always be retentive. The internal variable defined as **VAR_IN**, **VAR_OUT** and **VAR_IN_OUT** will be volatile in case the instance of the USERFB is associated to the **LOCAL** or **GLOBAL** group and retentive in case it is associated to the **LOCAL_RETAIN** or **GLOBAL_RETAIN** group.

8.2.5 Compile

In order to compile a Program (POU), there are four options:

1. Through the menu **Configuration** > **Build Main Resource**.

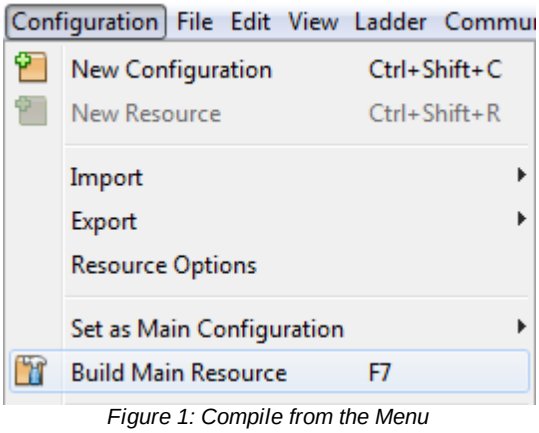


Figure 1: Compile from the Menu

2. Through the shortcut key **F4**.

3. Through the button on the **Toolbar**.

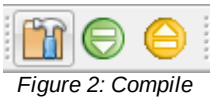


Figure 2: Compile from the Toolbar

4. Clicking the right button of the mouse on the name of the resource.

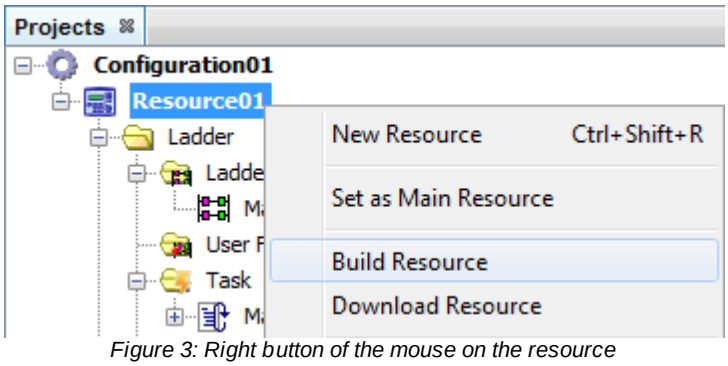


Figure 3: Right button of the mouse on the resource

The results of the compilation, indicating errors and warnings, can be viewed through the **Default Output** window.

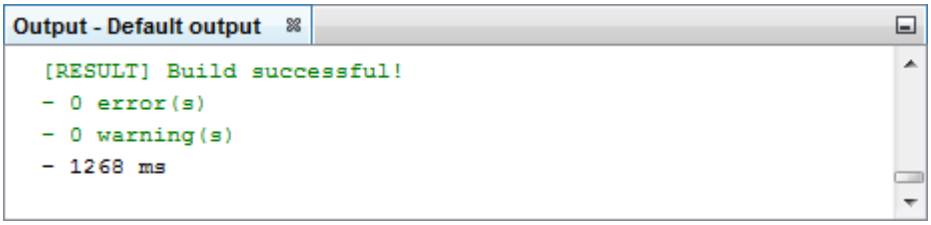


Figure 4: Results of the compilation

8.2.6 Transfer

In order to download a resource, there are four options:

1. Through the menu **Communication** > **Download Main Resource**.

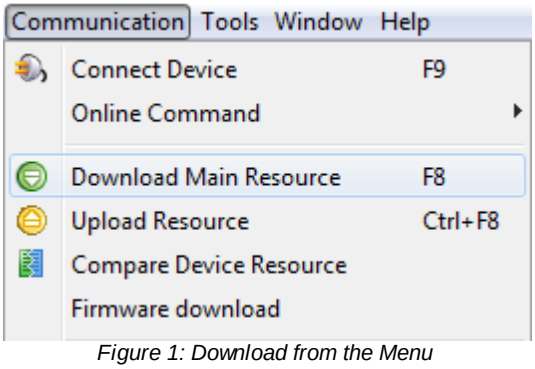


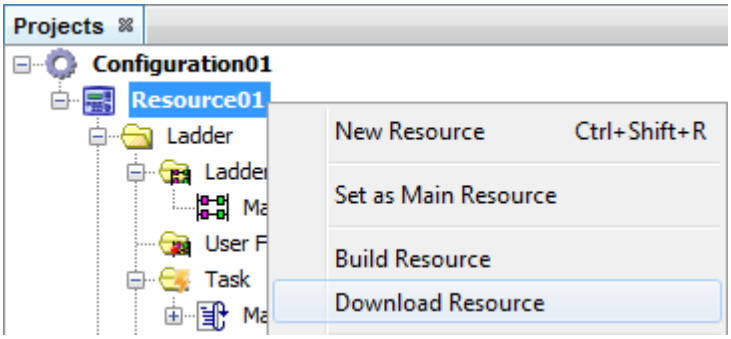
Figure 1: Download from the Menu

2. Through the shortcut key **F5**.

3. Through the button on the **Toolbar**.

Figure 2: Download from the Toolbar

4. Clicking the right button of the mouse on the name of the resource.



8.2.7 Online Monitoring

Overview

After the Ladder program is compiled and loaded on the device, it is possible to monitor the Ladder by pressing the **Connect Device** button  .
At this moment, the WPS v2.5X will try to establish communication with the device by testing the communication with it.

The online monitoring will graphically represent the logical state of the Ladder program. An example of online monitoring can be seen in the following figure.

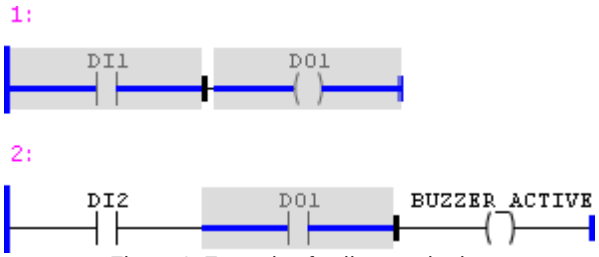


Figure 1: Example of online monitoring

For functional blocks, the values are presented as shown in the following figure:

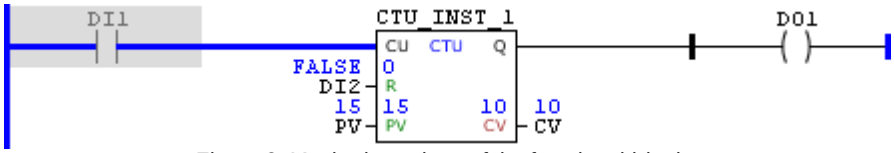


Figure 2: Monitoring values of the functional blocks

The variables values are shown on their respective variables; the internal values of the instance are shown on the name of the respective argument.

Writing of variables

In order to write variables, just double click the variable you wish to write and a value writing box will open as shown in the following figure.

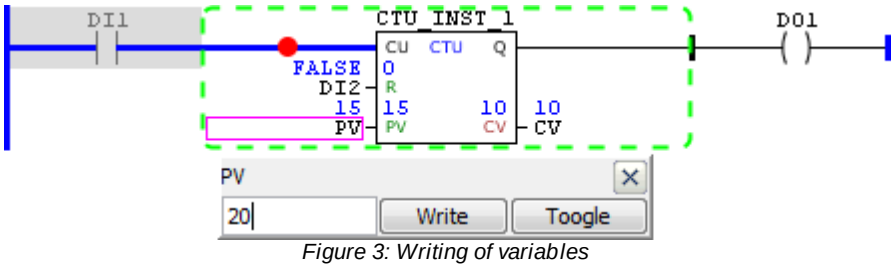


Figure 3: Writing of variables

In this box, you must enter the desired value and press the **Write** button in order to write the value. The **Toggle** button is used to toggle the value written between zero and the current value.

Monitoring of instances

In order to monitor, just double click the instance variable and a box related to instance monitoring will open as shown in the following figure.

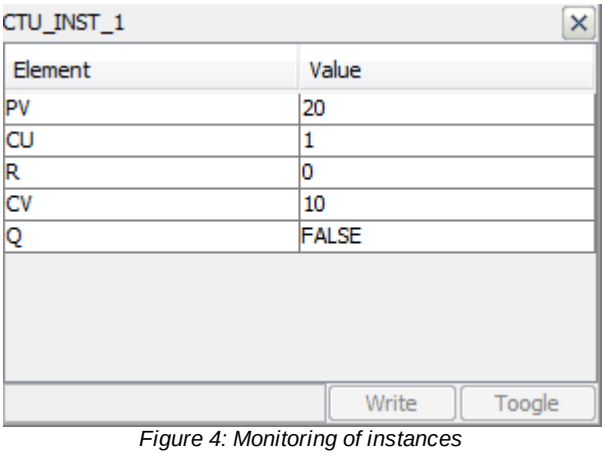


Figure 4: Monitoring of instances

In order to write on instance internal variables, just click on the corresponding line, use the value edition box and the **Write** and **Toggle** buttons as already mentioned in the previous item.

Monitoring of structures

For the variables created from structures defined in the resource, just click on the corresponding variable and a box similar to the instance monitoring one will open. See the following example.

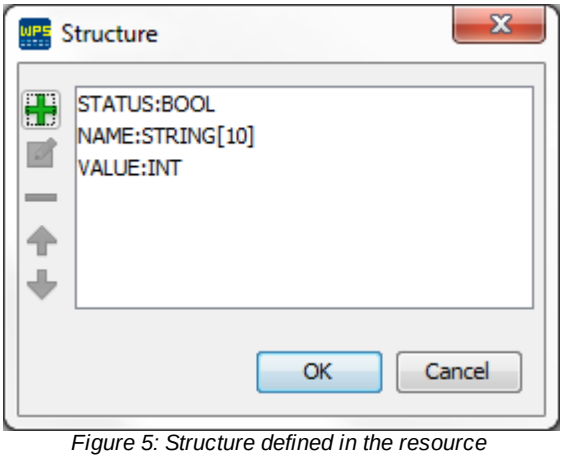


Figure 5: Structure defined in the resource

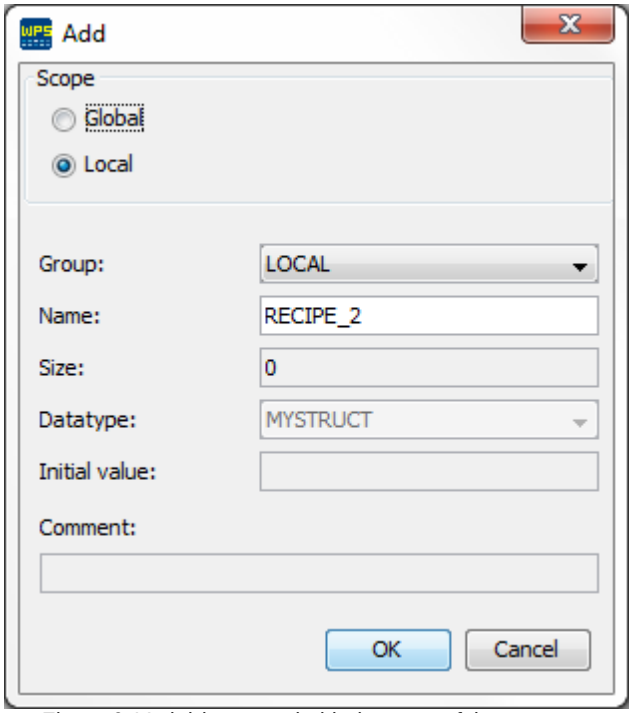


Figure 6: Variable created with data type of the structure defined in the resource

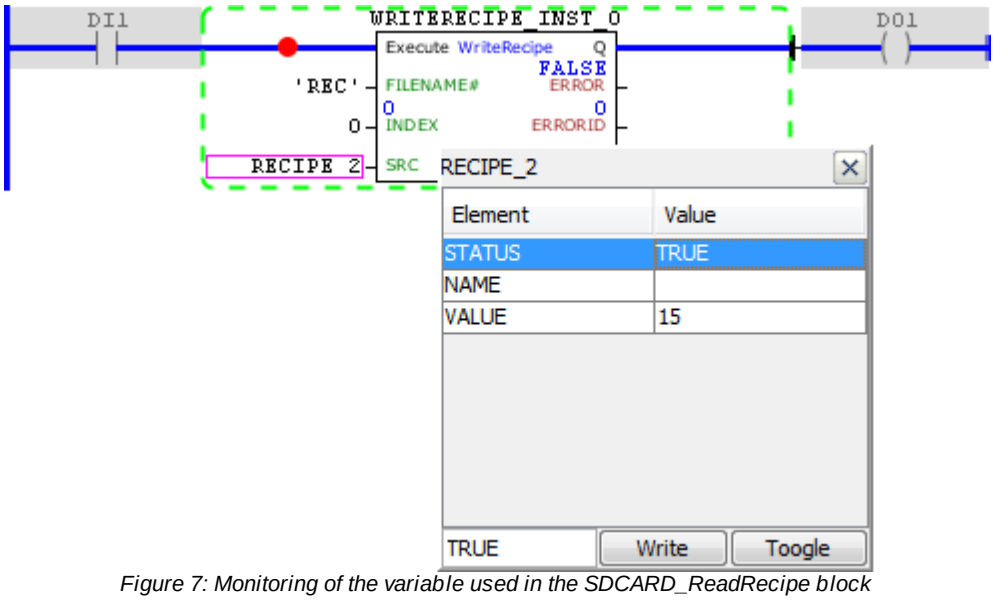


Figure 7: Monitoring of the variable used in the SDCARD_ReadRecipe block

Monitoring of arrays

For variables created with size greater than zero, it is possible to monitor all the data of their array. In order to do so, just click on the corresponding variable and a monitoring box will open. See the following example.

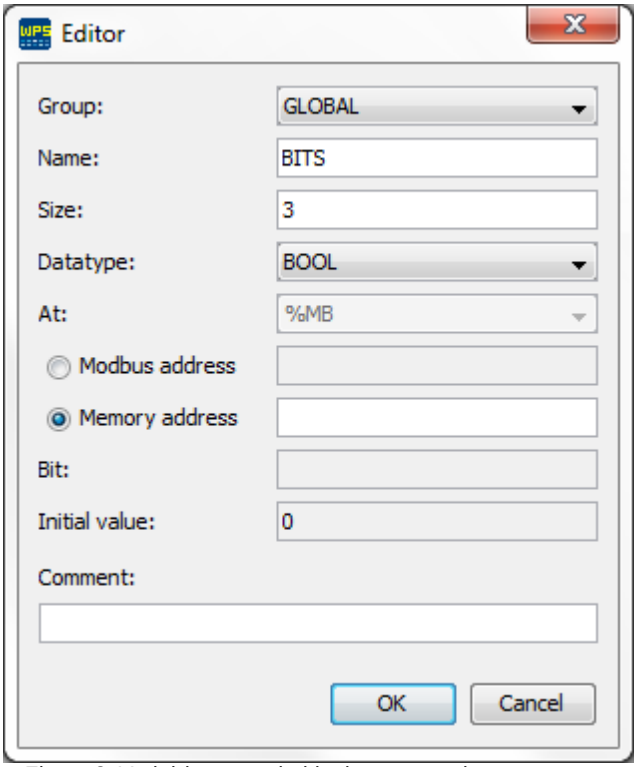


Figure 8: Variable created with size greater than zero, array

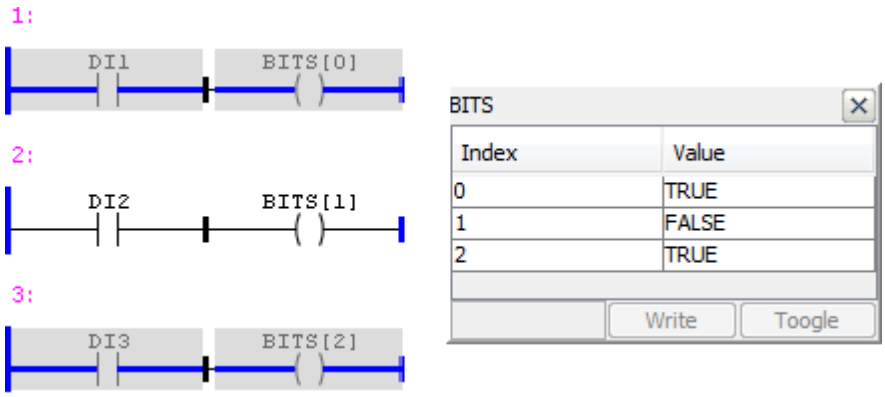


Figure 9: Monitoring of the variable used in MB_ReadRegister block

8.3 Working with USERFBs

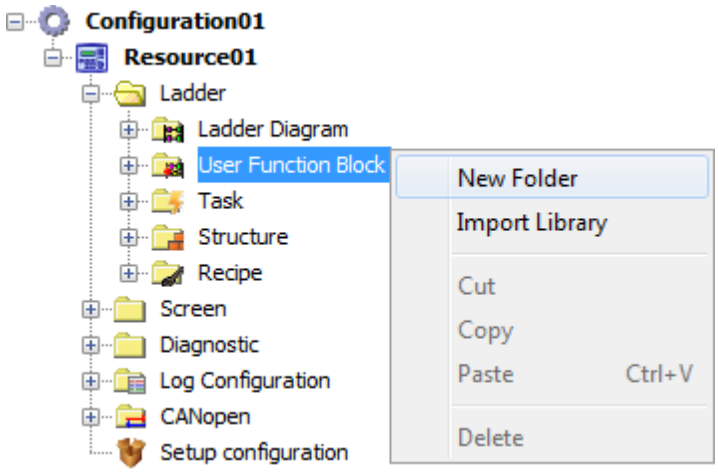
8.3.1 Creating USERFBs

USERFBs are cuser-customizable functional blocks. Its utilization is encouraged to make the Ladder program less bulky and polluted, abstracting information with which one does not want to work often and systematizing complex tasks.

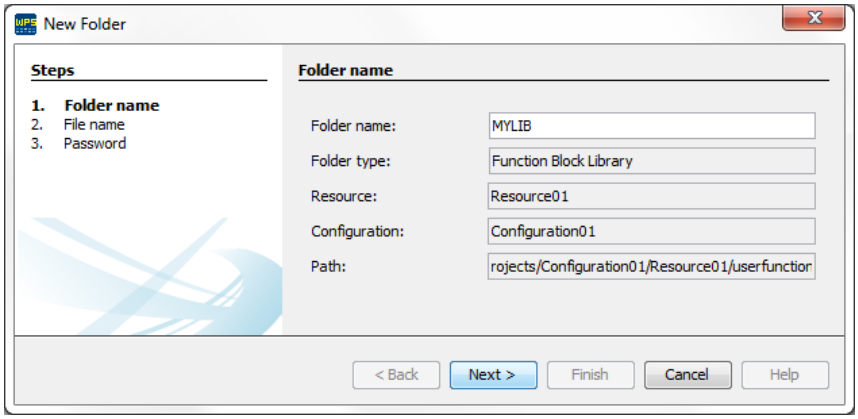
In these blocks, the inputs and outputs are defined by the user, who handles them in the Ladder diagram associated with the block. Here's how to create your USERFB.

1. In the **Projects** window, locate the resource in which you want to create the USERFB, right-click in **User**

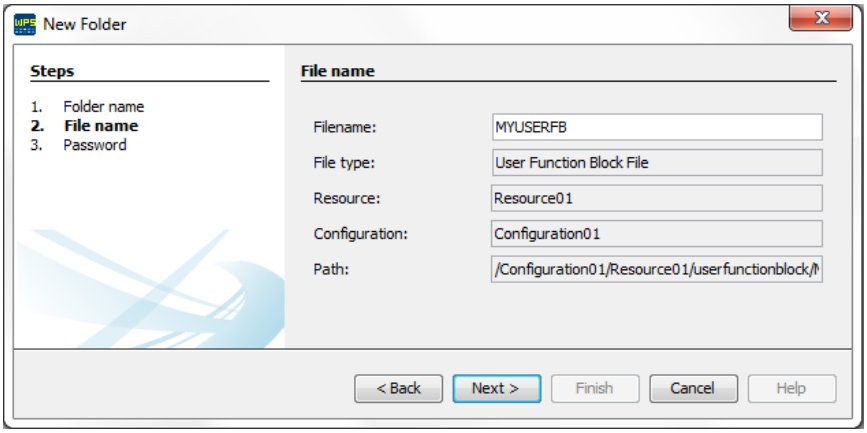
Function Block and click in **New Folder**.



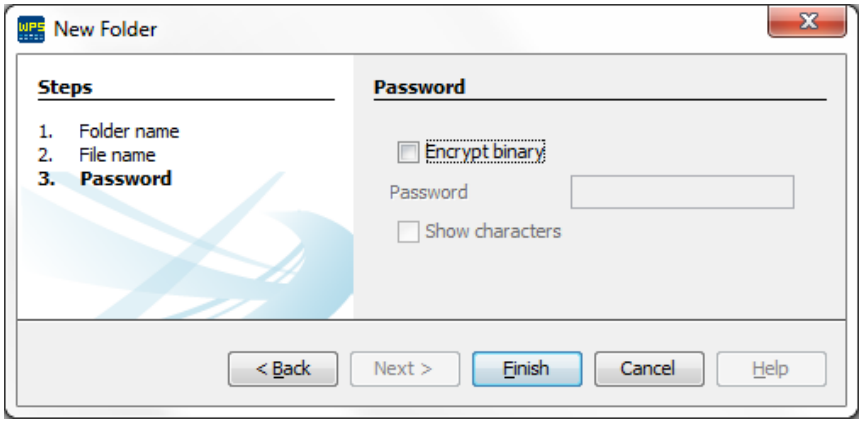
2. In the wizard, insert a name for the library to which the USERFB will belong and click **Next**.



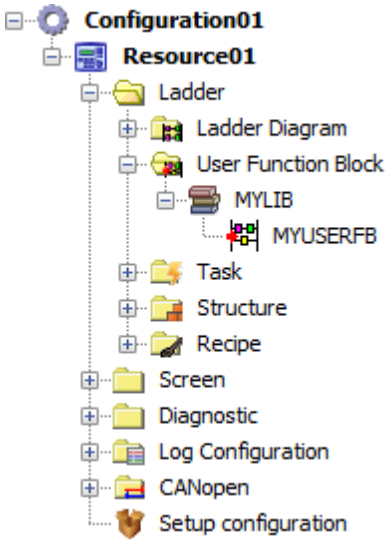
3. Insert a valid name for the USERFB and click **Next**.



4. If you want to insert a password to protect the block code, check the **Encrypt binary** checkbox and type a password. Otherwise, uncheck it. Click **Finish**.



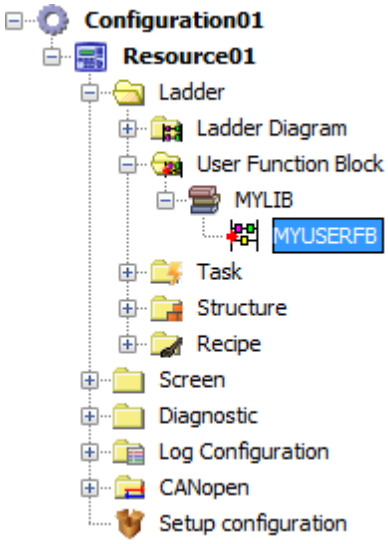
That's it! The USERFB has been succesfully created. You should see the following in the **Projects** window.



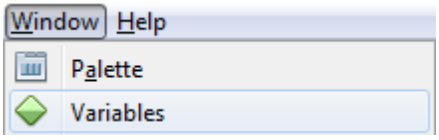
8.3.2 Adding input/output

Now we'll cover how to create inputs and outputs for the USERFB.

1. In the **Projects** window, double click the USERFB file in order to open its Ladder editor.

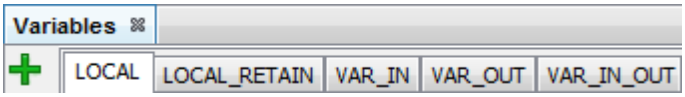


2. In the **Window** menu, click **Variables**.

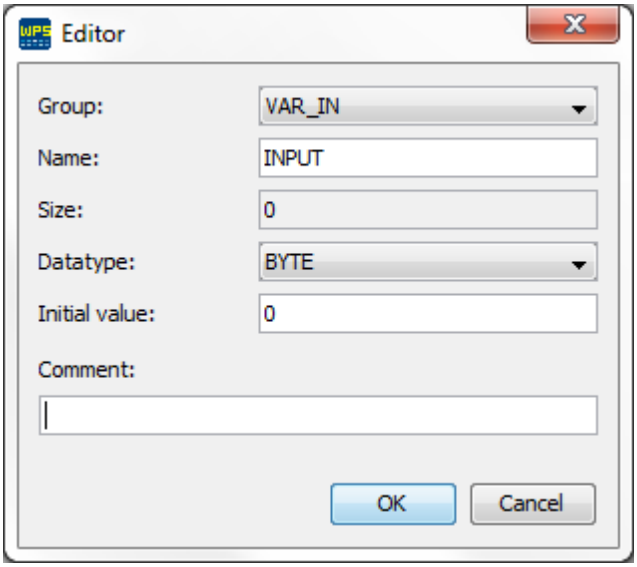


Analysing the following figure, we see that the USERFB **Variables** window is different from other Ladder files. It has only volatile and retain variables in **LOCAL** scope, which are the internal variables of the block, used in its subroutine. Besides these, it has three more groups: **VAR_IN**, **VAR_OUT** and **VAR_IN_OUT**.

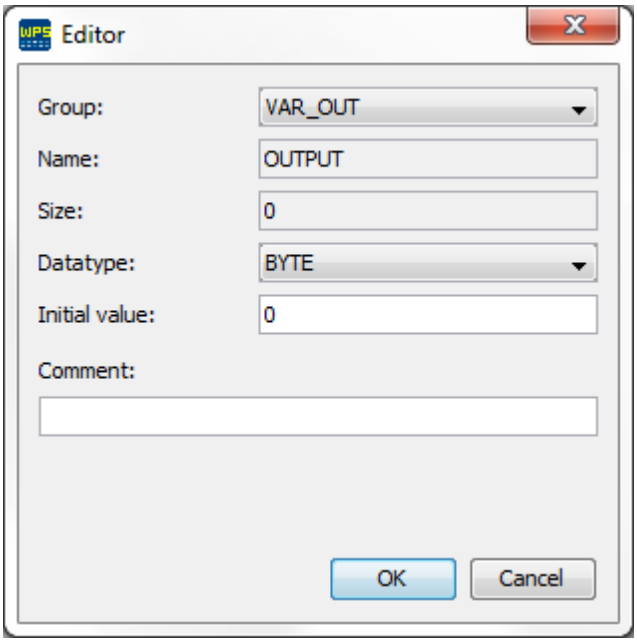
- **VAR_IN**: internal variables that repersent the input arguments for that block.
- **VAR_OUT**: internal variables that repersent the output arguments for that block.
- **VAR_IN_OUT**: internal variables that repersent the input/output arguments for that block.



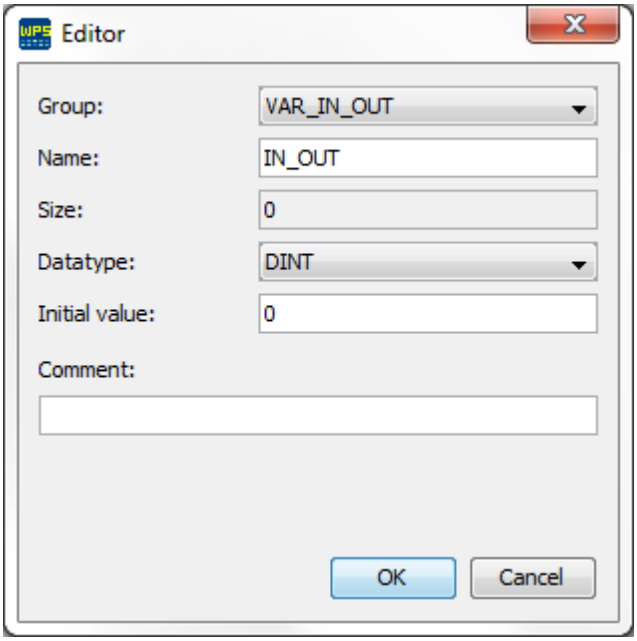
3. In order to create an input, click in the **VAR_IN** tab and click in the symbol. In the window, set a name and a datatype to this variable and click **OK**.



4. In order to create an output, click in the **VAR_OUT** tab and click in the symbol. In the window, set a name and a datatype to this variable and click **OK**.



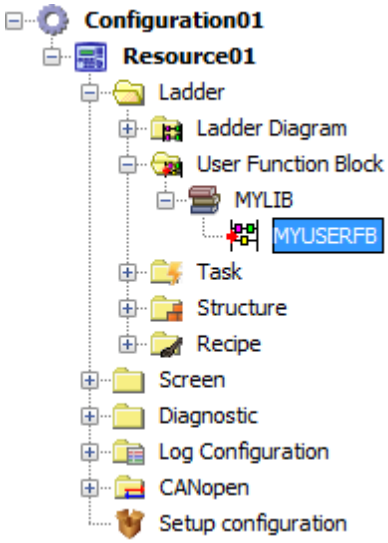
5. In order to create an input/output, click in the **VAR_IN_OUT** tab and click in the symbol. In the window, set a name and a datatype to this variable and click **OK**.



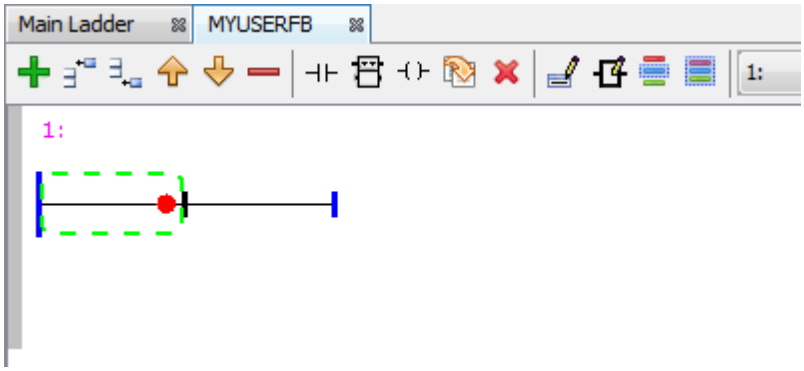
8.3.3 Editing the Ladder

Now we'll cover how to edit the USERFB subroutine.

1. In the **Projects** window, double click the USERFB file in order to open its Ladder editor.



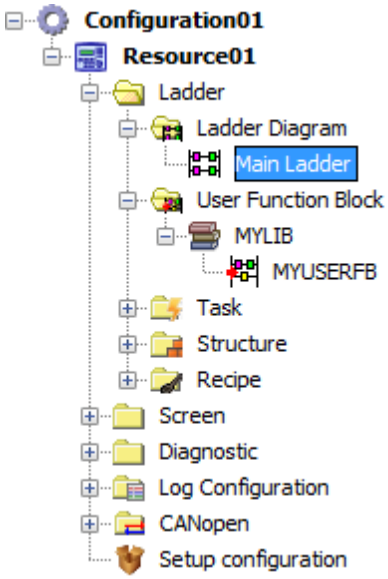
The **Ladder Editor** will open, like any other Ladder diagram. Any block may be inserted in it, including other USERFBs. Remember that only local variables may be used in it.



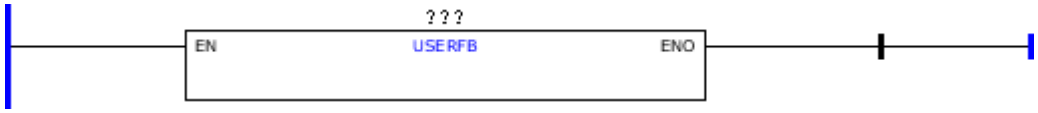
8.3.4 Using USERFBs

Lastly we'll cover how to make use of the USERFB, inserting it in other Ladder diagrams.

1. In the **Projects** window, double click the USERFB file in order to open its Ladder editor.

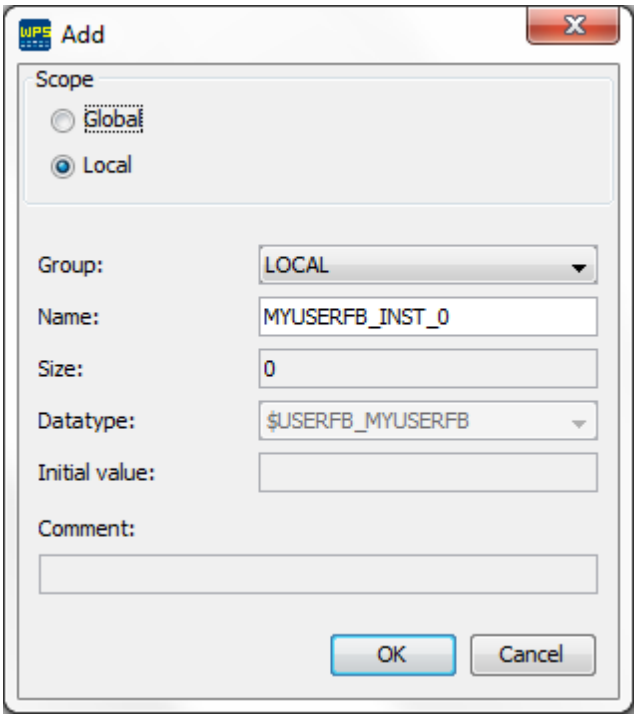


2. In the **Pallete** window, select the USERFB block from the **Module** category and drag it to the position where you want to use it in the Ladder diagram.

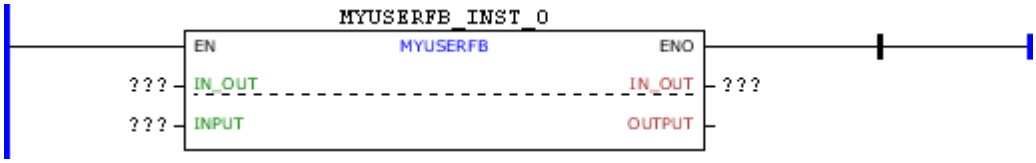


3. Double click the question marks (**???**) above the block in order to insert a instance variable for the USERFB. Type in the variable name and click **Edit**. In the confirmation dialog, click **Yes** to create the new variable.

4. In the Add dialog, type in a name for the variable and select its parameters. In the Datatype field, choose the name of the desired USERFB (if there is only one, the field will not be enabled).
For example, if your USERFB name is MYUSERFB, the correct datatype to be selected is $USERFB_MYUSERFB.



That's it! Your very own USERFB is inserted in the diagram and ready to work!

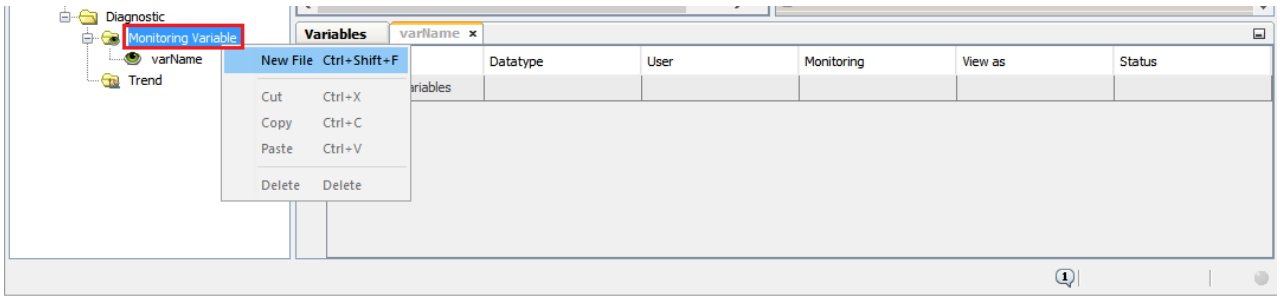


9 Diagnostic

9.1 Monitoring Variable

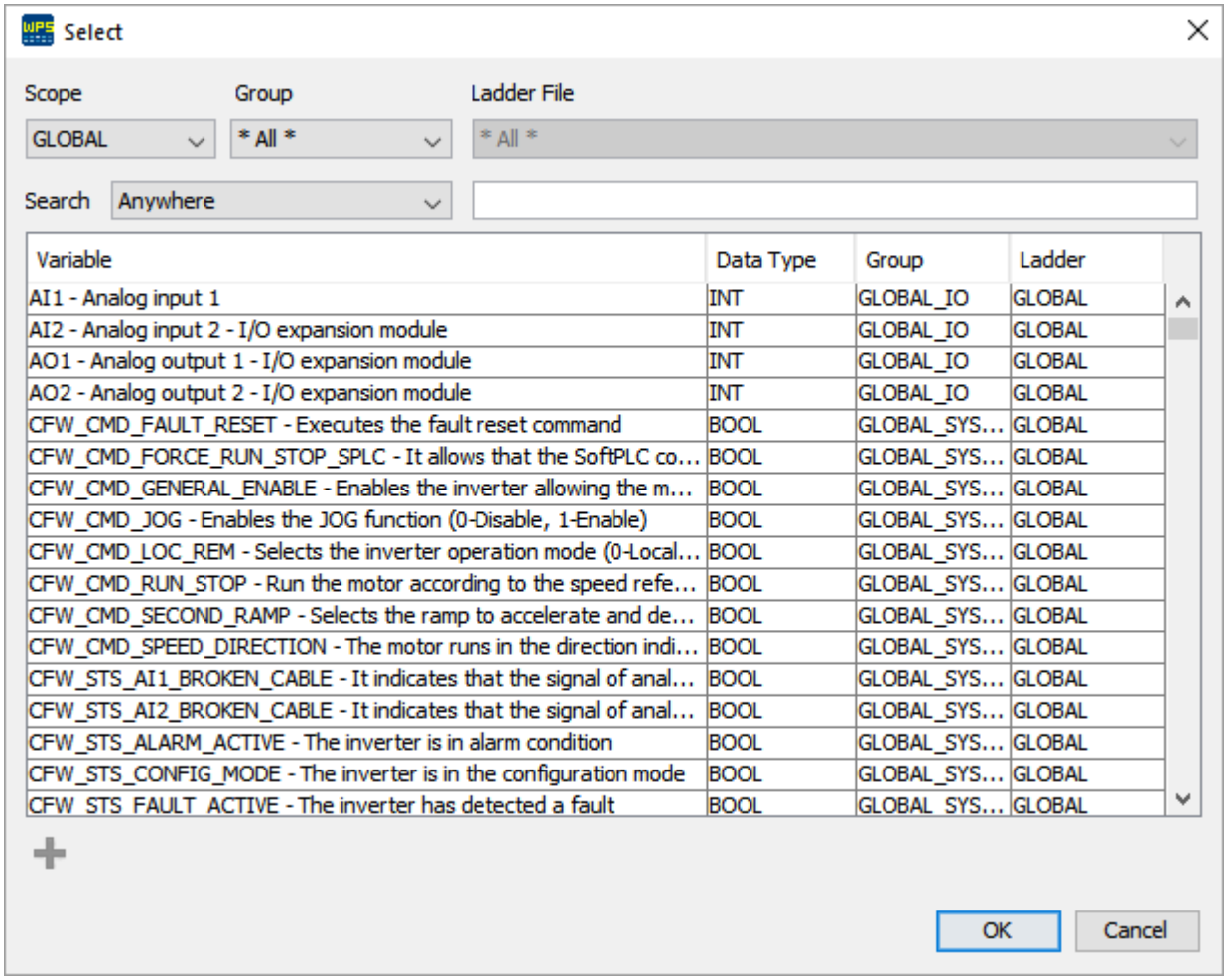
Monitoring Variable allows creating sets of variables for monitoring in a Resource.

To do this, right-click on Variable Monitoring and select **New File**, following the on-screen instructions for creating a new file.



After creating the file, use the  button to add a variable.

Select the desired variable and click **OK** to confirm adding:



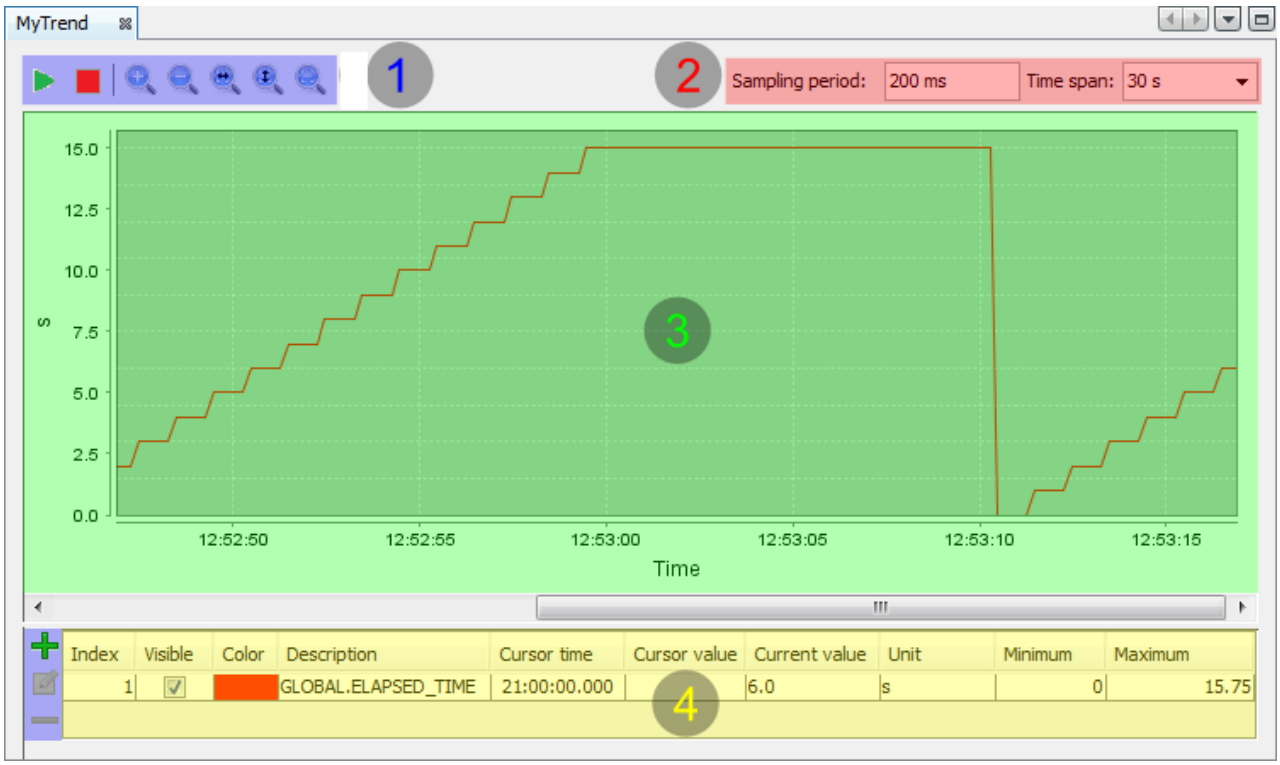
9.2 Trend

9.2.1 Overview

Trend is a graph of the values of variables versus time.

The **Trend** function has ten monitoring channels, which means it is possible to monitor up to ten variables and/or parameters at the same time.

Below is an overview of the trend function configuration screen.

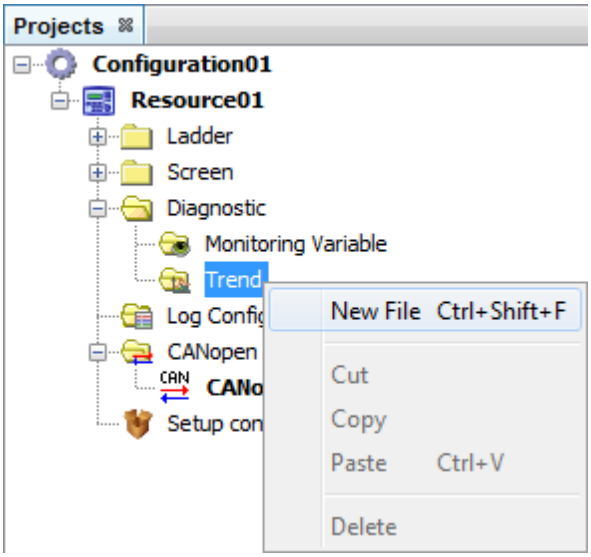


1. **Channels, Start, Stop and Graphic Zoom**: This bar contains the options to add, edit and remove channels, and also the options to control the chart, such as zoom in, zoom out, set width, set height and set all. There are buttons to start or stop data acquisition.
2. **Time**: This bar contains the options to configure the sampling periods and time range to be shown on the graphic.
3. **Chart**: This screen shows graphically the monitored values of the channels. In the lower part is the time of collection, and on the left is the range of values per unit of measurement of the channels.
4. **Channel Table**: This table shows the data of the chosen channels in the position where the cursor is, besides the possibility to hide channels (Visible), change the channel color (Color) and set the chart limits per unit of measurement (Maximum).

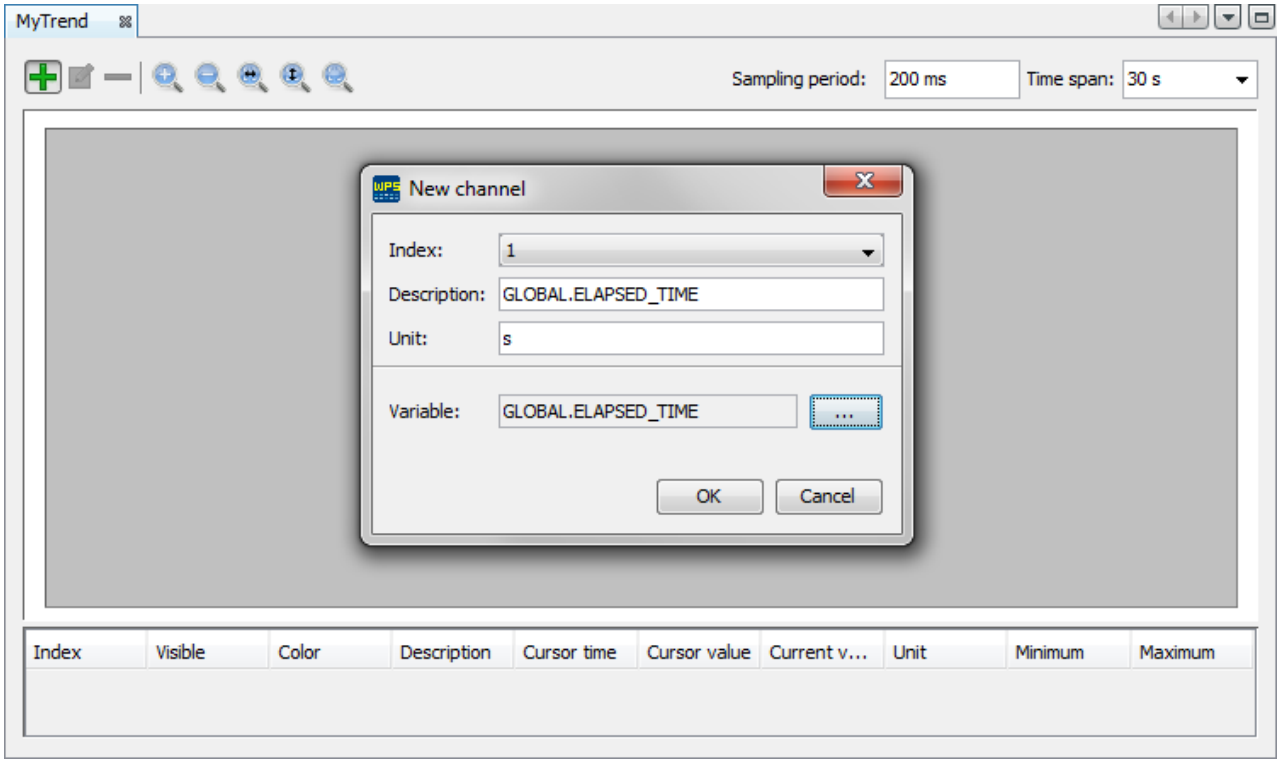
9.2.2 Configuration

Below is a list of the necessary steps to create a trend configuration.

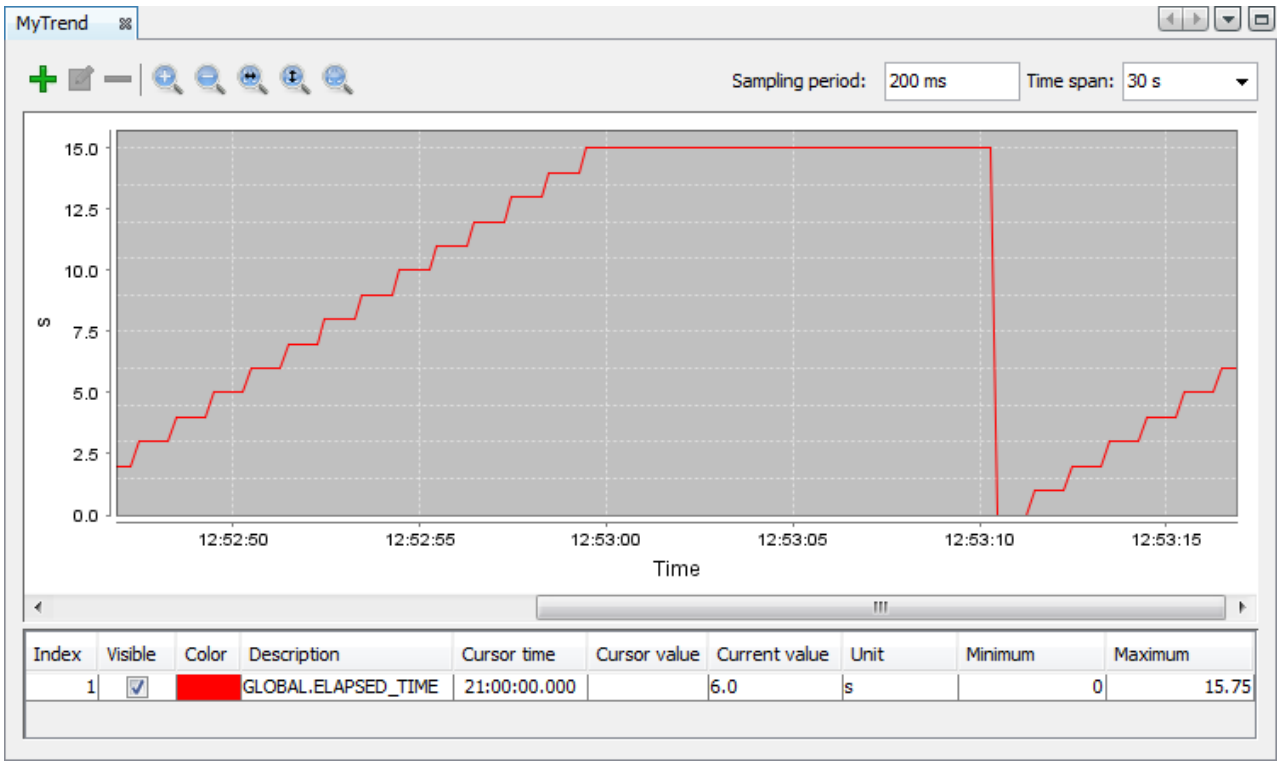
1. Creation of a new trend file.



2. Addition and configuration of the channels on the button on the upper left corner.



3. After adding the channel, just click on **Connect Device** and the trend will start automatically.

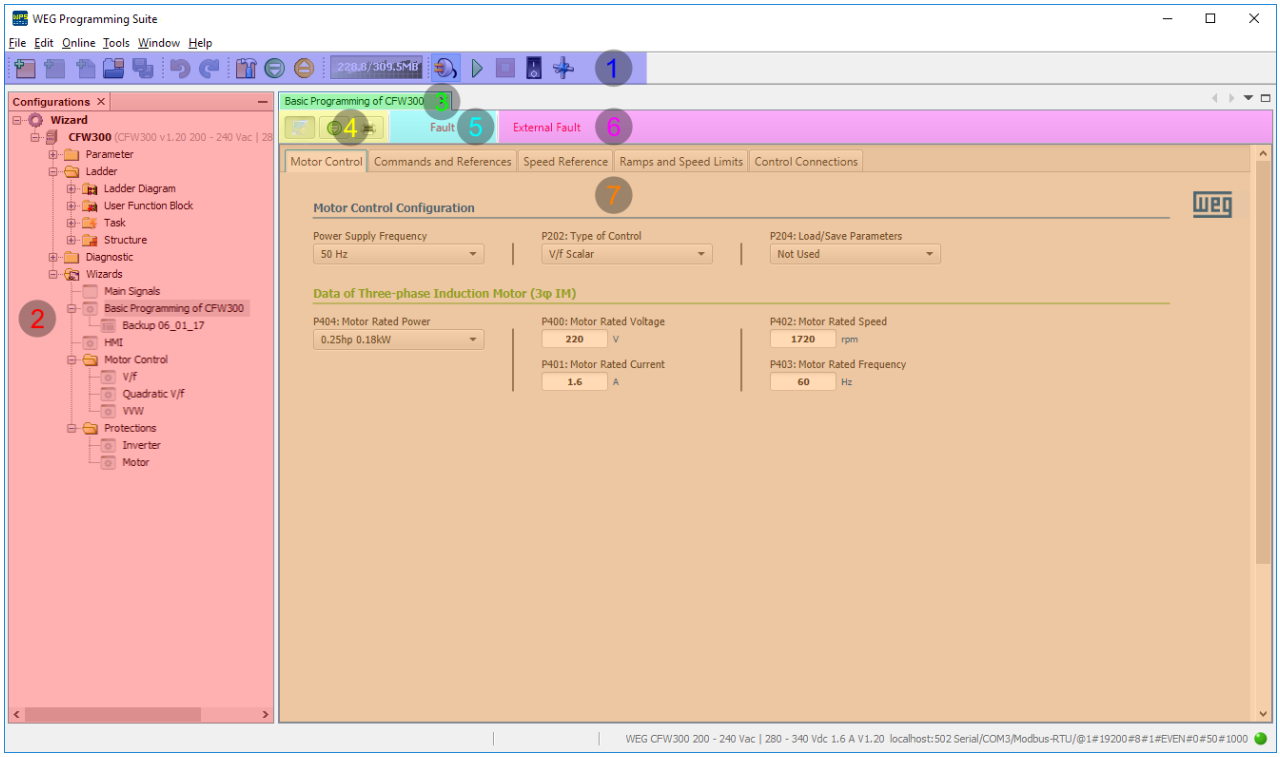

MyTrend
Sampling period:
200 ms
Time span:
30 s
15.0
12.5
10.0
7.5
5.0
2.5
0.0
s
12:52:50
12:52:55
12:53:00
12:53:05
12:53:10
12:53:15
Time
Index
Visible
Color
Description
Cursor time
Cursor value
Current value
Unit
Minimum
Maximum
1
GLOBAL.ELAPSED_TIME
21:00:00.000
6.0
s
0
15.75

10 Wizards

10.1 Overview

Wizards allow the user to configure and monitor variables of the configured equipment in the resource through predefined windows.

Below is an overview of the **Wizards** in WPS v2.5X.



1. **Toolbar**: displays the main commands of the software and contains the **Connect Device** button  required for writing and reading the equipment variables;
2. **Configurations Window**: displays the configurations, where each configuration contains resources, and the resource contains the monitoring and configuration wizards;
3. **Wizard Title**: displays the title of the open wizard;
4. **Wizard Toolbar**: displays the commands for upload and monitoring  , writing  and printing  values of a configured wizard variables (**Configuration Wizards** only);
5. **Equipment Status**: displays the status of the connected equipment (**Configuration Wizards** only);
6. **Message from an Equipment Status**: displays the message of an alarm, failure or configuration status of the connected equipment (**Configuration Wizards** only);
7. **Equipment Variable Area**: displays the writing and reading variables of the configured equipment and can contain tabs for better division of the equipment's features.

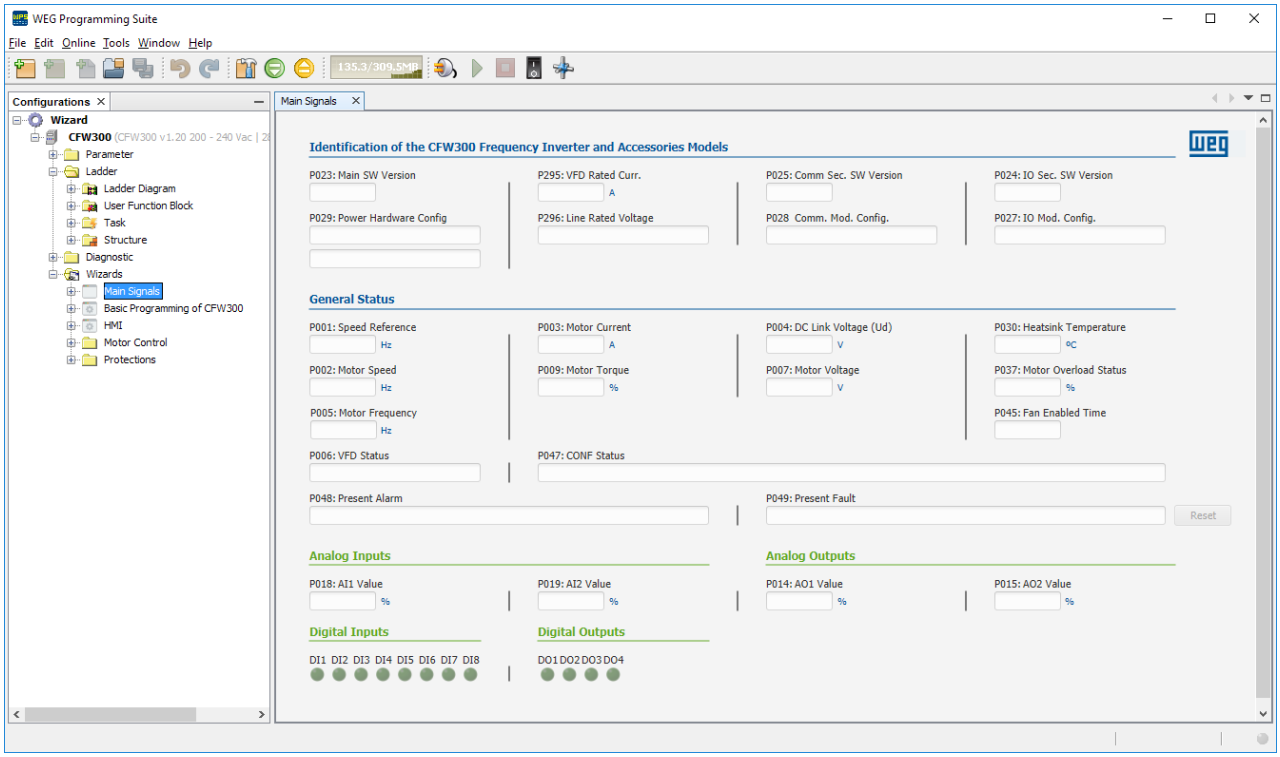
10.2 Monitoring Wizard

A **Monitoring Wizard** basically allows the user to read the equipment variables that WPS v2.5X is connected

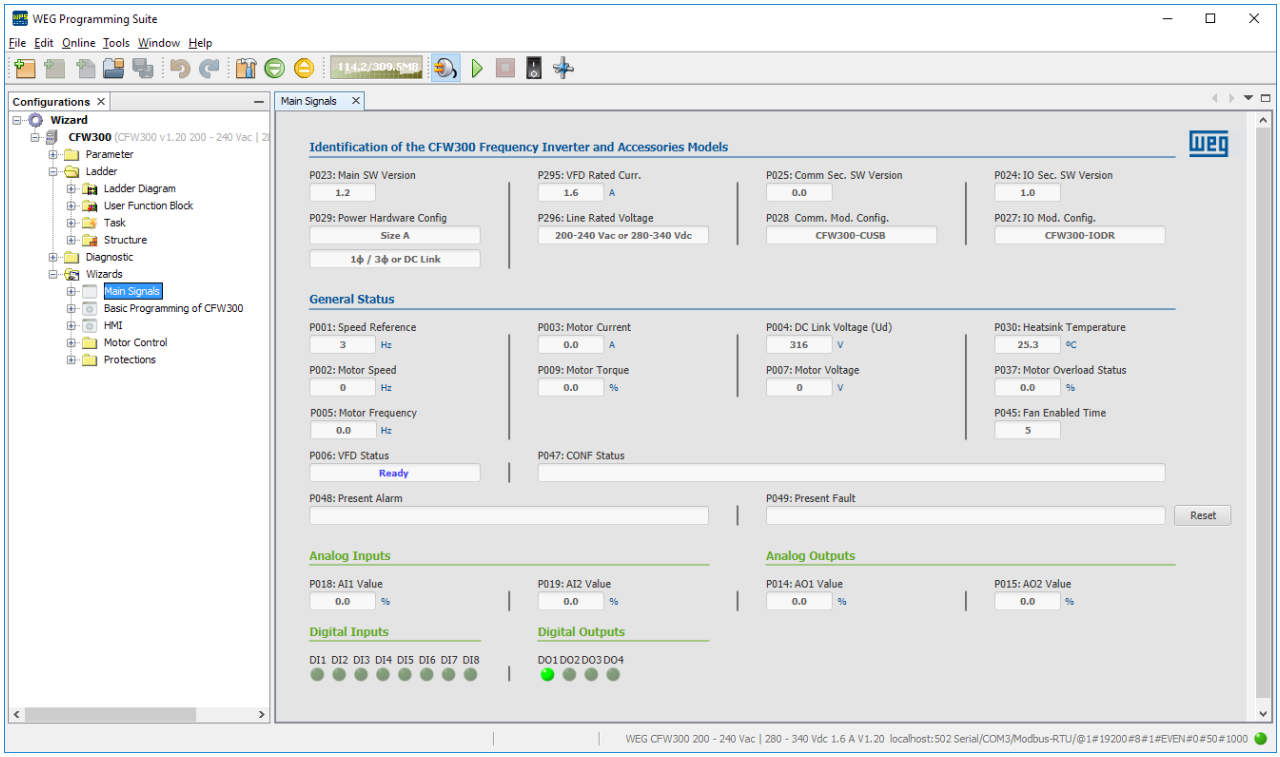
to.

The **Wizards** folder in the **Configurations Window** contains the list of wizards implemented for the equipment being a **Monitoring Wizard** identified by the icon .

When opening a **Monitoring Wizard** with communication with the equipment not established, a window will be presented to the user as below.



To start monitoring the window variables, press the **Connect Device** button  so that the communication with the device is established.



If the communication with the equipment is established the window background will change from light gray to dark gray. When closing the window no questioning will be done to the user.

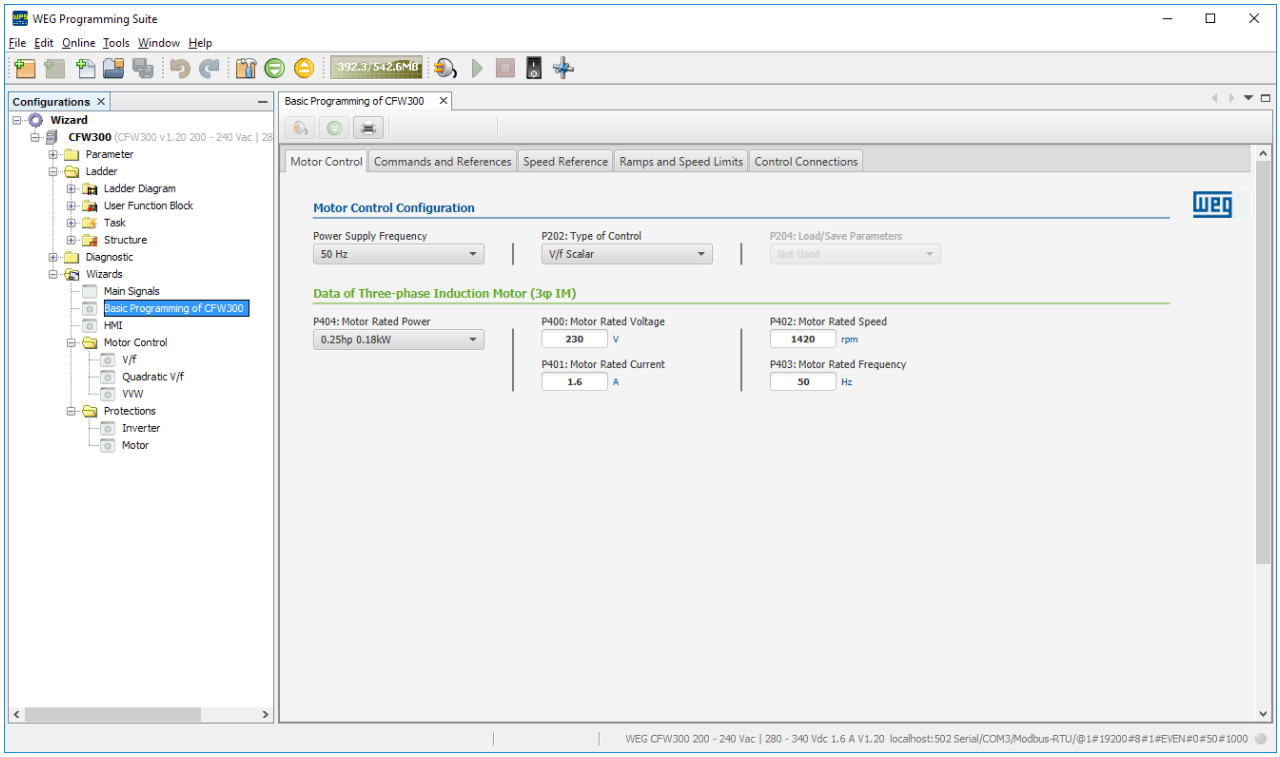
10.3 Configuration Wizard

A **Configuration Wizard** basically allows the user to configure the equipment's writing variables with WPS v2.5X connected (online) or not connected (offline) to the device.

The **Wizards** folder in the **Configurations Window** contains the list of wizards implemented for the equipment being a **Configuration Wizard** identified by the icon .

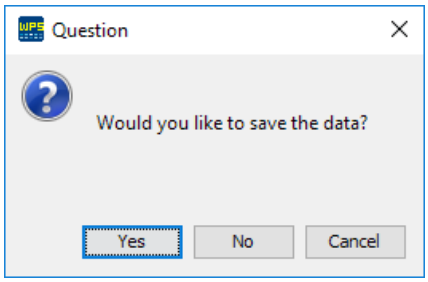
You can save the programmed values in a **Configuration Wizard** thus generating a new wizard file identified by the icon .

When opening a **Configuration Wizard** with communication with the equipment not established, a window will be presented to the user as below.

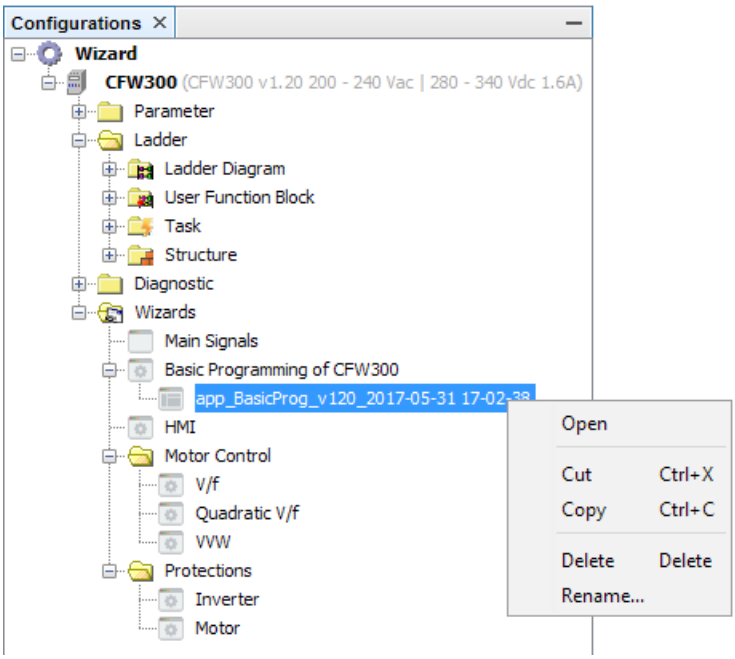


The configuration with the equipment not connected (offline) is done with the **Upload and Monitor Values** button  not active.

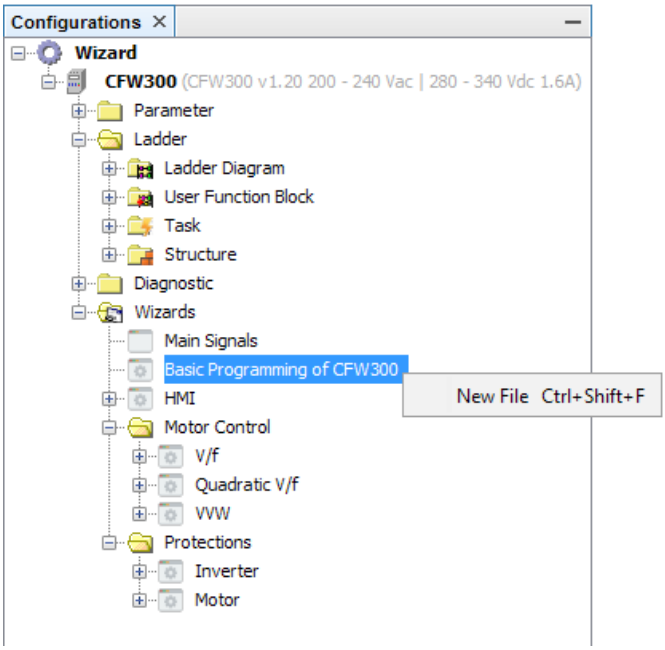
When the configuration is finished, the user can save the values of the wizard by closing the wizard and confirming the saving of the values.



A new wizard file with a generic name will then be saved, where the user can then manipulate this new file according to the options shown by right-clicking the file as shown below.



You can also create a new file based on the equipment wizard by right-clicking the wizard as shown below.



To print all the values of the **Configuration Wizard** you need to press the **Print Values** button  in the **Wizard Toolbar**. A print file will be generated in the format shown below.

Wizard - CFW300 - Basic Programming of CFW300 - 31/05/2017 17:42 - 1 / 2

PARAMETERS FOR BASIC PROGRAMMING OF CFW300 FREQUENCY INVERTER

Motor Control Configuration

PAR_202 (Type of Control): 0 - V/f Scalar

Data of Three-phase Induction Motor (3φ IM)

PAR_404 (Motor Rated Power): 1 - 0.25hp 0.18kW
PAR_400 (Motor Rated Voltage): 230 V
PAR_401 (Motor Rated Current): 1.6 A
PAR_402 (Motor Rated Speed): 1420 rpm
PAR_403 (Motor Rated Frequency): 50 Hz

Source Commands Configuration

PAR_220 (LOC/REM Selection Src): 0 - Always LOC

LOCAL Situation

PAR_221 (LOC Reference Sel.): 0 - HMI
PAR_224 (LOC Run/Stop Sel.): 0 - HMI Keys
PAR_223 (LOC FWD/REV Selection): 0 - Always FWD
PAR_225 (LOC JOG Selection): 1 - Not Used

REMOTE Situation

PAR_222 (REM Reference Sel.): 1 - AI1
PAR_227 (REM Run/Stop Sel.): 1 - DIx
PAR_226 (REM FWD/REV Selection): 4 - DIx
PAR_228 (REM JOG Selection): 2 - DIx

Speed Reference via HMI

PAR_121 (Keypad Reference): 3.0 Hz

Ramps and Speed Limits

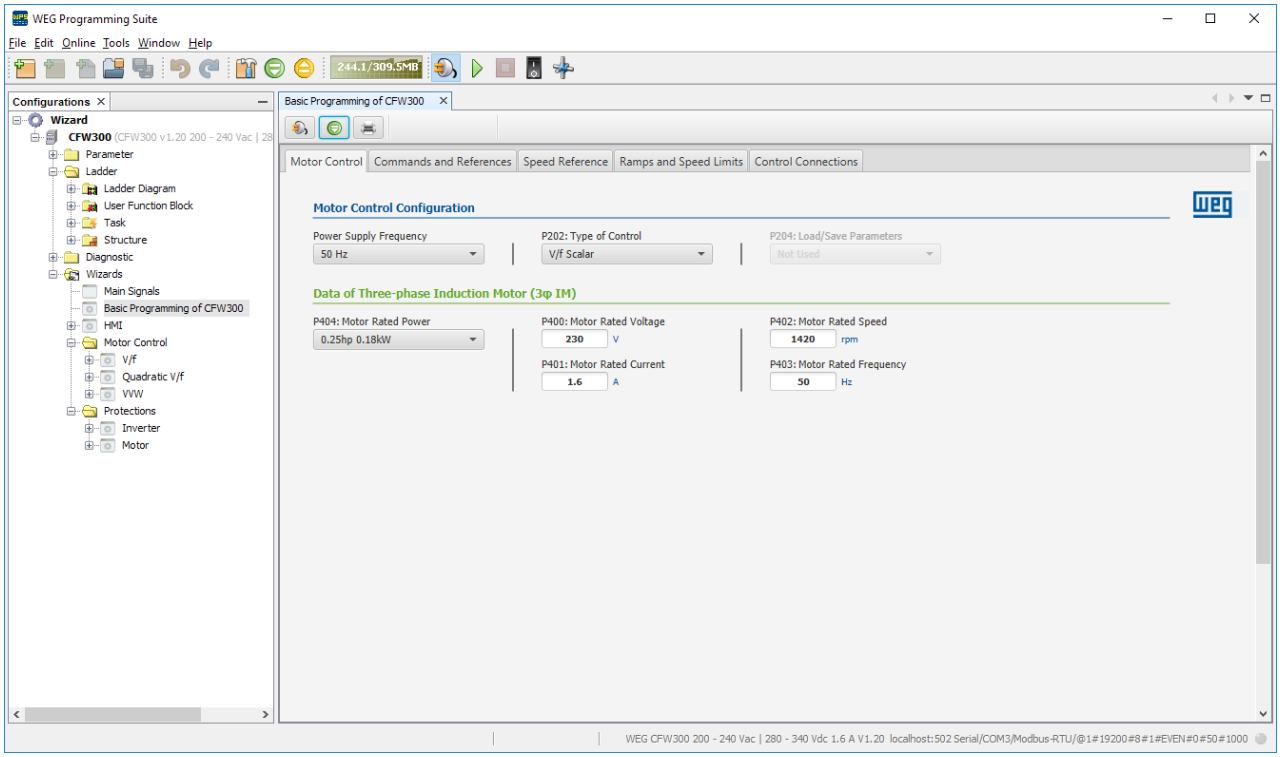
PAR_120 (Speed Ref. Backup): 1 - On
PAR_133 (Minimum Ref.): 3.0 Hz
PAR_134 (Maximum Ref.): 55.0 Hz
PAR_100 (Acceleration Time): 5.0 s
PAR_101 (Deceleration Time): 10.0 s

Control Connections - CFW300 Control Board

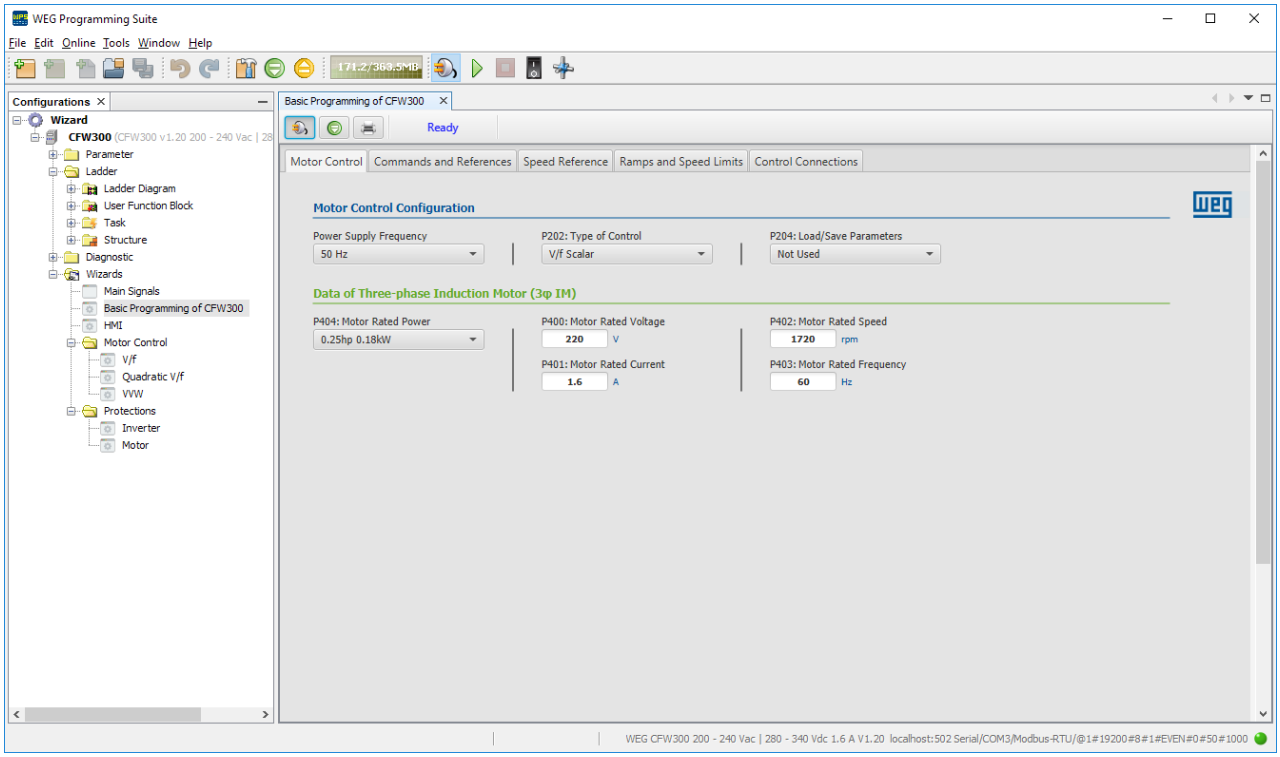
Digital Inputs

PAR_263 (DI1 Function): 1 - Run/Stop
PAR_264 (DI2 Function): 8 - Direction of Rotation
PAR_265 (DI3 Function): 0 - Not Used
PAR_266 (DI4 Function): 0 - Not Used
PAR_271 (DIs Function): 0 - (DI1..DI8)NPN

To send all the values of the **Configuration Wizard** to the equipment it is necessary to press the **Connect Device** button  so that the communication with the equipment is established and then press the **Write Values** button  in the **Wizard Toolbar**.



To start a **Configuration Wizard** with the connected equipment (online) it is necessary to press the **Monitor Values** button  so that the monitoring of the equipment variables is activated in the wizard. At this point the wizard variables are updated with the values that are on the connected equipment.



If the communication with the equipment is established the window background will change from light gray to dark gray. When closing the window will ask a question related to saving the values of the wizard.

11 Equipments (Devices)

11.1 CFW100

11.1.1 Description

The CFW100 frequency inverter is a high-performance product which allows speed control of three-phase induction motors. This product provides the user with the options of vector (VVW) or scalar (V/f) control, both programmable according to the application.

The scalar mode (V/f) is recommended for simpler applications, such as the activation of most pumps and fans. In such cases it is possible to reduce the losses in the motor and the inverter using the "V/f Quadratic", which results in energy savings. The V/f mode is used when more than a motor is activated by an inverter simultaneously (multimotor applications). In the vector mode (VVW), the operation is optimized for the motor in use, obtaining a better performance in terms of speed regulation.

The frequency inverter CFW100 also has functions of PLC (Programmable Logic Controller) by means of the SoftPLC (integrated) feature. It has a slot for connection of the accessories for input and output (I/Os) expansion, communication networks or remote HMI.

Refer to the user's manual of the CFW100 for further details about the product.

NOTE!

CFW100 versions below V3.00 do not have the Ladder tool available in WPS.
You can use the WLP application if this feature is required.

11.1.2 System Markers

The following variables contained in the **GLOBAL_SYSTEM** group of the variables table, have the fixed tag. The tag of system markers were divided into groups and subgroups, where:

Groups:
- CFW: reading and writing variables of the CFW100 frequency inverter.

Subgroups:
- STS: reading variable (status);
- CMD: writing variable (command).

Reading System Markers (Status)

Reading - Function Modbus 02 "Read Discrete Inputs"

Address	Bit	Modbus	Tag	Description
Ladder				
%SB6000	0	0	SYS_FREQ_2HZ	Oscillator with frequency of 2 Hz

%SB6000	1	1	SYS_PULSE_1SCAN	Pulse during the first scan cycle
%SB6000	2	2	SYS_FALSE	Always in 0
%SB6000	3	3	SYS_TRUE	Always in 1
Logical Status				
%SB6002	1	17	CFW_STS_RUN_COMMAND	The run motor command is active in the inverter
%SB6002	2	18	CFW_STS_FIRE_MODE_ACTIVE	Fire Mode Function is active
%SB6002	5	21	CFW_STS_SEC_RAMP_ACTIVE	The inverter is configured to use the first or second ramp values (0-First, 1-Second)
%SB6002	6	22	CFW_STS_CONFIG_MODE	The inverter is in the configuration mode
%SB6002	7	23	CFW_STS_ALARM_ACTIVE	The inverter is in alarm condition
%SB6003	0	24	CFW_STS_MOTOR_RUNNING	The inverter is running the motor at the speed reference, or executing either the acceleration or the deceleration ramp
%SB6003	1	25	CFW_STS_GENERAL_ENABLED	General Enable is active and the inverter is ready to run the motor
%SB6003	2	26	CFW_STS_FWD_REV_DIRECTION	The motor is running in the reverse or forward direction (0-Reverse, 1-Forward)
%SB6003	3	27	CFW_STS_JOG_ACTIVE	The JOG function is active
%SB6003	4	28	CFW_STS_LOC_REM_MODE	The inverter is in local or remote mode (0-Local, 1-Remote)
%SB6003	5	29	CFW_STS_UNDERVOLTAGE	The inverter is in undervoltage
%SB6003	7	31	CFW_STS_FAULT_ACTIVE	The inverter has detected a fault
%SB6004	0	32	CFW_STS_AI1_BROKEN_CABLE	It indicates that the signal of analog input AI1 in 4 to 20 mA or 20 to 4 mA is below 2 mA
HMI keys				
%SB6006	0	48	CFW_STS_KEY_START_STOP	START/STOP key (I)/(0) pressed
%SB6006	2	50	CFW_STS_KEY_UP	UP key pressed
%SB6006	3	51	CFW_STS_KEY_DOWN	DOWN key pressed
Infrared Remote Control (IRC 1)				
%SB6010	0	80	CFW_STS_IRC_1_KEY_ON	Start/Stop Motor key pressed
%SB6010	1	81	CFW_STS_IRC_1_KEY_DOWN	Browse Downwards key pressed
%SB6010	2	82	CFW_STS_IRC_1_KEY_UP	Browse Upwards key pressed
%SB6010	3	83	CFW_STS_IRC_1_KEY_CHANGE	Commute view key pressed. This key allows commute view between two parameters (values) defined by parameters P842 and P843
%SB6010	4	84	CFW_STS_IRC_1_KEY_P	Confirm/Program key pressed

%SB6010	5	85	CFW_STS_IRC_1_KEY_SFK1	Special Function key 1 pressed
%SB6010	6	86	CFW_STS_IRC_1_KEY_SFK2	Special Function key 2 pressed
%SB6010	7	87	CFW_STS_IRC_1_KEY_SFK3	Special Function key 3 pressed

Reading - Function Modbus 04 "Read Input Registers"

Speed				
% SW6200	--	3100	CFW_STS_MOTOR_SPEED_13BITS	Motor speed in 13 bits (8192)
% SW6202	--	3101	CFW_STS_MOTOR_SYNC_SPEED	Motor synchronous speed in rpm
% SW6204	--	3102	CFW_STS_MOTOR_SPEED_RPM	Motor speed in rpm
% SW6206	--	3103	CFW_STS_SPEED_REFERENCE	Speed reference after ramp in rpm
Alarm and Fault				
% SW6208	--	3104	CFW_STS_PRES_ALARM	Alarm number that may be present in the inverter
% SW6210	--	3105	CFW_STS_PRES_FAULT	Fault number that may be present in the inverter
Current and Torque				
% SW6212	--	3106	CFW_STS_RATED_CURRENT	Inverter rated current (HD) in A (x10)
% SW6214	--	3107	CFW_STS_MOTOR_CURRENT	Motor current without filter in A (x10)
% SW6216	--	3108	CFW_STS_MOTOR_TORQUE	Motor torque without filter in % (x10)

Writing / Reading System Markers (Command)

Reading - Function Modbus 01 "Read Coils"
Writing - Function Modbus 05 "Write Single Coil" and 15 "Write Multiple Coils"

Address	Bit	Modbus	Tag	Description
Logical Command				
%CB6008	0	16	CFW_CMD_RUN_STOP	Run the motor according to the speed reference value (0-Stop, 1-Run)
%CB6008	1	17	CFW_CMD_GENERAL_ENABLE	Enables the inverter allowing the motor operation (0-Disable, 1-Enable)
%CB6008	2	18	CFW_CMD_SPEED_DIRECTION	The motor runs in the direction indicated by the speed reference (0-Reverse, 1-Forward)
%CB6008	3	19	CFW_CMD_JOG	Enables the JOG function (0-Disable, 1-Enable)
%CB6008	4	20	CFW_CMD_LOC_REM	Selects the inverter operation mode (0-Local, 1-Remote)
%CB6008	5	21	CFW_CMD_SECOND_RAMP	Selects the ramp to accelerate and decelerate the motor (0-First, 1-Second)
%CB6008	6	22	CFW_CMD_FORCE_RUN_STOP_SPLC	It allows that the SoftPLC command CFW_CMD_RUN_STOP change the inverter command Run/Stop regardless of source programmed for Start/Stop via P224 or P227
%CB6008	7	23	CFW_CMD_FAULT_RESET	Executes the fault reset command

11.1.3 I/O's

Hardware information can be found in the Manual of the CFW100 at the website www.weg.net.

Digital Inputs

Address	Bit	Modbus	Tag	Description
%IB0	0	16000	DI1	Digital input 1
%IB0	1	16001	DI2	Digital input 2
%IB0	2	16002	DI3	Digital input 3
%IB0	3	16003	DI4	Digital input 4
%IB0	4	16004	DI5	Digital input 5 - I/O expansion module
%IB0	5	16005	DI6	Digital input 6 - I/O expansion module
%IB0	6	16006	DI7	Digital input 7 - I/O expansion module
%IB0	7	16007	DI8	Digital input 8 - I/O expansion module

Analog Inputs

Address	Bit	Modbus	Tag	Description
%IW2	--	5001	AI1	Analog input 1 - I/O expansion module
%IW6	--	5003	AIP	Analog input (Potentiometer) - I/O expansion module
%IW8	--	5004	FI1	Frequency Input 1

Digital Outputs

Address	Bit	Modbus	Tag	Description
%QB0	0	16000	DO1	Digital output 1 - I/O expansion module
%QB0	1	16001	DO2	Digital output 2 - I/O expansion module
%QB0	2	16002	DO3	Digital output 3 - I/O expansion module

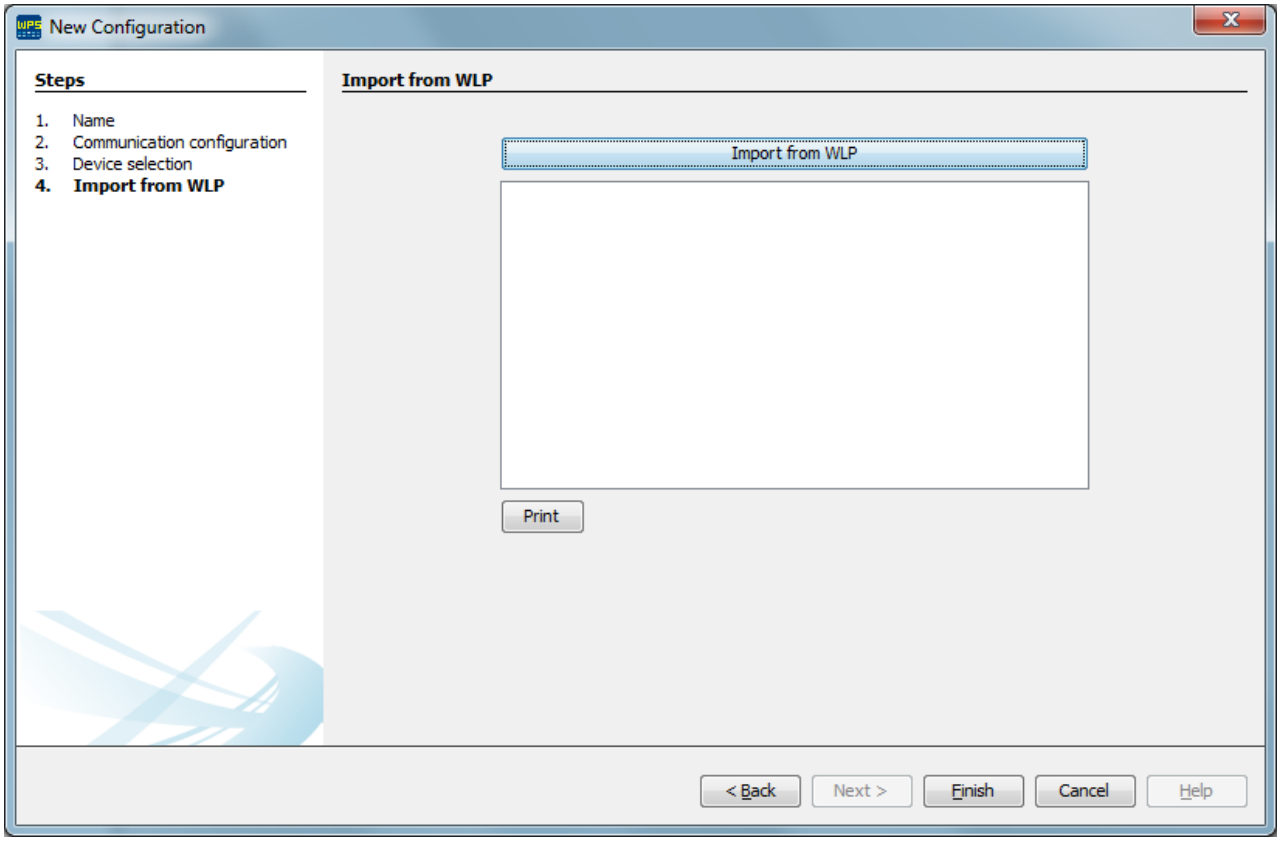
Analog Outputs

Address	Bit	Modbus	Tag	Description
%QW2	--	5001	AO1	Analog output 1 - I/O expansion module

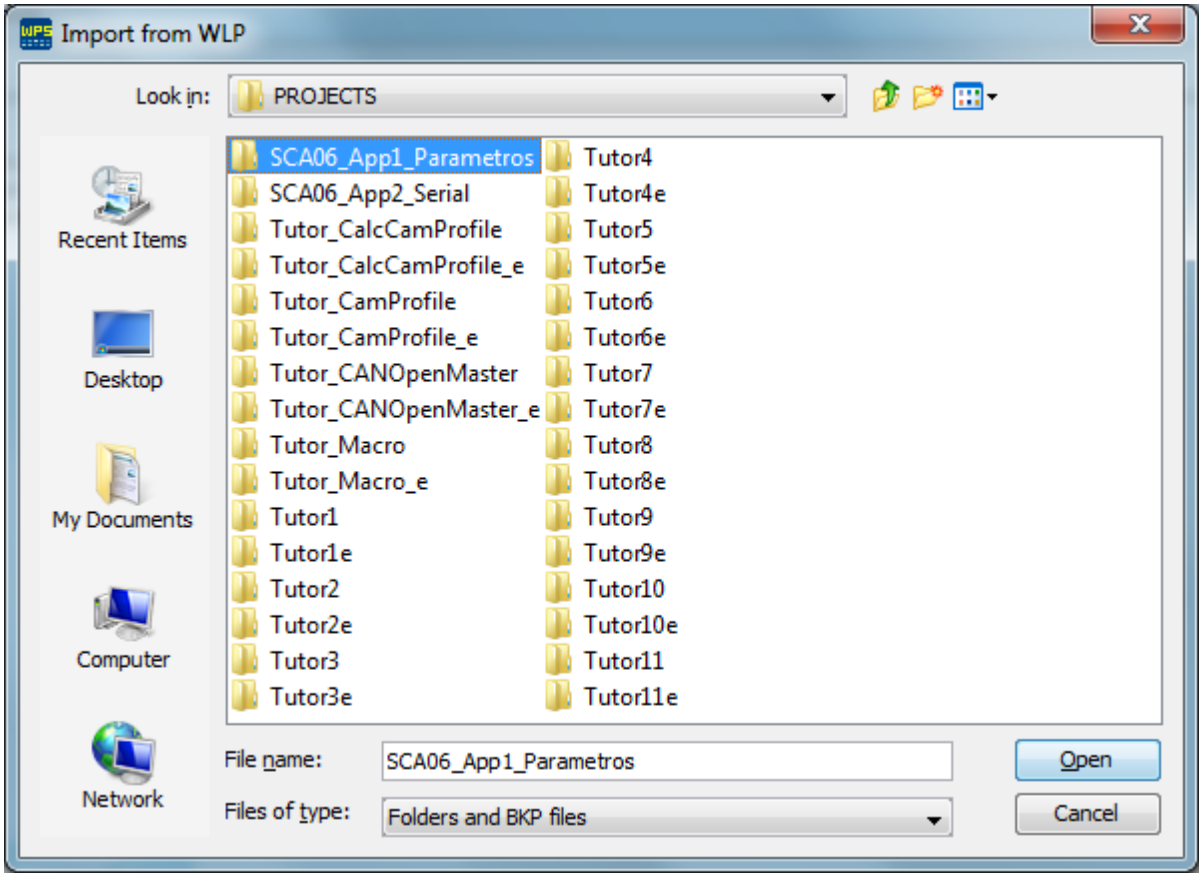
11.1.4 Import from WLP

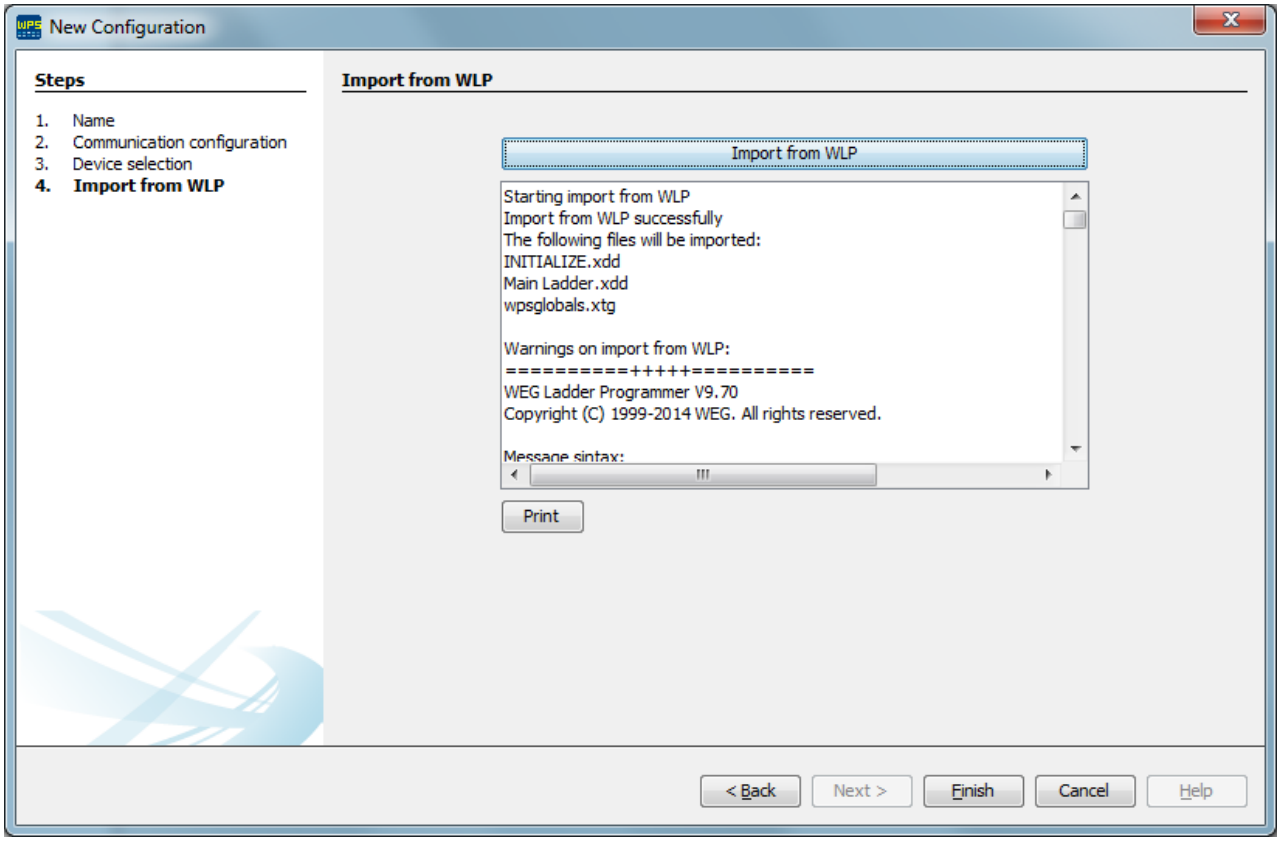
The function import from WLP is utilized to import Ladder developed on WLP software to equipment (device).

The import from WLP can be executed during the resource creation.



1. To execute the import WLP function click the Import from WLP button and select the WLP project folder or the WLP BKP file.


Import from WLP
Look in:
PROJECTS
SCA06_App1_Parametros
SCA06_App2_Serial
Tutor_CalcCamProfile
Tutor_CalcCamProfile_e
Tutor_CamProfile
Tutor_CamProfile_e
Tutor_CANOpenMaster
Tutor_CANOpenMaster_e
Tutor_Macro
Tutor_Macro_e
Tutor1
Tutor1e
Tutor2
Tutor2e
Tutor3
Tutor3e
Tutor4
Tutor4e
Tutor5
Tutor5e
Tutor6
Tutor6e
Tutor7
Tutor7e
Tutor8
Tutor8e
Tutor9
Tutor9e
Tutor10
Tutor10e
Tutor11
Tutor11e
Recent Items
Desktop
My Documents
Computer
Network
File name:
SCA06_App1_Parametros
Files of type:
Folders and BKP files
Open
Cancel



2. After import from WLP completed successfully click the Finish button to copy the imported files to new resource.

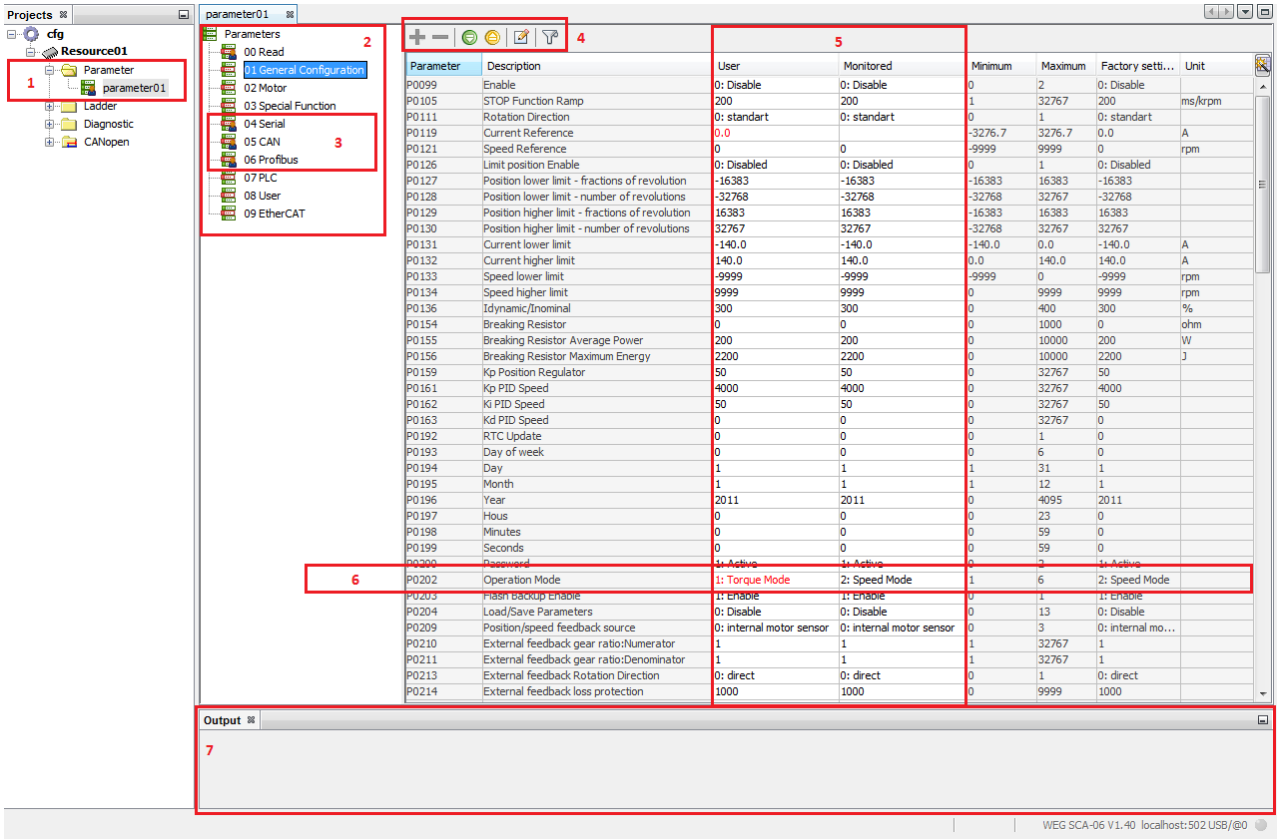
11.1.5 Parameters

11.1.5.1 Overview

The parameter configuration screen is used to configure and monitor all the parameters of the equipment, including the user parameters.

NOTE!
The reading and writing of such parameters is done on this screen; only the user parameter configuration must be sent the first time or whenever modified by means of the resource download routine.

Below is an overview of the parameter configuration screen.



Parameter	Description	User	Monitored	Minimum	Maximum	Factory setti...	Unit
P0099	Enable	0: Disable	0: Disable	0	2	0: Disable	
P0105	STOP Function Ramp	200	200	1	32767	200	ms/krpm
P0111	Rotation Direction	0: standart	0: standart	0	1	0: standart	
P0119	Current Reference	0.0		-3276.7	3276.7	0.0	A
P0121	Speed Reference	0	0	-9999	9999	0	rpm
P0126	Limit position Enable	0: Disabled	0: Disabled	0	1	0: Disabled	
P0127	Position lower limit - fractions of revolution	-16383	-16383	-16383	16383	-16383	
P0128	Position lower limit - number of revolutions	-32768	-32768	-32768	32767	-32768	
P0129	Position higher limit - fractions of revolution	16383	16383	-16383	16383	16383	
P0130	Position higher limit - number of revolutions	32767	32767	-32768	32767	32767	
P0131	Current lower limit	-140.0	-140.0	-140.0	0.0	-140.0	A
P0132	Current higher limit	140.0	140.0	0.0	140.0	140.0	A
P0133	Speed lower limit	-9999	-9999	-9999	0	-9999	rpm
P0134	Speed higher limit	9999	9999	0	9999	9999	rpm
P0136	Idynamic/Inominal	300	300	0	400	300	%
P0154	Breaking Resistor	0	0	0	1000	0	ohm
P0155	Breaking Resistor Average Power	200	200	0	10000	200	W
P0156	Breaking Resistor Maximum Energy	2200	2200	0	10000	2200	J
P0159	Kp Position Regulator	50	50	0	32767	50	
P0161	Kp PID Speed	4000	4000	0	32767	4000	
P0162	Ki PID Speed	50	50	0	32767	50	
P0163	Kd PID Speed	0	0	0	32767	0	
P0192	RTC Update	0	0	0	1	0	
P0193	Day of week	0	0	0	6	0	
P0194	Day	1	1	1	31	1	
P0195	Month	1	1	1	12	1	
P0196	Year	2011	2011	0	4095	2011	
P0197	Hous	0	0	0	23	0	
P0198	Minutes	0	0	0	59	0	
P0199	Seconds	0	0	0	59	0	
P0200	Password	1: Active	1: Active	0	2	1: Active	
P0202	Operation Mode	1: Torque Mode	2: Speed Mode	1	6	2: Speed Mode	
P0203	Flash Backup Enable	1: Enable	1: Enable	0	1	1: Enable	
P0204	Load/Save Parameters	0: Disable	0: Disable	0	13	0: Disable	
P0209	Position/speed feedback source	0: internal motor sensor	0: internal motor sensor	0	3	0: internal mo...	
P0210	External feedback gear ratio:Numerator	1	1	1	32767	1	
P0211	External feedback gear ratio:Denominator	1	1	1	32767	1	
P0213	External feedback Rotation Direction	0: direct	0: direct	0	1	0: direct	
P0214	External feedback loss protection	1000	1000	0	9999	1000	

1. **Parameter files**. In this part are all the parameter configuration files created by the user. Notice that when the file features a person figure on the table, it means this parameter table contains hidden parameters/ group of parameters.
2. **Group of parameters**. This tree shows all the group of parameters. Notice that the same parameter can be in more than one group, and when its value is modified, it will be modified in all the groups to which it belongs.
3. **Modified group of parameters**. Group of parameters which contain the figure of a person on the table means they have hidden parameters.
4. **Commands**. The commands are described below in the order they appear:
   - 4.1. **Unhide parameter**: In case some parameter has been hidden, this button allows making it visible again.
   - 4.2. **Hide parameter**: Just select one or more parameters on the table and trigger this command to hide them.
   - 4.3. **Save table**: It saves the values of the parameters shown on the equipment screen; the sent values are the ones in the User column. The flow is User -> Monitored (equipment)
   - 4.4. **Read table**: It reads the parameters of the equipment shown in the Monitored column and saves them in the parameter file in the User column. The flow is Monitored (equipment) -> User
   - 4.5. **User parameters**: It opens a screen to edit the user parameters.
   - 4.6. **Filter**: It opens a parameter filter option, and it can filter by parameter number or description.
   - 4.7. **User Parameters and Monitored Parameters**. These two columns show the off-line and on-line parameters, so to speak. The **User** column shows the values contained in the file located on the computer and the **Monitored** column shows the values that are effectively saved on the equipment. Whenever you use the **Save Parameter** option, the sent values will be from the **User** column to the **Monitored** column, that is, File -> Equipment. In case of reading, the flow is the opposite, from the **Monitored** column to the **User** column, that is, Equipment -> File. In case you wish to change the values directly on the equipment without changing it in the file, just click on the monitored column,

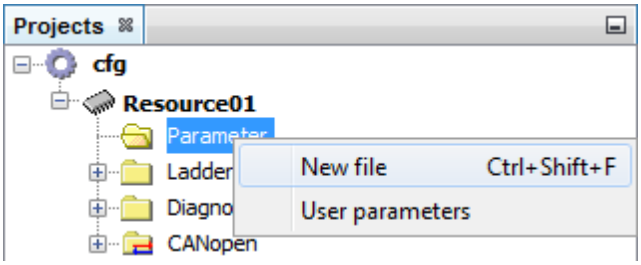
change the values and the modification will occur on-line.

5. **Modified parameters**: Whenever a parameter value in the **User** column is different from the **Monitored** column, it will be shown in red.
6. **Output**. This screen shows error information in case they occur during the writing or reading of the parameters.

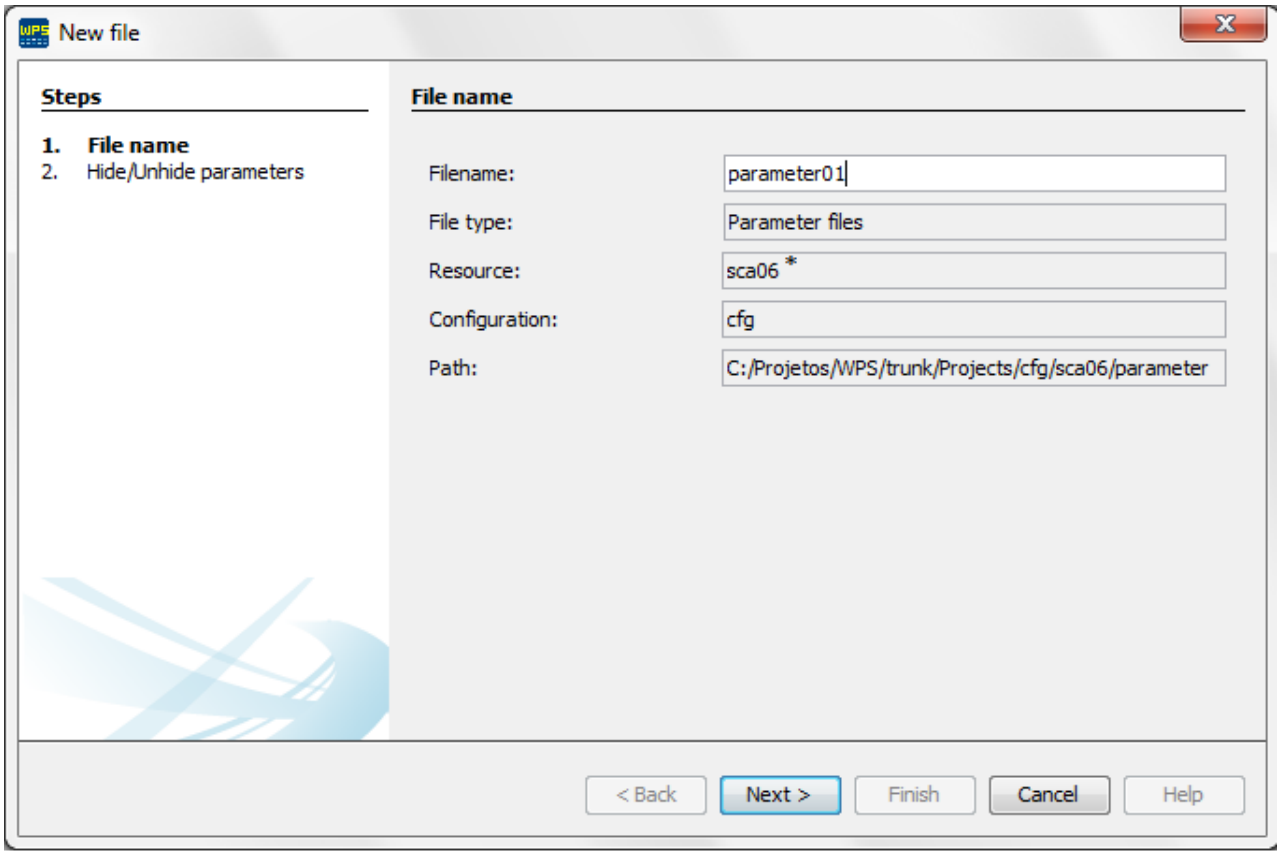
11.1.5.2 Configuration

Below is the list of the required steps to create a parameter file.

1. Create a new parameter file.

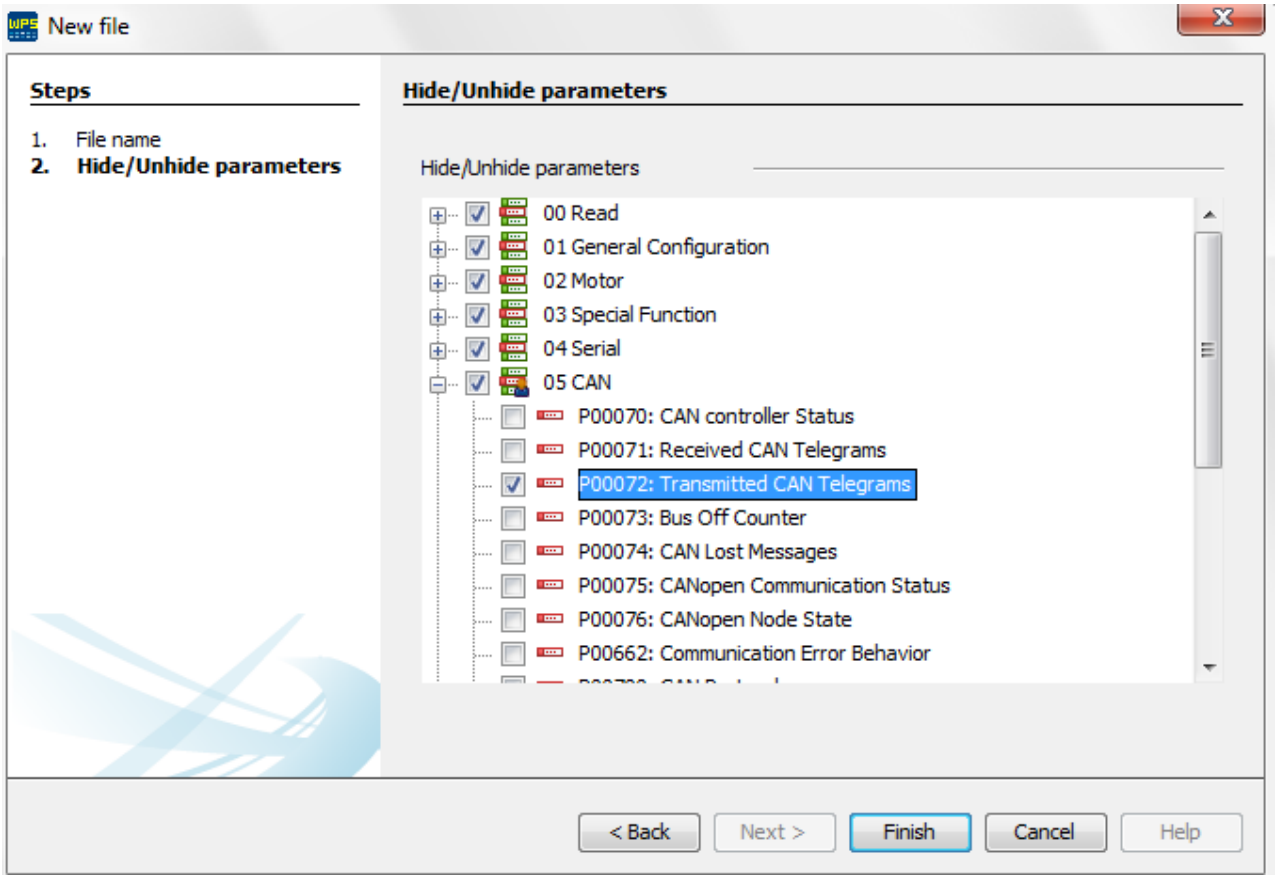


2. Define a name for the parameter file

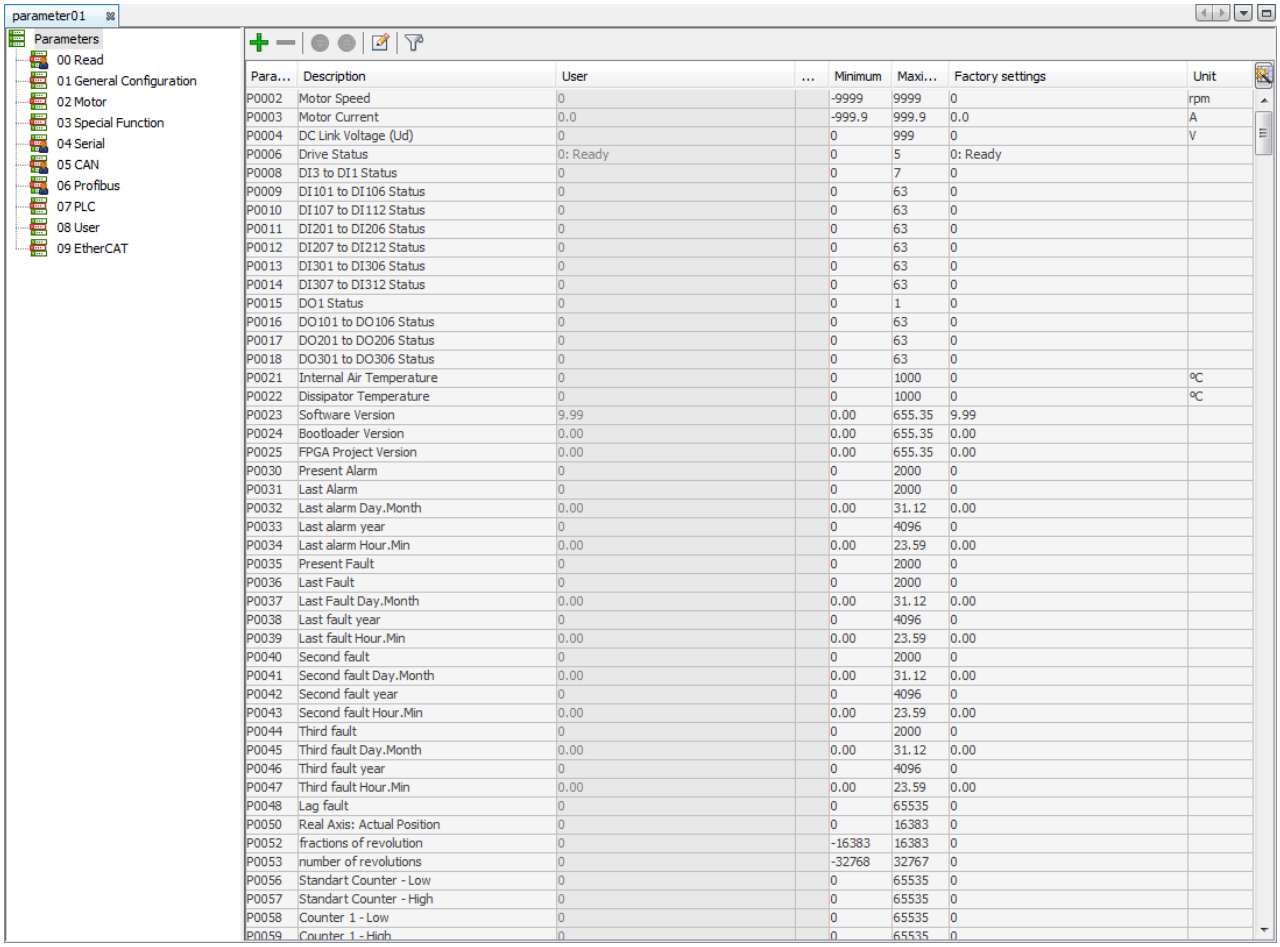


*** Resource: Resource01, SCA06, CFW300, etc.**

3. Configure which parameters you wish to view in your parameter table



4. After performing the steps above, the parameter file will be created and the equipment can be parameterized.

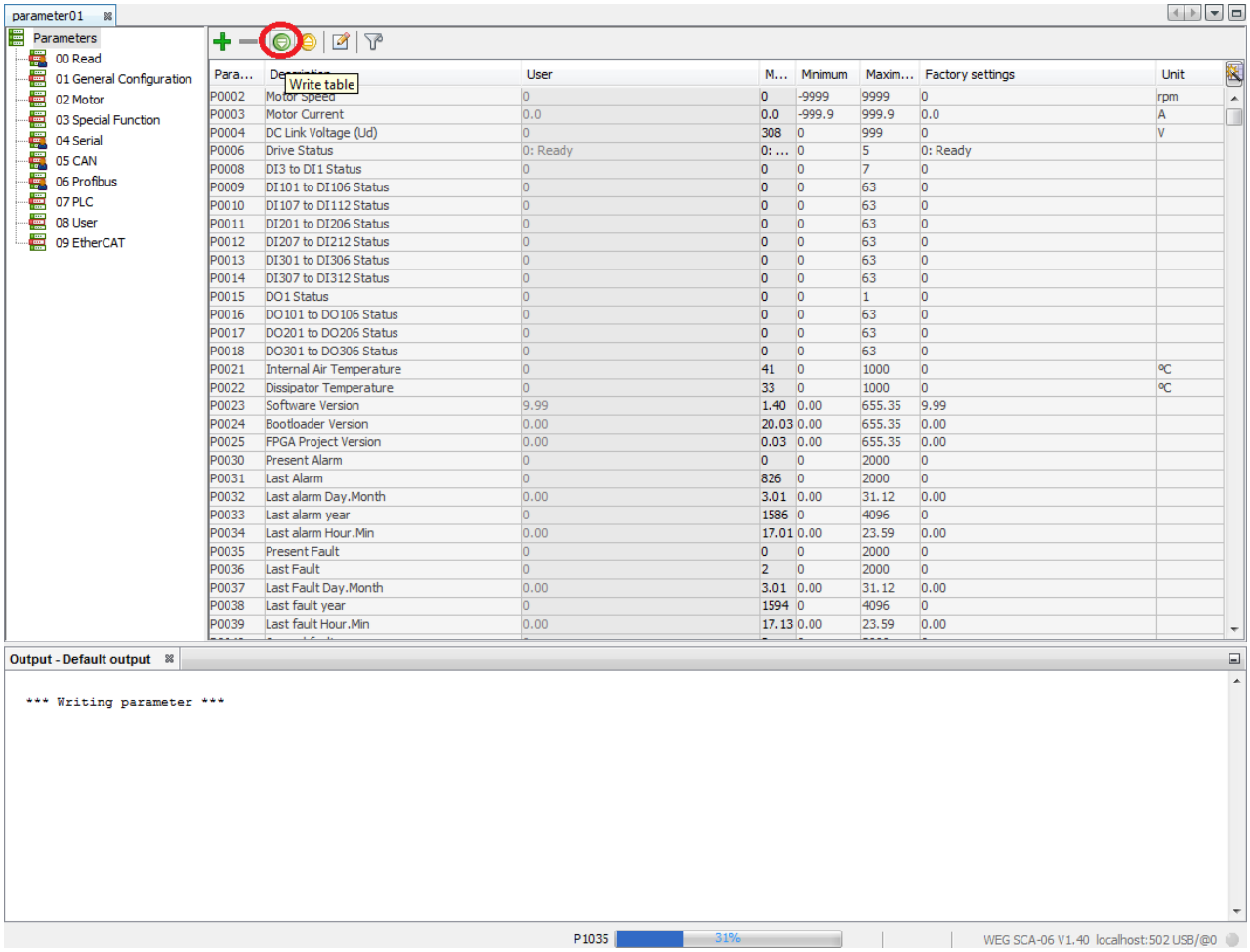


Para...	Description	User	...	Minimum	Maxi...	Factory settings	Unit
P0002	Motor Speed	0		-9999	9999	0	rpm
P0003	Motor Current	0.0		-999.9	999.9	0.0	A
P0004	DC Link Voltage (Ud)	0		0	999	0	V
P0006	Drive Status	0: Ready		0	5	0: Ready	
P0008	DI3 to DI1 Status	0		0	7	0	
P0009	DI101 to DI106 Status	0		0	63	0	
P0010	DI107 to DI112 Status	0		0	63	0	
P0011	DI201 to DI206 Status	0		0	63	0	
P0012	DI207 to DI212 Status	0		0	63	0	
P0013	DI301 to DI306 Status	0		0	63	0	
P0014	DI307 to DI312 Status	0		0	63	0	
P0015	DO1 Status	0		0	1	0	
P0016	DO101 to DO106 Status	0		0	63	0	
P0017	DO201 to DO206 Status	0		0	63	0	
P0018	DO301 to DO306 Status	0		0	63	0	
P0021	Internal Air Temperature	0		0	1000	0	ºC
P0022	Dissipator Temperature	0		0	1000	0	ºC
P0023	Software Version	9.99		0.00	655.35	9.99	
P0024	Bootloader Version	0.00		0.00	655.35	0.00	
P0025	FPGA Project Version	0.00		0.00	655.35	0.00	
P0030	Present Alarm	0		0	2000	0	
P0031	Last Alarm	0		0	2000	0	
P0032	Last alarm Day.Month	0.00		0.00	31.12	0.00	
P0033	Last alarm year	0		0	4096	0	
P0034	Last alarm Hour.Min	0.00		0.00	23.59	0.00	
P0035	Present Fault	0		0	2000	0	
P0036	Last Fault	0		0	2000	0	
P0037	Last Fault Day.Month	0.00		0.00	31.12	0.00	
P0038	Last fault year	0		0	4096	0	
P0039	Last fault Hour.Min	0.00		0.00	23.59	0.00	
P0040	Second fault	0		0	2000	0	
P0041	Second fault Day.Month	0.00		0.00	31.12	0.00	
P0042	Second fault year	0		0	4096	0	
P0043	Second fault Hour.Min	0.00		0.00	23.59	0.00	
P0044	Third fault	0		0	2000	0	
P0045	Third fault Day.Month	0.00		0.00	31.12	0.00	
P0046	Third fault year	0		0	4096	0	
P0047	Third fault Hour.Min	0.00		0.00	23.59	0.00	
P0048	Lag fault	0		0	65535	0	
P0050	Real Axis: Actual Position	0		0	16383	0	
P0052	fractions of revolution	0		-16383	16383	0	
P0053	number of revolutions	0		-32768	32767	0	
P0056	Standart Counter - Low	0		0	65535	0	
P0057	Standart Counter - High	0		0	65535	0	
P0058	Counter 1 - Low	0		0	65535	0	
P0059	Counter 1 - High	0		0	65535	0	

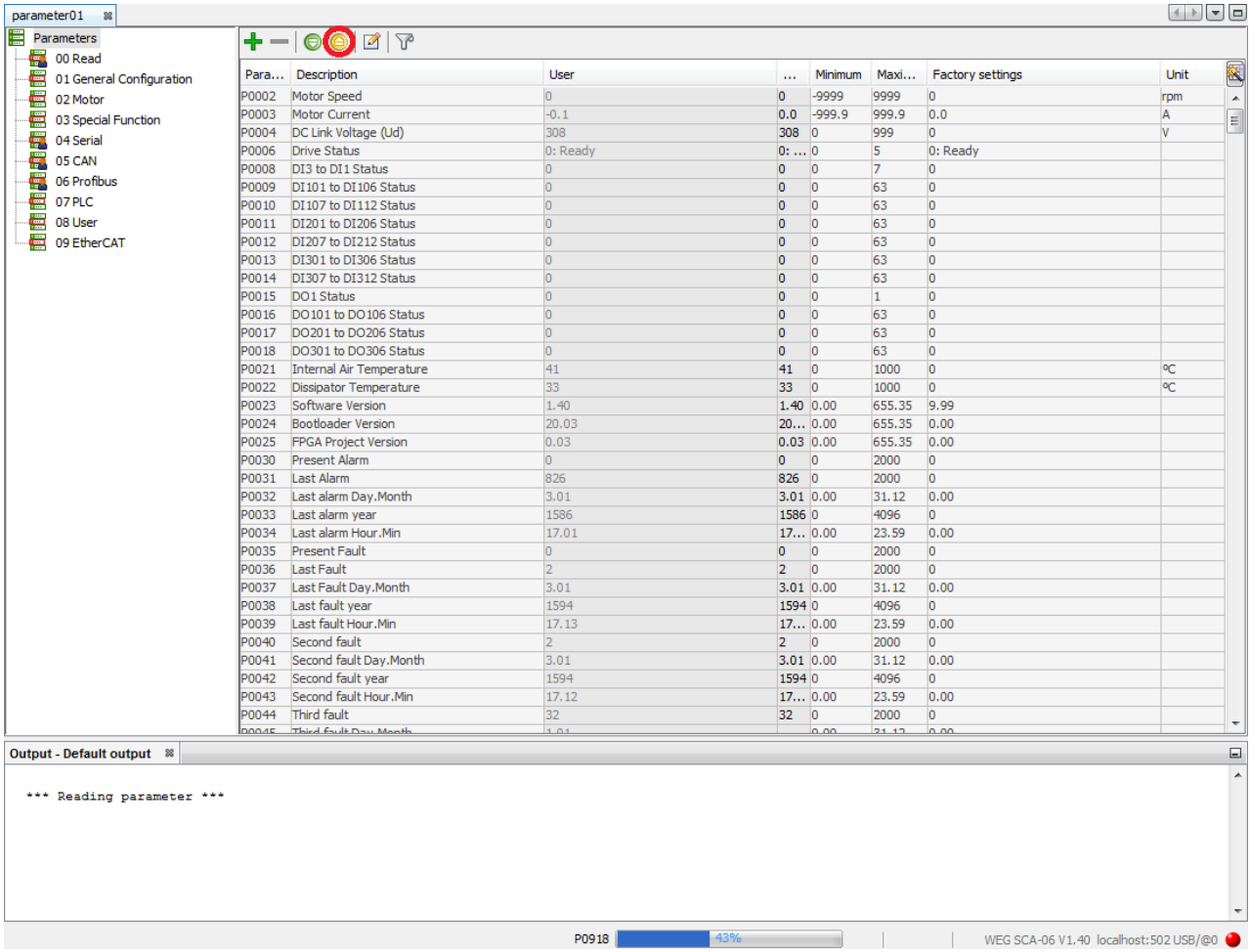
11.1.5.3 Read and Write of Parameters

There are 3 (three) ways to do the reading and writing of the parameters: by means of table, selection and group.

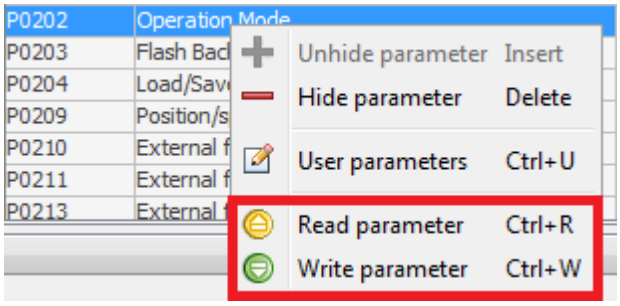
1. **Table writing**. The table writing command will send all visible parameters on the equipment screen. If and error occurs during the sending of some specific parameter, a message will be shown on the output window informing the error. It is important to notice that only visible parameters will be sent; therefore, it is necessary attention to which node of the group of parameters tree you are viewing. Example: If you wish to write all of them without filtering per group, just select the tree root.



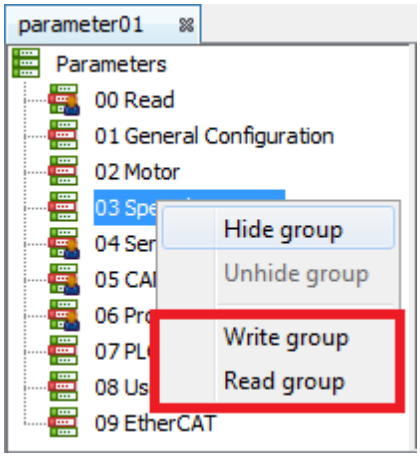
2. **Table reading**. The table reading command will read all the parameters of the equipment. If a error occurs during the reading of some specific parameter, a message will be shown on the output window informing the error. It is important to notice that only visible parameters will be read; therefore, it is necessary attention to which node of the group of parameters tree you are viewing. Example: If you wish to read all of them without filtering per group, just select the tree root.



3. **Reading/writing of specific parameters**. In order to read/write one or more specific parameters, just select them on the table, right click and choose the desired option: read or write parameter.



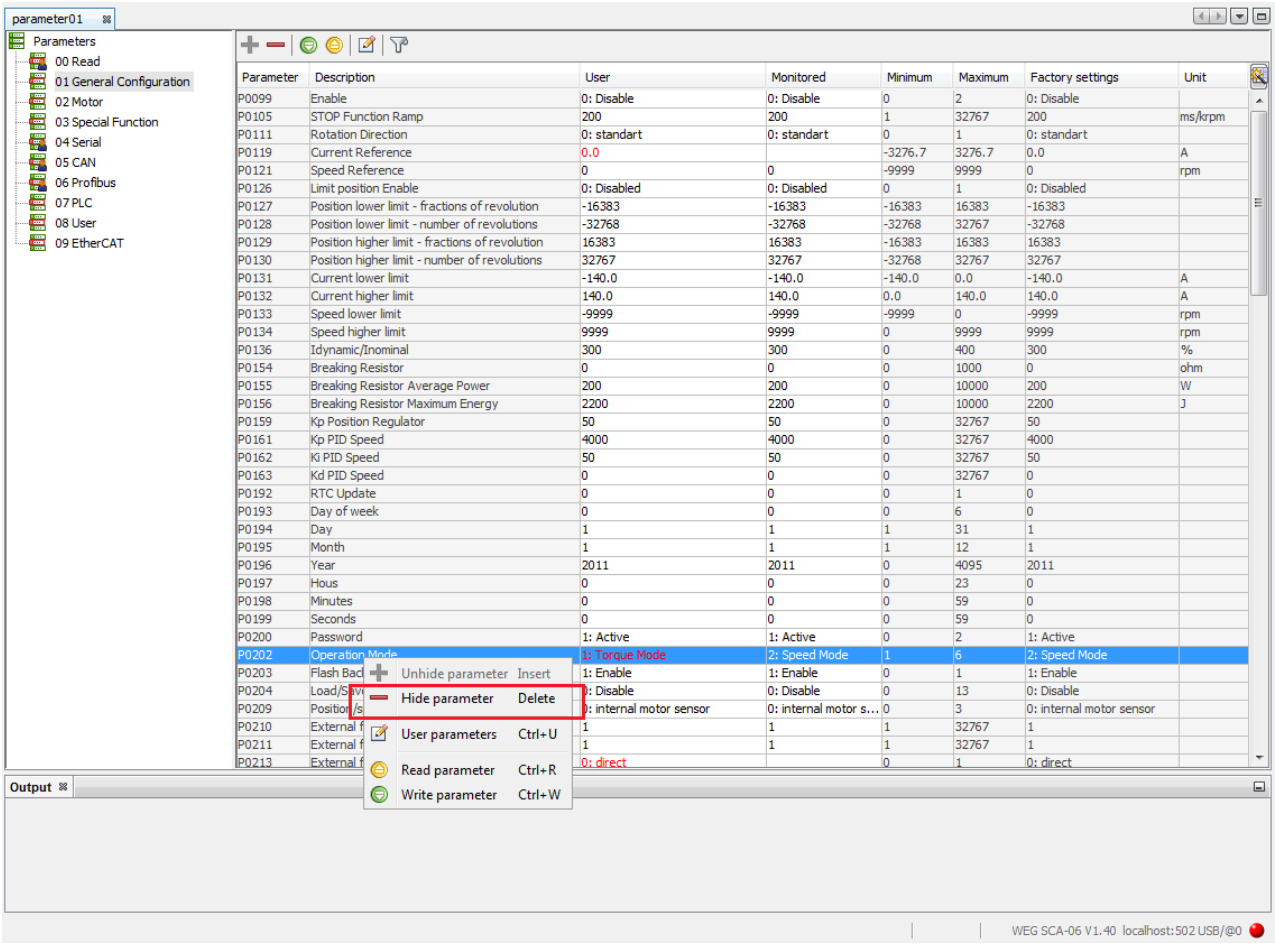
4. **Reading/writing of group of parameters**. In order to read/write only one group of parameters, just select it on the group tree, right click and choose the desired option: read or write group.



11.1.5.4 Hide/Unhide Parameters and Group of Parameters

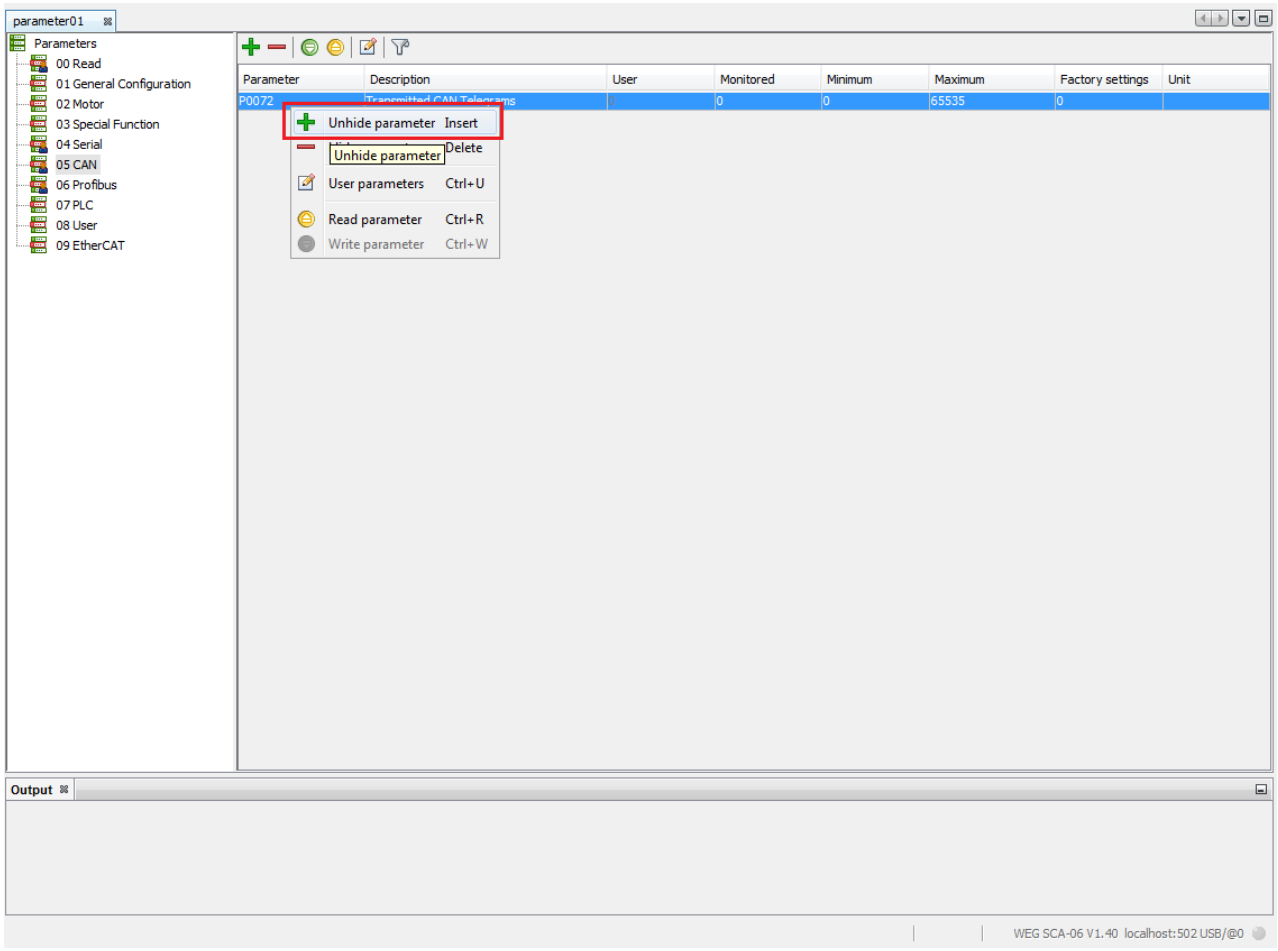
The parameter can be hidden/unhidden in two ways: individually or in group.

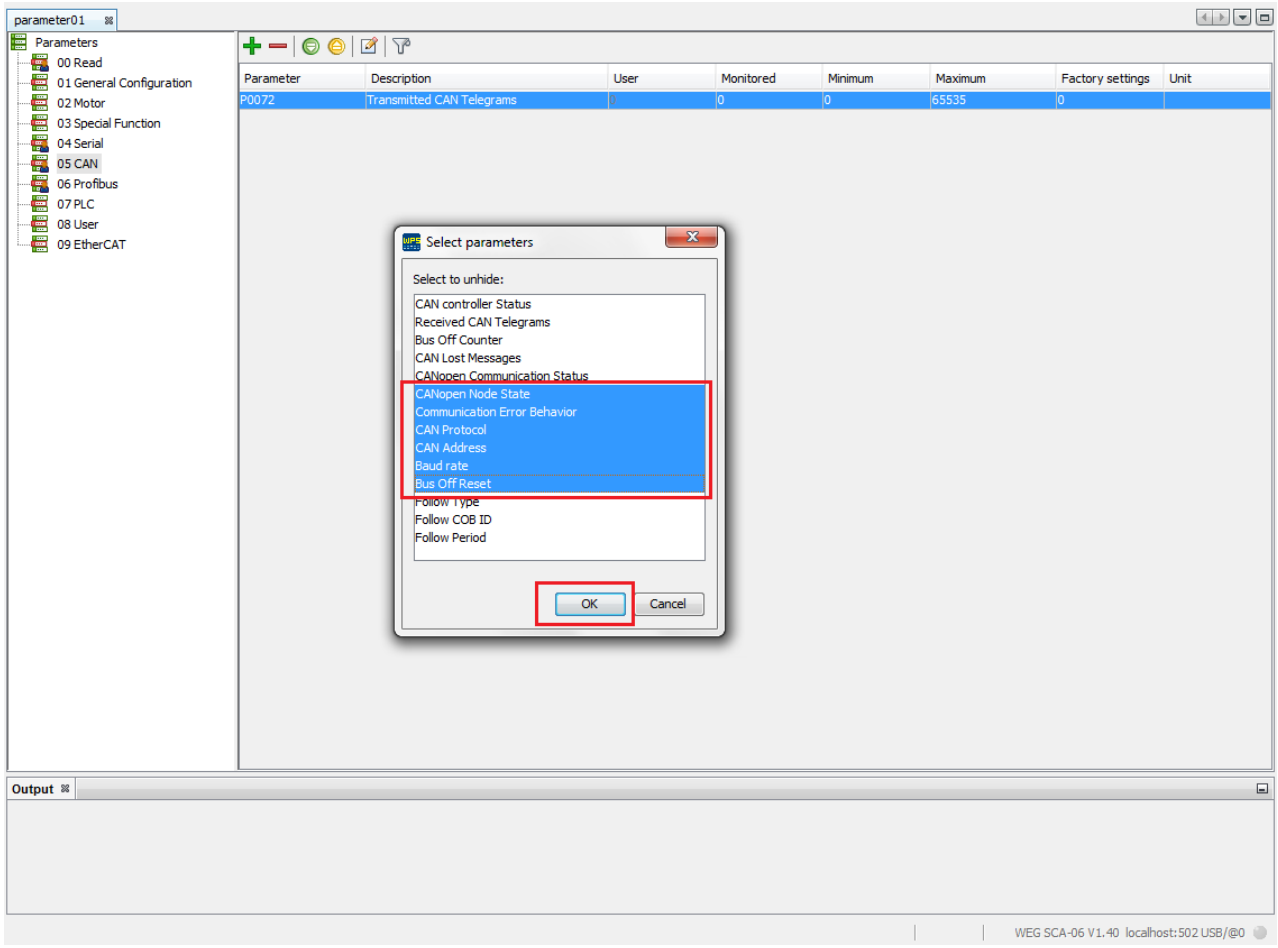
1. **Hide parameters**. In order to hide a parameter individually, just right click on the desired parameters and select the **Hide Parameter** option. You can also press the **Delete** key.

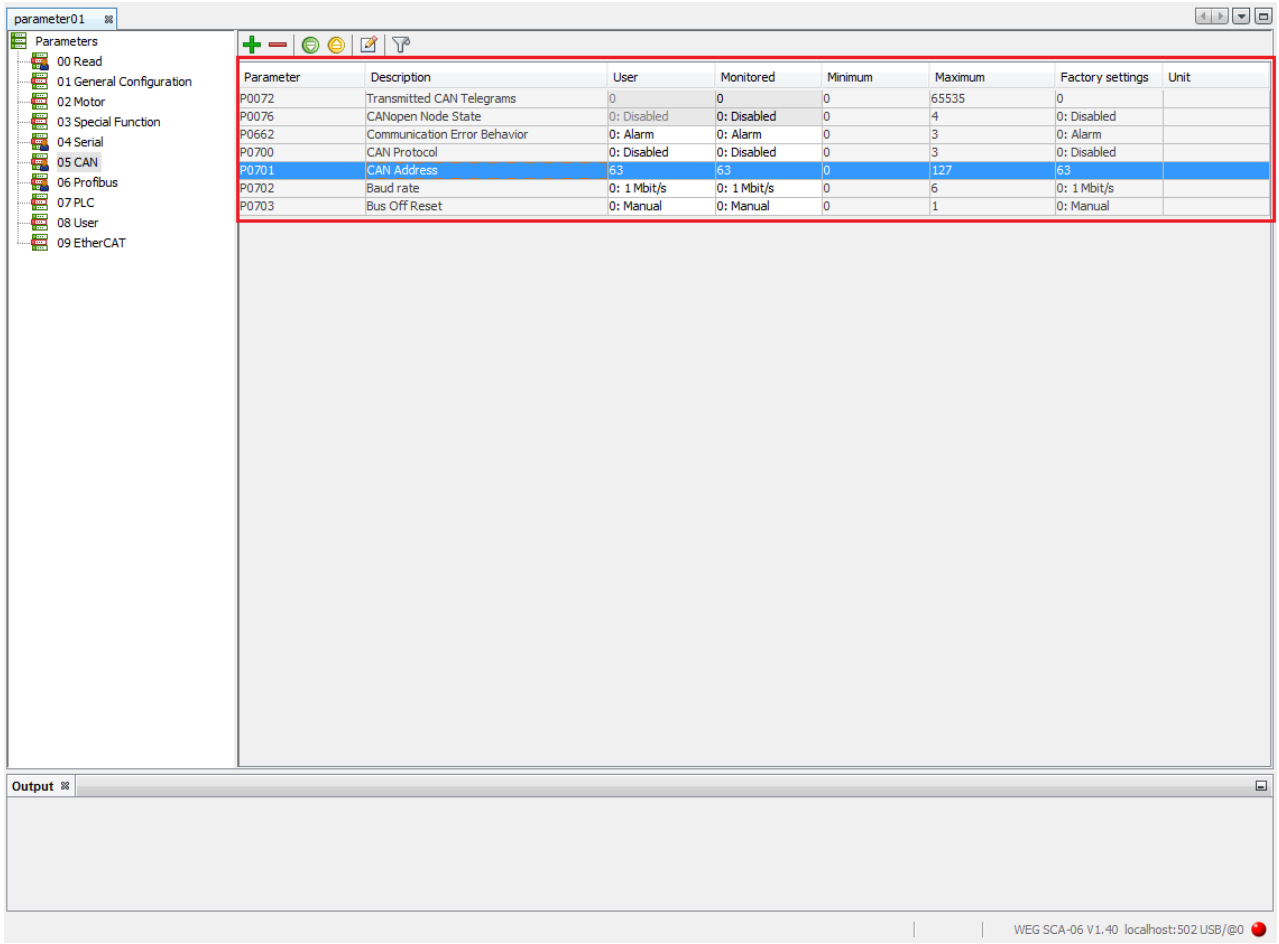


2. **Unhide Parameters**. In order to show hidden parameters, right click and choose the **Unhide Parameters**

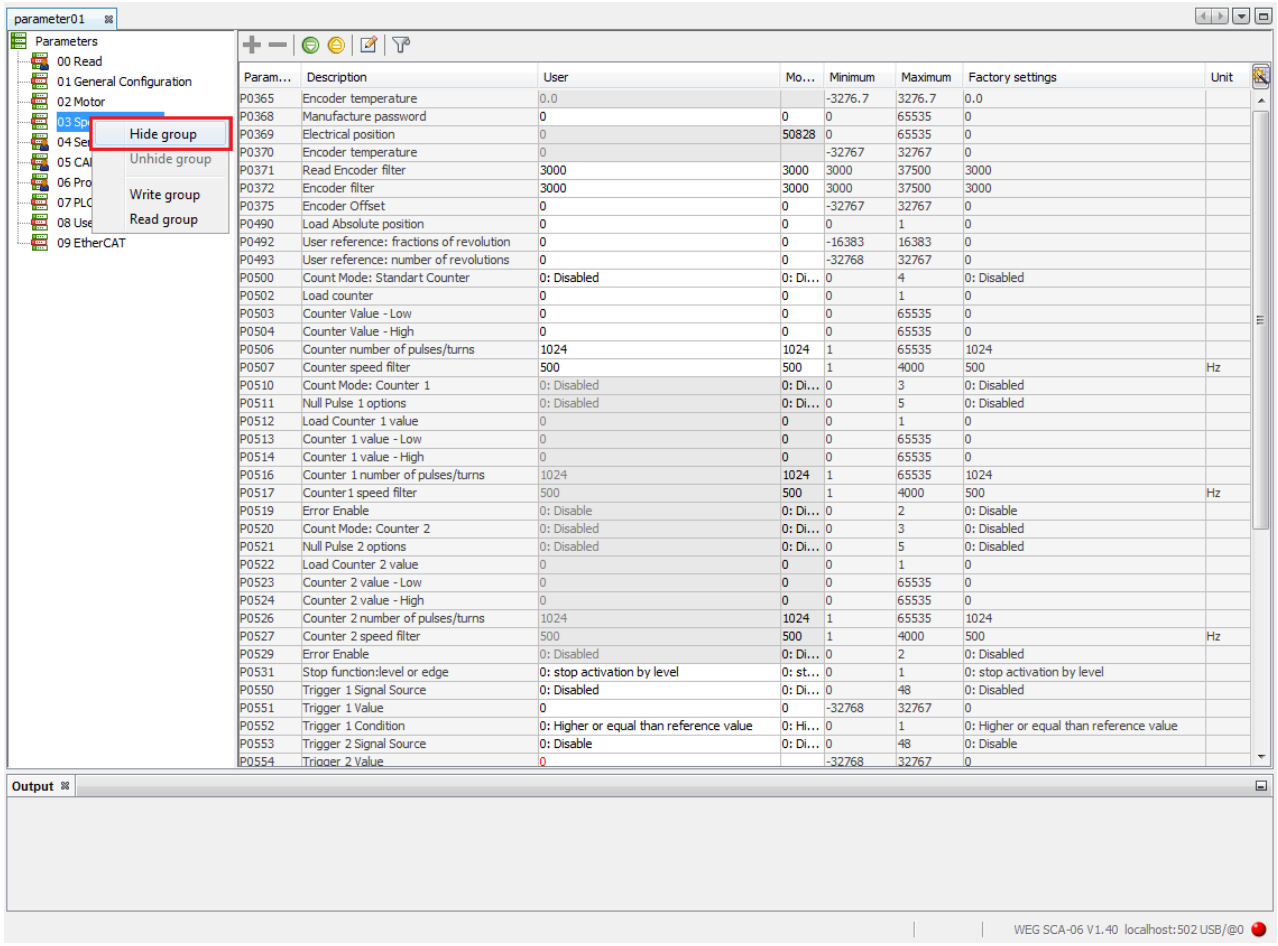
or press the **Insert** key. Then, a window will open and show the hidden parameters. Now, you just have to select the desired parameters and confirm.
Note: The parameters shown on this new window are only those which belong to the current filter according to the selection on the parameter group tree. In the figures below, the CAN group is selected; that means that only the hidden parameters of this group will be shown.

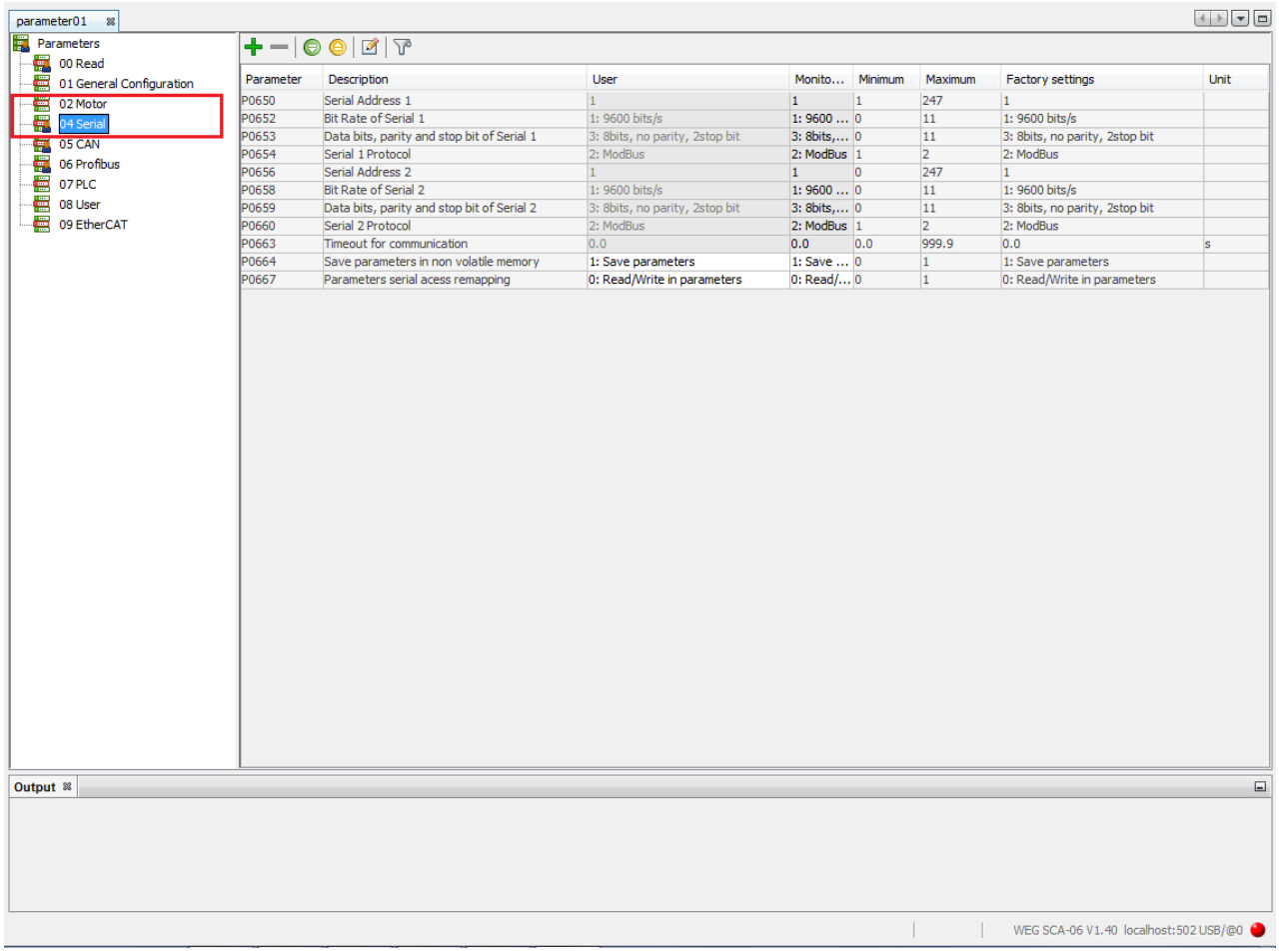



parameter01
Parameters
00 Read
01 General Configuration
02 Motor
03 Special Function
04 Serial
05 CAN
06 Profibus
07 PLC
08 User
09 EtherCAT
Parameter
Description
User
Monitored
Minimum
Maximum
Factory settings
Unit
P0072
Transmitted CAN Telegrams
0
0
65535
0
Select parameters
Select to unhide:
CAN controller Status
Received CAN Telegrams
Bus Off Counter
CAN Lost Messages
CANopen Communication Status
CANopen Node State
Communication Error Behavior
CAN Protocol
CAN Address
Baud rate
Bus Off Reset
Follow Type
Follow COB ID
Follow Period
OK
Cancel
Output
WEG SCA-06 V1.40 localhost:502 USB/@0

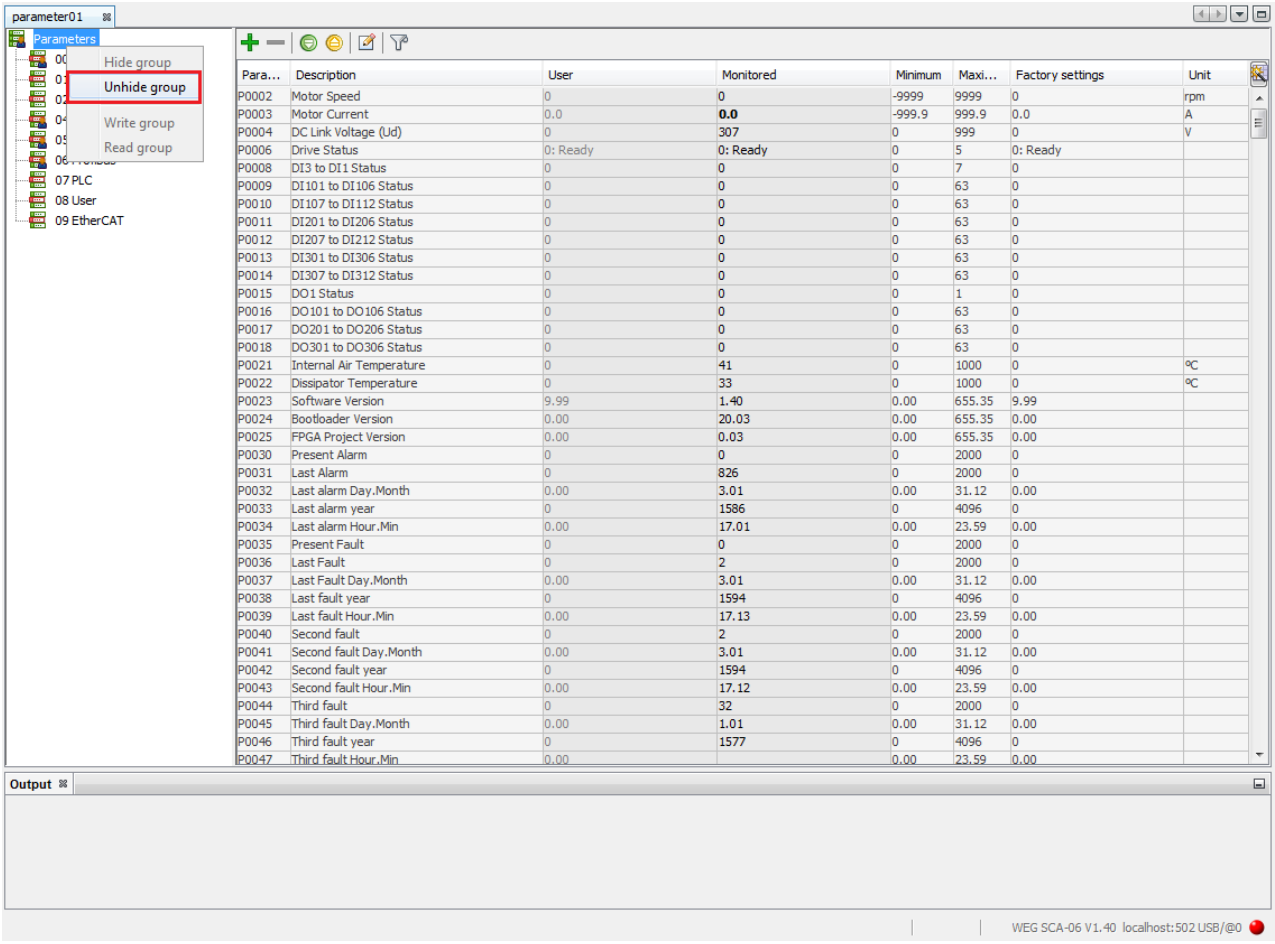


3. **Hide Group of Parameters**. In order to hide a group of parameters, just select the group on the tree and use the **Hide Group** option.

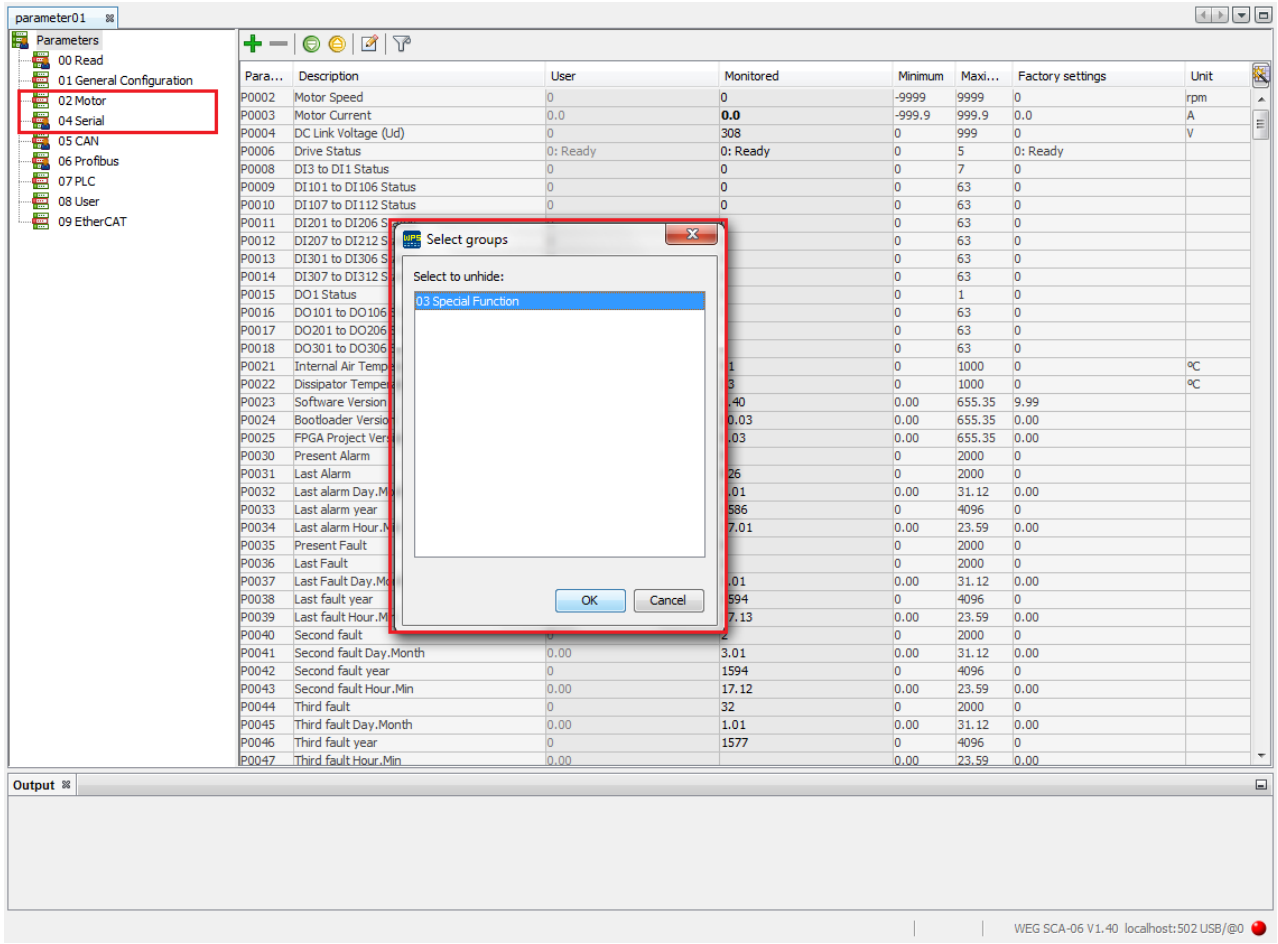

parameter01
Parameters
00 Read
01 General Configuration
02 Motor
03 Sp
04 Se
05 CA
06 Pro
07 PLC
08 Use
09 EtherCAT
Hide group
Unhide group
Write group
Read group
Param...
Description
User
Mo...
Minimum
Maximum
Factory settings
Unit
P0365 Encoder temperature 0.0 -3276.7 3276.7 0.0
P0368 Manufacture password 0 0 0 65535 0
P0369 Electrical position 0 50828 0 65535 0
P0370 Encoder temperature 0 -32767 32767 0
P0371 Read Encoder filter 3000 3000 3000 37500 3000
P0372 Encoder filter 3000 3000 3000 37500 3000
P0375 Encoder Offset 0 0 -32767 32767 0
P0490 Load Absolute position 0 0 0 1 0
P0492 User reference: fractions of revolution 0 0 -16383 16383 0
P0493 User reference: number of revolutions 0 0 -32768 32767 0
P0500 Count Mode: Standart Counter 0: Disabled 0: Di... 0 4 0: Disabled
P0502 Load counter 0 0 0 1 0
P0503 Counter Value - Low 0 0 0 65535 0
P0504 Counter Value - High 0 0 0 65535 0
P0506 Counter number of pulses/turns 1024 1024 1 65535 1024
P0507 Counter speed filter 500 500 1 4000 500 Hz
P0510 Count Mode: Counter 1 0: Disabled 0: Di... 0 3 0: Disabled
P0511 Null Pulse 1 options 0: Disabled 0: Di... 0 5 0: Disabled
P0512 Load Counter 1 value 0 0 0 1 0
P0513 Counter 1 value - Low 0 0 0 65535 0
P0514 Counter 1 value - High 0 0 0 65535 0
P0516 Counter 1 number of pulses/turns 1024 1024 1 65535 1024
P0517 Counter 1 speed filter 500 500 1 4000 500 Hz
P0519 Error Enable 0: Disable 0: Di... 0 2 0: Disable
P0520 Count Mode: Counter 2 0: Disabled 0: Di... 0 3 0: Disabled
P0521 Null Pulse 2 options 0: Disabled 0: Di... 0 5 0: Disabled
P0522 Load Counter 2 value 0 0 0 1 0
P0523 Counter 2 value - Low 0 0 0 65535 0
P0524 Counter 2 value - High 0 0 0 65535 0
P0526 Counter 2 number of pulses/turns 1024 1024 1 65535 1024
P0527 Counter 2 speed filter 500 500 1 4000 500 Hz
P0529 Error Enable 0: Disabled 0: Di... 0 2 0: Disabled
P0531 Stop function:level or edge 0: stop activation by level 0: st... 0 1 0: stop activation by level
P0550 Trigger 1 Signal Source 0: Disabled 0: Di... 0 48 0: Disabled
P0551 Trigger 1 Value 0 0 -32768 32767 0
P0552 Trigger 1 Condition 0: Higher or equal than reference value 0: Hi... 0 1 0: Higher or equal than reference value
P0553 Trigger 2 Signal Source 0: Disable 0: Di... 0 48 0: Disable
P0554 Trigger 2 Value 0 -32768 32767 0
Output
WEG SCA-06 V1.40 localhost:502 USB/@0

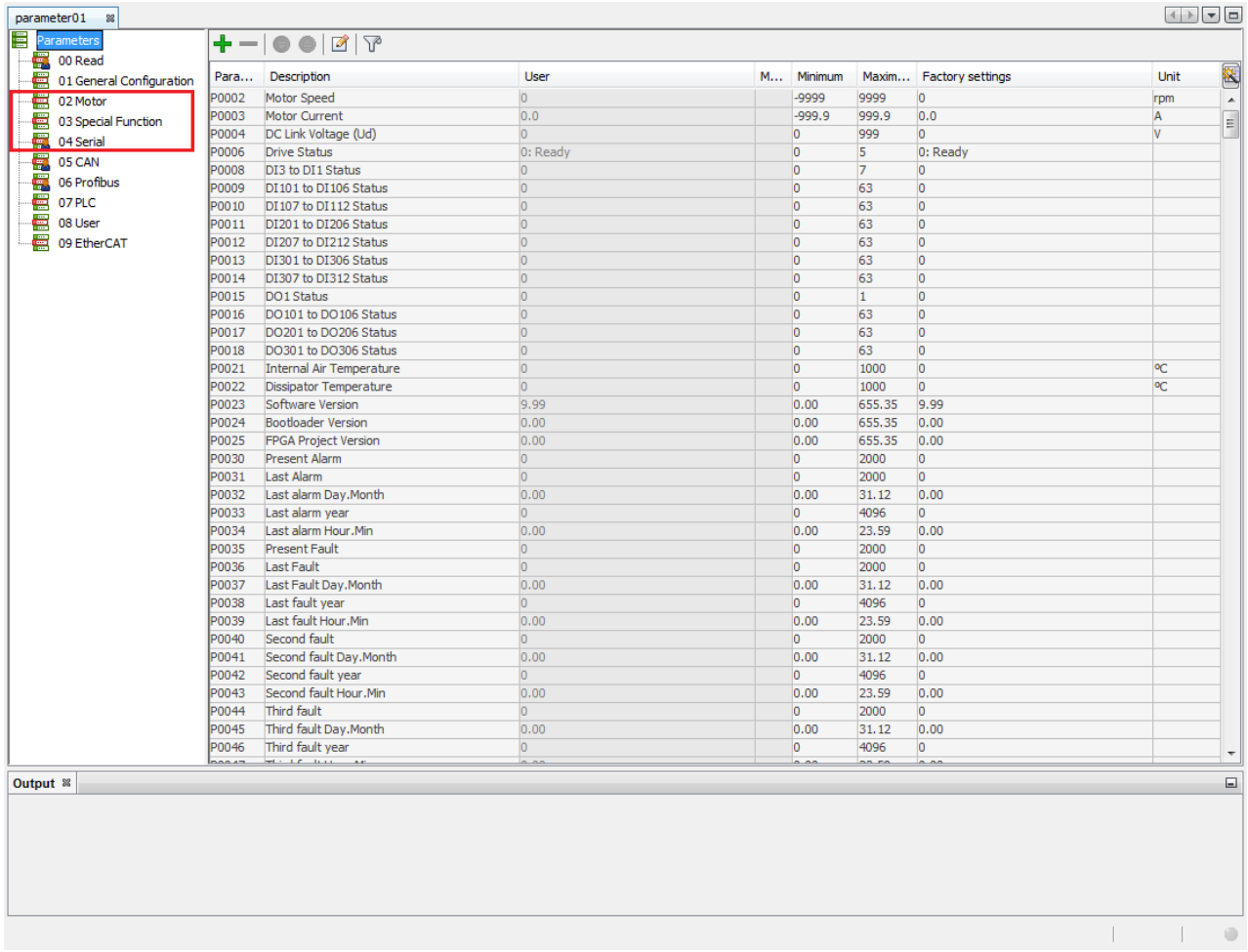


4. **Unhide Group of Parameters**. In order to show a hidden group of parameters, just select the root of the group tree and select the **Unhide Group** option. A window will open showing the groups that are hidden; then just select the group you wish to unhide.



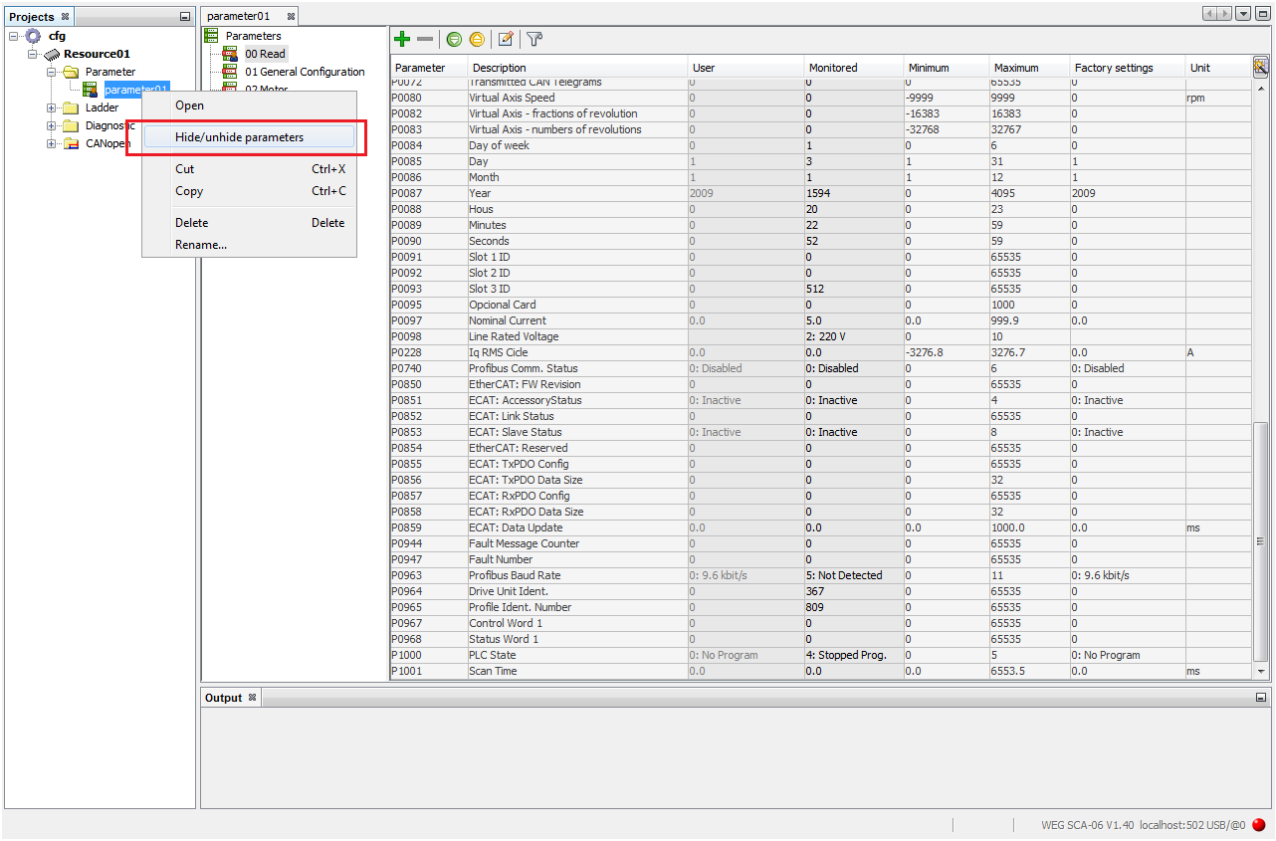
Para...	Description	User	Monitored	Minimum	Maxi...	Factory settings	Unit
P0002	Motor Speed	0	0	-9999	9999	0	rpm
P0003	Motor Current	0.0	**0.0**	-999.9	999.9	0.0	A
P0004	DC Link Voltage (Ud)	0	307	0	999	0	V
P0006	Drive Status	0: Ready	0: Ready	0	5	0: Ready	
P0008	DI3 to DI1 Status	0	0	0	7	0	
P0009	DI101 to DI106 Status	0	0	0	63	0	
P0010	DI107 to DI112 Status	0	0	0	63	0	
P0011	DI201 to DI206 Status	0	0	0	63	0	
P0012	DI207 to DI212 Status	0	0	0	63	0	
P0013	DI301 to DI306 Status	0	0	0	63	0	
P0014	DI307 to DI312 Status	0	0	0	63	0	
P0015	DO1 Status	0	0	0	1	0	
P0016	DO101 to DO106 Status	0	0	0	63	0	
P0017	DO201 to DO206 Status	0	0	0	63	0	
P0018	DO301 to DO306 Status	0	0	0	63	0	
P0021	Internal Air Temperature	0	41	0	1000	0	°C
P0022	Dissipator Temperature	0	33	0	1000	0	°C
P0023	Software Version	9.99	1.40	0.00	655.35	9.99	
P0024	Bootloader Version	0.00	20.03	0.00	655.35	0.00	
P0025	FPGA Project Version	0.00	0.03	0.00	655.35	0.00	
P0030	Present Alarm	0	0	0	2000	0	
P0031	Last Alarm	0	826	0	2000	0	
P0032	Last alarm Day.Month	0.00	3.01	0.00	31.12	0.00	
P0033	Last alarm year	0	1586	0	4096	0	
P0034	Last alarm Hour.Min	0.00	17.01	0.00	23.59	0.00	
P0035	Present Fault	0	0	0	2000	0	
P0036	Last Fault	0	2	0	2000	0	
P0037	Last Fault Day.Month	0.00	3.01	0.00	31.12	0.00	
P0038	Last fault year	0	1594	0	4096	0	
P0039	Last fault Hour.Min	0.00	17.13	0.00	23.59	0.00	
P0040	Second fault	0	2	0	2000	0	
P0041	Second fault Day.Month	0.00	3.01	0.00	31.12	0.00	
P0042	Second fault year	0	1594	0	4096	0	
P0043	Second fault Hour.Min	0.00	17.12	0.00	23.59	0.00	
P0044	Third fault	0	32	0	2000	0	
P0045	Third fault Day.Month	0.00	1.01	0.00	31.12	0.00	
P0046	Third fault year	0	1577	0	4096	0	
P0047	Third fault Hour.Min	0.00		0.00	23.59	0.00	

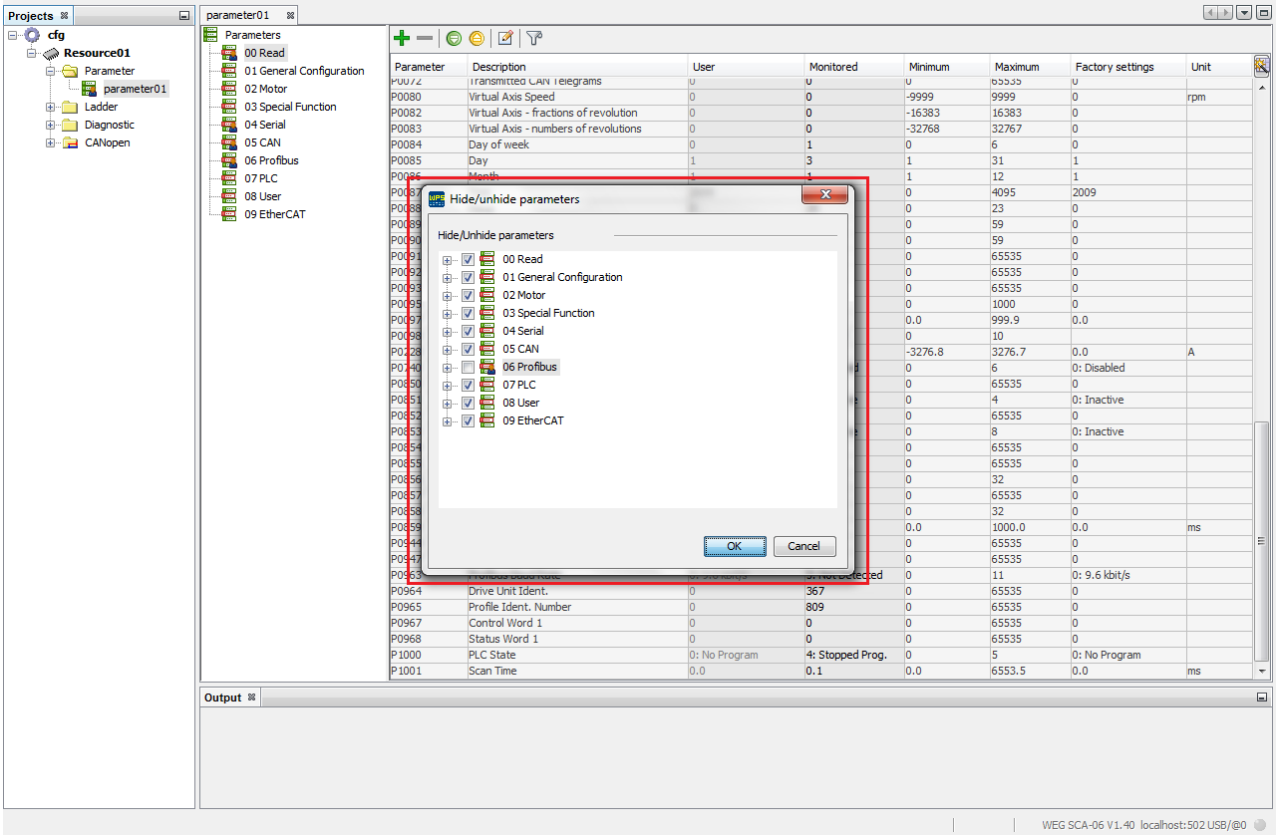

parameter01
Parameters
00 Read
01 General Configuration
02 Motor
04 Serial
05 CAN
06 Profibus
07 PLC
08 User
09 EtherCAT
Select groups
Select to unhide:
03 Special Function
OK
Cancel
Output
WEG SCA-06 V1.40 localhost:502 USB/@0



Para...	Description	User	M...	Minimum	Maxim...	Factory settings	Unit
P0002	Motor Speed	0		-9999	9999	0	rpm
P0003	Motor Current	0.0		-999.9	999.9	0.0	A
P0004	DC Link Voltage (Ud)	0		0	999	0	V
P0006	Drive Status	0: Ready		0	5	0: Ready	
P0008	DI3 to DI1 Status	0		0	7	0	
P0009	DI101 to DI106 Status	0		0	63	0	
P0010	DI107 to DI112 Status	0		0	63	0	
P0011	DI201 to DI206 Status	0		0	63	0	
P0012	DI207 to DI212 Status	0		0	63	0	
P0013	DI301 to DI306 Status	0		0	63	0	
P0014	DI307 to DI312 Status	0		0	63	0	
P0015	DO1 Status	0		0	1	0	
P0016	DO101 to DO106 Status	0		0	63	0	
P0017	DO201 to DO206 Status	0		0	63	0	
P0018	DO301 to DO306 Status	0		0	63	0	
P0021	Internal Air Temperature	0		0	1000	0	°C
P0022	Dissipator Temperature	0		0	1000	0	°C
P0023	Software Version	9.99		0.00	655.35	9.99	
P0024	Bootloader Version	0.00		0.00	655.35	0.00	
P0025	FPGA Project Version	0.00		0.00	655.35	0.00	
P0030	Present Alarm	0		0	2000	0	
P0031	Last Alarm	0		0	2000	0	
P0032	Last alarm Day.Month	0.00		0.00	31.12	0.00	
P0033	Last alarm year	0		0	4096	0	
P0034	Last alarm Hour.Min	0.00		0.00	23.59	0.00	
P0035	Present Fault	0		0	2000	0	
P0036	Last Fault	0		0	2000	0	
P0037	Last Fault Day.Month	0.00		0.00	31.12	0.00	
P0038	Last fault year	0		0	4096	0	
P0039	Last fault Hour.Min	0.00		0.00	23.59	0.00	
P0040	Second fault	0		0	2000	0	
P0041	Second fault Day.Month	0.00		0.00	31.12	0.00	
P0042	Second fault year	0		0	4096	0	
P0043	Second fault Hour.Min	0.00		0.00	23.59	0.00	
P0044	Third fault	0		0	2000	0	
P0045	Third fault Day.Month	0.00		0.00	31.12	0.00	
P0046	Third fault year	0		0	4096	0	

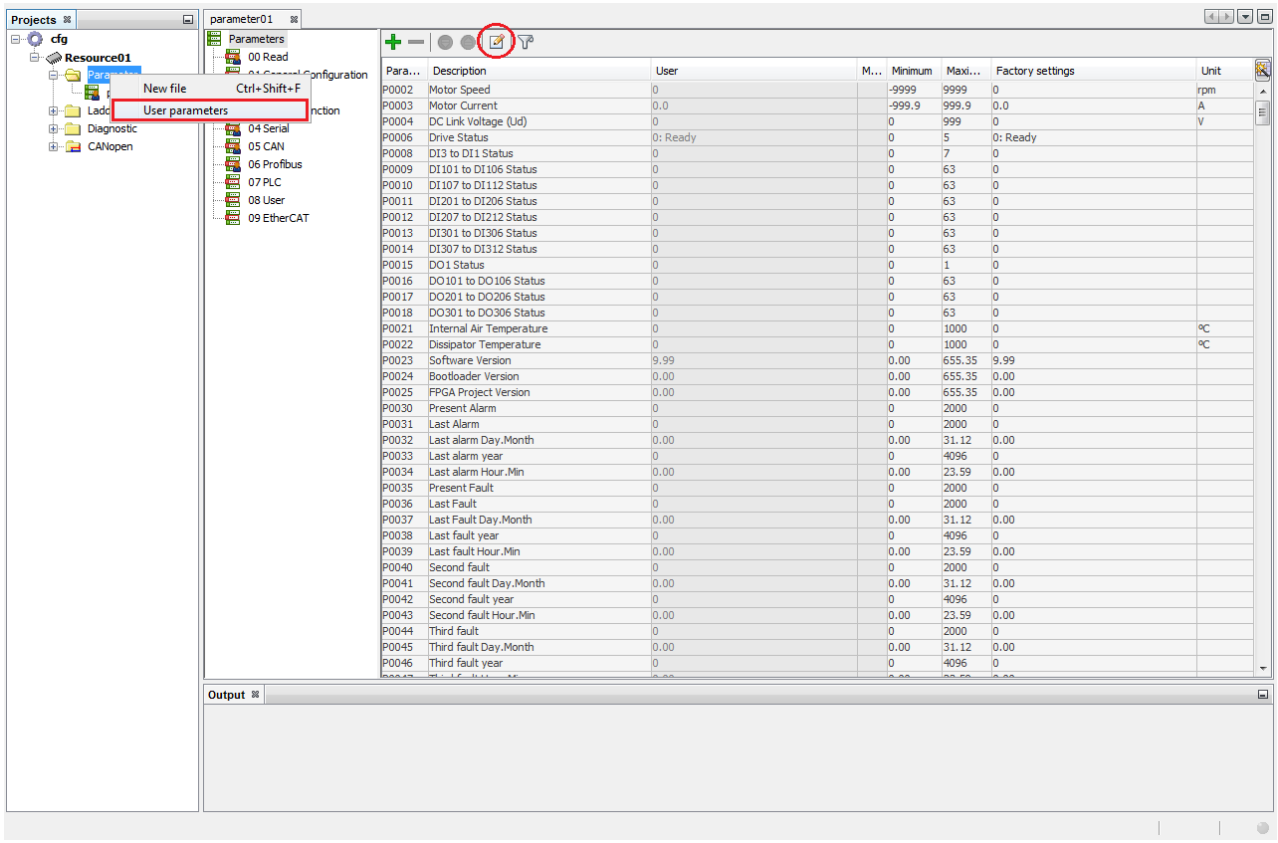
5. **Hide and Show Parameters and Groups of Parameters**. By means of this option, you have full control of the parameters and groups of parameters. It is possible to hide and unhide individual parameters, multiple parameters, individual groups and multiple groups in the same action.


Projects
cfg
Resource01
Parameter
parameter01
Ladder
Diagnostic
CANopen
Open
Hide/unhide parameters
Cut Ctrl+X
Copy Ctrl+C
Delete Delete
Rename...
parameter01
Parameters
00 Read
01 General Configuration
Parameter
Description
User
Monitored
Minimum
Maximum
Factory settings
Unit
P0080 Virtual Axis Speed
P0082 Virtual Axis - fractions of revolution
P0083 Virtual Axis - numbers of revolutions
P0084 Day of week
P0085 Day
P0086 Month
P0087 Year
P0088 Hous
P0089 Minutes
P0090 Seconds
P0091 Slot 1 ID
P0092 Slot 2 ID
P0093 Slot 3 ID
P0095 Opcional Card
P0097 Nominal Current
P0098 Line Rated Voltage
P0228 Iq RMS Cicle
P0740 Profibus Comm. Status
P0850 EtherCAT: FW Revision
P0851 ECAT: AccessoryStatus
P0852 ECAT: Link Status
P0853 ECAT: Slave Status
P0854 EtherCAT: Reserved
P0855 ECAT: TxPDO Config
P0856 ECAT: TxPDO Data Size
P0857 ECAT: RxPDO Config
P0858 ECAT: RxPDO Data Size
P0859 ECAT: Data Update
P0944 Fault Message Counter
P0947 Fault Number
P0963 Profibus Baud Rate
P0964 Drive Unit Ident.
P0965 Profile Ident. Number
P0967 Control Word 1
P0968 Status Word 1
P1000 PLC State
P1001 Scan Time
Output
WEG SCA-06 V1.40 localhost:502 USB/@0


Projects
cfg
Resource01
Parameter
parameter01
Ladder
Diagnostic
CANopen
parameter01
Parameters
00 Read
01 General Configuration
02 Motor
03 Special Function
04 Serial
05 CAN
06 Profibus
07 PLC
08 User
09 EtherCAT
Hide/unhide parameters
Hide/Unhide parameters
OK
Cancel
Output
WEG SCA-06 V1.40 localhost:502 USB/@0

11.1.5.5 User Parameters

In order to open the configuration screen of the user parameters, just click on the **User Parameters** option on the Parameter node of the project tree or click on the icon indicated on the tool bar of the parameter file.



Para...	Description	User	M...	Minimum	Maxi...	Factory settings	Unit
P0002	Motor Speed	0		-9999	9999	0	rpm
P0003	Motor Current	0.0		-999.9	999.9	0.0	A
P0004	DC Link Voltage (Ud)	0		0	999	0	V
P0006	Drive Status	0: Ready		0	5	0: Ready	
P0008	DI3 to DI1 Status	0		0	7	0	
P0009	DI101 to DI106 Status	0		0	63	0	
P0010	DI107 to DI112 Status	0		0	63	0	
P0011	DI201 to DI206 Status	0		0	63	0	
P0012	DI207 to DI212 Status	0		0	63	0	
P0013	DI301 to DI306 Status	0		0	63	0	
P0014	DI307 to DI312 Status	0		0	63	0	
P0015	DO1 Status	0		0	1	0	
P0016	DO101 to DO106 Status	0		0	63	0	
P0017	DO201 to DO206 Status	0		0	63	0	
P0018	DO301 to DO306 Status	0		0	63	0	
P0021	Internal Air Temperature	0		0	1000	0	°C
P0022	Dissipator Temperature	0		0	1000	0	°C
P0023	Software Version	9.99		0.00	655.35	9.99	
P0024	Bootloader Version	0.00		0.00	655.35	0.00	
P0025	FPGA Project Version	0.00		0.00	655.35	0.00	
P0030	Present Alarm	0		0	2000	0	
P0031	Last Alarm	0		0	2000	0	
P0032	Last alarm Day.Month	0.00		0.00	31.12	0.00	
P0033	Last alarm year	0		0	4096	0	
P0034	Last alarm Hour.Min	0.00		0.00	23.59	0.00	
P0035	Present Fault	0		0	2000	0	
P0036	Last Fault	0		0	2000	0	
P0037	Last Fault Day.Month	0.00		0.00	31.12	0.00	
P0038	Last fault year	0		0	4096	0	
P0039	Last fault Hour.Min	0.00		0.00	23.59	0.00	
P0040	Second fault	0		0	2000	0	
P0041	Second fault Day.Month	0.00		0.00	31.12	0.00	
P0042	Second fault year	0		0	4096	0	
P0043	Second fault Hour.Min	0.00		0.00	23.59	0.00	
P0044	Third fault	0		0	2000	0	
P0045	Third fault Day.Month	0.00		0.00	31.12	0.00	
P0046	Third fault year	0		0	4096	0	

Configuration Table.

On the user parameter configuration table, it is possible to define several attributes to the parameters, such as description, minimum and maximum values, unit, digits, data type, etc.

NOTE!
These settings will be automatically displayed in the parameter table. However, to be sent to the device, you need to *download* the resource.

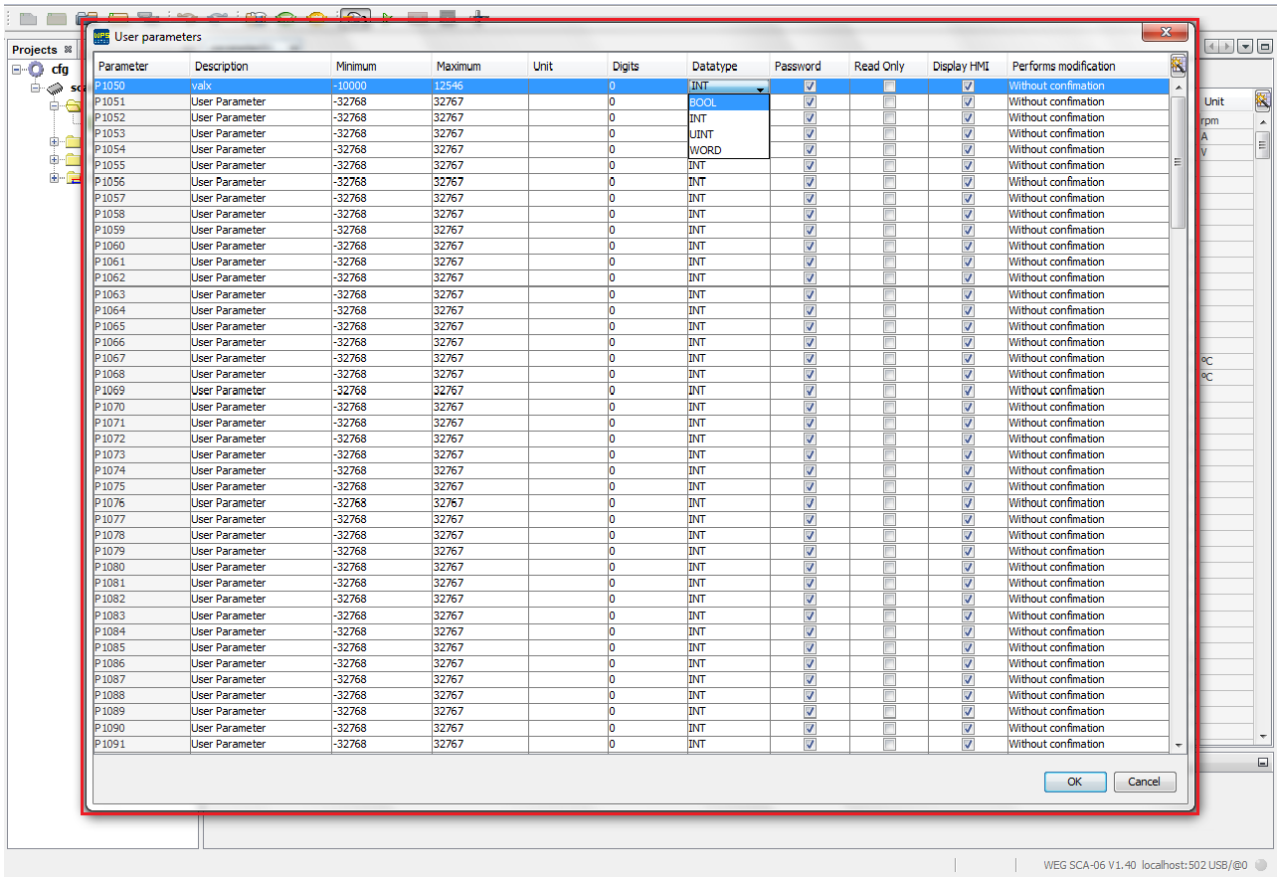


Table fields:

- Parameter: User parameter identification.
- Description: Description of the user parameter in the parameter table. On devices that have text-based HMIs, the description is sent to the machine and displayed on the HMI.
- Minimum: Minimum input value for parameter.
- Maximum: Maximum input value for parameter.
- Unit: Unit displayed on the device's HMI.
- Default: Value loaded when restore factory default is selected.
- Retentive: Retain value after rebooting devices.
- Hexadecimal: Displays the value in hexadecimal.
- Digits: Number of decimal digits for displaying value.
- Datatype: Parameter datatype used by the ladder application.
- Password: Enables password request by changing parameter value.

- Read only: It does not allow the writing of values in the parameter by the communication network or the HMI. Writing is done only by the ladder application.

- Display HMI: Displays the parameter in the HMI.

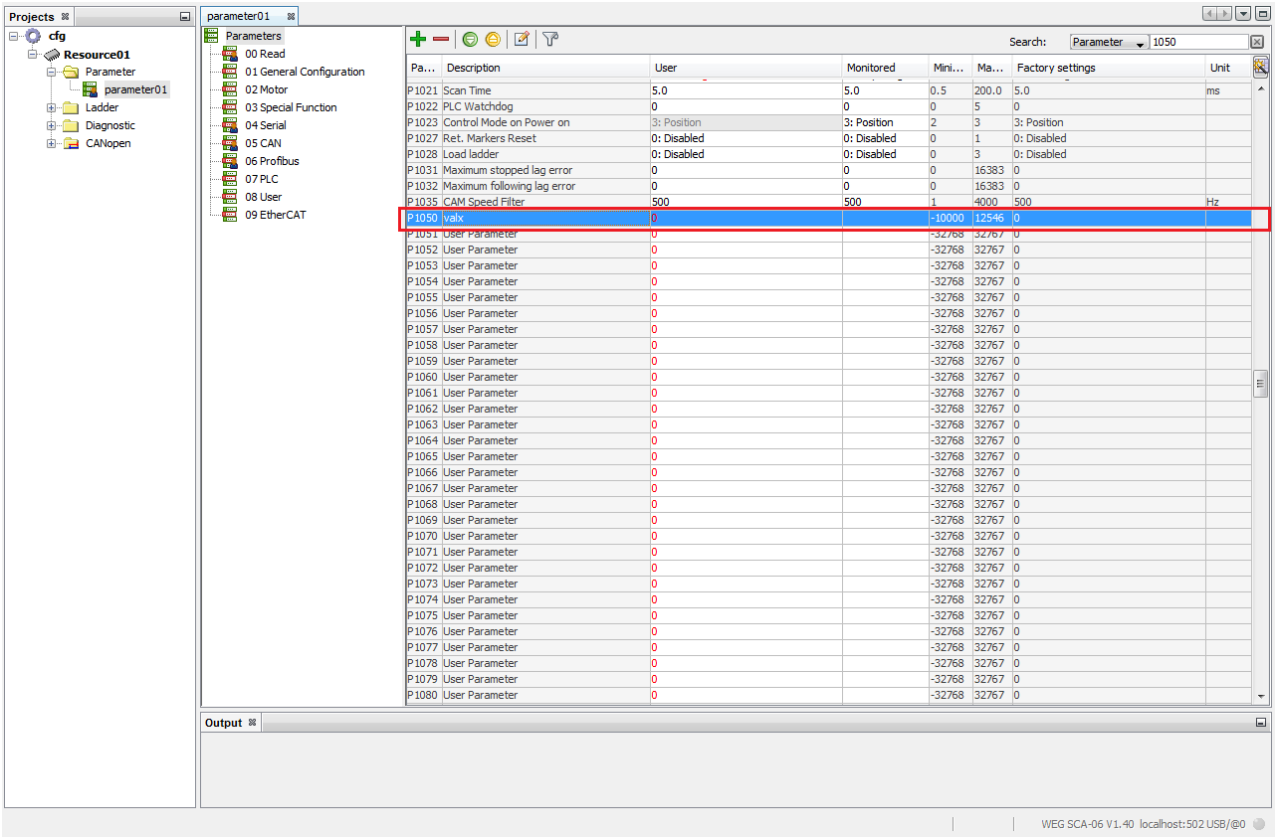
- Performs modification: Confirmation options when changing the parameter:
o No confirmation: Does not prompt for confirmation when changing parameter.
o With confirmation and engine stopped: Request confirmation and allow change only with engine stopped.
o With confirmation: Prompt for confirmation when changing parameter.

- Stopped motor: Perform change only with motor stopped.

- Help: On devices that have text-based HMI, you can edit a help text for the parameter.

View the user parameter

In the parameter table, the user parameters will be shown as they are configured on the configuration screen.



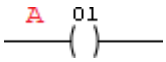
11.1.6 Ladder

11.1.6.1 Coil

11.1.6.1.1 DIRECTCOIL

Logical block used to assign direct values of the output variables.

Ladder Representation



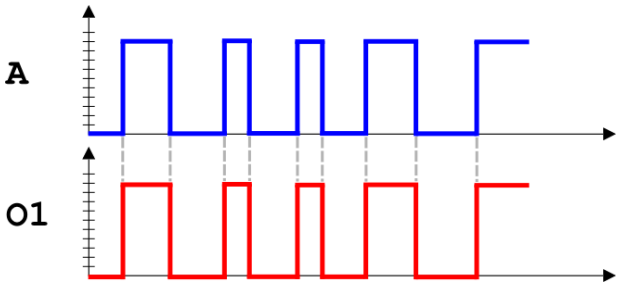
Block Structure

Variable Type	Name	Data Type	Description
VAR_OUTPUT	O1	BOOL	Block log output

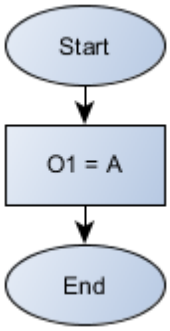
Operation

The block transfers the value of A for the memory address corresponding to O1.

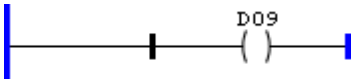
Diagram



Block Flowchart



Example

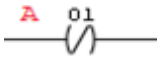


The above example keeps the digital output DO9 permanently connected, because the value of A in this case is the value of the left bus which is always considered high logic level (TRUE).

11.1.6.1.2 INVERTEDCOIL

Logical block used for assigning values denied to output variables.

Ladder Representation



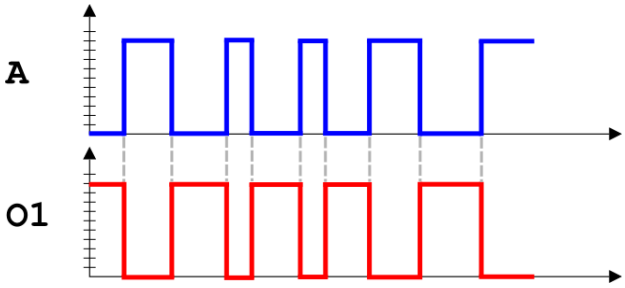
Block Structure

Variable Type	Name	Data Type	Description
VAR_OUTPUT	O1	BOOL	Block log output

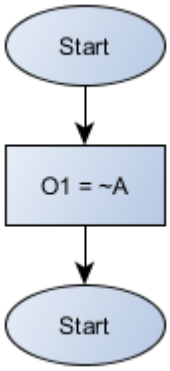
Operation

The block transfers the denied value of A for the memory address corresponding to O1.

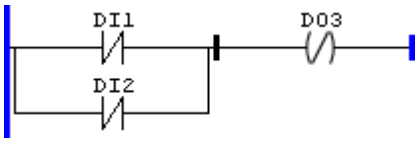
Diagram



Block Flowchart



Example

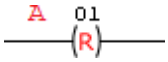


The above example disables the digital output DO3 when some of the digital inputs DI1 and DI2 are with FALSE value. When both inputs are with a TRUE value, DO3 activates.

11.1.6.1.3 RESETCOIL

Logical block used for indefinite disabling of output variables.

Ladder Representation



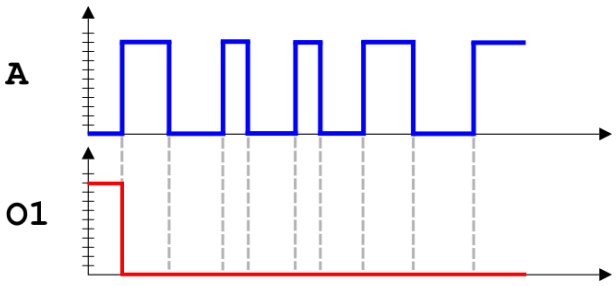
Block Structure

Variable Type	Name	Data Type	Description
VAR_OUTPUT	O1	BOOL	Block log output

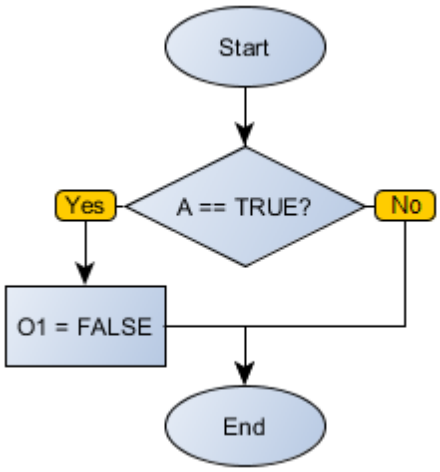
Operation

When identifying a TRUE value in A, this block transfers a FALSE value to the memory address corresponding to O1.
When identifying a FALSE value in A, this block performs no operation.

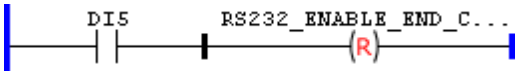
Diagram



Block Flowchart



Example

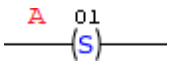


The example above activates permanently the system control marker that enables end-of-message character in RS232 communication to identify a TRUE level at the digital input DI5.

11.1.6.1.4 SETCOIL

Logical block used for indefinite enabling of output variables.

Ladder Representation



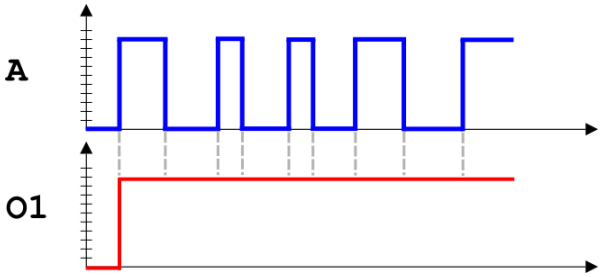
Block Structure

Variable Type	Name	Data Type	Description
VAR_OUTPUT	O1	BOOL	Block log output

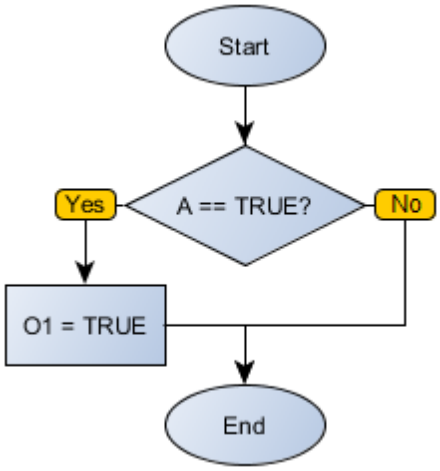
Operation

When identifying a TRUE value in A, this block transfers the value of A for the memory address corresponding to O1.
When identifying a FALSE value in A, this block performs no operation.

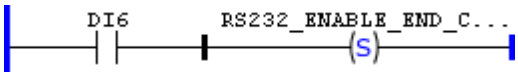
Diagram



Block Flowchart



Example

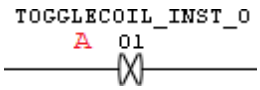


The example above activates permanently the system control marker that enables end-of-message character in RS232 communication to identify a TRUE level at the digital input DI6.

11.1.6.1.5 TOGGLECOIL

Logical block used for output variables alternance.

Ladder Representation



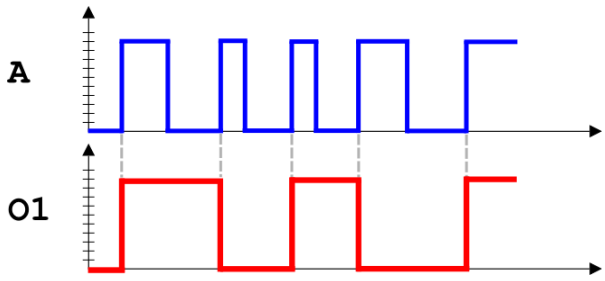
Block Structure

Variable Type	Name	Data Type	Description
VAR_OUTPUT	O1	BOOL	Block log output
VAR	TOGGLECOIL_INST_0	TOGGLECOIL	Instance of access to block structure

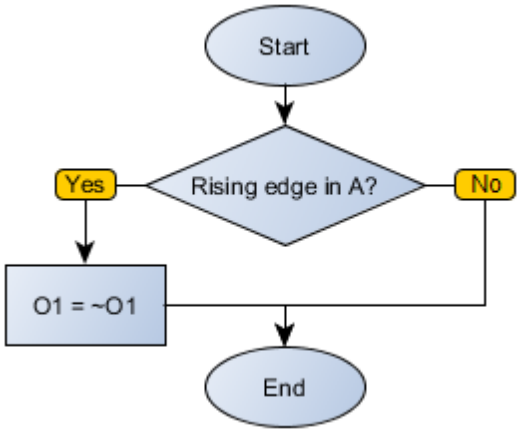
Operation

When identifying a transition from FALSE to TRUE (leading edge) on A, the block reverses the status of O1.

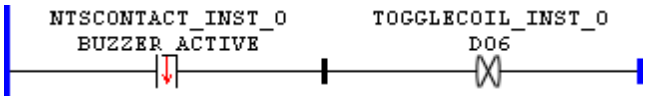
Diagram



Block Flowchart



Example



The above example inverts the state of the digital output DO6 to each disabling the internal buzzer.

11.1.6.2 Communication Network

11.1.6.2.1 Modbus RTU

11.1.6.2.1.1 Modbus RTU Overview

Operation in the Modbus RTU Network - Master Mode

The CFW300 allows operation as a master for the Modbus RTU network. For this operation, it is necessary to observe the following points:

- Only interface RS485 allows operation as a network master.
- It is necessary to program, in product configurations, the operation mode as "Master", besides the communication rate, parity, and stop bits, which must be the same for the whole equipment in the network.
- The Modbus RTU network master does not have an address, so the address configured in the CFW300 is not used.
- Sending and receiving telegrams via RS485 interface using the Modbus RTU is programmed by using blocks in Ladder programming language. It is necessary to know the available blocks and the Ladder programming software in order to be able to program the network master.

- The following functions are available for the sending of requisitions by the Modbus master:
  - Function 01: Read Coils
  - Function 02: Read Discrete Inputs
  - Function 03: Read Holding Registers
  - Function 04: Read Input Registers
  - Function 05: Write Single Coil
  - Function 06: Write Single Register
  - Function 15: Write Multiple Coils
  - Function 16: Write Multiple Registers

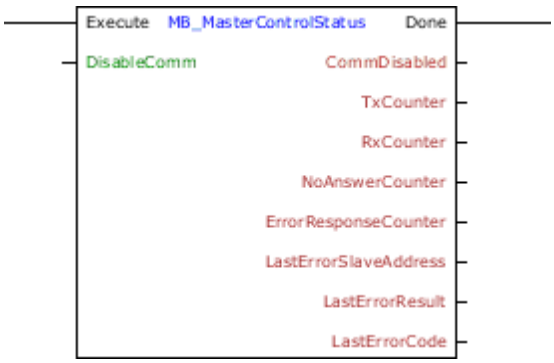
Blocks to program the master

In order to control and monitor the Modbus RTU communication using the CFW300, the following blocks were developed, and they must be used when programming in Ladder.

11.1.6.2.1.2 MB_MasterControlStatus

Block that allows monitoring various statuses of the Modbus RTU network master.

Ladder Representation



Block Structure

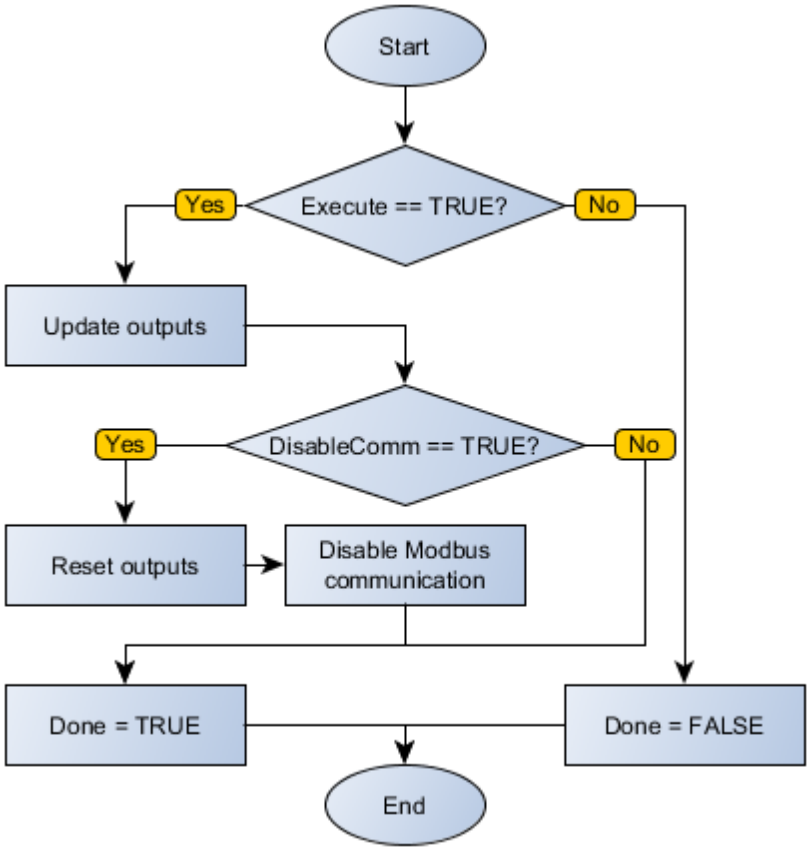
Variable Type	Name	Data Type	Description
VAR_INPUT	Execute	BOOL	Block enabling
	DisableComm	BOOL	Disables Modbus RTU communication
VAR_OUTPUT	Done	BOOL	Output enabling
	CommDisabled	BOOL	Disabled communication flag
	TxCounter	WORD UINT	Counter of requests sent
	RxCounter	WORD UINT	Counter of telegrams received
	NoAnswerCounter	WORD UINT	Counter of requests not answered
	ErrorResponseCounter	WORD UINT	Counter of responses received with error information
	LastErrorSlaveAddress	BYTE USINT	Slave address in which the last communication error was detected
	LastErrorResult	BYTE USINT	Operation result of the last communication error received (0 = No error) (4 – Response Timeout) (5 = Slave returned error)
	LastErrorCode	BYTE USINT	Code of the last communication error received

Operation

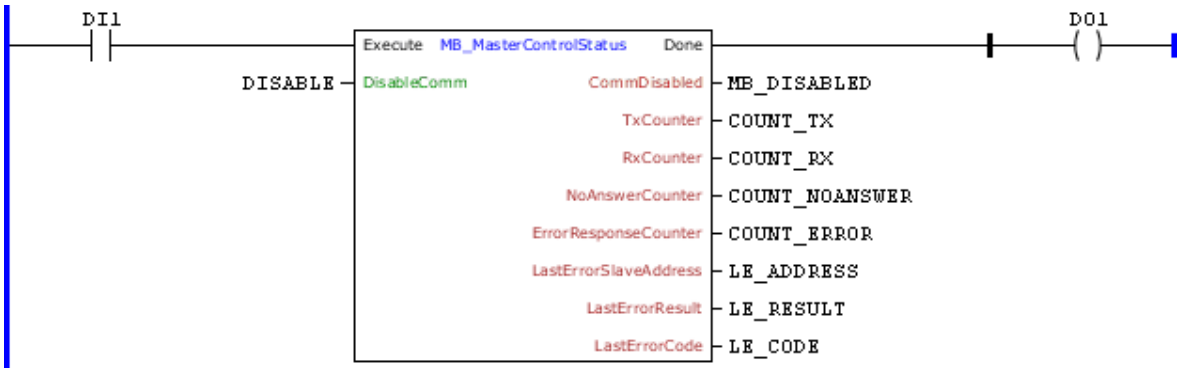
This block remains active while Execute is at TRUE level, updating its outputs according to the monitoring of the master and input requests. When Execute receives FALSE level, the inputs are ignored and the outputs are zeroed. The Done output receives TRUE level when Execute has TRUE level and block finished its execution.

A TRUE level DisableComm disables the Modbus RTU communication and resets the status counters and markers of the master. These markers and counters are displayed in the output block each having some data corresponding to its description. Their values are also cleared at shutdown of the master.

Block Flowchart



Example

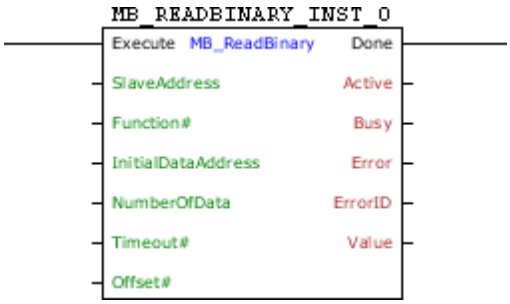


The example above requests status data of the Modbus RTU network master, and allows disabling communication through DISABLE. The block ends successfully, Done output is activated.

11.1.6.2.1.3 MB_ReadBinary

Block that performs a reading of up to 128 binary data (via Read Coils or Read Discrete Inputs) of a slave on the Modbus RTU network.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	Execute	BOOL	Block enabling
	SlaveAddress	BYTE	Slave address
	Function#	BYTE	Reading function code
	InitialDataAddress	WORD	Initial bit address of the data to be read
	NumberOfData	BYTE	Number of bits to be read (1 to 128)
	Timeout#	WORD	Maximum w aiting time for the slave response [ms]
	Offset#	BOOL	Offset Indication in InitialDataAddress, i.e., need to subtract 1 from this number
VAR_OUTPUT	Done	BOOL	Output enabling
	Active	BOOL	Aw aiting response flag
	Busy	BOOL	Flag indicating the RS485 interface is busy w ith another request
	Error	BOOL	Error in the execution flag
	ErrorID	BYTE	Identifier of the occurred error
	Value	BOOL	Variable that stores the received data
VAR	MB_READBINARY_INST_0	MB_READBINARY	Instance of access to block structure

Operation

When this block detects a leading edge on Execute, it checks whether the Modbus slave RTU in specified address in SlaveAddress is free to send data (Busy variable at FALSE level). If so, it sends the reading request of a number of bits indicated by NumberOfData in InitialDataAddress address using chosen function in Function# and sets the Active output, resetting it when receiving the response from the slave. The received data is stored in the Value variable. If the slave is not free, the block waits Busy go to FALSE level to resubmit the request.

NOTE!
If Execute goes to FALSE level and Busy is still at TRUE level, the request is canceled.

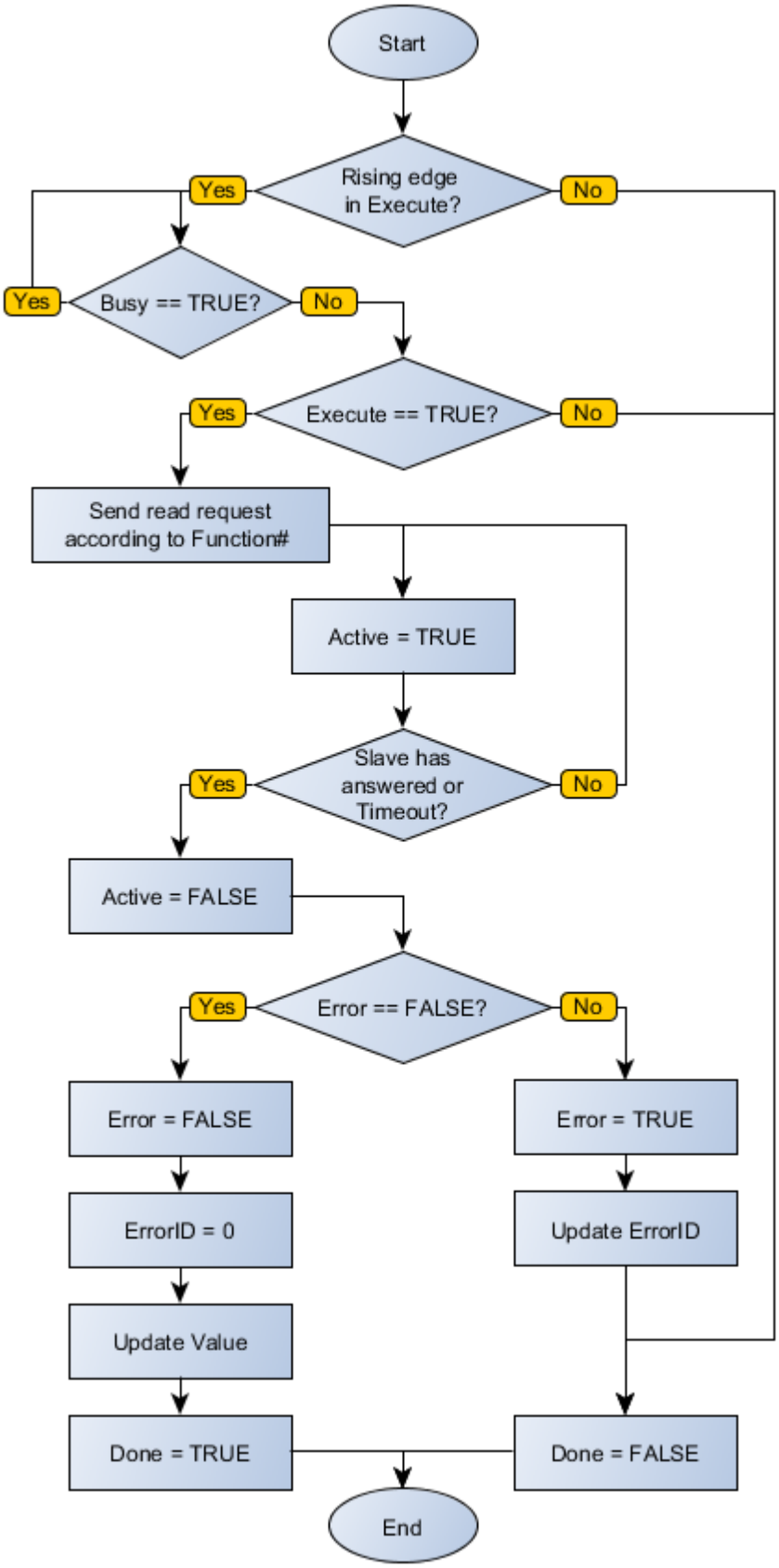
NOTE!
Value is an array of size equal to NumberOfData. It is important to check this compatibility not to generate errors in the block.

When Execute has FALSE value, Done remains FALSE. The Done output is only activated when the block finishes executing successfully, remaining at TRUE level until Execute receives FALSE.

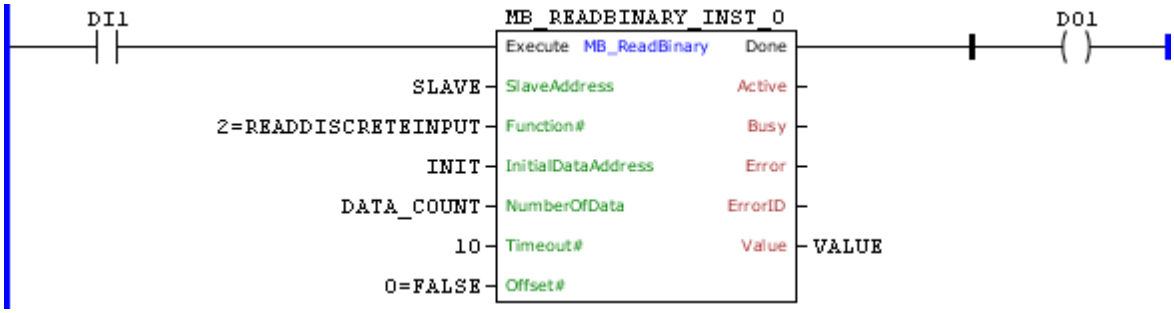
If there is any error in the execution, the Error output is enabled and ErrorID displays an error code according to the table below.

Code	Description
0	Executed successfully
1	Invalid input data
2	Master not enabled
4	Timeout in slave response
5	Slave returned error

Block Flowchart


Start
Rising edge in Execute?
Yes
No
Busy == TRUE?
Yes
No
Execute == TRUE?
Yes
No
Send read request according to Function#
Active = TRUE
Slave has answered or Timeout?
Yes
No
Active = FALSE
Error == FALSE?
Yes
No
Error = FALSE
ErrorID = 0
Update Value
Done = TRUE
Error = TRUE
Update ErrorID
Done = FALSE
End

Example

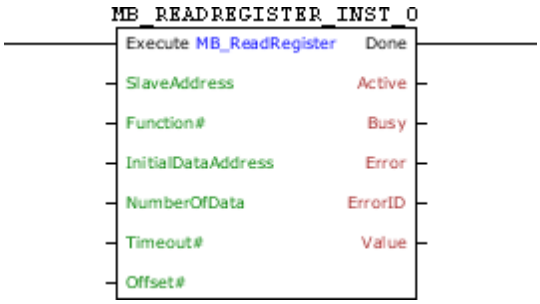


The above example requests reading of a number of binary data described by DATA_COUNT positioned in the INIT Modbus RTU slave of SLAVE address through the Read Discrete Input function. These data are forwarded to VALUE. The block ends successfully, Done output is activated.

11.1.6.2.1.4 MB_ReadRegister

Block that performs a reading of up to 64 16-bit registers (via Read Holding Registers or Read Input Registers) of a slave on the Modbus RTU network.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	Execute	BOOL	Block enabling
	SlaveAddress	BYTE	Slave address
	Function#	BYTE	Reading function code
	InitialDataAddress	WORD	Initial register address to be read
	NumberOfData	BYTE	Number of registers to be read (1 to 64)
	Timeout#	WORD	Maximum waiting time for the slave response [ms]
	Offset#	BOOL	Offset Indication in InitialDataAddress, i.e., need to subtract 1 from this number
VAR_OUTPUT	Done	BOOL	Output enabling
	Active	BOOL	Awaiting response flag
	Busy	BOOL	Flag indicating the RS485 interface is busy with another request
	Error	BOOL	Error in the execution flag
	ErrorID	BYTE	Identifier of the occurred error
	Value	BYTE SINT USINT WORD UINT INT DWORD UDINT DINT REAL	Variable that stores the received data
VAR	MB_READREGISTER_INST_0	MB_READREGISTER	Instance of access to block structure

Operation

When this block detects a leading edge on Execute, it checks whether the Modbus RTU slave in specified address in SlaveAddress is free to send data (Busy variable at FALSE level). If so, it sends the reading request of a number of registers indicated by NumberOfData in InitialDataAddress address using chosen function in Function# and sets the Active output, resetting them when receiving the response from the slave. The received data is stored in the Value variable. If the slave is not free, the block waits Busy go to FALSE level to resubmit the request.

NOTE!
If Execute goes to FALSE level and Busy is still at TRUE level, the request is canceled.

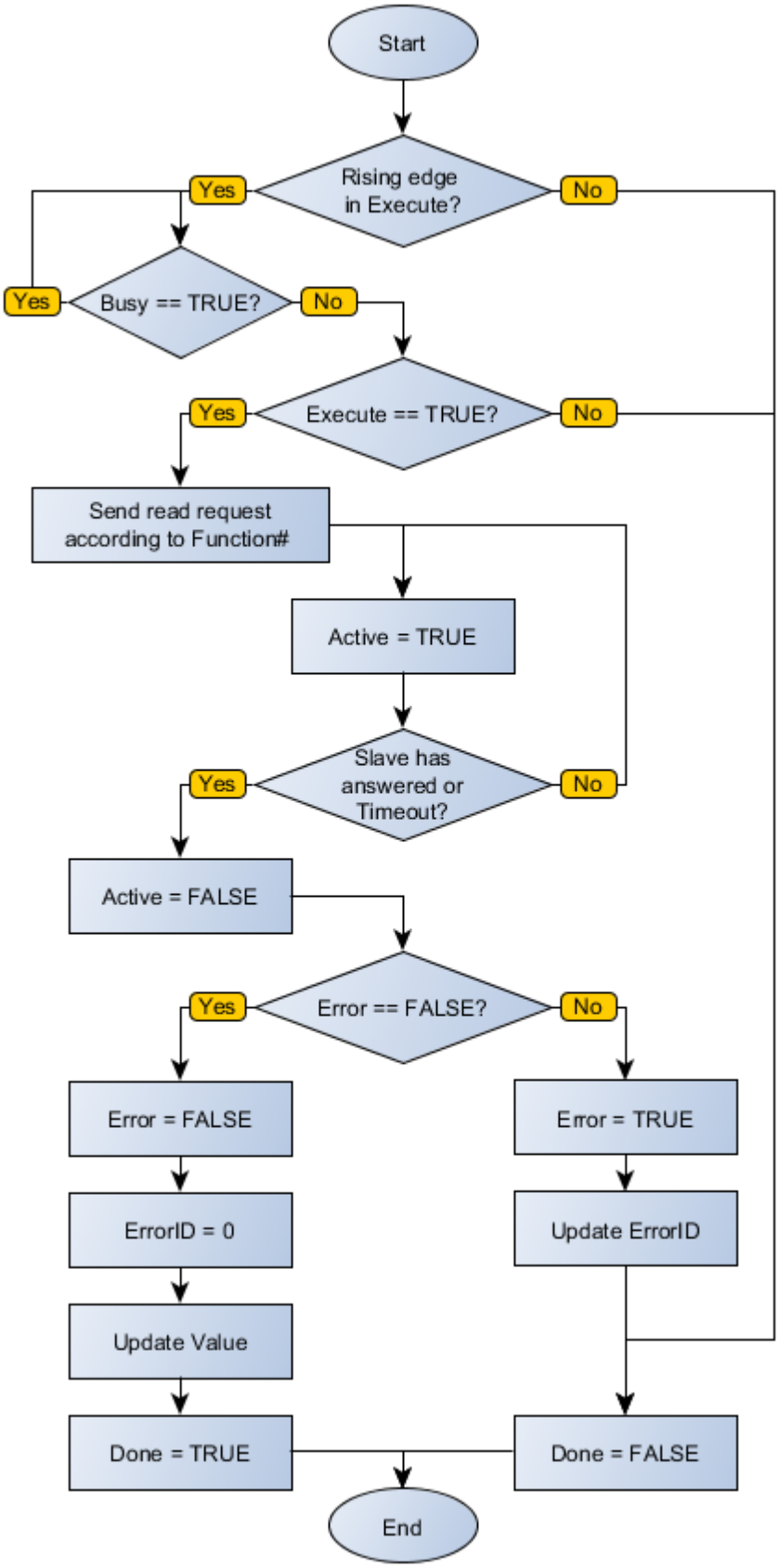
NOTE!
Value is an array of number of bits NumberOfData multiplied by 16. That is, if NumberOfData is 16, Value can be an array of 32 BYTE positions, 16 WORD positions or 8 DWORD positions. It is important to check this compatibility not to generate errors in the block.

When Execute has FALSE value, Done remains FALSE. The Done output is only activated when the block finishes executing successfully, remaining at TRUE level until Execute receives FALSE.

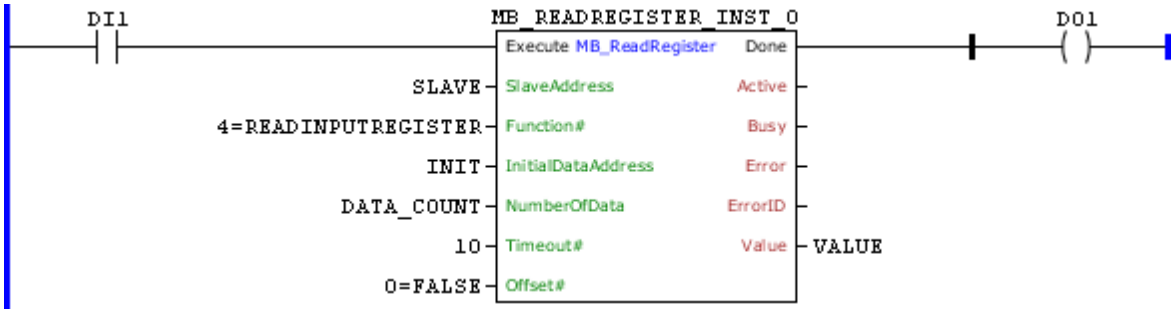
If there is any error in the execution, the Error output is enabled and ErrorID displays an error code according to the table below.

Code	Description
0	Executed successfully
1	Invalid input data
2	Master not enabled
4	Timeout in slave response
5	Slave returned error

Block Flowchart


Start
Rising edge in Execute?
Yes
No
Busy == TRUE?
Yes
No
Execute == TRUE?
Yes
No
Send read request according to Function#
Active = TRUE
Slave has answered or Timeout?
Yes
No
Active = FALSE
Error == FALSE?
Yes
No
Error = FALSE
ErrorID = 0
Update Value
Done = TRUE
Error = TRUE
Update ErrorID
Done = FALSE
End

Example



The above example requests reading of a number of binary data described by DATA_COUNT positioned in the INIT in the Modbus RTU slave of SLAVE address through the Read Input Register function. These data are forwarded to VALUE. The block ends successfully, Done output is activated.

11.1.6.2.1.5 MB_SlaveStatus

Block that allows monitoring the status of 4 slaves of the Modbus RTU network.

Ladder Representation

Block Structure

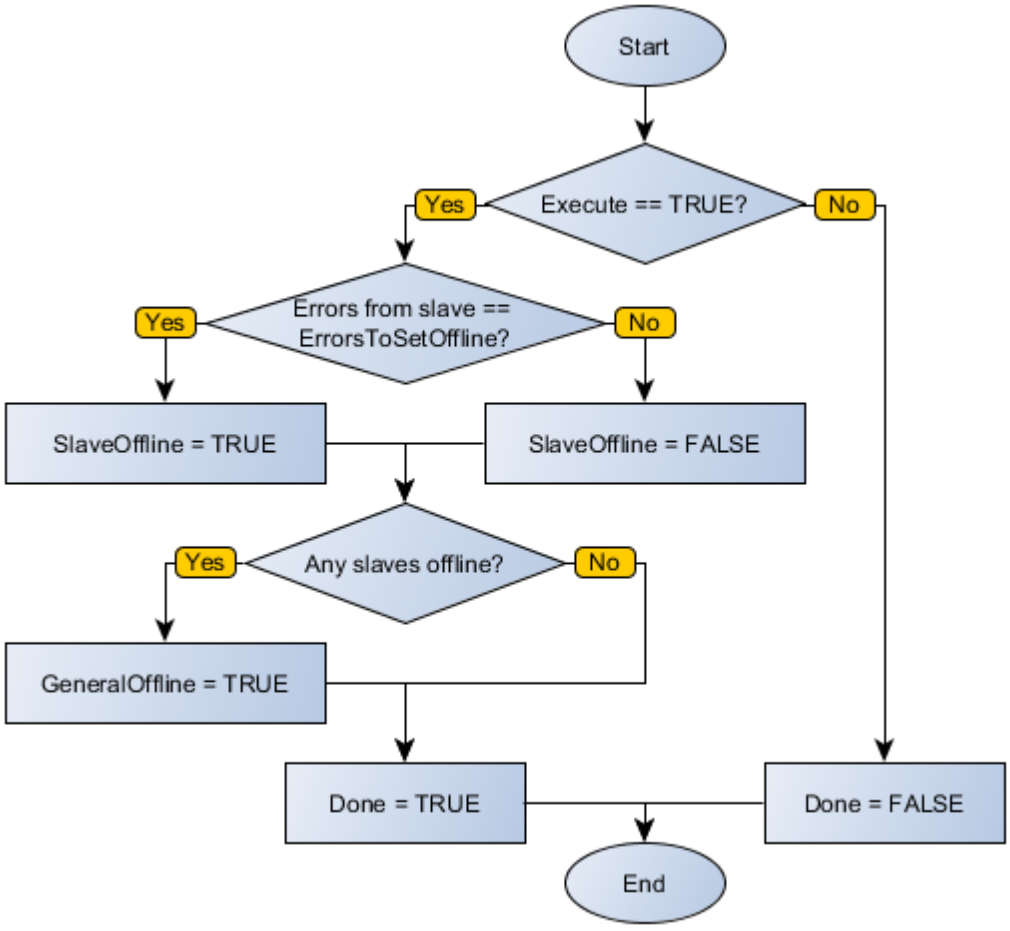
Variable Type	Name	Data Type	Description
VAR_INPUT	Execute	BOOL	Block enabling
	ErrorsToSetOffline#	BYTE	Amount of errors that the master must identify until it considers communication w ith an offline slave
	AddressSlave1#	BYTE	Slave address 1 to be monitored
	AddressSlave2#	BYTE	Slave address 2 to be monitored
	AddressSlave3#	BYTE	Slave address 3 to be monitored
	AddressSlave4#	BYTE	Slave address 4 to be monitored
VAR_OUTPUT	Done	BOOL	Output enabling
	GeneralOffline	BOOL	Flag indicating any one of the monitored communication is offline
	Slave1Offline	BOOL	Flag of offline status slave 1
	Slave2Offline	BOOL	Flag of offline status slave 2
	Slave3Offline	BOOL	Flag of offline status slave 3
	Slave4Offline	BOOL	Flag of offline status slave 4

Operation

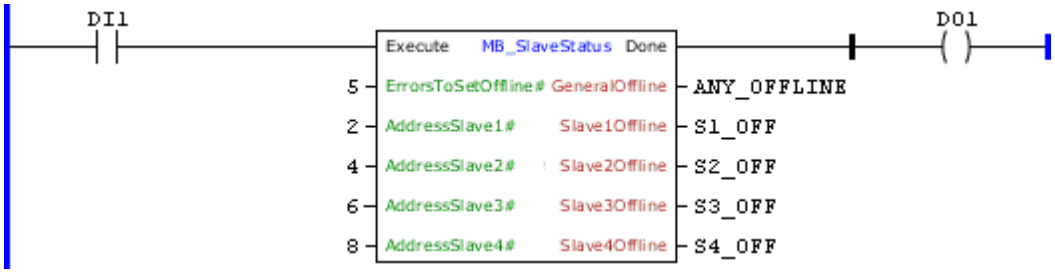
This block remains active while Execute is at TRUE level, updating its outputs according to the number of errors recorded for each slave. When Execute receives FALSE level, the inputs are ignored and the outputs are zeroed. The Done output receives TRUE level when Execute has TRUE level and block finished its execution.

The ErrorsToSetOffline # input allows registering the number of errors identified in a slave that will feature an offline communication. AddressSlave inputs allow inserting four slave addresses to be monitored. When this monitored slave reports the programmed number of errors, its corresponding SlaveOffline output is set to TRUE level. If any of SlaveOffline outputs is at TRUE level, GeneralOffline also receives TRUE level.

Block Flowchart



Example

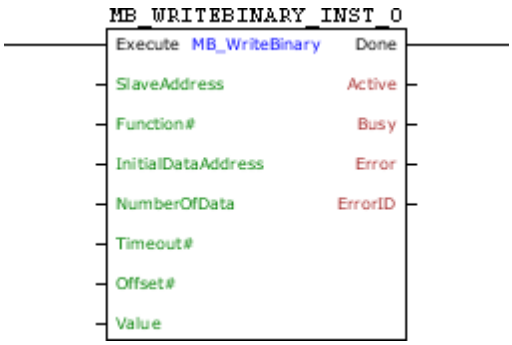


The above example checks the number of error responses sent by the slaves 2, 4, 6 and 8 of the Modbus RTU. If any of them is greater than 5, its SX_OFF status is led to TRUE level. The block ends successfully, Done output is activated.

11.1.6.2.1.6 MB_WriteBinary

Block that performs a writing of up to 128 binary data (via Write Single Coil or Write Multiple Coils) in a slave on the Modbus RTU network.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	Execute	BOOL	Block enabling
	SlaveAddress	BYTE	Slave address
	Function#	BYTE	Writing function code
	InitialDataAddress	WORD	Initial bit address where the data will be written
	NumberOfData	BYTE	Number of bits to be written (1 to 128)
	Timeout#	WORD	Maximum waiting time for the slave response [ms]
	Offset#	BOOL	Offset Indication in InitialDataAddress, i.e., need to subtract 1 from this number
	Value	BOOL	Variable that stores the data to be written
VAR_OUTPUT	Done	BOOL	Output enabling
	Active	BOOL	Awaiting response flag
	Busy	BOOL	Flag indicating the RS485 interface is busy with another request
	Error	BOOL	Error in the execution flag
	ErrorID	BYTE	Identifier of the occurred error
VAR	MB_WRITEBINARY_INST_0	MB_WRITEBINARY	Instance of access to block structure

Operation

When this block detects a leading edge on Execute, it checks whether the Modbus RTU slave in specified address in SlaveAddress is free to send data (Busy variable at FALSE level). If so, it sends the writing request of a number of bits indicated by NumberOfData in InitialDataAddress address using chosen function in Function# and sets the Active output, resetting it when receiving the response from the slave. If the slave is not free, the block waits Busy go to FALSE level to resubmit the request.

NOTE!
If Execute goes to FALSE level and Busy is still at TRUE level, the request is canceled.

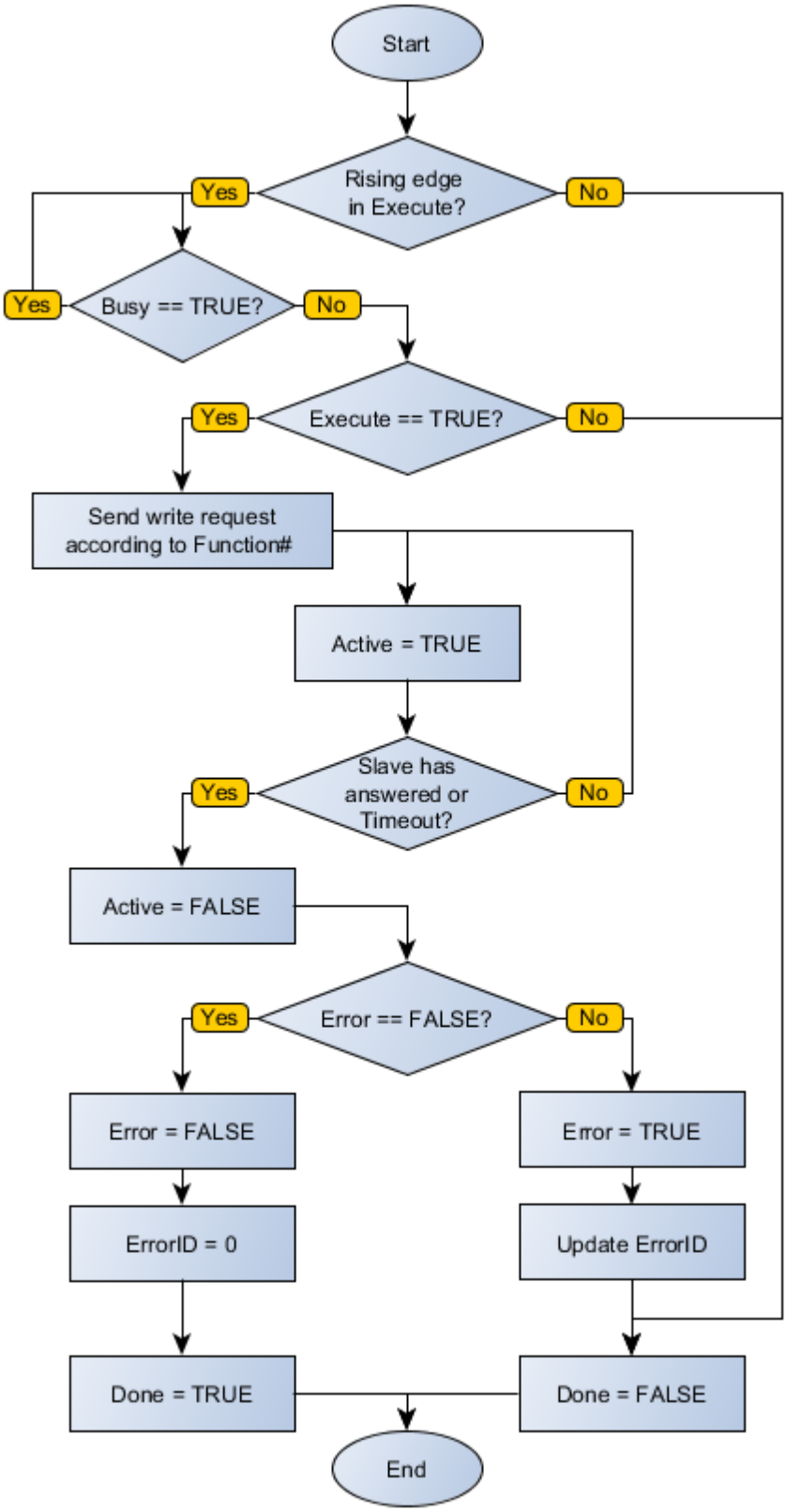
NOTE!
Value is an array of size equal to NumberOfData. It is important to check this compatibility not to generate errors in the block.

When Execute has FALSE value, Done remains FALSE. The Done output is only activated when the block finishes executing successfully, remaining at TRUE level until Execute receives FALSE.

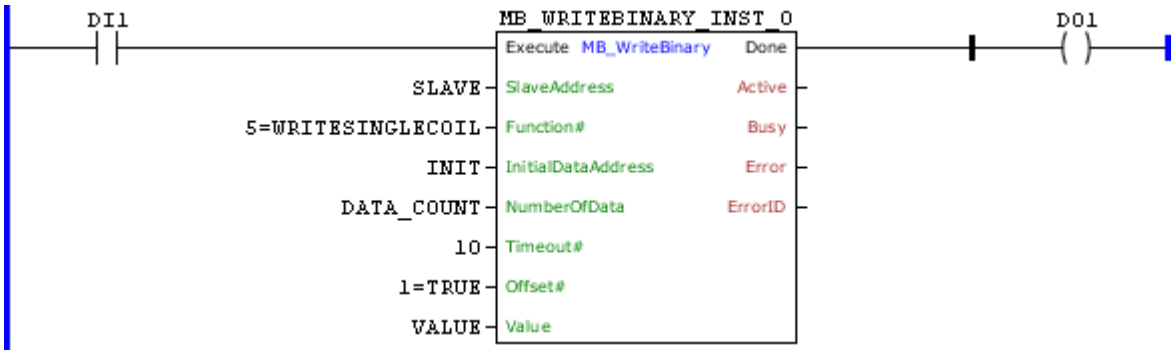
If there is any error in the execution, the Error output is enabled and ErrorID displays an error code according to the table below.

Code	Description
0	Executed successfully
1	Invalid input data
2	Master not enabled
4	Timeout in slave response
5	Slave returned error

Block Flowchart


Start
Rising edge in Execute?
Yes
No
Busy == TRUE?
Yes
No
Execute == TRUE?
Yes
No
Send write request according to Function#
Active = TRUE
Slave has answered or Timeout?
Yes
No
Active = FALSE
Error == FALSE?
Yes
No
Error = FALSE
Error = TRUE
ErrorID = 0
Update ErrorID
Done = TRUE
Done = FALSE
End

Example

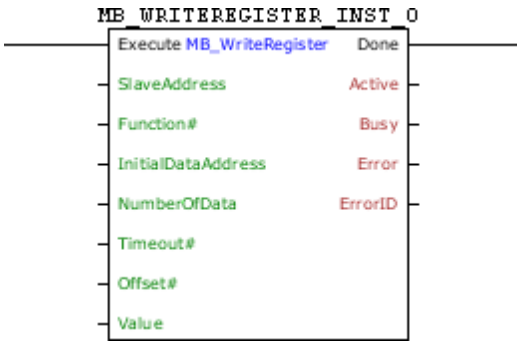


The example above requests written data contained in VALUE, with size described by DATA_COUNT, at addresses positioned from INIT on Modbus RTU slave at address SLAVE using the function Write Single Coil. The block ends successfully, Done output is activated.

11.1.6.2.1.7 MB_WriteRegister

Block that performs a reading of up to sixteen 16-bit registers (via Write Single Register or Write Multiple Registers) of a slave on the Modbus RTU network.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	Execute	BOOL	Block enabling
	SlaveAddress	BYTE	Slave address
	Function#	BYTE	Writing function code
	InitialDataAddress	WORD	Initial register address to be written
	NumberOfData	BYTE	Number of registers to be written (1 to 16)
	Timeout#	WORD	Maximum waiting time for the slave response [ms]
	Offset#	BOOL	Offset Indication in InitialDataAddress, i.e., need to subtract 1 from this number
	Value	BYTE SINT USINT WORD UINT INT DWORD UDINT DINT REAL	Variable that stores the data to be written
VAR_OUTPUT	Done	BOOL	Output enabling
	Active	BOOL	Awaiting response flag
	Busy	BOOL	Flag indicating the RS485 interface is busy with another request
	Error	BOOL	Error in the execution flag
	ErrorID	BYTE	Identifier of the occurred error
VAR	MB_WRITEREGISTER_INST_0	MB_WRITEREGISTER	Instance of access to block structure

Operation

When this block detects a leading edge on Execute, it checks whether the Modbus RTU slave in specified address in SlaveAddress is free to send data (Busy variable at FALSE level). If so, it sends the writing request of Value values in a number of registers indicated by NumberOfData in InitialDataAddress address using chosen function in Function# and sets the Active output, resetting it when receiving the response from the slave. If the slave is not free, the block waits Busy go to FALSE level to resubmit the request.

NOTE!
If Execute goes to FALSE level and Busy is still at TRUE level, the request is canceled.

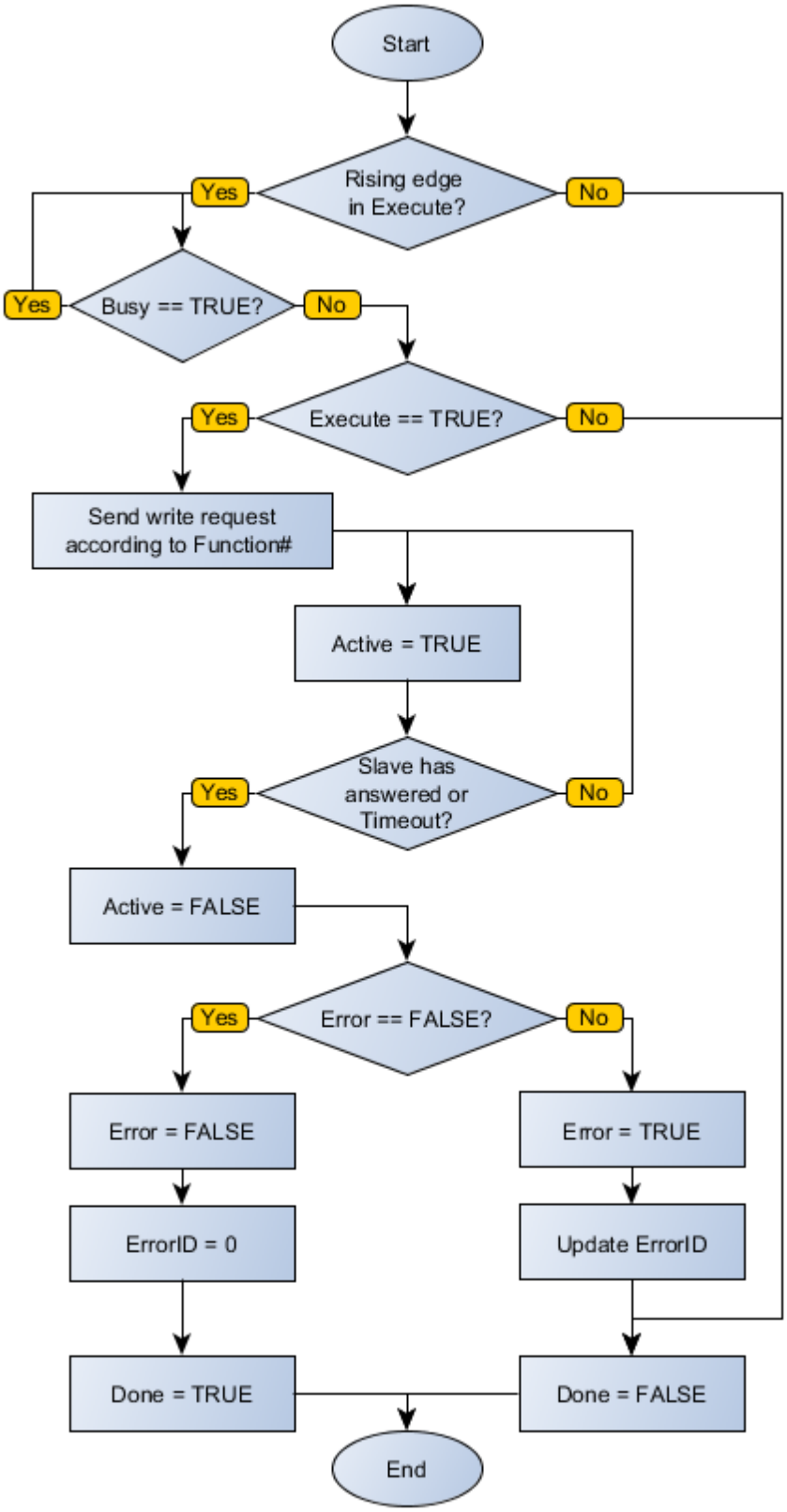
NOTE!
Value is an array of number of bits NumberOfData multiplied by 16. That is, if NumberOfData is 16, Value can be an array of 32 BYTE positions, 16 WORD positions or 8 DWORD positions. It is important to check this compatibility not to generate errors in the block.

When Execute has FALSE value, Done remains FALSE. The Done output is only activated when the block finishes executing successfully, remaining at TRUE level until Execute receives FALSE.

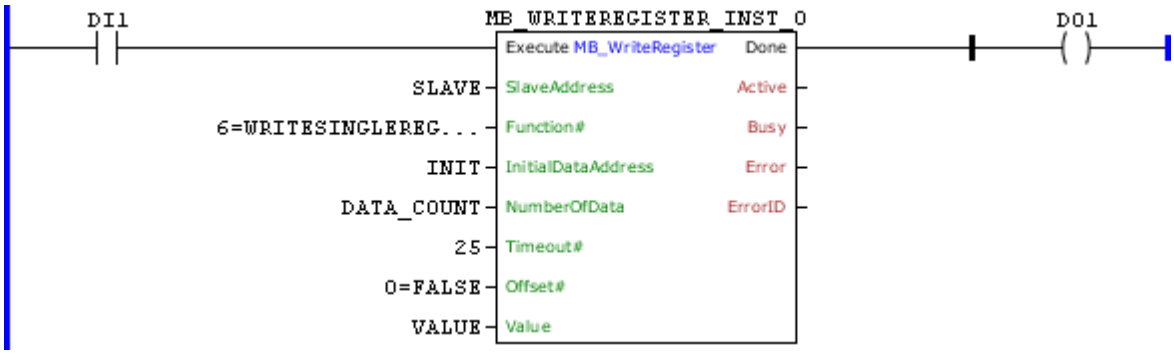
If there is any error in the execution, the Error output is enabled and ErrorID displays an error code according to the table below.

Code	Description
0	Executed successfully
1	Invalid input data
2	Master not enabled
4	Timeout in slave response
5	Slave returned error

Block Flowchart


Start
Rising edge in Execute?
Yes
No
Busy == TRUE?
Yes
No
Execute == TRUE?
Yes
No
Send write request according to Function#
Active = TRUE
Slave has answered or Timeout?
Yes
No
Active = FALSE
Error == FALSE?
Yes
No
Error = FALSE
Error = TRUE
ErrorID = 0
Update ErrorID
Done = TRUE
Done = FALSE
End

Example



The example above requests written data contained in VALUE, with size described by DATA_COUNT, at addresses positioned from INIT on Modbus RTU slave at address SLAVE using the function Write Single Register. The block ends successfully, Done output is activated.

11.1.6.3 Compare

11.1.6.3.1 COMP_EQ

Block that compares the values of Value1 and Value2, enabling the output Q if both are equal.

Ladder Representation

Block Structure

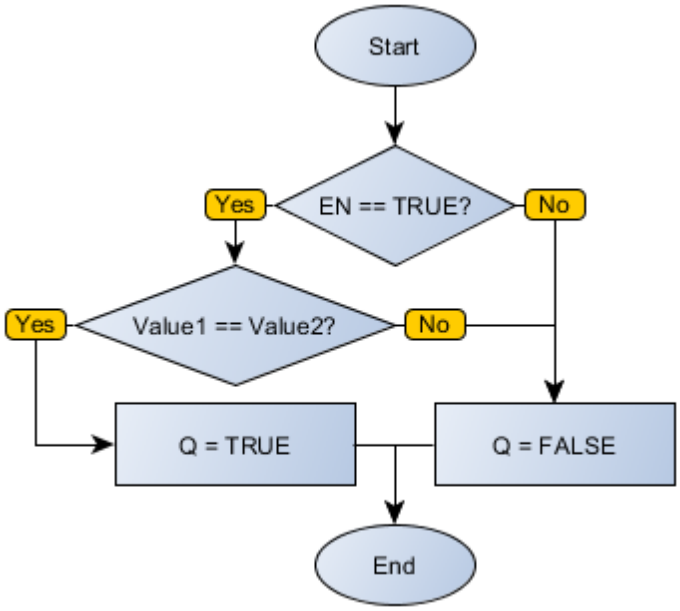
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	First value of comparison
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Second value of comparison
VAR_OUTPUT	Q	BOOL	Indicator of equality

Operation

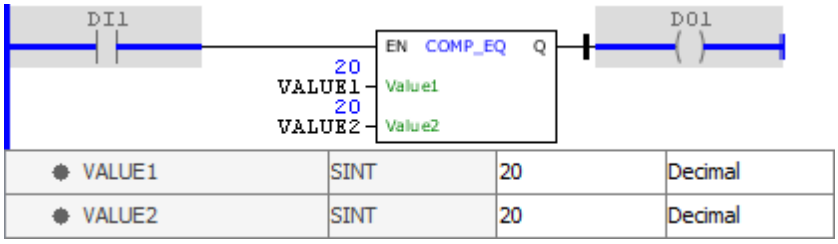
When this block has a TRUE value in EN, it sends to the output Q the TRUE value if Value1 and Value2 are the same. Otherwise, Q receives FALSE.

When EN has FALSE value, Q remains in FALSE.

Block Flowchart

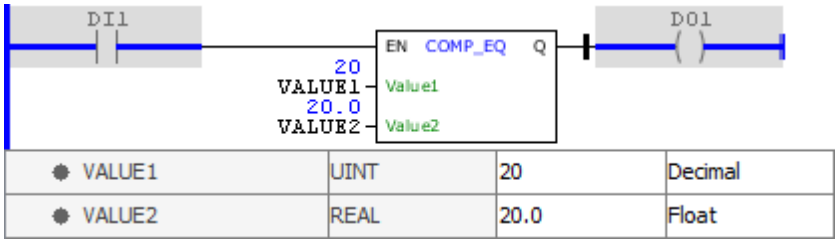


Example



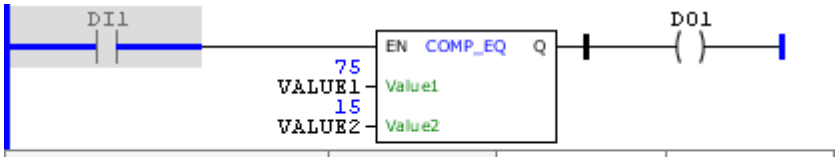
VALUE1	SINT	20	Decimal
VALUE2	SINT	20	Decimal

The example above checks equality between VALUE1 and VALUE2. Since both variables have the same value, the Q output is activated.



VALUE1	UINT	20	Decimal
VALUE2	REAL	20.0	Float

The example above checks equality between VALUE1 and VALUE2. Since both variables have the same value, the Q output is activated. Notice that the types of the input variables can be different without causing execution problems.



VALUE1	UINT	75	Decimal
VALUE2	UINT	15	Decimal

The example above checks equality between VALUE1 and VALUE2. Since both variables have different values, the Q output is disabled.

11.1.6.3.2 COMP_GE

Block that compares the values of Value1 and Value2, enabling the output Q if Value1 is higher than or equal to Value2.

Ladder Representation

Block Structure

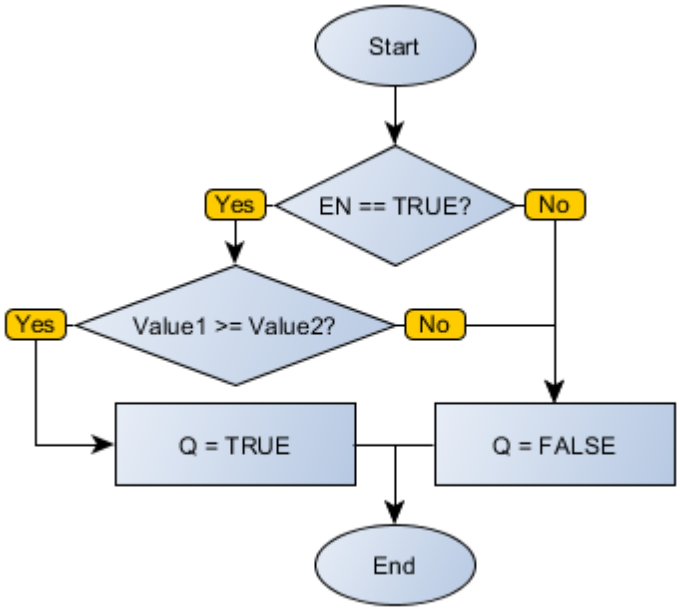
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	First value of comparison
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Second value of comparison
VAR_OUTPUT	Q	BOOL	Indicator of equality or majority of Value1

Operation

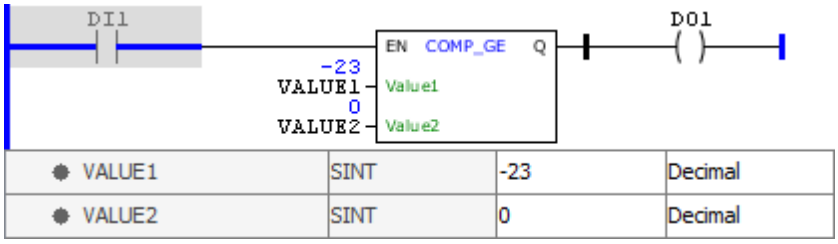
When this block has a TRUE value in EN it sends the Q output to the TRUE value if Value1 is higher than or equal to Value2. Otherwise, Q receives FALSE.

When EN has FALSE value, Q remains in FALSE.

Block Flowchart

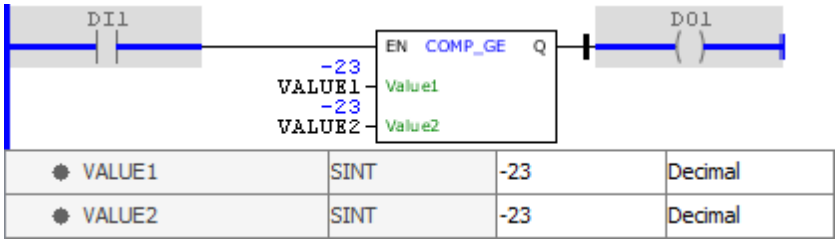


Example



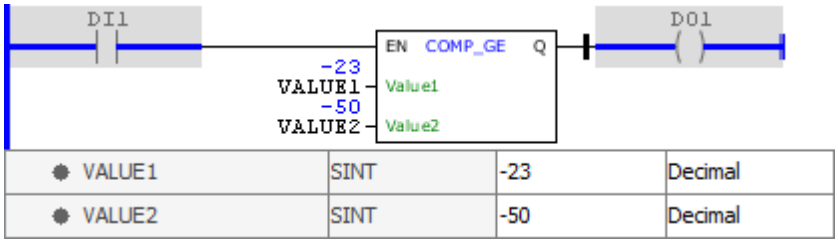
VALUE1	SINT	-23	Decimal
VALUE2	SINT	0	Decimal

The example above checks equality or majority of VALUE1 in relation to VALUE2. Since VALUE1 has lower value than VALUE2, the Q output is disabled.



VALUE1	SINT	-23	Decimal
VALUE2	SINT	-23	Decimal

The example above checks equality or majority of VALUE1 in relation to VALUE2. Since both variables have the same value, the Q output is activated.



VALUE1	SINT	-23	Decimal
VALUE2	SINT	-50	Decimal

The example above checks equality or majority of VALUE1 in relation to VALUE2. Since VALUE1 has higher value than VALUE2, the Q output is activated.

11.1.6.3.3 COMP_GT

Block that compares the values of Value1 and Value2, enabling the output Q if Value1 is higher than Value2.

Ladder Representation

Block Structure

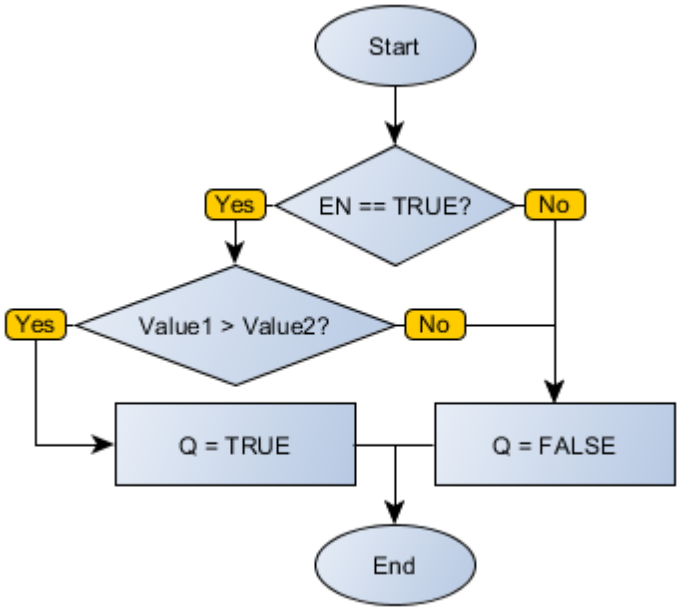
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	First value of comparison
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Second value of comparison
VAR_OUTPUT	Q	BOOL	Indicator of majority of Value1

Operation

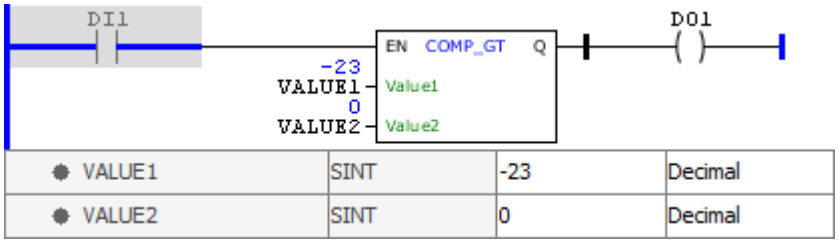
When this block has a TRUE value in EN, it sends to the Q output the TRUE value if Value1 is higher than Value2. Otherwise, Q receives FALSE.

When EN has FALSE value, Q remains in FALSE.

Block Flowchart

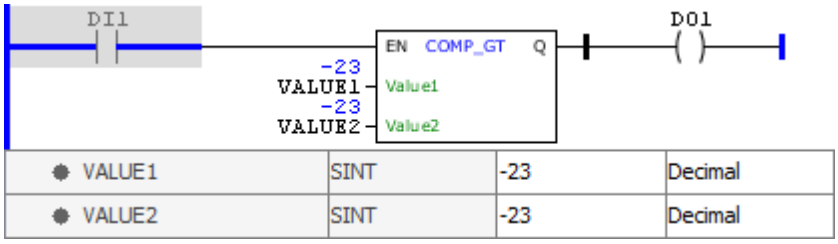


Example



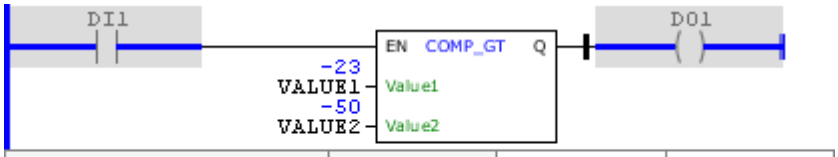
VALUE1	SINT	-23	Decimal
VALUE2	SINT	0	Decimal

The example above checks the majority of VALUE1 in relation to VALUE2. Since VALUE1 has lower value than VALUE2, the Q output is disabled.



VALUE1	SINT	-23	Decimal
VALUE2	SINT	-23	Decimal

The example above checks the majority of VALUE1 in relation to VALUE2. Since both variables have the same value, the Q output is disabled.



VALUE1	SINT	-23	Decimal
VALUE2	SINT	-50	Decimal

The example above checks the majority of VALUE1 in relation to VALUE2. Since VALUE1 has higher value than VALUE2, the Q output is activated.

11.1.6.3.4 COMP_LE

Block that compares the values of Value1 and Value2, enabling the output Q if Value1 is lower than or equal to Value2.

Ladder Representation

Block Structure

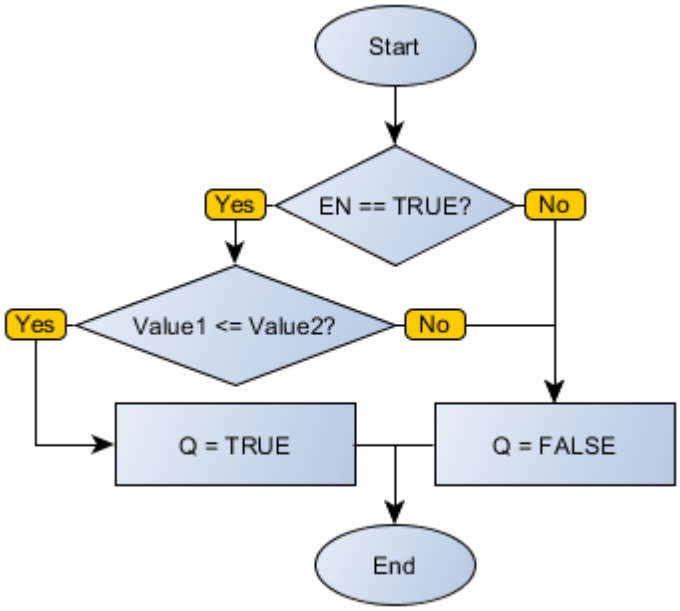
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	First value of comparison
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Second value of comparison
VAR_OUTPUT	Q	BOOL	Indicator of equality or minority of Value1

Operation

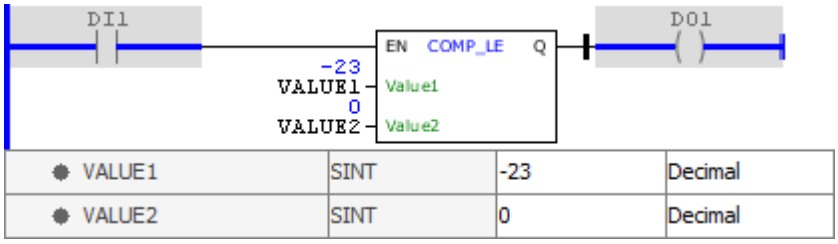
When this block has a TRUE value in EN, it sends to the Q output the TRUE value if Value1 is lower than or equal to Value2. Otherwise, Q receives FALSE.

When EN has FALSE value, Q remains in FALSE.

Block Flowchart

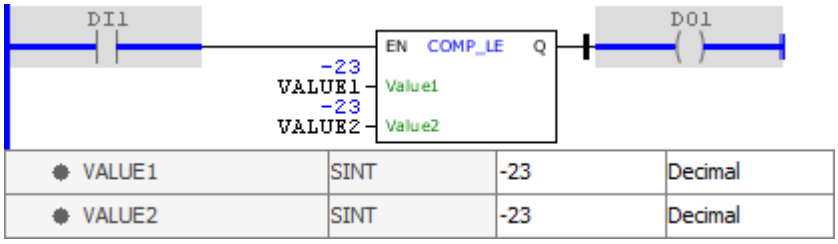


Example



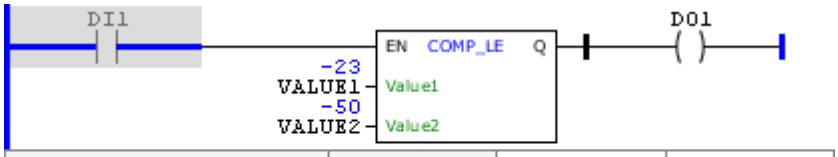
VALUE1	SINT	-23	Decimal
VALUE2	SINT	0	Decimal

The example above checks equality or minority of VALUE1 in relation to VALUE2. Since VALUE1 has lower value than VALUE2, the Q output is activated.



VALUE1	SINT	-23	Decimal
VALUE2	SINT	-23	Decimal

The example above checks equality or minority of VALUE1 in relation to VALUE2. Since both variables have the same value, the Q output is activated.



VALUE1	SINT	-23	Decimal
VALUE2	SINT	-50	Decimal

The example above checks equality or minority of VALUE1 in relation to VALUE2. Since VALUE1 has higher value than VALUE2, the Q output is disabled.

11.1.6.3.5 COMP_LT

Block that compares the values of Value1 and Value2, enabling the output Q if Value1 is lower than Value2.

Ladder Representation

Block Structure

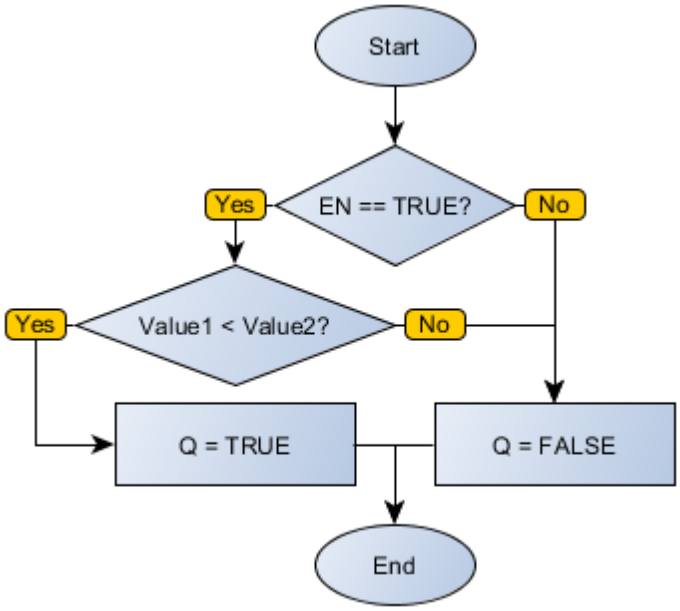
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	First value of comparison
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Second value of comparison
VAR_OUTPUT	Q	BOOL	Indicator of minority of Value1

Operation

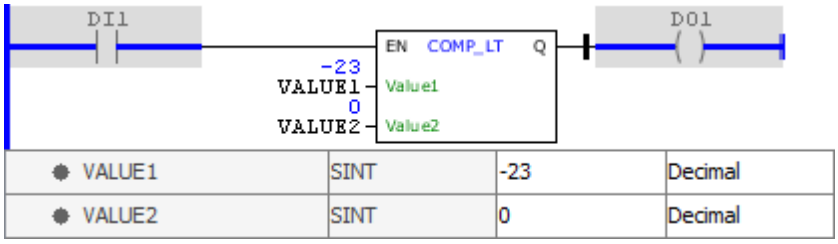
When this block has a TRUE value in EN, it sends to the Q output the TRUE value if Value1 is lower than or equal to Value2. Otherwise, Q receives FALSE.

When EN has FALSE value, Q remains in FALSE.

Block Flowchart

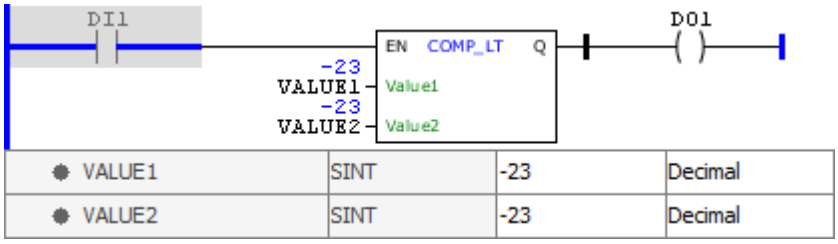


Example



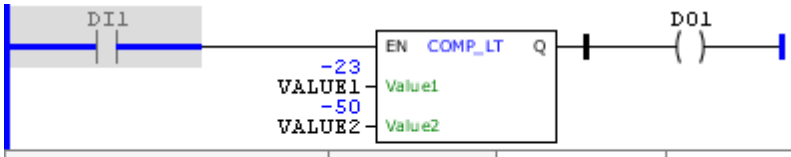
VALUE1	SINT	-23	Decimal
VALUE2	SINT	0	Decimal

The example above checks minority of VALUE1 in relation to VALUE2. Since VALUE1 has lower value than VALUE2, the Q output is activated.



VALUE1	SINT	-23	Decimal
VALUE2	SINT	-23	Decimal

The example above checks the minority of VALUE1 in relation to VALUE2. Since both variables have the same value, the Q output is disabled.



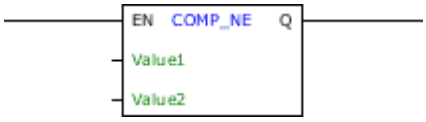
VALUE1	SINT	-23	Decimal
VALUE2	SINT	-50	Decimal

The example above checks the minority of VALUE1 in relation to VALUE2. Since VALUE1 has higher value than VALUE2, the Q output is disabled.

11.1.6.3.6 COMP_NE

Block that compares the values of Value1 and Value2, enabling the Q output if Value1 is different from Value2.

Ladder Representation



Block Structure

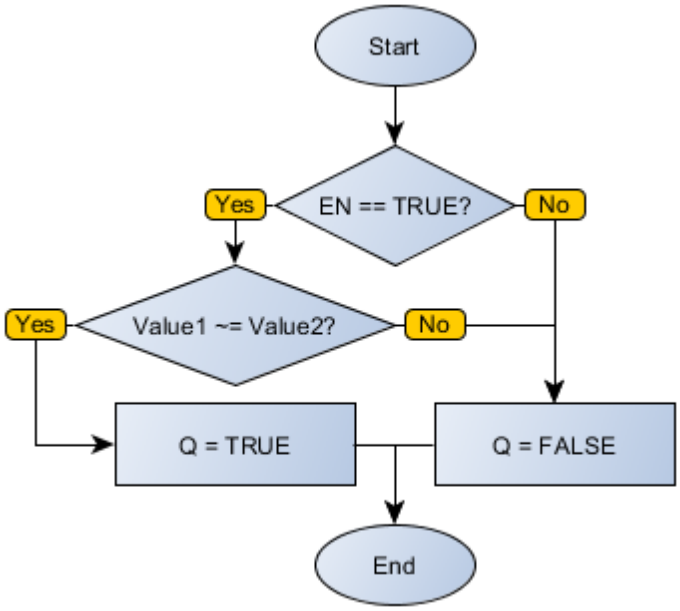
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	First value of comparison
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Second value of comparison
VAR_OUTPUT	Q	BOOL	Indicator of inequality

Operation

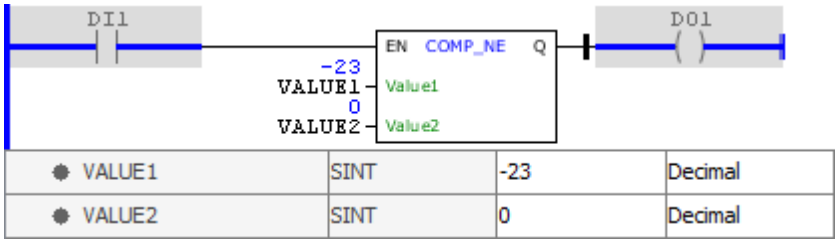
When this block has a TRUE value in EN, it sends to the Q output the TRUE value if Value1 is different from Value2. Otherwise, Q receives FALSE.

When EN has FALSE value, Q remains in FALSE.

Block Flowchart

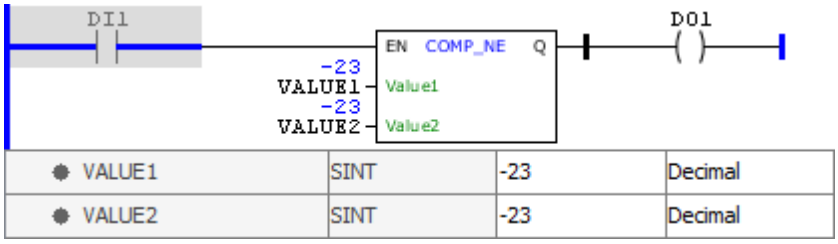


Example



VALUE1	SINT	-23	Decimal
VALUE2	SINT	0	Decimal

The example above checks inequality between VALUE1 and VALUE2. Since both variables have different values, the Q output is activated.



VALUE1	SINT	-23	Decimal
VALUE2	SINT	-23	Decimal

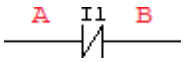
The example above checks equality between VALUE1 and VALUE2. Since both variables have the same value, the Q output is disabled.

11.1.6.4 Contact

11.1.6.4.1 NCCONTACT

Normally closed contact.

Ladder Representation



Block Structure

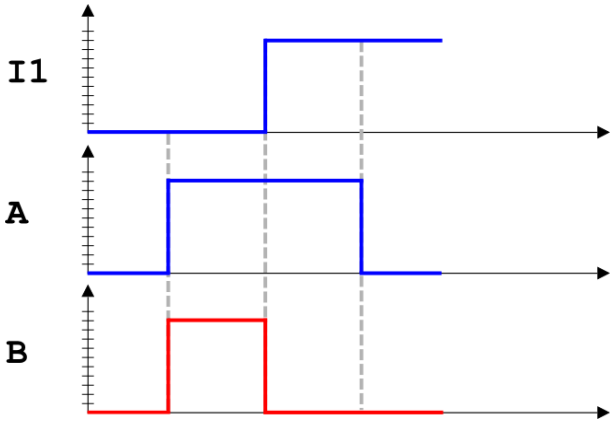
Variable Type	Name	Data Type	Description
VAR_OUTPUT	I1	BOOL	Block control input

Operation

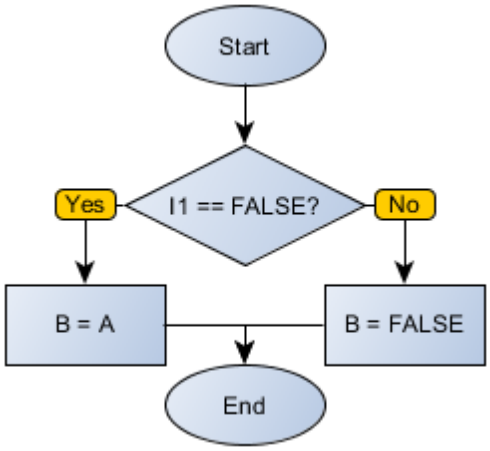
When variable I1 is with TRUE value, B receives FALSE.
When variable I1 is with FALSE value, B receives the value of A.

NOTE!
Watch out for series and parallel associations of contacts. Refer to section Contact Logic for further information.

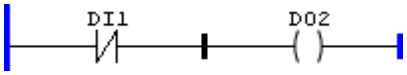
Diagram



Block Flowchart



Example

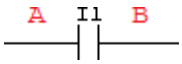


The above example performs the transfer of the opposite value of digital input DI1 to the digital output DO2.

11.1.6.4.2 NOCONTACT

Normally open contact.

Ladder Representation



Block Structure

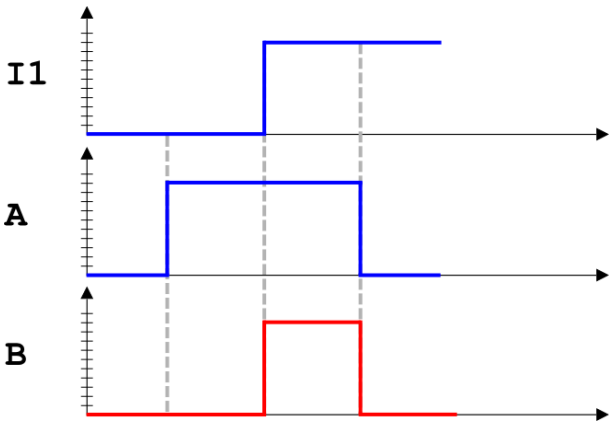
Variable Type	Name	Data Type	Description
VAR_OUTPUT	I1	BOOL	Block control input

Operation

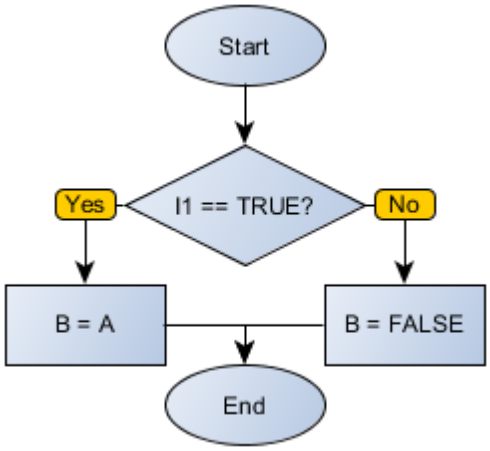
When variable I1 is with FALSE value, B receives FALSE.
When variable I1 is with TRUE value, B receives the value of A.

NOTE!
Watch out for series and parallel associations of contacts. Refer to section Contact Logic for further information.

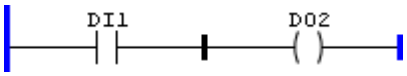
Diagram



Block Flowchart



Example



The above example performs the transfer of the value of digital input DI1 to the digital output DO2.

11.1.6.4.3 NTSCONTACT

Falling edge transition contact.

Ladder Representation

Block Structure

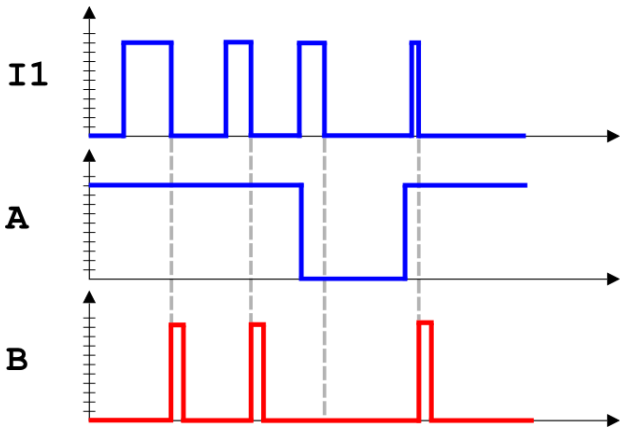
Variable Type	Name	Data Type	Description
VAR_INPUT	I1	BOOL	Block control input
VAR	NTSCONTACT_INST_0	NTSCONTACT	Instance of access to block structure

Operation

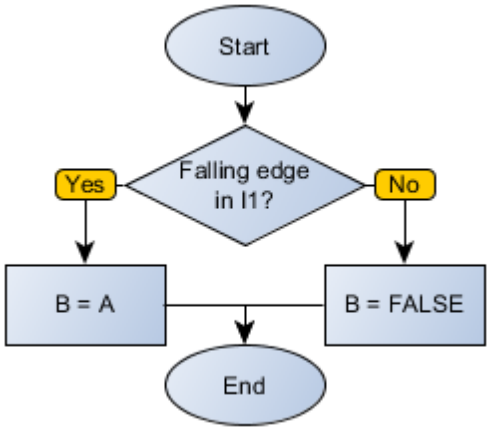
At the instant the variable I1 transitions from TRUE to FALSE (falling edge or negative edge transition), B receives the value of A for a scan cycle.
At all other times, B receives the FALSE value.

NOTE!
Watch out for series and parallel associations of contacts. Refer to section Contact Logic for further information.

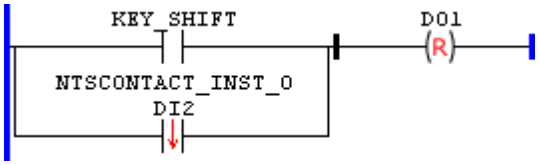
Diagram



Block Flowchart



Example

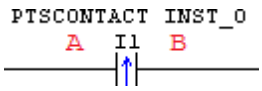


The above example resets the digital output DO1 if the SHIFT key is pressed or a positive pulse on the digital input DI2 is given.

11.1.6.4.4 PTSCONTACT

Leading edge transition contact.

Ladder Representation



Block Structure

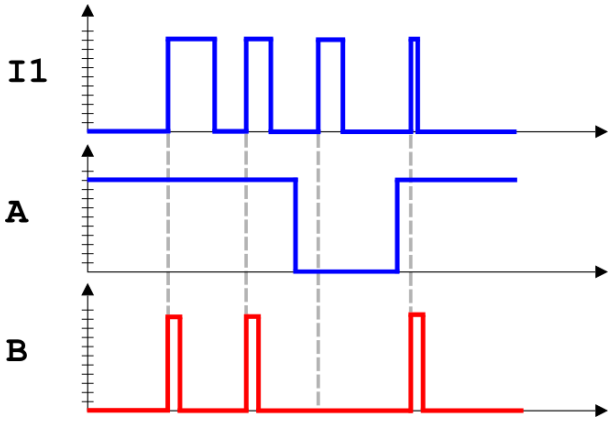
Variable Type	Name	Data Type	Description
VAR_INPUT	I1	BOOL	Block control input
VAR	PTSCONTACT_INST_0	PTSCONTACT	Instance of access to block structure

Operation

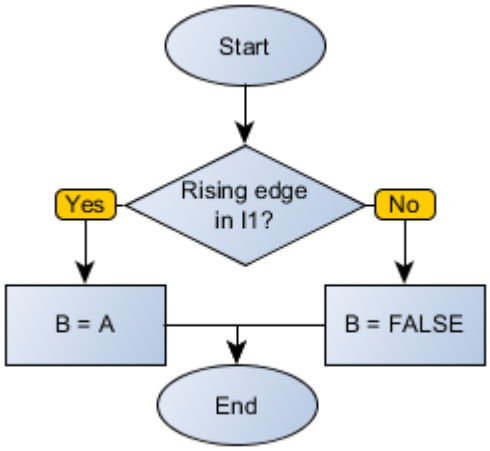
At the instant the variable I1 transitions from FALSE to TRUE (leading edge or positive edge transition), B receives the value of A for a scan cycle.
At all other times, B receives the FALSE value.

NOTE!
Watch out for series and parallel associations of contacts. Refer to section Contact Logic for further information.

Diagram



Block Flowchart



Example

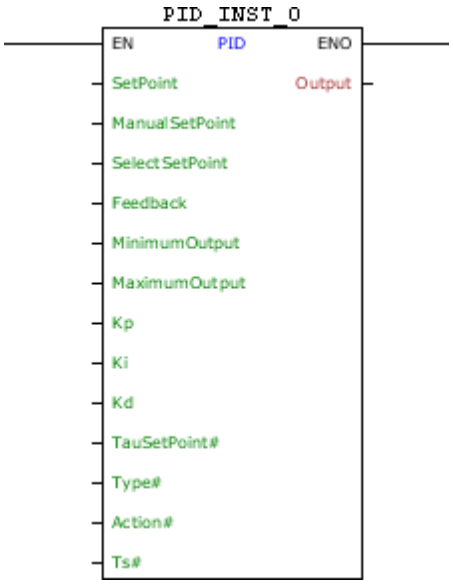
The above example resets the digital output DO1 if the SHIFT key is pressed and a positive pulse on the digital input DI2 is given.

11.1.6.5 Control

11.1.6.5.1 PID

Block that performs the function of a discrete PID controller. From the input variables, it calculates the corresponding controller output.

Ladder Representation



Block Structure

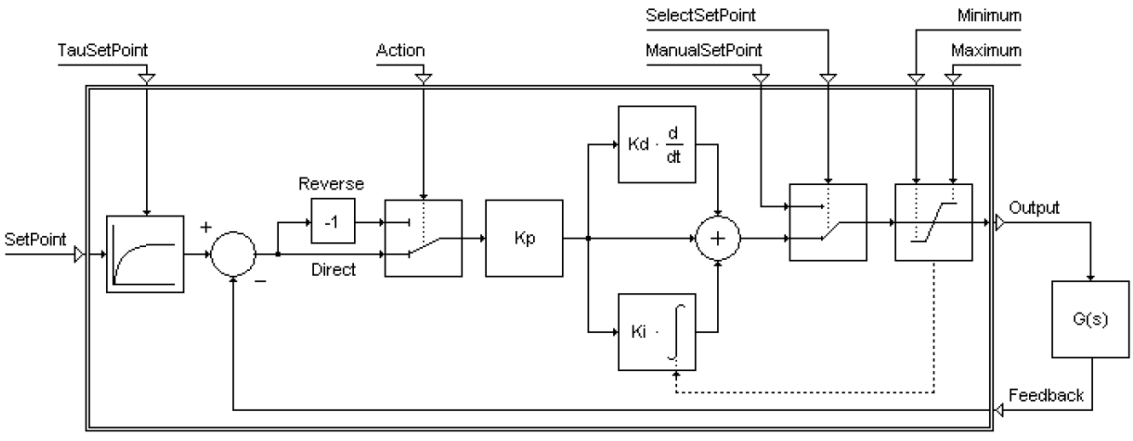
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	SetPoint	REAL	Automatic reference (pre-control)
	ManualSetPoint	REAL	Forced reference (post control)
	SelectSetPoint	BOOL	Selects w hich reference to use
	Feedback	REAL	Feedback loop variable
	MinimumOutput	REAL	Minimum value of the controller output
	MaximumOutput	REAL	Maximum value of the controller output
	Kp	REAL	Proportional gain
	Ki	REAL	Integral gain
	Kd	REAL	Derivative gain
	TauSetPoint#	REAL	Time constant of the automatic reference in put filter
	Type#	BYTE	Controller type
	Action#	BYTE	Control action
	Ts#	UINT	Sampling time [ms]
VAR_OUTPUT	ENO	BOOL	Output enabling
	Output	REAL	Controller output
VAR	PID_INST_0	PID	Instance of access to block structure

Operation

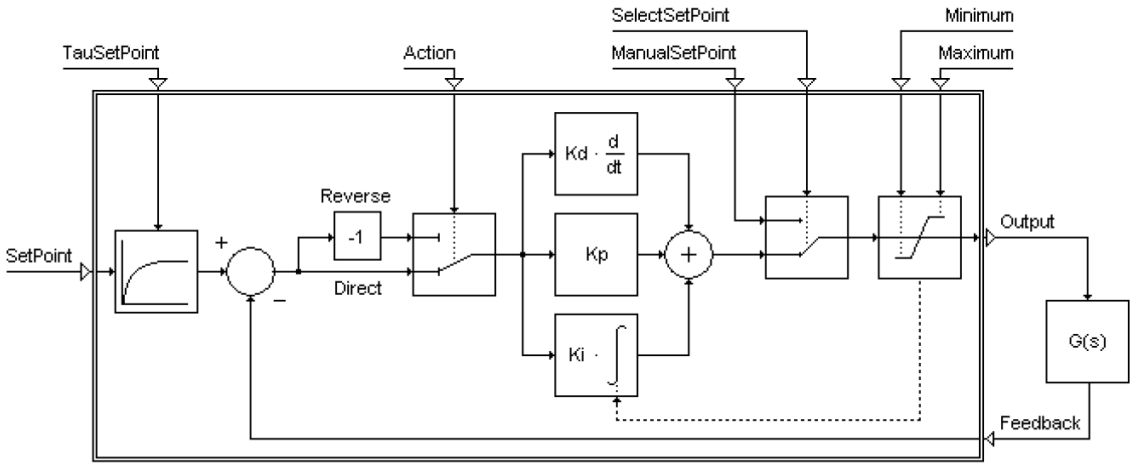
On the positive transition edge in EN, Output receives zero value, and the block executes its functionality as EN is at TRUE level.

When enabled, this block performs a routine PID control with the Kp, Ki and Kd parameters chosen. The PID topology used may be the Academic or Parallel, depending on what is chosen in Type#.

Academic Form:



Parallel Form:



The levels of the output signal of the controller are saturated at value MinimumOutput and MaximumOutput. The SelectSetPoint input level FALSE causes the SetPoint reference be adopted, allowing the controller maintains control over the process. When SelectSetPoint goes to TRUE level, the controller has no more domain, and ManualSetPoint becomes to be considered the output signal of the controller.

Action# will define the feedback operation. If Action# is DIRECT, the operation will be SetPoint – Feedback. If Action# is REVERSE, the operation will be Feedback – SetPoint.

Feedback receives the process variable considered as the plant output. Ts# receives the sampling time for the controller and # TauSetPoint receives the time constant for the input filter of the automatic reference.

When EN has FALSE value, Output remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

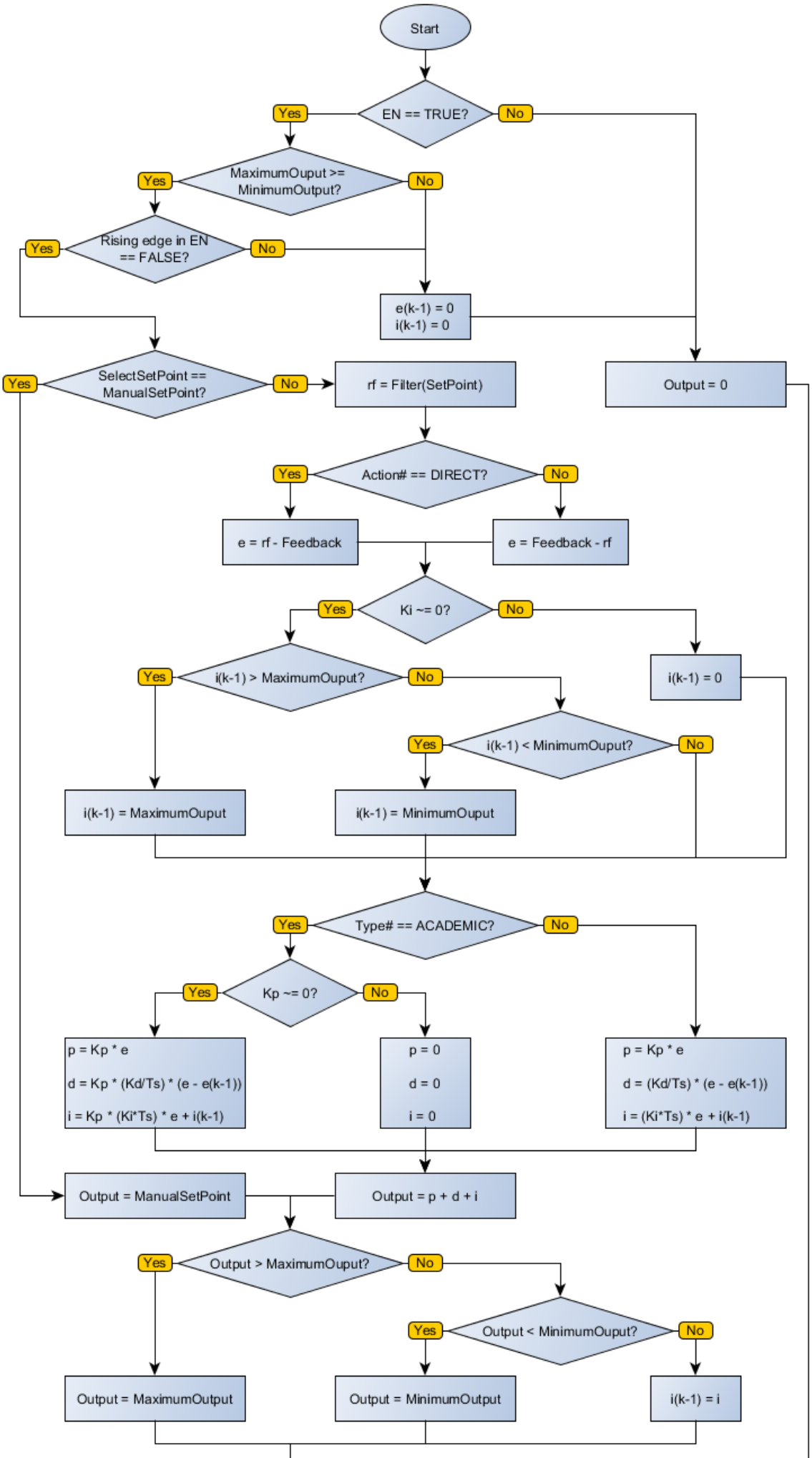
NOTE!
Effects of the alteration of gains on the process

- If Kp decreases, the process becomes slower; generally more stable or less oscillating; it has less overshoot.
- If Kp increases, the process responds faster; it may become more unstable or more oscillating; it has more overshoot.
- If Ki decreases, the process becomes slower, lagging to reach the "SetPoint"; it becomes more stable or less oscillating; it has less overshoot.
- If Ki increases, the process becomes faster, quickly reaching the "SetPoint"; it becomes more unstable or more oscillating; it has more overshoot.
- If Kd decreases, the process becomes slower; it has less overshoot.
- If Kd increases, it has more overshoot.

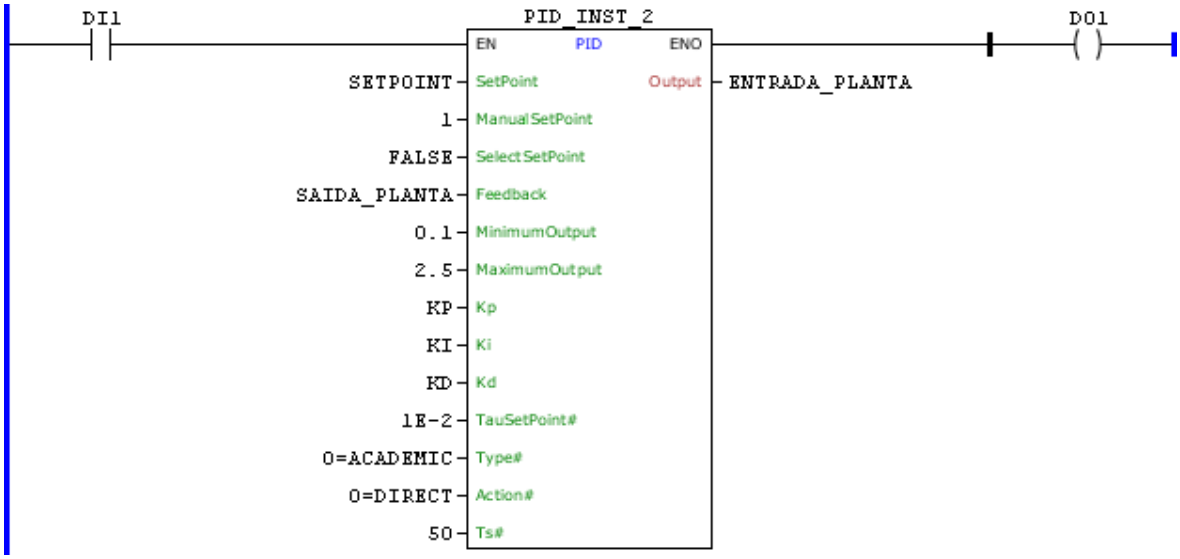
NOTE!
How to improve the performance of the process through the adjustment of gains (valid for the Academic PID)

- If the performance of the process is almost good, but the overshoot is a bit high, try to: (1) decrease Kp 20%, (2) decrease Ki 20% and/or (3) decrease Kd 50%.
- If the performance of the process is almost good, but it does not have overshoot and lags to reach the "SetPoint", try to: (1) increase Kp 20%, (2) increase Ki 20% and/or (3) increase Kd 50%.
- If the performance of the process is good, but the process output is varying too much, try to: (1) increase Kd 50%, (2) decrease Kp 20%.
- If the performance of the process is bad, i.e. after start up, the transitory lasts several periods of oscillation that reduce very slowly or never reduce at all, try to: (1) decrease Kp 50%.
- If the performance of the process is bad, i.e. after start up it slowly moves towards the "SetPoint" without overshoot, but is still very far and the process output is less than the rated value, try to: (1) increase Kp 50%, (2) increase Ki 50%, (3) increase Kd 70%.

Block Flowchart


Start
EN == TRUE?
Yes
No
MaximumOuput >= MinimumOutput?
Yes
No
Rising edge in EN == FALSE?
Yes
No
e(k-1) = 0
i(k-1) = 0
SelectSetPoint == ManualSetPoint?
Yes
No
rf = Filter(SetPoint)
Output = 0
Action# == DIRECT?
Yes
No
e = rf - Feedback
e = Feedback - rf
Ki ~= 0?
Yes
No
i(k-1) > MaximumOuput?
Yes
No
i(k-1) = 0
i(k-1) < MinimumOuput?
Yes
No
i(k-1) = MaximumOuput
i(k-1) = MinimumOuput
Type# == ACADEMIC?
Yes
No
Kp ~= 0?
Yes
No
p = Kp * e
d = Kp * (Kd/Ts) * (e - e(k-1))
i = Kp * (Ki*Ts) * e + i(k-1)
p = 0
d = 0
i = 0
p = Kp * e
d = (Kd/Ts) * (e - e(k-1))
i = (Ki*Ts) * e + i(k-1)
Output = ManualSetPoint
Output = p + d + i
Output > MaximumOuput?
Yes
No
Output < MinimumOuput?
Yes
No
Output = MaximumOutput
Output = MinimumOutput
i(k-1) = i

Example



The above example creates a loop of a digital PID form with sampling time 50 ms, using the constants KP, KI and KD for control. Automatic reference SETPOINT, filtered by a first order filter with time constant of 0:01 is used. The error signal is calculated as the difference between the filtered reference and variable SAIDA_PLANTA. The controller output is saturated between the values 0.1 and 2.5 and sent to the variable ENTRADA_PLANTA.

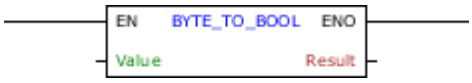
11.1.6.6 Conversion

11.1.6.6.1 BOOL

11.1.6.6.1.1 BYTE_TO_BOOL

Block that performs the conversion of a BYTE value into a BOOL value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT	Value in BYTE
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BOOL	Value in BOOL

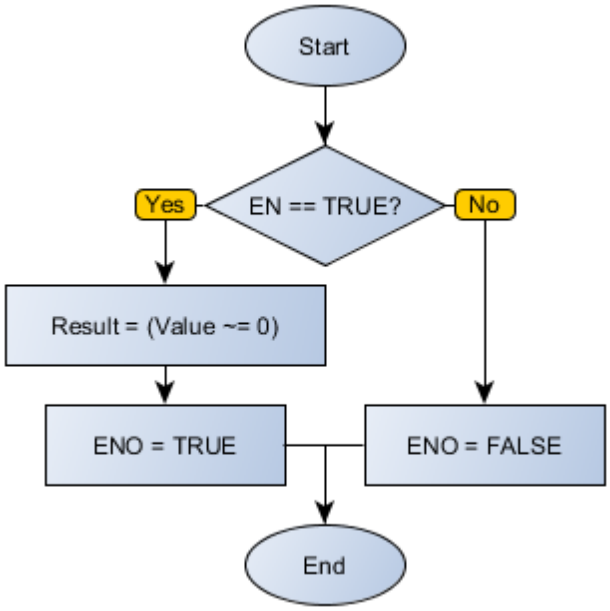
Operation

When this block has a TRUE value in EN, it interprets the Value value as BYTE and converts it into BOOL, storing in Result.

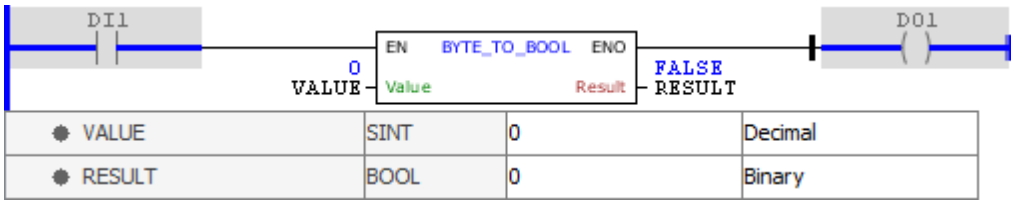
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

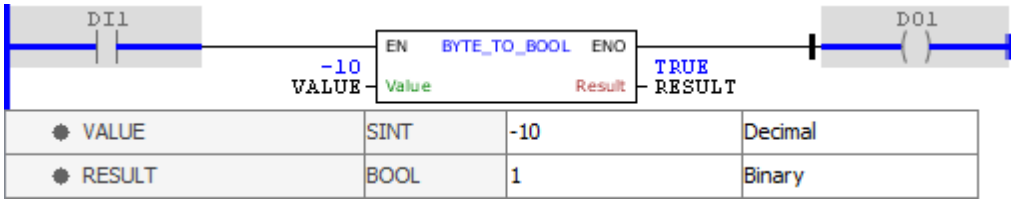
Block Flowchart



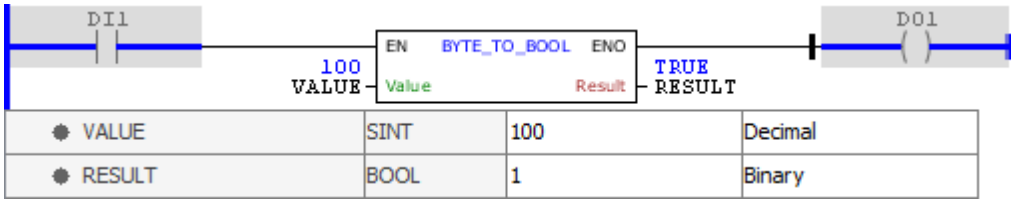
Example



VALUE	SINT	0	Decimal
RESULT	BOOL	0	Binary



VALUE	SINT	-10	Decimal
RESULT	BOOL	1	Binary



VALUE	SINT	100	Decimal
RESULT	BOOL	1	Binary

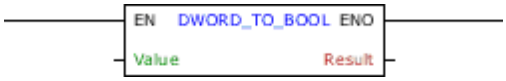
The examples above perform the conversion of VALUE variable, in BYTE, into a BOOL value storing

the final result in RESULT. The block ends with success and ENO output is activated.

11.1.6.6.1.2 DWORD_TO_BOOL

Block that performs the conversion of a DWORD value into a BOOL value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	DWORD UDINT DINT	Value in DWORD
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BOOL	Value in BOOL

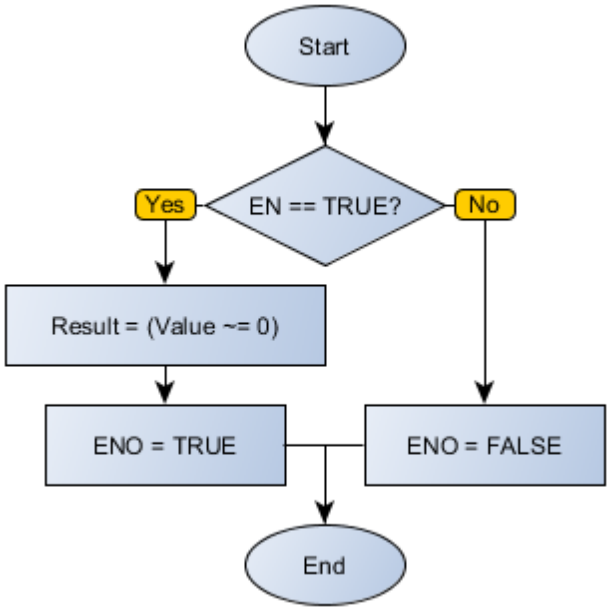
Operation

When this block has a TRUE value in EN, it interprets the Value value as DWORD and converts it into BOOL, storing in Result.

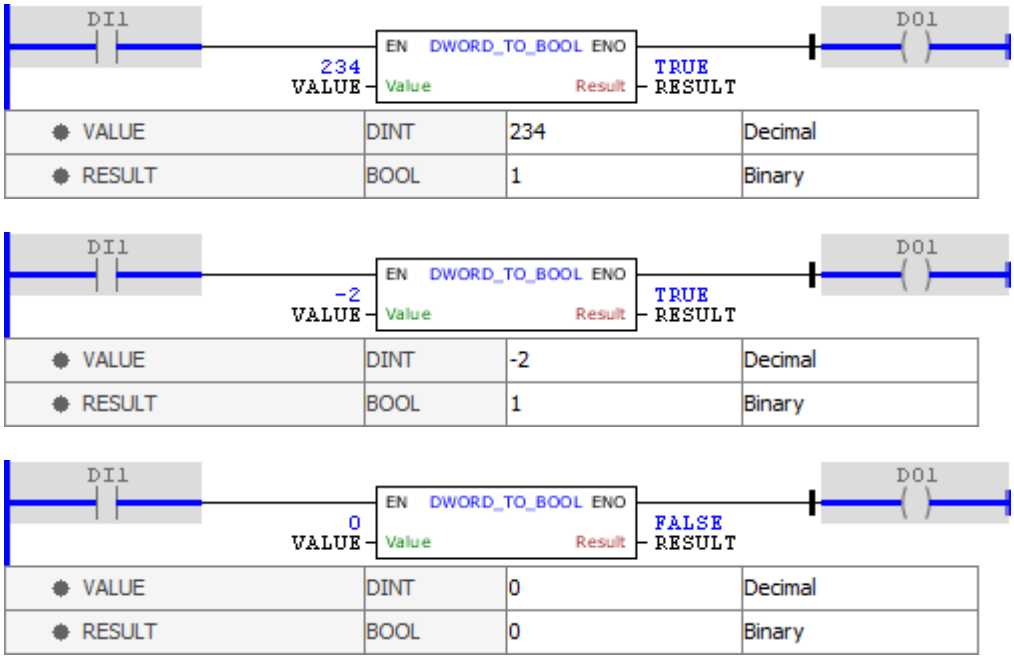
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example

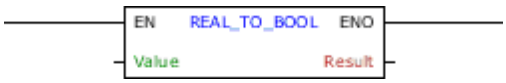


The examples above perform the conversion of VALUE variable, in DWORD, into a BOOL value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.1.6.6.1.3 REAL_TO_BOOL

Block that performs the conversion of a REAL value into a BOOL value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Value in REAL
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BOOL	Value in BOOL

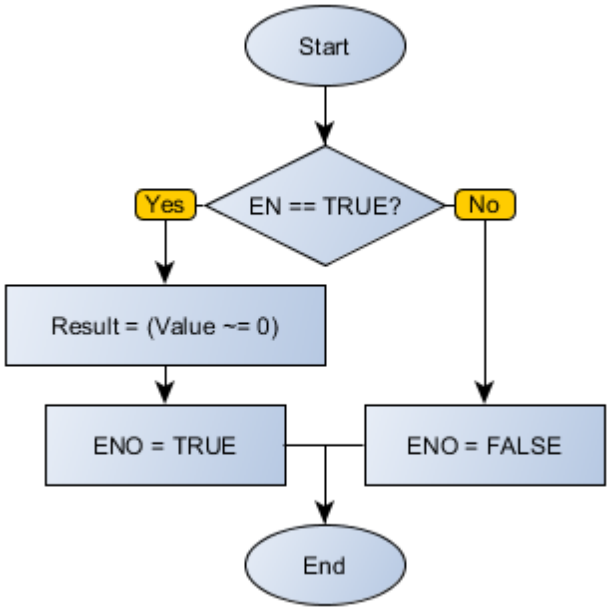
Operation

When this block has a TRUE value in EN, it interprets the Value value as REAL and converts it into BOOL, storing in Result.

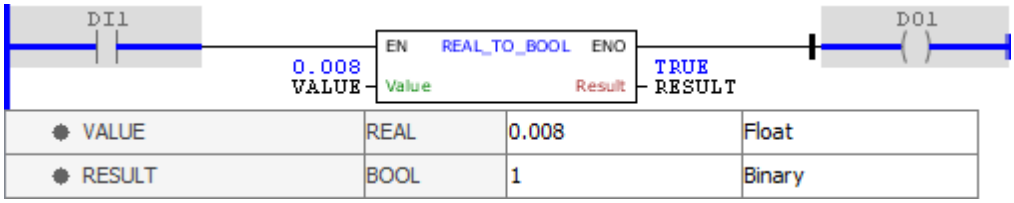
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

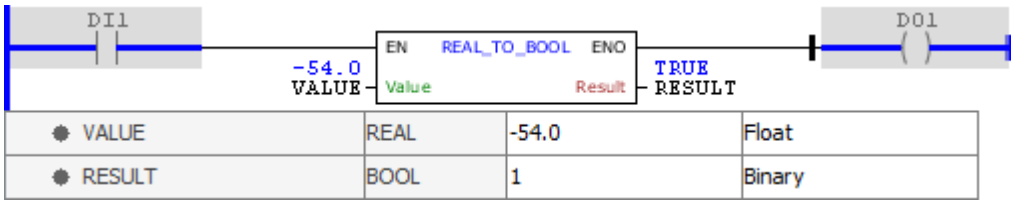
Block Flowchart



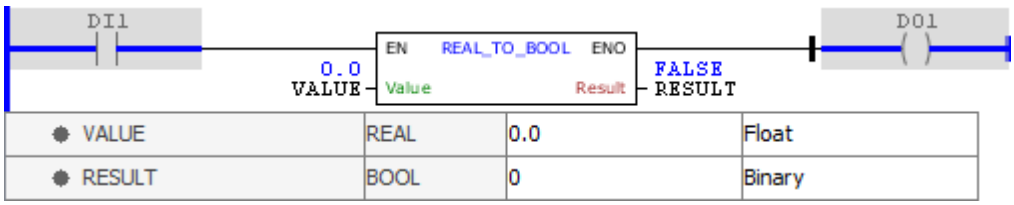
Example



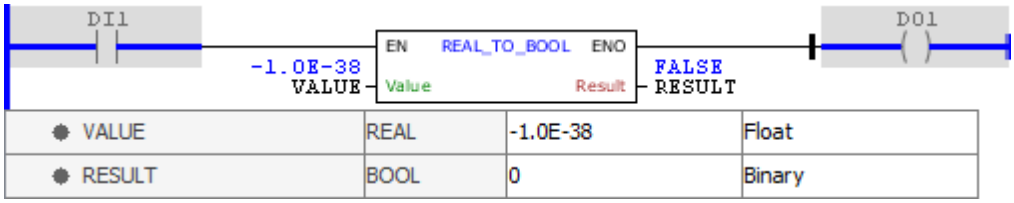
VALUE	REAL	0.008	Float
RESULT	BOOL	1	Binary



VALUE	REAL	-54.0	Float
RESULT	BOOL	1	Binary



VALUE	REAL	0.0	Float
RESULT	BOOL	0	Binary



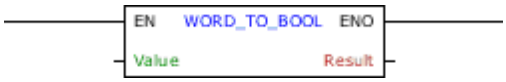
VALUE	REAL	-1.0E-38	Float
RESULT	BOOL	0	Binary

The examples above perform the conversion of VALUE variable, in REAL, into a BOOL value storing the final result in RESULT. The block ends with success and ENO output is activated. Notice in the last example that the values very close to the machine epsilon may result in an interpretation of the FALSE value.

11.1.6.6.1.4 WORD_TO_BOOL

Block that performs the conversion of a WORD value into a BOOL value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	WORD UINT INT	Value in WORD
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BOOL	Value in BOOL

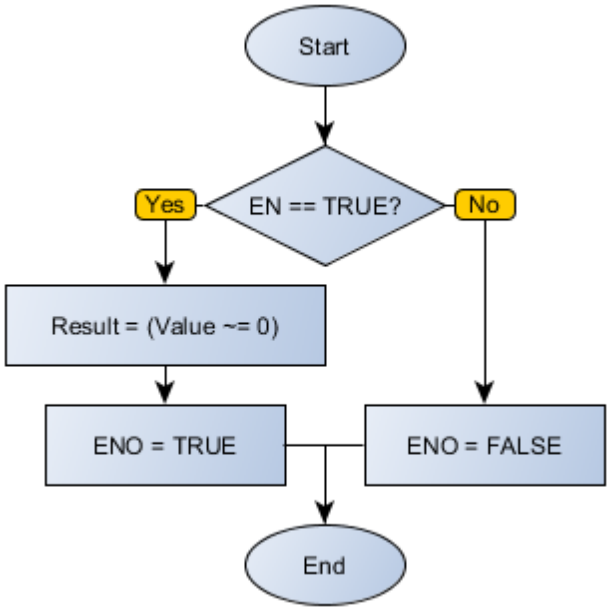
Operation

When this block has a TRUE value in EN, it interprets the Value value as WORD and converts it into BOOL, storing in Result.

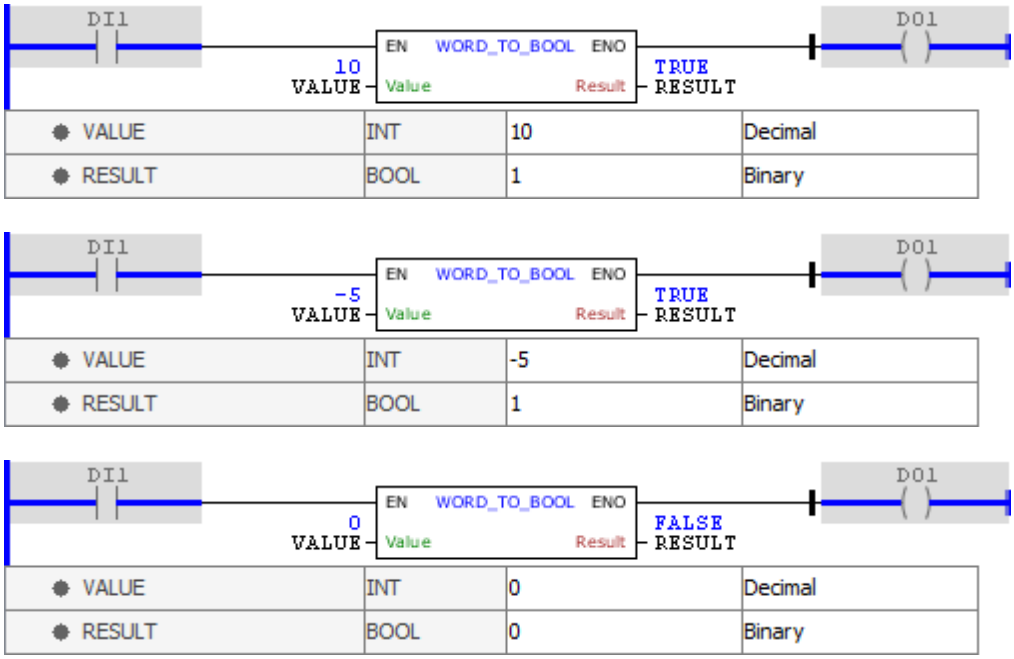
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



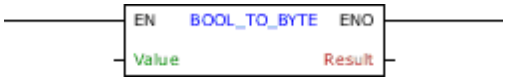
The examples above perform the conversion of VALUE variable, in WORD, into a BOOL value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.1.6.6.2 BYTE

11.1.6.6.2.1 BOOL_TO_BYTE

Block that performs the conversion of a BOOL value into a BYTE value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BOOL	Value in BOOL
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT	Value in BYTE

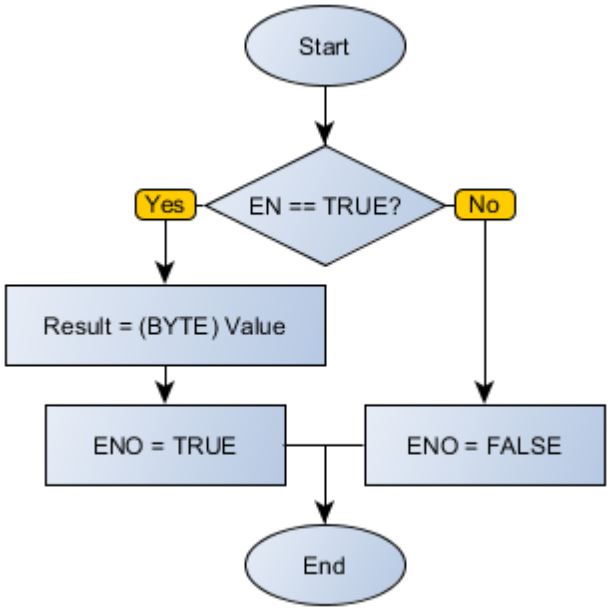
Operation

When this block has a TRUE value in EN, it interprets the Value value as BOOL and converts it into BYTE, storing in Result.

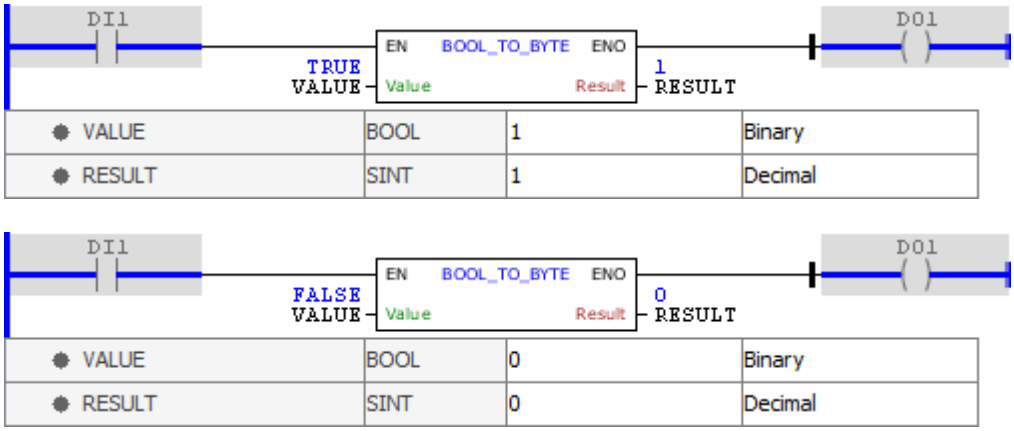
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



VALUE	BOOL	1	Binary
RESULT	SINT	1	Decimal

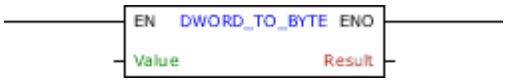
VALUE	BOOL	0	Binary
RESULT	SINT	0	Decimal

The examples above perform the conversion of variable VALUE, in BOOL, into a BYTE value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.1.6.6.2.2 DWORD_TO_BYTE

Block that performs the conversion of a DWORD value into a BYTE value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	DWORD UDINT DINT	Value in DWORD
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT	Value in BYTE

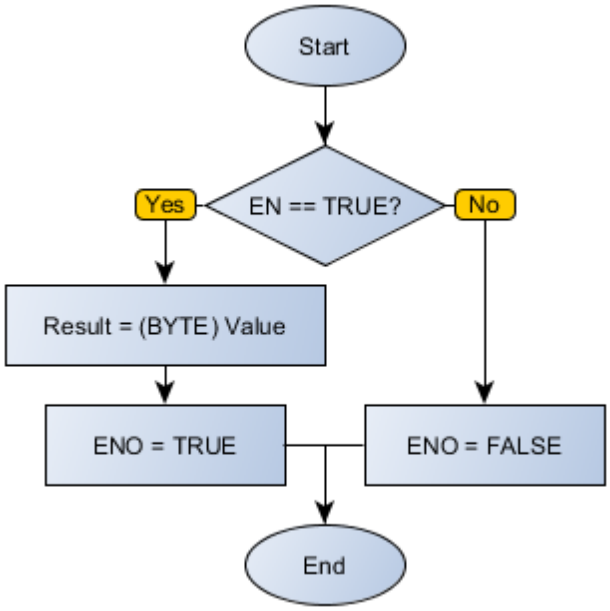
Operation

When this block has a TRUE value in EN, it interprets the Value value as DWORD and converts it into BYTE, storing in Result.

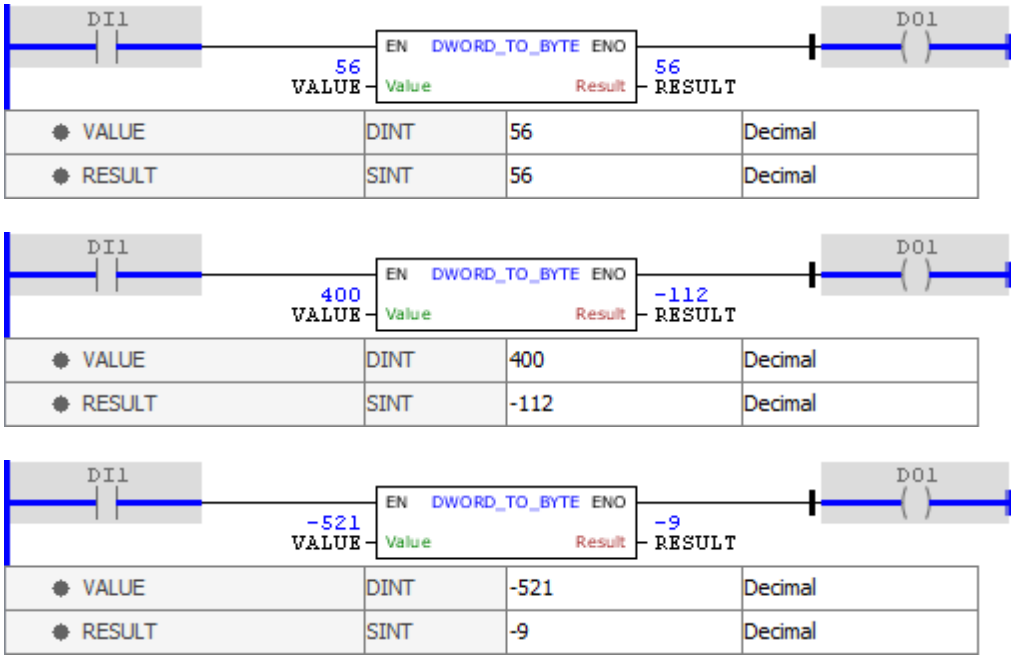
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example

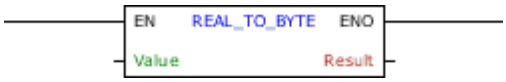


The examples above perform the conversion of variable VALUE, in DWORD, into a BYTE value storing the final result in RESULT. The block ends with success and ENO output is activated. Notice that only the eight least significant bits are taken into account.

11.1.6.6.2.3 REAL_TO_BYTE

Block that performs the conversion of a REAL value into a BYTE value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Value in REAL
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT	Value in BYTE

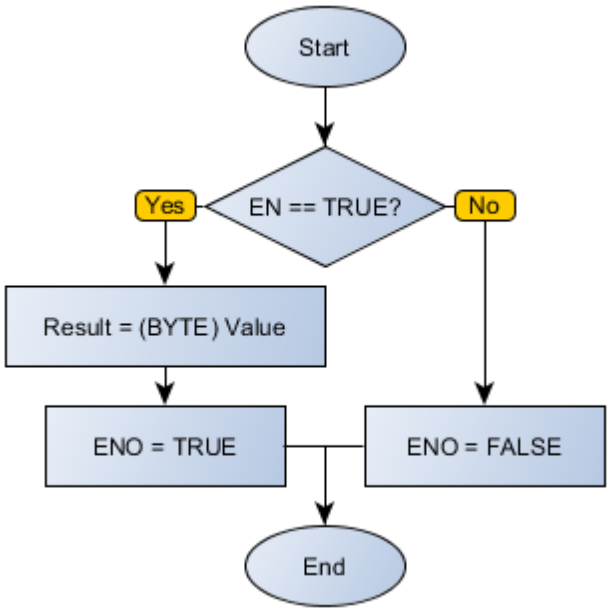
Operation

When this block has a TRUE value in EN, it interprets the Value value as REAL and converts it into BYTE, storing in Result.

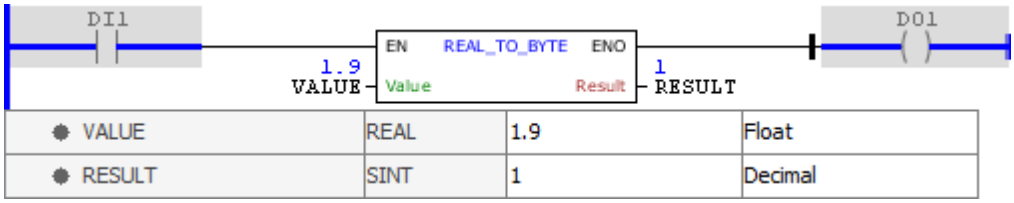
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

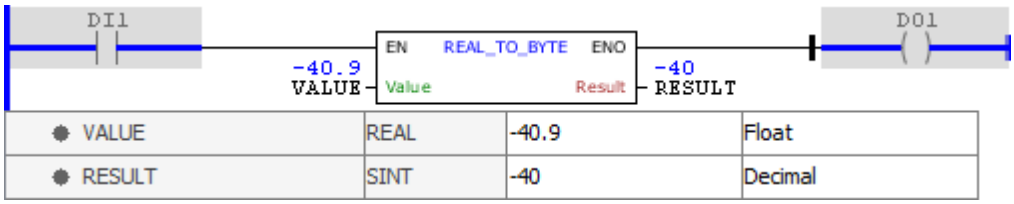
Block Flowchart



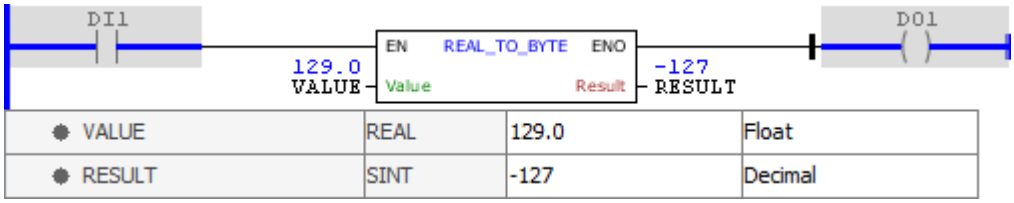
Example



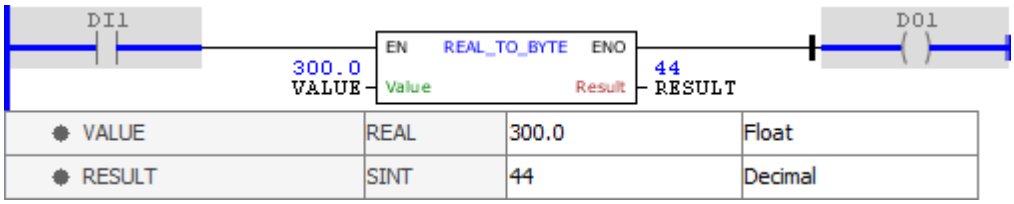
VALUE	REAL	1.9	Float
RESULT	SINT	1	Decimal



VALUE	REAL	-40.9	Float
RESULT	SINT	-40	Decimal



VALUE	REAL	129.0	Float
RESULT	SINT	-127	Decimal



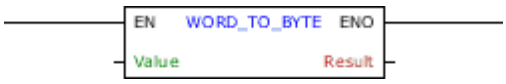
VALUE	REAL	300.0	Float
RESULT	SINT	44	Decimal

The examples above perform the conversion of variable VALUE, in REAL, into a BYTE value storing the final result in RESULT. The block ends with success and ENO output is activated. Notice that the results are truncated in decimal and only the eight least significant bits are taken into account.

11.1.6.6.2.4 WORD_TO_BYTE

Block that performs the conversion of a WORD value into a BYTE value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	WORD UINT INT	Value in WORD
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT	Value in BYTE

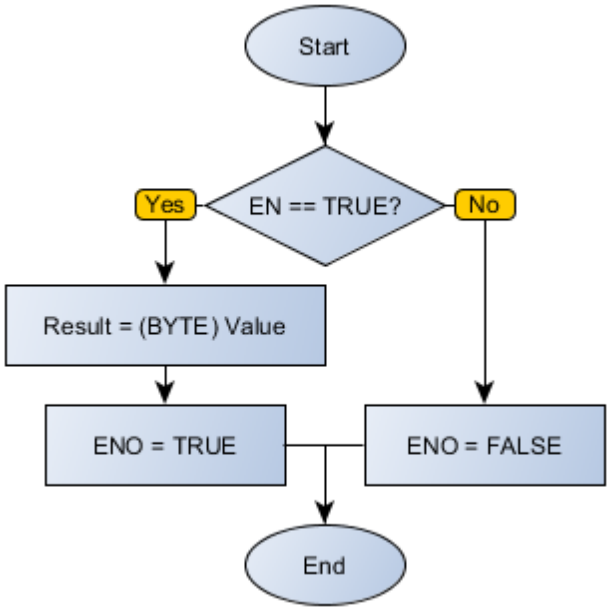
Operation

When this block has a TRUE value in EN, it interprets the Value value as WORD and converts it into BYTE, storing in Result.

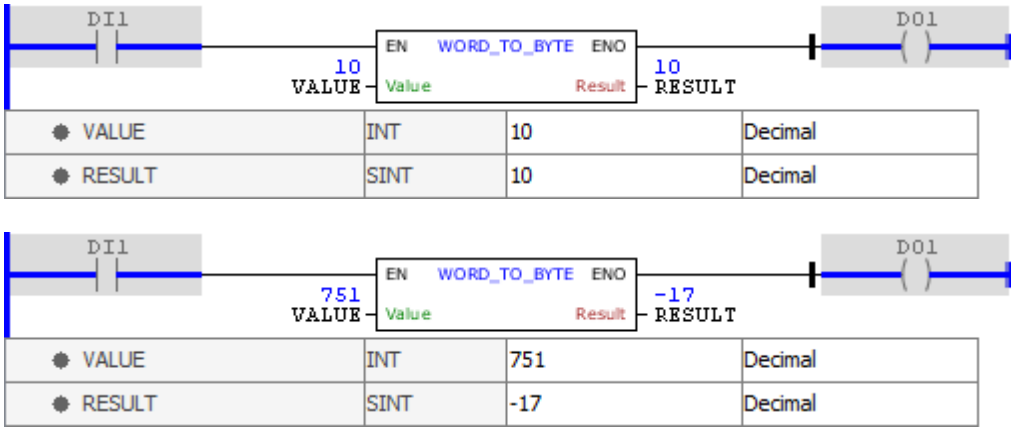
When EN has FALSE value, Result remains unchanged.

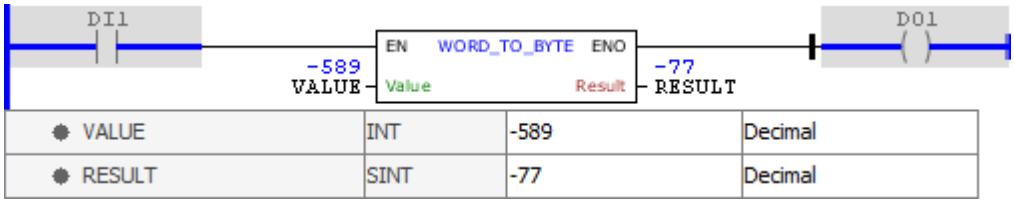
The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example





VALUE	INT	-589	Decimal
RESULT	SINT	-77	Decimal

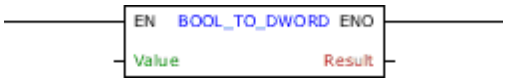
The examples above perform the conversion of variable VALUE, in WORD, into a BYTE value storing the final result in RESULT. The block ends with success and ENO output is activated. Notice that only the eight least significant bits are taken into account.

11.1.6.6.3 DWORD

11.1.6.6.3.1 BOOL_TO_DWORD

Block that performs the conversion of a BOOL value into a DWORD value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BOOL	Value in BOOL
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	DWORD UDINT DINT	Value in DWORD

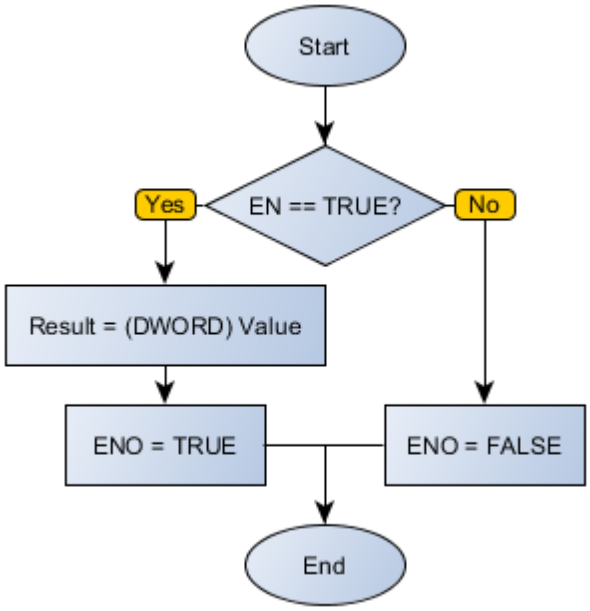
Operation

When this block has a TRUE value in EN, it interprets the Value value as BOOL and converts it into DWORD, storing in Result.

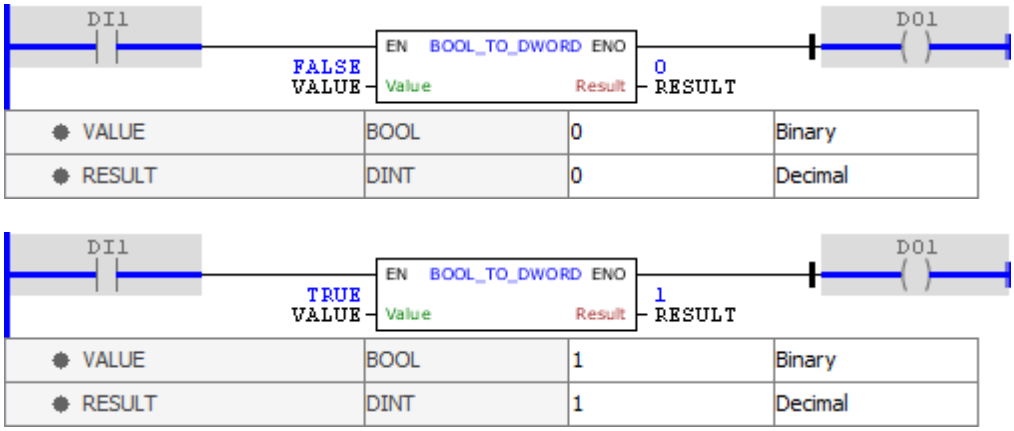
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example

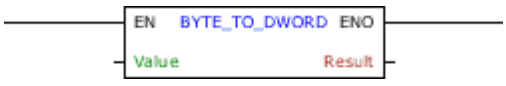


The examples above perform the conversion of VALUE variable, in BOOL, into a DWORD value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.1.6.6.3.2 BYTE_TO_DWORD

Block that performs the conversion of a BYTE value into a DWORD value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT	Value in BYTE
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	DWORD UDINT DINT	Value in DWORD

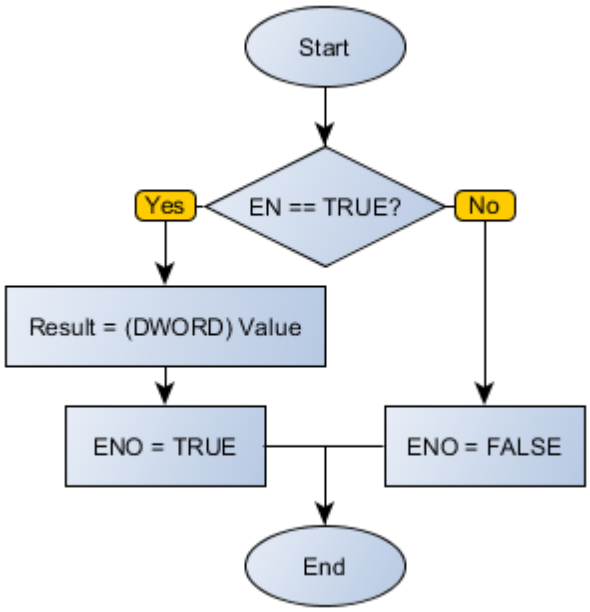
Operation

When this block has a TRUE value in EN, it interprets the Value value as BYTE and converts it into DWORD, storing in Result.

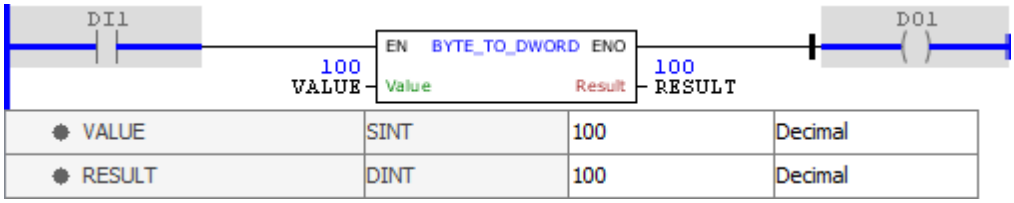
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

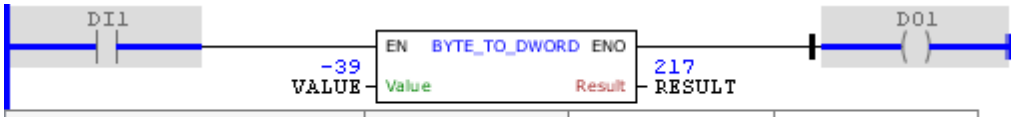
Block Flowchart



Example



VALUE	SINT	100	Decimal
RESULT	DINT	100	Decimal



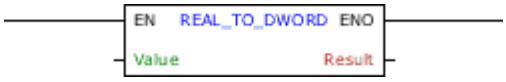
VALUE	SINT	-39	Decimal
RESULT	DINT	217	Decimal

The examples above perform the conversion of variable VALUE, in BYTE, into a DWORD value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.1.6.6.3.3 REAL_TO_DWORD

Block that performs the conversion of a REAL value into a DWORD value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Value in REAL
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	DWORD UDINT DINT	Value in DWORD

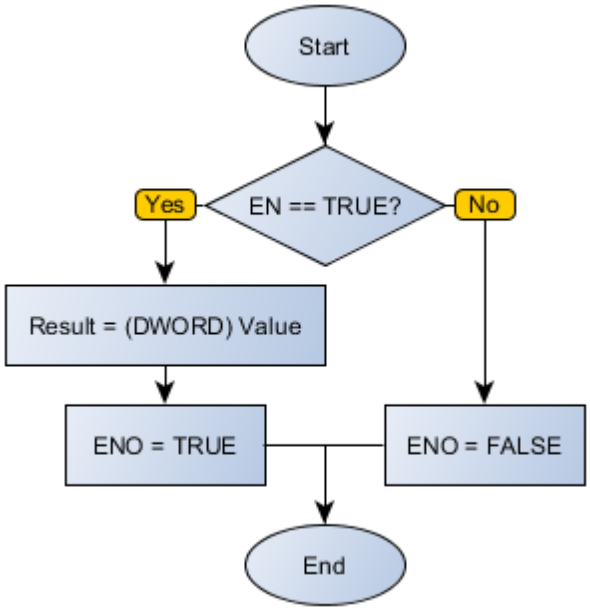
Operation

When this block has a TRUE value in EN, it interprets the Value value as REAL and converts it into DWORD, storing in Result.

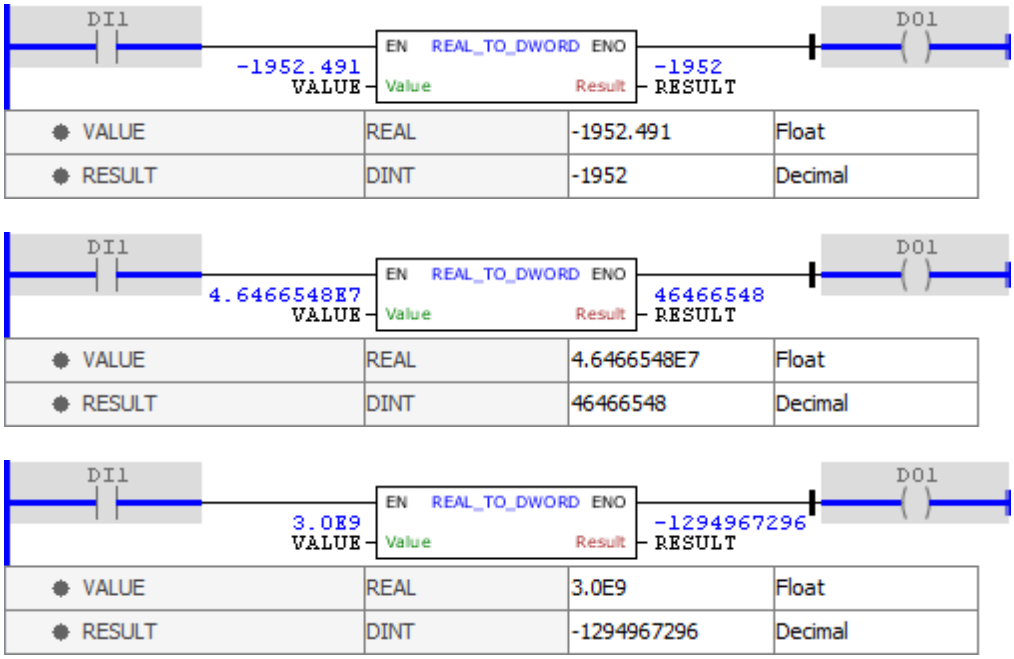
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example

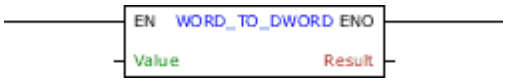


The examples above perform the conversion of variable VALUE, in REAL, into a DWORD value storing the final result in RESULT. The block ends with success and ENO output is activated. Note that the results are truncated in decimal and only the thirty-two least significant bits are taken into account.

11.1.6.6.3.4 WORD_TO_DWORD

Block that performs the conversion of a WORD value into a DWORD value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	WORD UINT INT	Value in WORD
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	DWORD UDINT DINT	Value in DWORD

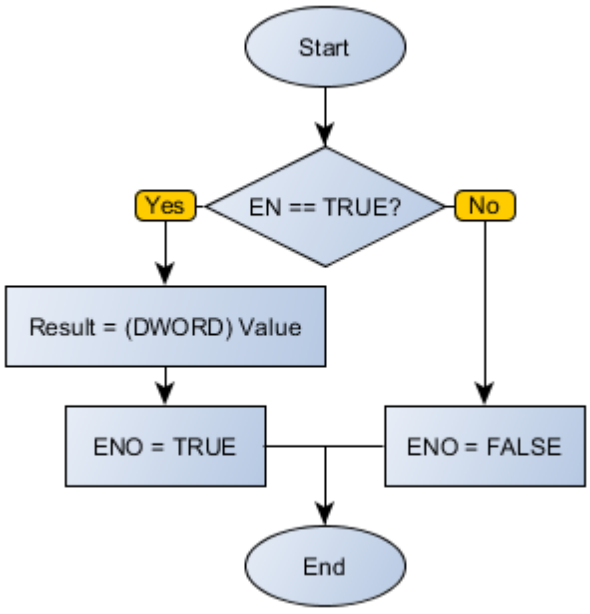
Operation

When this block has a TRUE value in EN, it interprets the Value value as WORD and converts it into DWORD, storing in Result.

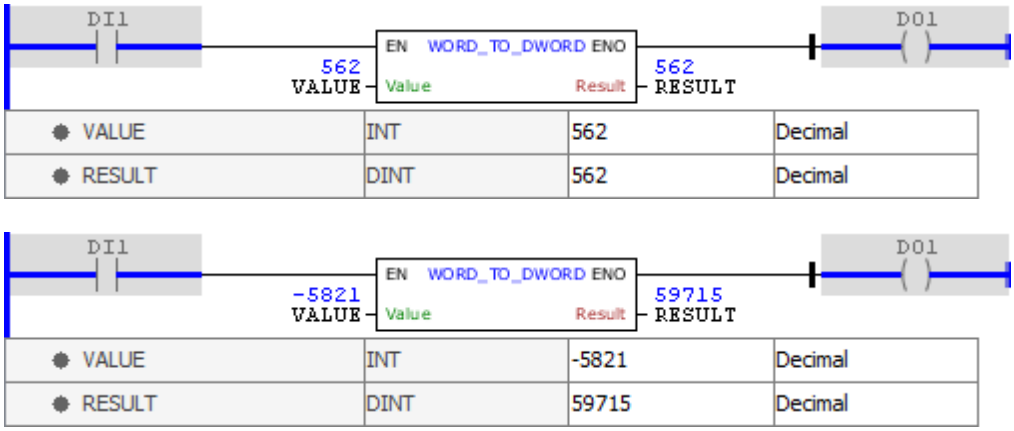
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



VALUE	INT	562	Decimal
RESULT	DINT	562	Decimal

VALUE	INT	-5821	Decimal
RESULT	DINT	59715	Decimal

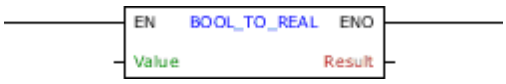
The examples above convert the VALUE variable, in WORD, into a DWORD value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.1.6.6.4 REAL

11.1.6.6.4.1 BOOL_TO_REAL

Block that performs the conversion of a BOOL value into a REAL value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BOOL	Value in BOOL
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	REAL	Value in REAL

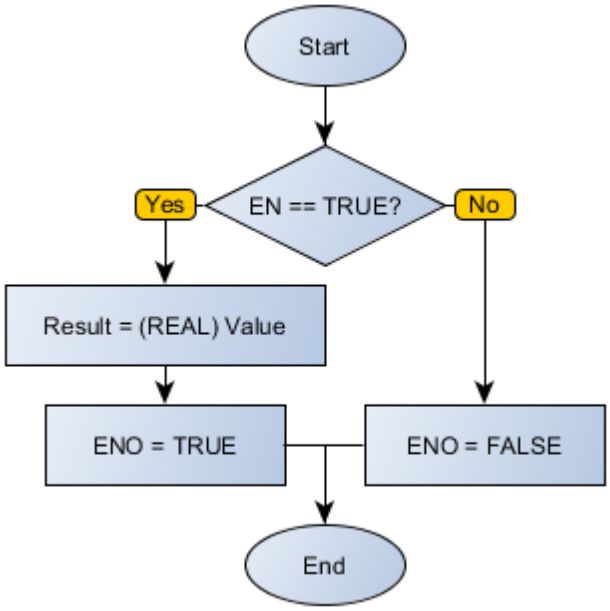
Operation

When this block has a TRUE value in EN, it interprets the Value value as BOOL and converts it into REAL, storing in Result.

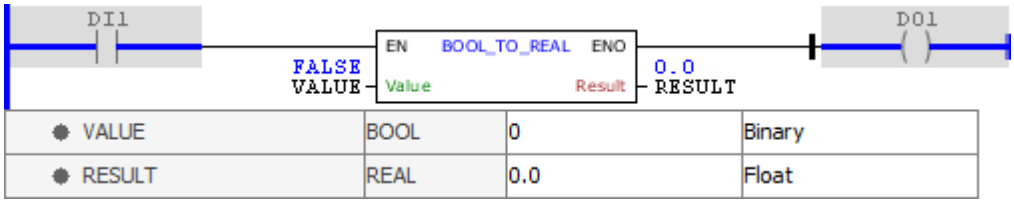
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

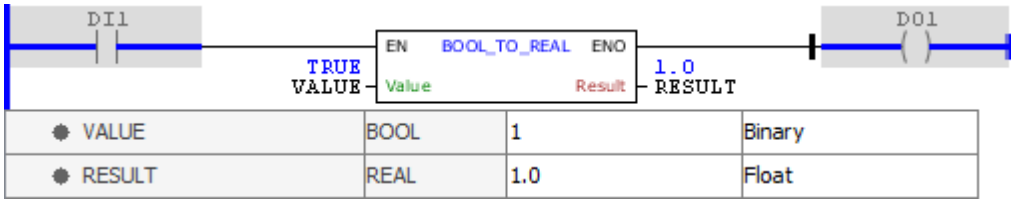
Block Flowchart



Example



VALUE	BOOL	0	Binary
RESULT	REAL	0.0	Float



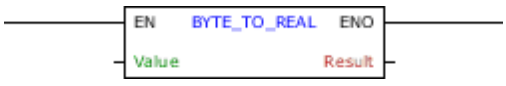
VALUE	BOOL	1	Binary
RESULT	REAL	1.0	Float

The examples above perform the conversion of variable VALUE, in BOOL, into a REAL value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.1.6.6.4.2 BYTE_TO_REAL

Block that performs the conversion of a BYTE value into a REAL value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT	Value in BYTE
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	REAL	Value in REAL

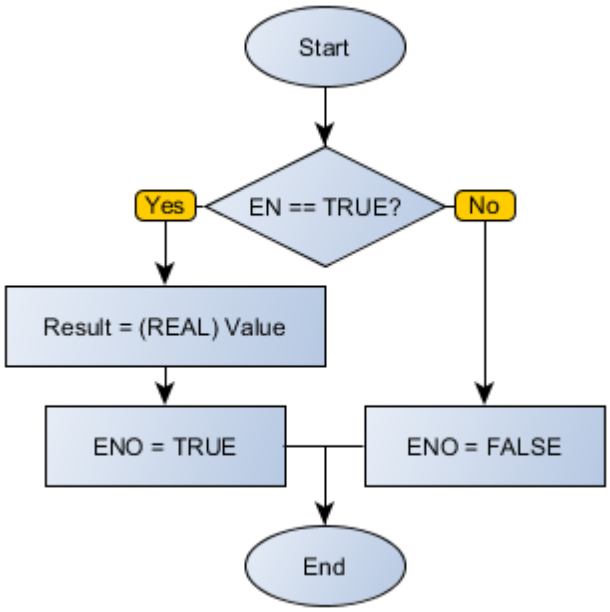
Operation

When this block has a TRUE value in EN, it interprets the Value value as BYTE and converts it into REAL, storing in Result.

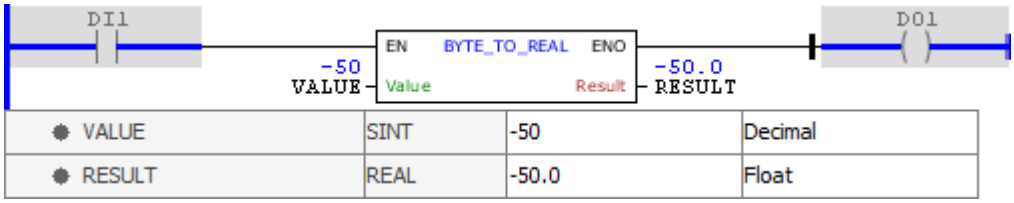
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

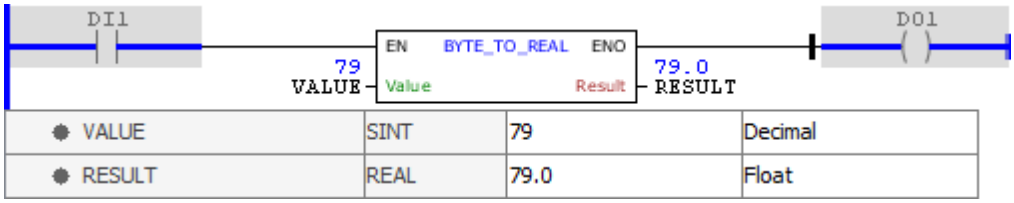
Block Flowchart



Example



VALUE	SINT	-50	Decimal
RESULT	REAL	-50.0	Float



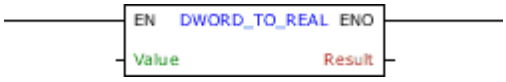
VALUE	SINT	79	Decimal
RESULT	REAL	79.0	Float

The examples above perform the conversion of variable VALUE, in BYTE, into a REAL value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.1.6.6.4.3 DWORD_TO_REAL

Block that performs the conversion of a DWORD value into a REAL value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	DWORD UDINT DINT	Value in DWORD
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	REAL	Value in REAL

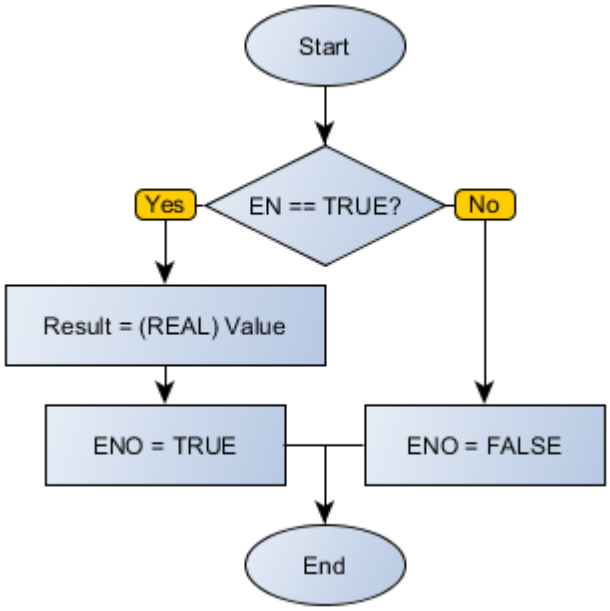
Operation

When this block has a TRUE value in EN, it interprets the Value value as DWORD and converts it into REAL, storing in Result.

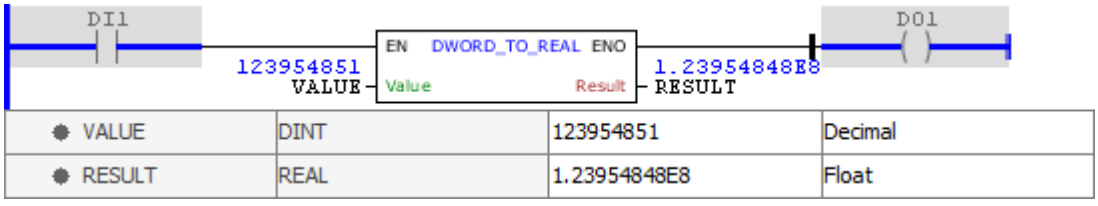
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

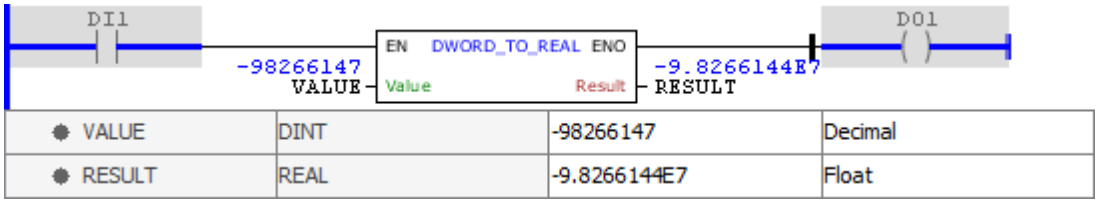
Block Flowchart



Example



VALUE	DINT	123954851	Decimal
RESULT	REAL	1.23954848E8	Float



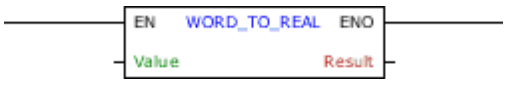
VALUE	DINT	-98266147	Decimal
RESULT	REAL	-9.8266144E7	Float

The examples above perform the conversion of variable VALUE, in DWORD, into a REAL value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.1.6.6.4.4 WORD_TO_REAL

Block that performs the conversion of a WORD value into a REAL value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	WORD UINT INT	Value in WORD
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	REAL	Value in REAL

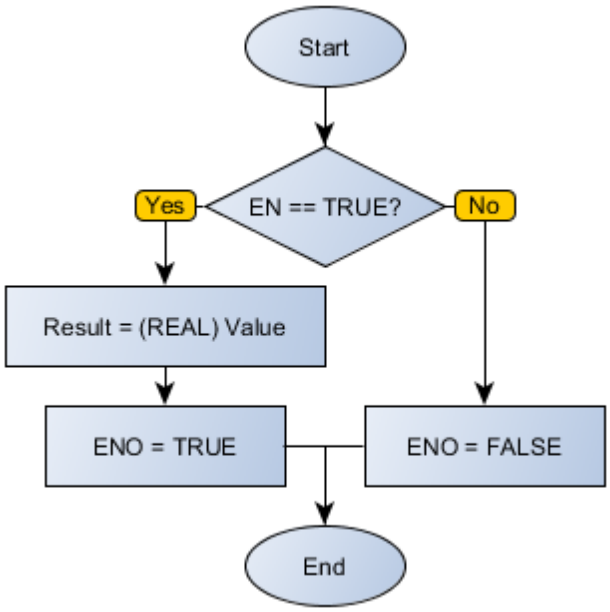
Operation

When this block has a TRUE value in EN, it interprets the Value value as WORD and converts it into REAL, storing in Result.

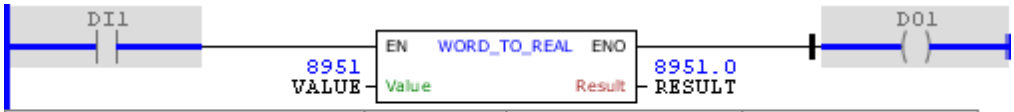
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

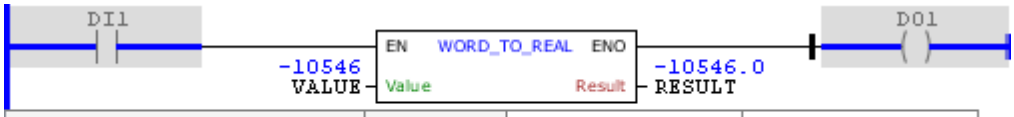
Block Flowchart



Example



VALUE	INT	8951	Decimal
RESULT	REAL	8951.0	Float



VALUE	INT	-10546	Decimal
RESULT	REAL	-10546.0	Float

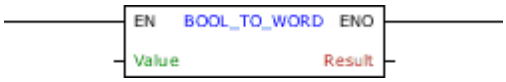
The examples above perform the conversion of variable VALUE, in WORD, into a REAL value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.1.6.6.5 WORD

11.1.6.6.5.1 BOOL_TO_WORD

Block that performs the conversion of a BOOL value into a WORD value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BOOL	Value in BOOL
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	WORD UINT INT	Value in WORD

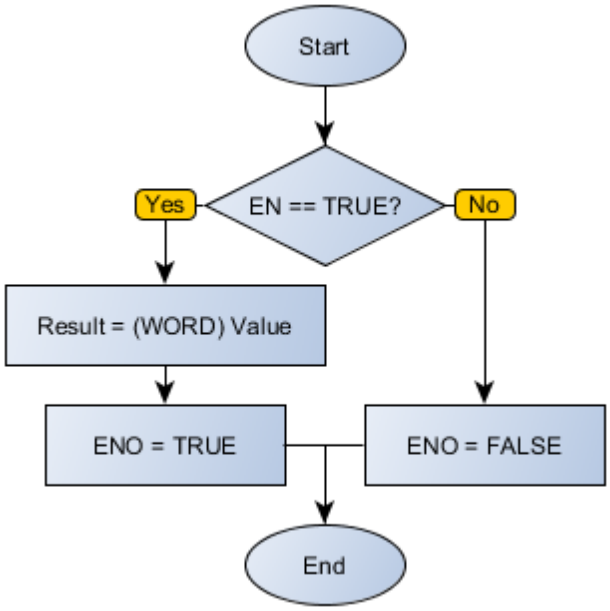
Operation

When this block has a TRUE value in EN, it interprets the Value value as BOOL and converts it into WORD, storing in Result.

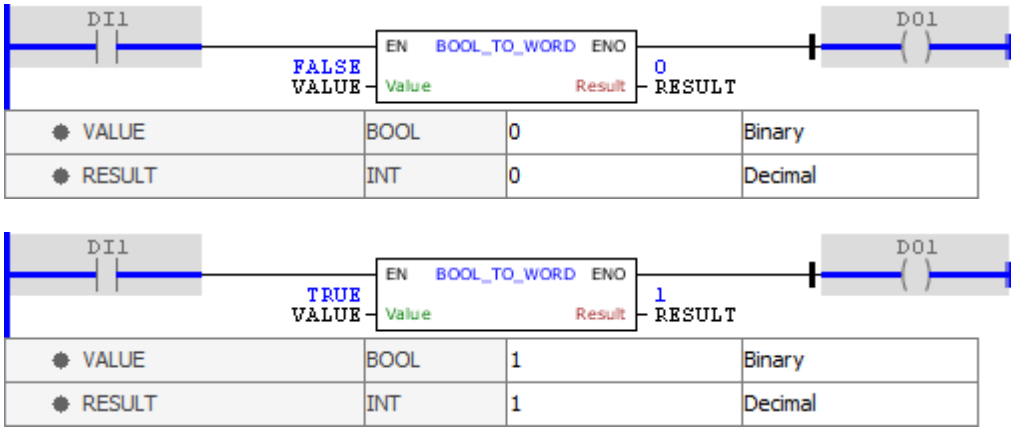
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example

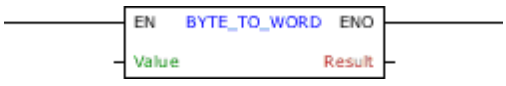


The examples above perform the conversion of VALUE variable, in BOOL, into a WORD value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.1.6.6.5.2 BYTE_TO_WORD

Block that performs the conversion of a BYTE value into a WORD value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT	Value in BYTE
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	WORD UINT INT	Value in WORD

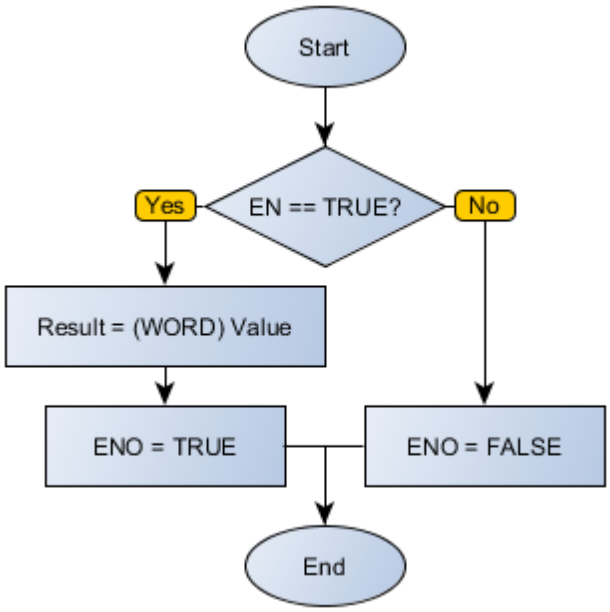
Operation

When this block has a TRUE value in EN, it interprets the Value value as BYTE and converts it into WORD, storing in Result.

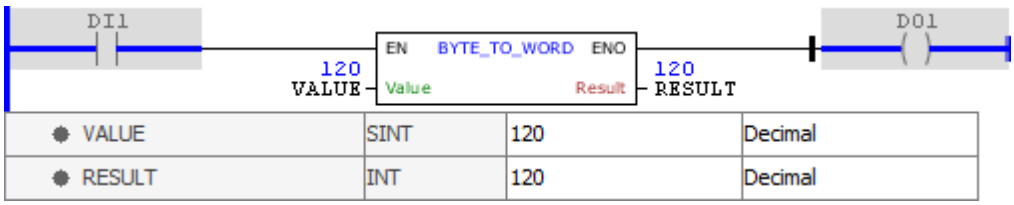
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

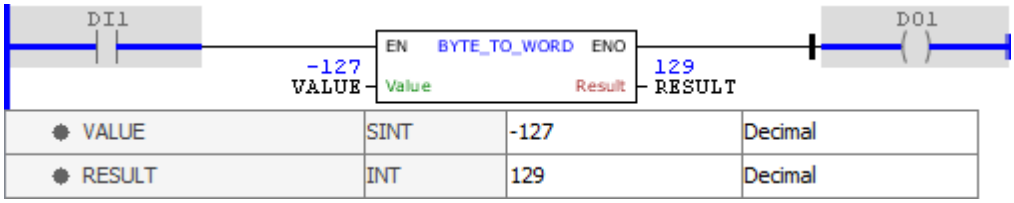
Block Flowchart



Example



VALUE	SINT	120	Decimal
RESULT	INT	120	Decimal



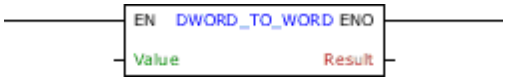
VALUE	SINT	-127	Decimal
RESULT	INT	129	Decimal

The examples above perform the conversion of variable VALUE, in BYTE, into a WORD value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.1.6.6.5.3 DWORD_TO_WORD

Block that performs the conversion of a DWORD value into a WORD value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	DWORD UDINT DINT	Value in DWORD
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	WORD UINT INT	Value in WORD

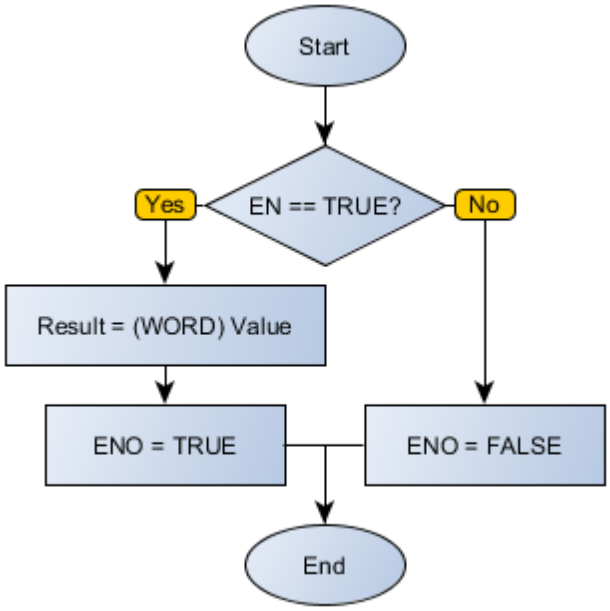
Operation

When this block has a TRUE value in EN, it interprets the Value value as DWORD and converts it into WORD, storing in Result.

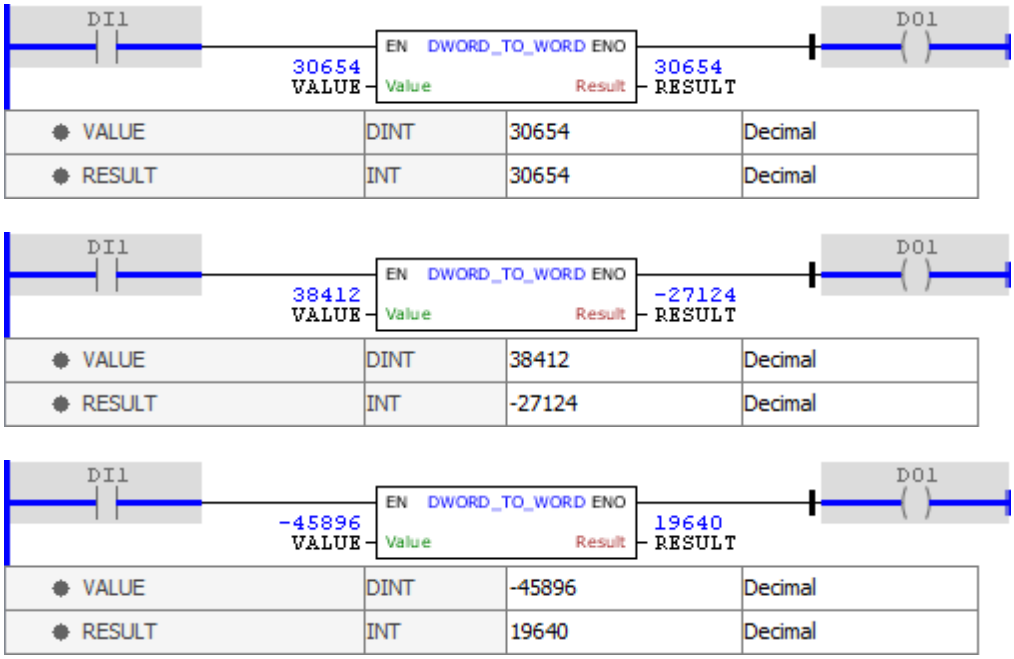
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example

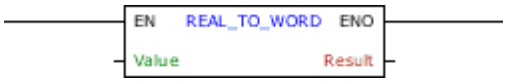


The examples above convert the VALUE variable, in DWORD, into a WORD value storing the final result in RESULT. The block ends with success and ENO output is activated. Notice that only the sixteen least significant bits are taken into account.

11.1.6.6.5.4 REAL_TO_WORD

Block that performs the conversion of a REAL value into a WORD value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Value in REAL
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	WORD UINT INT	Value in WORD

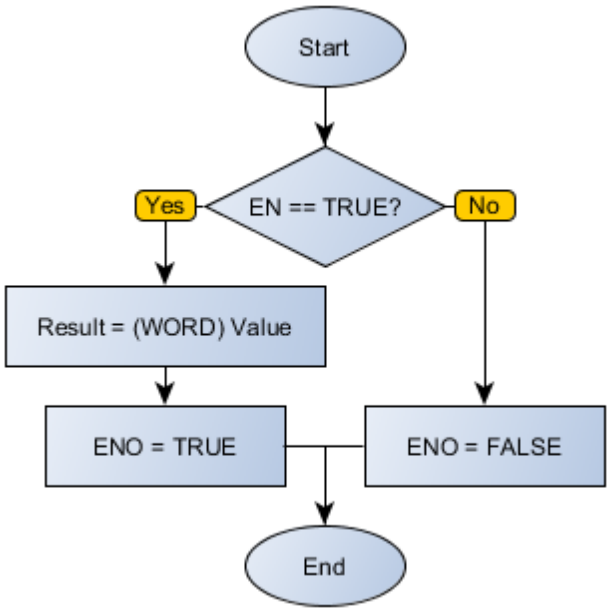
Operation

When this block has a TRUE value in EN, it interprets the Value value as REAL and converts it into WORD, storing in Result.

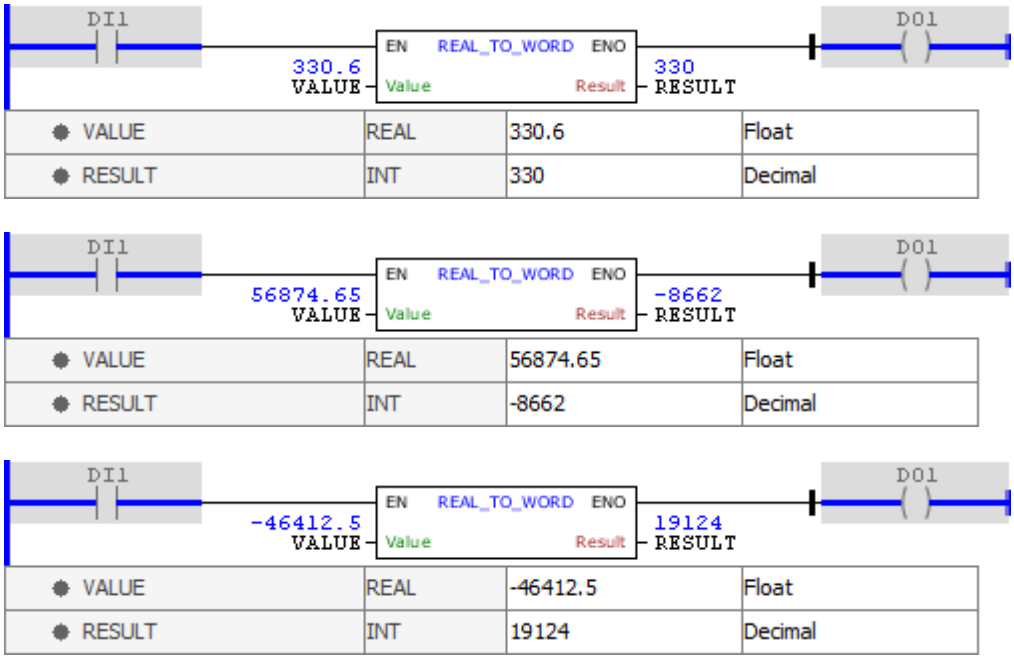
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



VALUE	REAL	330.6	Float
RESULT	INT	330	Decimal

VALUE	REAL	56874.65	Float
RESULT	INT	-8662	Decimal

VALUE	REAL	-46412.5	Float
RESULT	INT	19124	Decimal

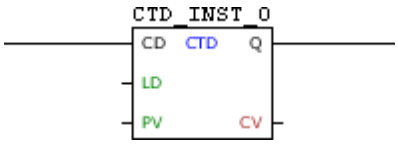
The examples above convert the VALUE variable, in DWORD, into a WORD value storing the final result in RESULT. The block ends with success and ENO output is activated. Note that the results are truncated in decimal and only the sixteen least significant bits are taken into account.

11.1.6.7 Counter

11.1.6.7.1 CTD

Countdown block of input pulses.

Ladder Representation



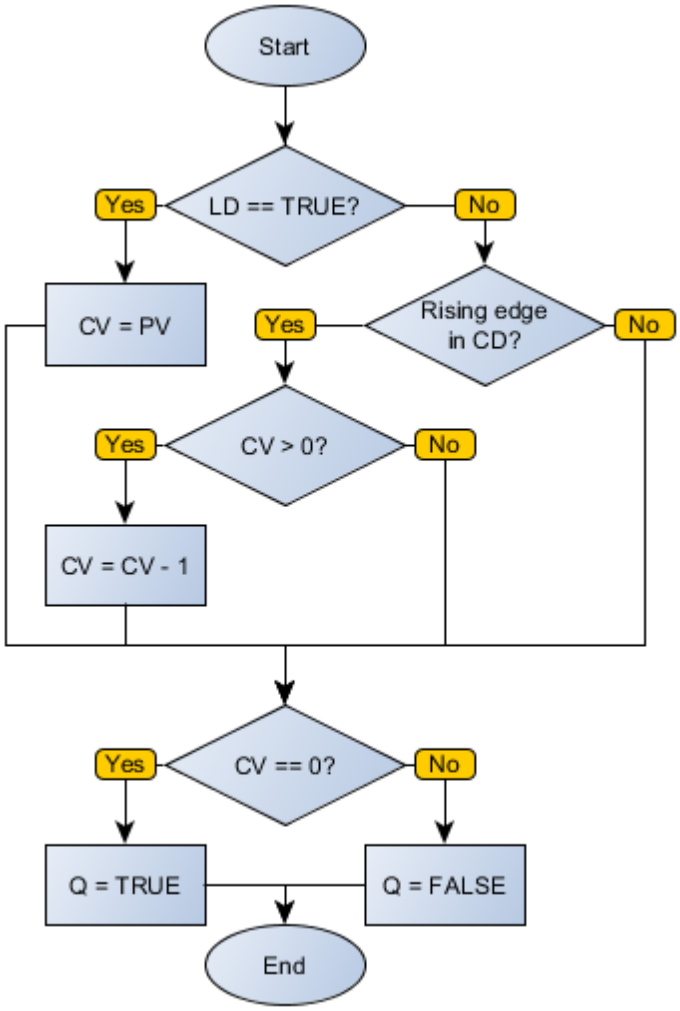
Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	CD	BOOL	Pulse identifier
	LD	BOOL	Loads the value of PV in CV
	PV	WORD UINT	Value of initial configuration
VAR_OUTPUT	Q	BOOL	Counter zeroed flag
	CV	WORD UINT	Current count value
VAR	CTD_INST_0	CTD	Instance of access to block structure

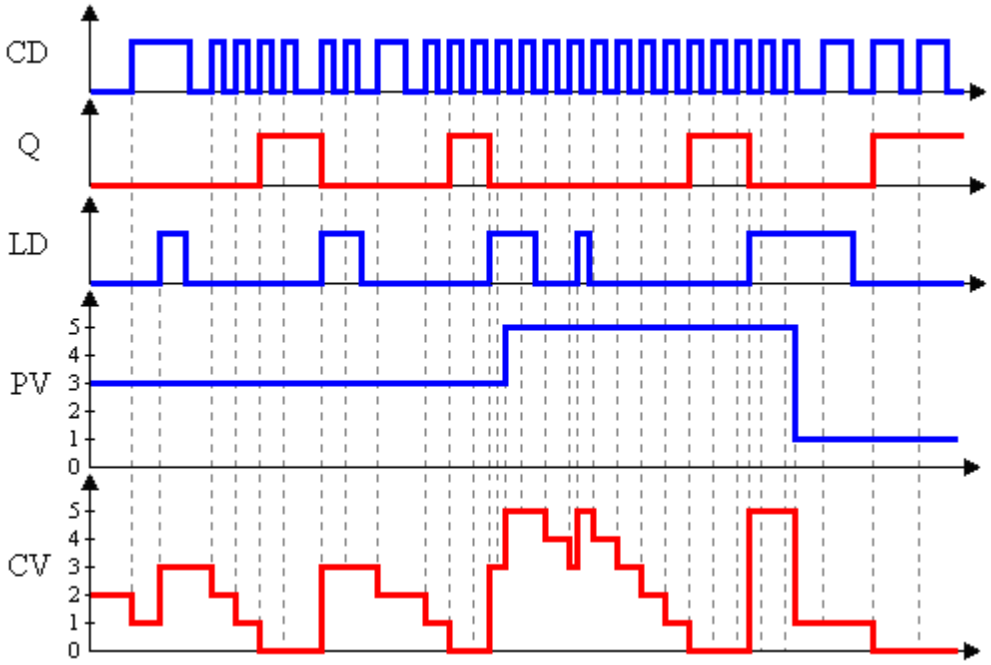
Operation

When this block identifies a leading edge in CD, it decrements the CV variable until it is zero. While CV equals zero, the output Q remains at TRUE level. By detecting high-level LD, the block loads the PV value in CV.

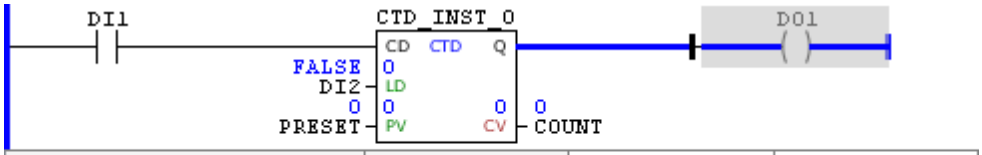
Block Flowchart



Operation Diagram

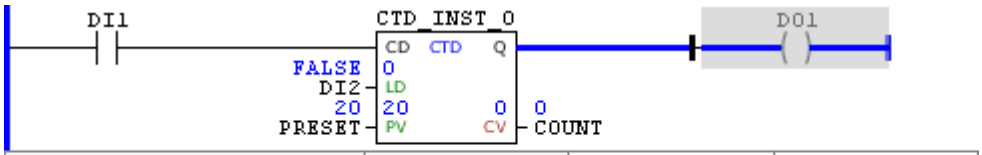


Example



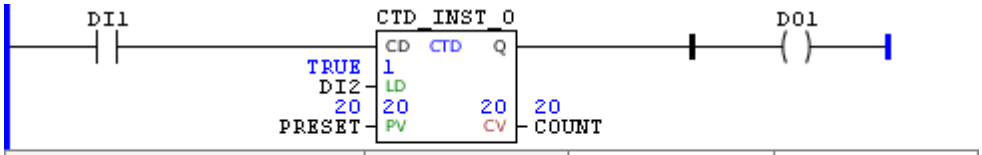
PRESET	UINT	0	Decimal
COUNT	UINT	0	Decimal

The above example shows the initial conditions of routine. As CV has a value of zero, the Q output is enabled.



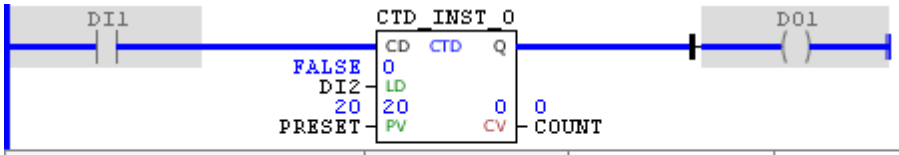
PRESET	UINT	20	Decimal
COUNT	UINT	0	Decimal

The value of the PV variable was changed to 20, but not yet loaded.



PRESET	UINT	20	Decimal
COUNT	UINT	20	Decimal

By identifying TRUE level in LD, the block loads the PV value to CV. Since this value is greater than zero, the Q output is disabled.



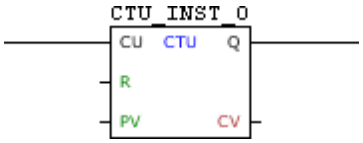
PRESET	UINT	20	Decimal
COUNT	UINT	0	Decimal

At each leading edge identified in CD, the value of COUNT is decremented until it reaches zero, when the Q output is enabled.

11.1.6.7.2 CTU

Block for gradual count of input pulses.

Ladder Representation



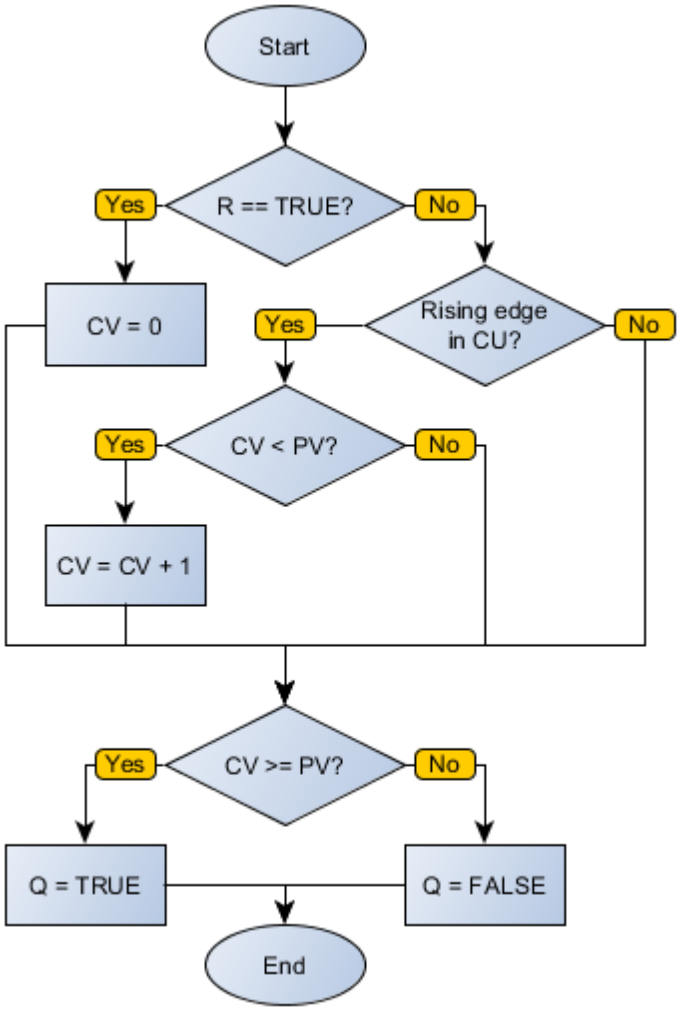
Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	CU	BOOL	Pulse identifier
	R	BOOL	Loads the zero value in CV
	PV	WORD UINT	Maximum count value
VAR_OUTPUT	Q	BOOL	Counter overrun flag
	CV	WORD UINT	Current count value
VAR	CTU_INST_0	CTU	Instance of access to block structure

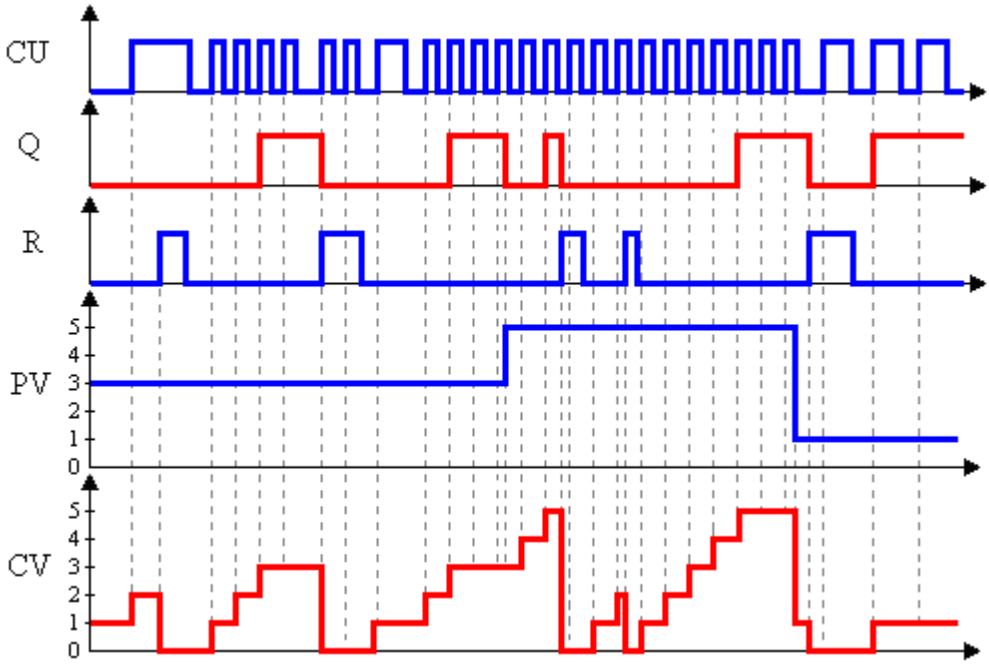
Operation

When this block identifies a leading edge in CD, it increments the CV variable until it is equal to PV. While CV equals PV, the output Q remains at TRUE level. By detecting high-level R, the block loads the zero value in CV.

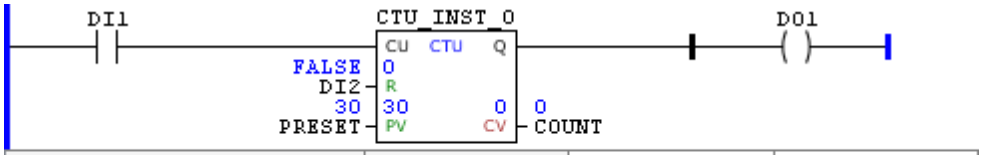
Block Flowchart



Operation Diagram

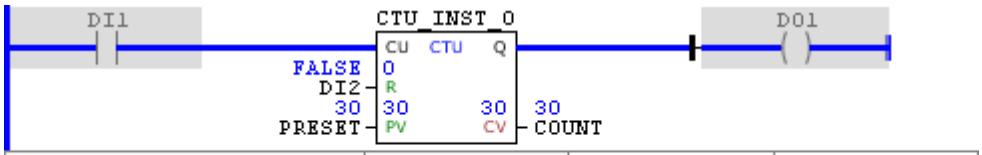


Example



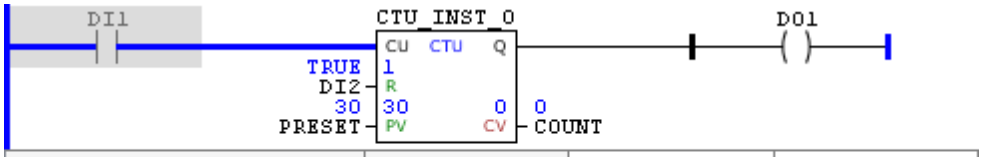
PRESET	UINT	30	Decimal
COUNT	UINT	0	Decimal

The above example shows the initial conditions of routine. Since CV has a lower value than of PV, the Q output is disabled.



PRESET	UINT	30	Decimal
COUNT	UINT	30	Decimal

At each leading edge identified in CU, the value of CV is incremented until it reaches the PV value, when the Q output is enabled.



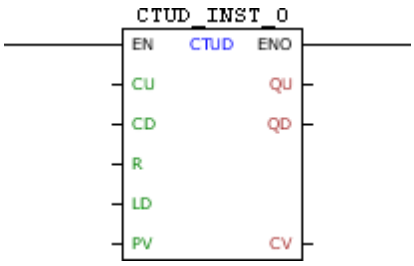
PRESET	UINT	30	Decimal
COUNT	UINT	0	Decimal

By identifying TRUE level in R, the block loads the zero value to CV. Since this value is lower than of PV, the Q output is disabled.

11.1.6.7.3 CTUD

Block for gradual count and countdown of input pulses.

Ladder Representation



Block Structure

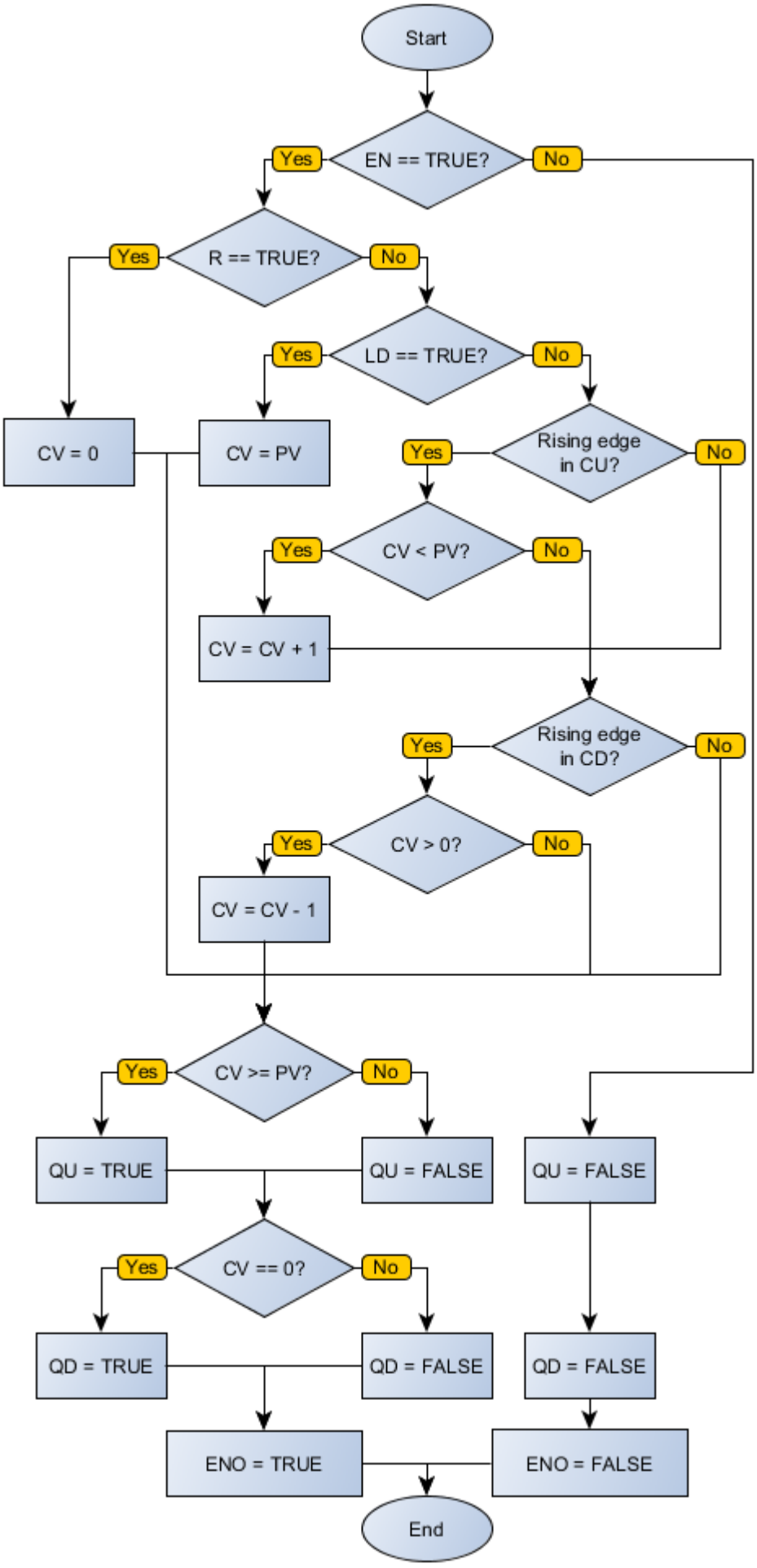
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	CU	BOOL	Pulse identifier for incremental
	CD	BOOL	Pulse identifier for decremental
	R	BOOL	Loads the zero value in CV
	LD	BOOL	Loads the value of PV in CV
	PV	WORD UINT	Reference value
VAR_OUTPUT	ENO	BOOL	Output enabling
	QU	BOOL	Counter overrun flag
	QD	BOOL	Counter zeroed flag
	CV	WORD UINT	Current count value
VAR	CTUD_INST_0	CTUD	Instance of access to block structure

Operation

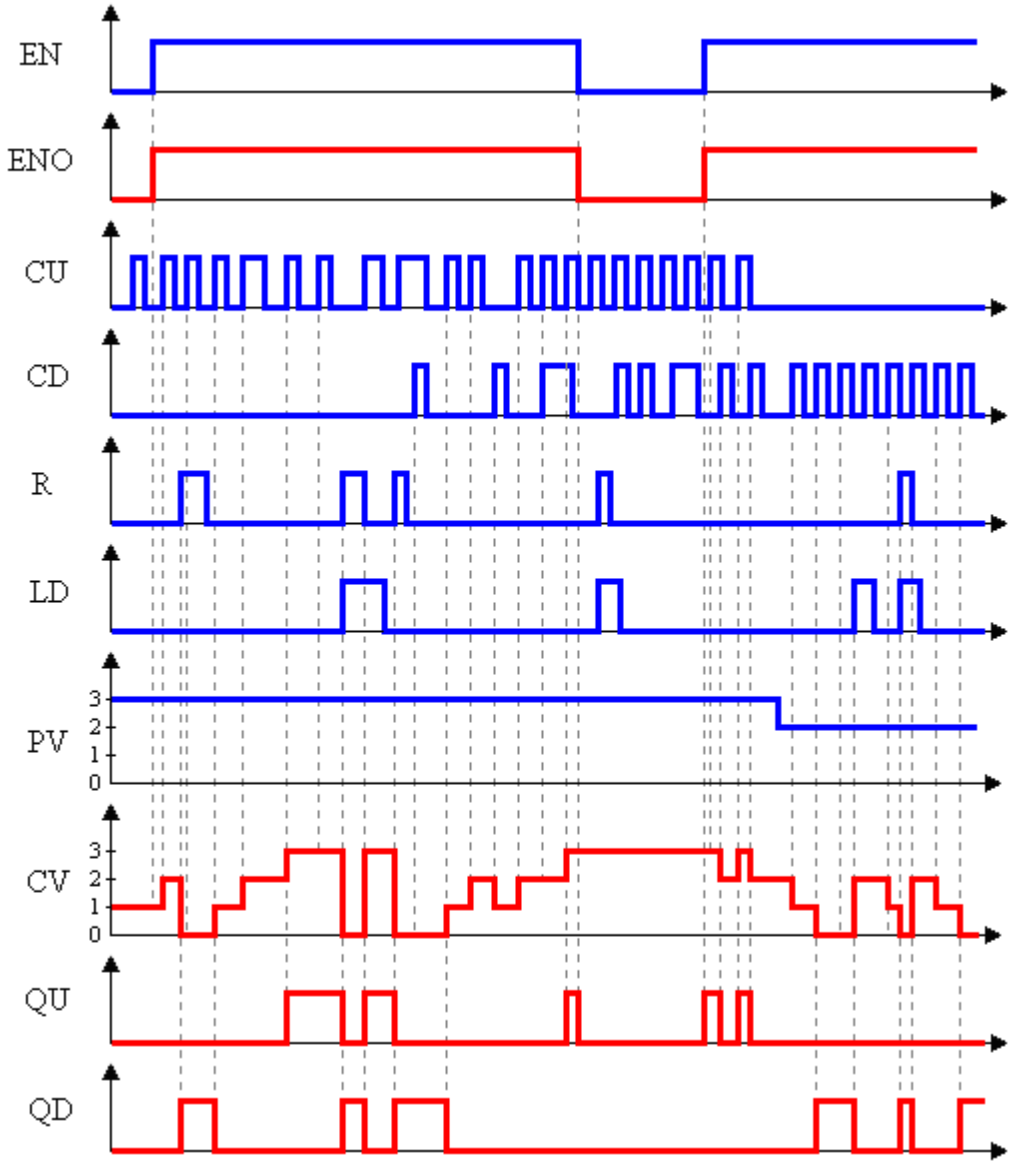
When this block has a TRUE value in EN, it acts as a CTD block and block CTU at the same time acting on the same CV counter. That is: increments CV until it reaches PV to the leading edges in CU and decrements CV until it reaches zero to the leading edges in CD. A positive transition in R carries zero in CV, while a leading edge in LD loads the PV value in CV. If CV has zero value, QD receives TRUE, and if CV has value equal to PV, QU receives TRUE.

The ENO value forwards to the next Ladder block the EN value.

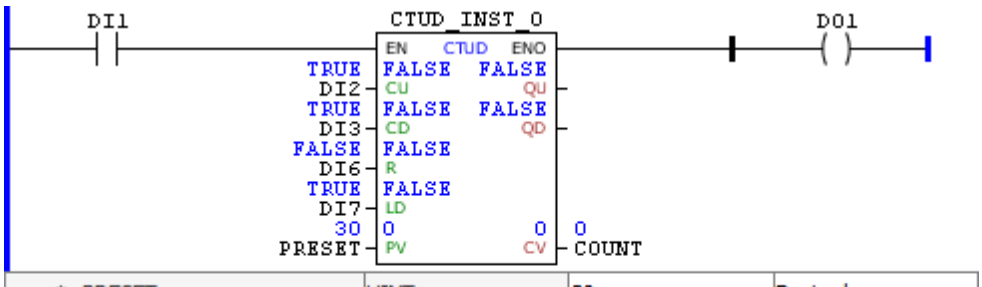
Block Flowchart


Start
EN == TRUE?
Yes
No
R == TRUE?
Yes
No
LD == TRUE?
Yes
No
CV = 0
CV = PV
Rising edge in CU?
Yes
No
CV < PV?
Yes
No
CV = CV + 1
Rising edge in CD?
Yes
No
CV > 0?
Yes
No
CV = CV - 1
CV >= PV?
Yes
No
QU = TRUE
QU = FALSE
QU = FALSE
CV == 0?
Yes
No
QD = TRUE
QD = FALSE
QD = FALSE
ENO = TRUE
ENO = FALSE
End

Operation Diagram

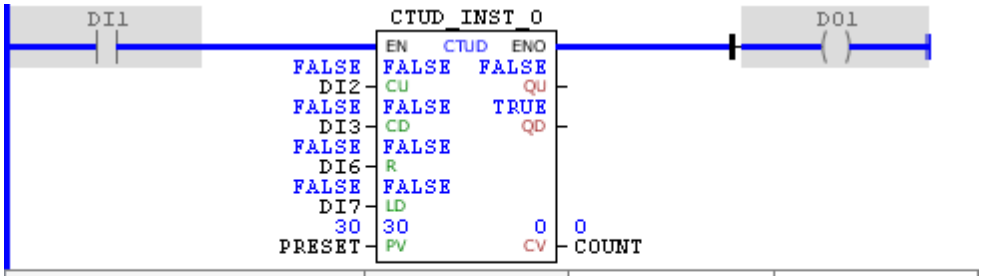


Example



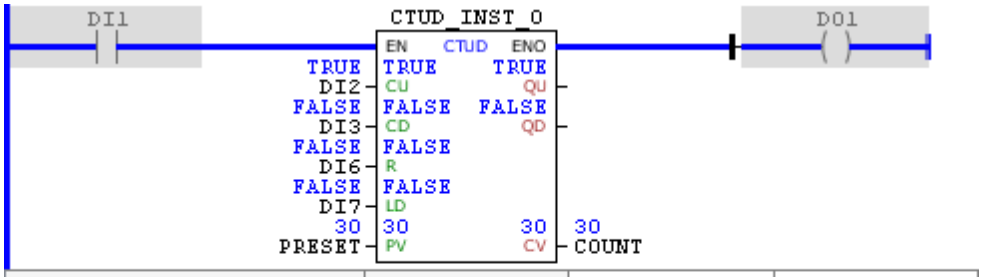
PRESET	UINT	30	Decimal
COUNT	UINT	0	Decimal

The example above shows the disabled block, with all its internal variables zeroed. Although the external controls are activated, these values are not forwarded to the instance of the block.



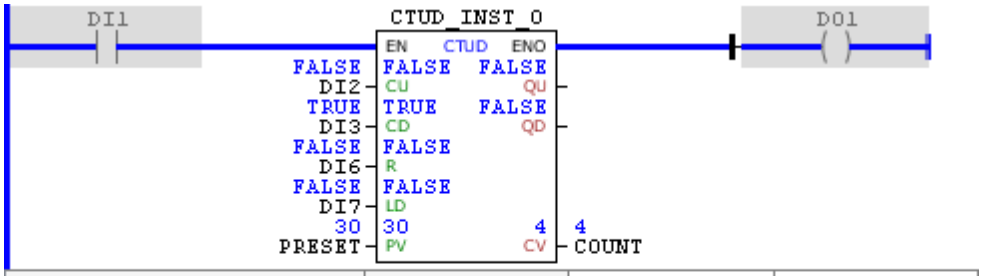
PRESET	UINT	30	Decimal
COUNT	UINT	0	Decimal

When activated, the block identifies the value of PRESET, loading it in PV, and identifies that the output is at zero, enabling the QD output. When execution is completed, the ENO output is activated.



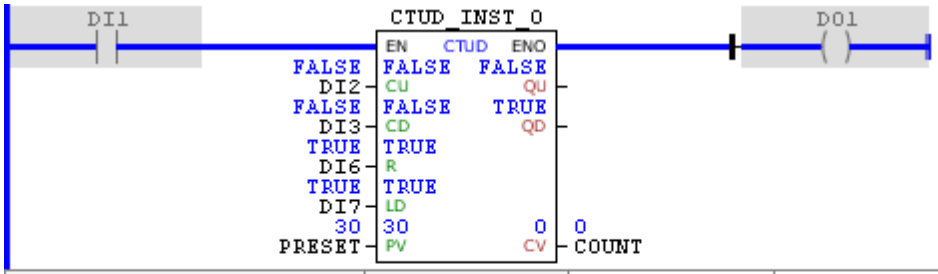
PRESET	UINT	30	Decimal
COUNT	UINT	30	Decimal

At each leading edge identified in CU, the value of CV is incremented until it reaches the PV value, when the QU output is enabled. When execution is completed, the ENO output is activated.



PRESET	UINT	30	Decimal
COUNT	UINT	4	Decimal

At each leading edge detected in CD, the CV value is decremented. When CV is a value between zero and PV, both QD and QU outputs are deactivated. When execution is completed, the ENO output is activated.



PRESET	UINT	30	Decimal
COUNT	UINT	0	Decimal

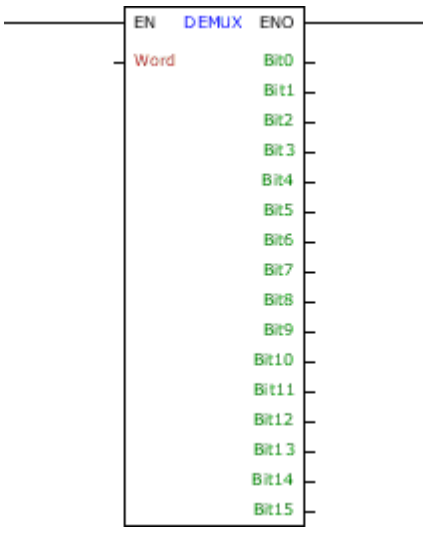
A TRUE value in R resets CV, while a TRUE value in LD loads the value of PV to CV. As we can see, R prevails over LD, leaving CV and enabling the QD output. When execution is completed, the ENO output is activated.

11.1.6.8 Data Transfer

11.1.6.8.1 DEMUX

Block that creates 16 new BOOL variables from the decomposition of a WORD variable.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Word	WORD UINT INT	Input variable of 15 bits
VAR_OUTPUT	ENO	BOOL	End of operation
	Bit0 – Bit15	BOOL	Bit of the corresponding position of Word

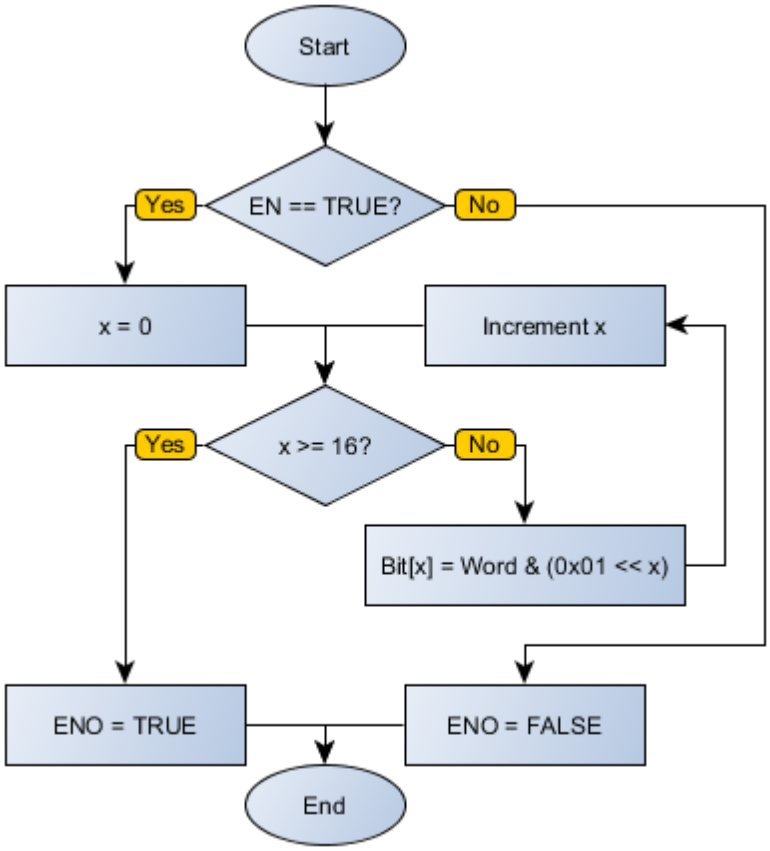
Operation

When this block has a TRUE value in EN, it decomposes the input variable in Word 15 Boolean values stored in Bit0 to Bit15 variables. Bit0 corresponds to the LSB (least significant bit) and Bit15 corresponds to the MSB (most significant bit).

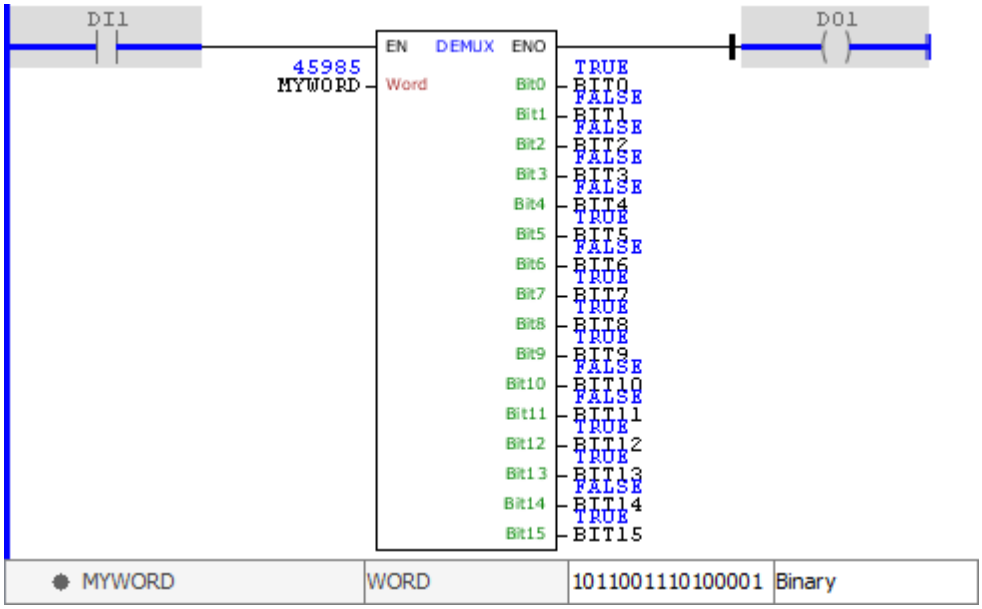
When EN has FALSE value, output variables remain unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example

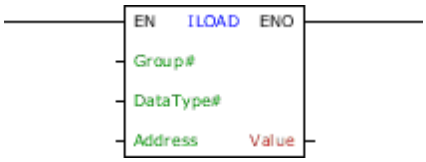


The example above decomposes the value of MYWORD in Boolean values, which are stored in the output variables BIT0 to Bit15. The block ends successfully and the ENO output is activated.

11.1.6.8.2 ILOAD

Block which indirectly loads the value of a variable and transfers it to Value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Group#	BYTE	Group w here the variable is stored
	DataType#	BYTE	Data type of the selected variable
	Address	DWORD UDINT DINT	Address of the global variable, as its group
VAR_OUTPUT	ENO	BOOL	End of operation
	Value	As selected in DataType#	Value of the selected variable

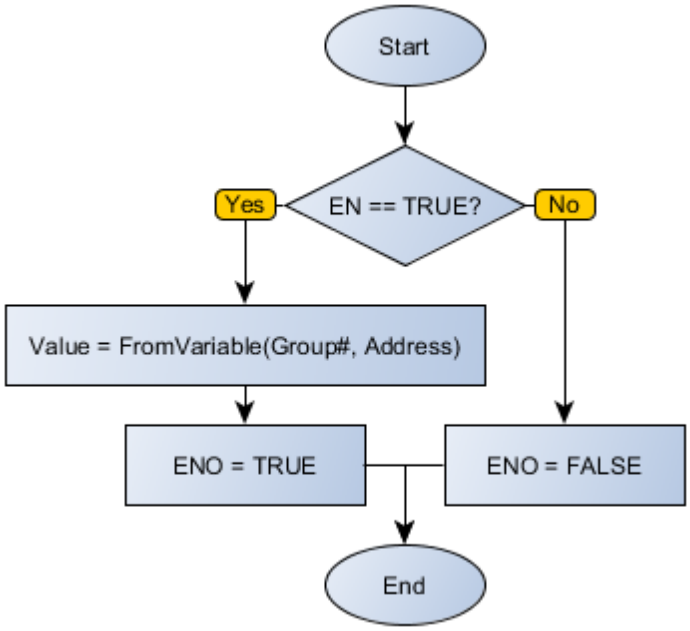
Operation

When this block has a TRUE value in EN, it loads, in Value, the of the Address variable belonging to the Group# group, as the selected DataType#.

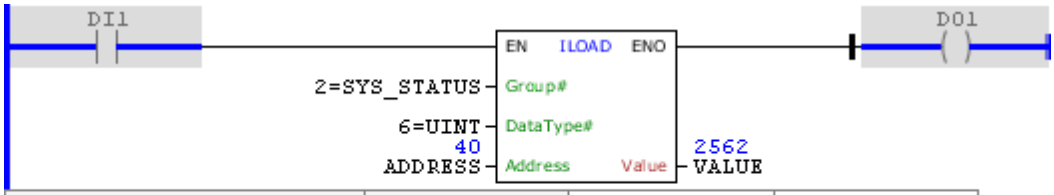
When EN has FALSE value, Value remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



ADDRESS	DINT	40	Decimal
VALUE	UINT	2562	Decimal

The above example loads the value of the address 40 of group 2 (GLOBAL_SYSTEM%S), which represents the status of ESC key in UINT format for the VALUE variable. The block ends with success and ENO output is activated.

11.1.6.8.3 ILOADBOOL

Block that indirectly loads the value of a bit in a global variable address.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Group#	BYTE	Group w here the variable is stored
	Address	DWORD UDINT DINT	Address of the global variable, as its group
	Bit	BYTE USINT SINT	Position of the bit to be checked
VAR_OUTPUT	ENO	BOOL	End of operation
	Value	BOOL	Value of the bit selected by the input arguments

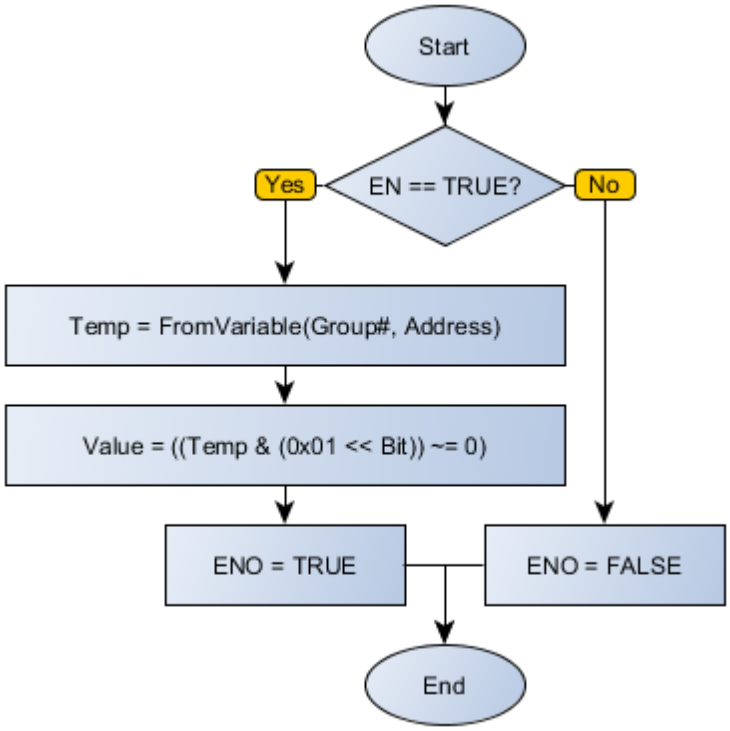
Operation

When this block has a TRUE value in EN, it loads, in Value, the Bit contents of the Address variable belonging to the Group# group.

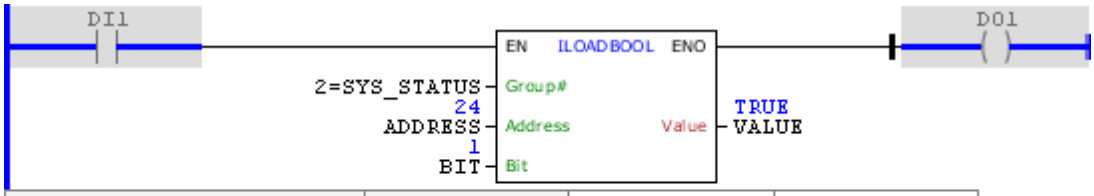
When EN has FALSE value, Value remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



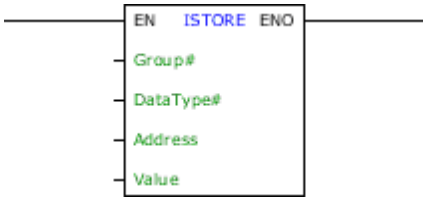
ADDRESS	DINT	24	Decimal
BIT	BYTE	1	Decimal
VALUE	BOOL	1	Binary

The above example loads the value of bit 1 of the address 24 of group 2 (S GLOBAL_SYSTEM%), which represents the status of ESC key for the VALUE variable. The block ends with success and ENO output is activated.

11.1.6.8.4 ISTORE

Block that indirectly loads the Value value in a variable.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Group#	BYTE	Group where the variable is stored
	DataType#	BYTE	Data type of the selected variable
	Address	DWORD UDINT DINT	Address of the global variable, as its group
	Value	Depending on the selection of the DataType#	Value to be written in the selected variable
VAR_OUTPUT	ENO	BOOL	End of operation

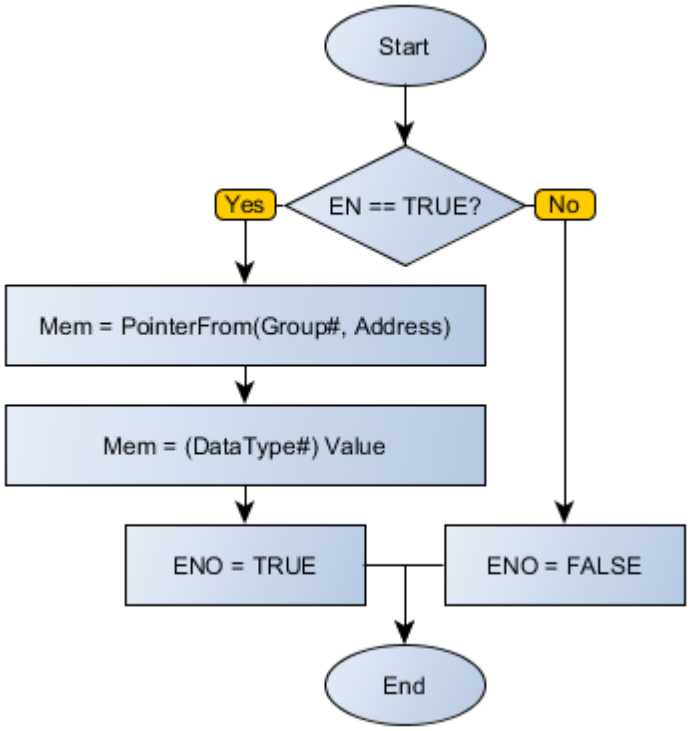
Operation

When this block has a TRUE value in EN, it loads the Value value in the contents of the Address variable belonging to the Group# group, as the selected DataType#.

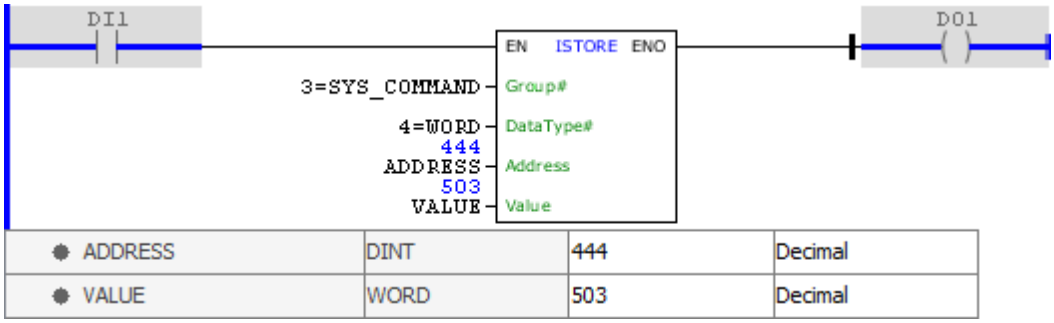
When EN has FALSE value, Value remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



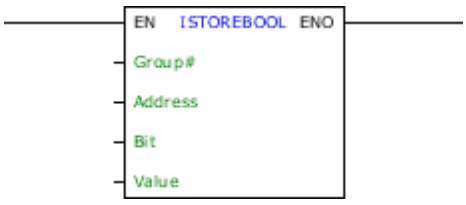
ADDRESS	DINT	444	Decimal
VALUE	WORD	503	Decimal

The example above stores the VALUE value in WORD format in address 444 of group 3 (GLOBAL_SYSTEM% C), which represents the index of the communication port Modbus TCP. The block ends with success and ENO output is activated.

11.1.6.8.5 ISTOREBOOL

Block that indirectly loads the Value value in a bit in a global variable address.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Group#	BYTE	Group where the variable is stored
	Address	DWORD UDINT DINT	Address of the global variable, as its group
	Bit	BYTE USINT SINT	Position of the bit to be modified
	Value	BOOL	New value of the selected bit
VAR_OUTPUT	ENO	BOOL	End of operation

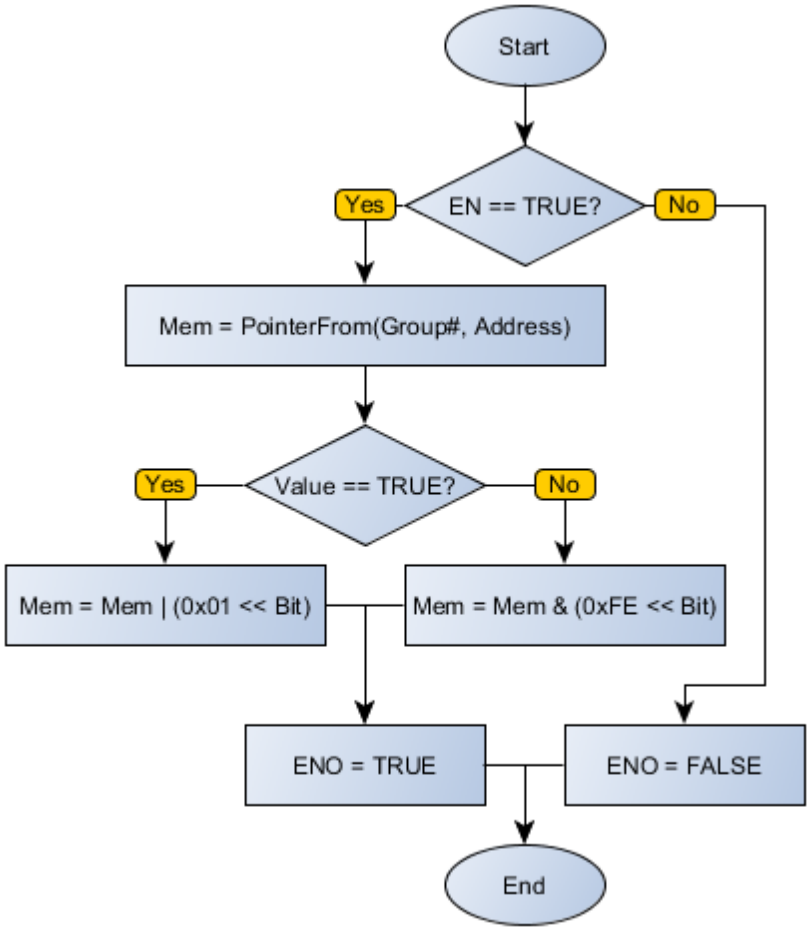
Operation

When this block has a TRUE value in EN, it loads the Value value in the Bit contents of the Address variable belonging to the Group# group.

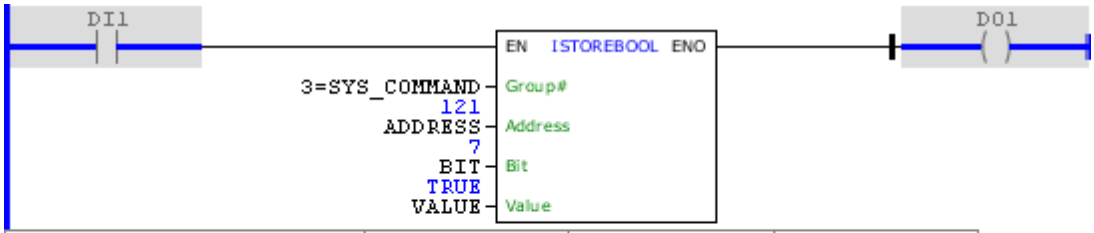
When EN has FALSE value, Value remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



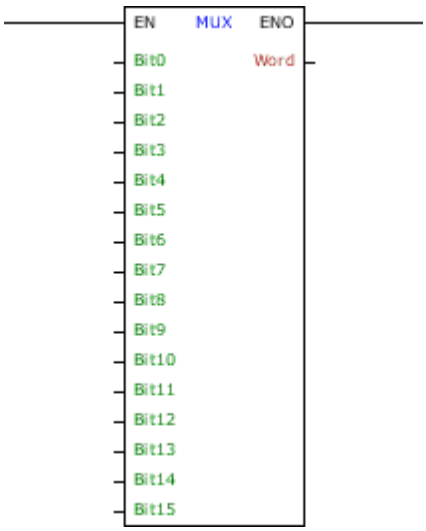
ADDRESS	DINT	121	Decimal
BIT	BYTE	7	Decimal
VALUE	BOOL	1	Binary

The example above stores the value of VALUE in bit 7 of the address 121 in group 3 (GLOBAL_SYSTEM% C), which represents the disable command of CANopen communication. The block ends with success and ENO output is activated.

11.1.6.8.6 MUX

Block that creates a new WORD variable from the concatenation of 16 BOOL variables.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Bit0 – Bit15	BOOL	Bit of the corresponding position in the new word
VAR_OUTPUT	ENO	BOOL	End of operation
	Word	WORD UINT INT	New word formed from the input bits

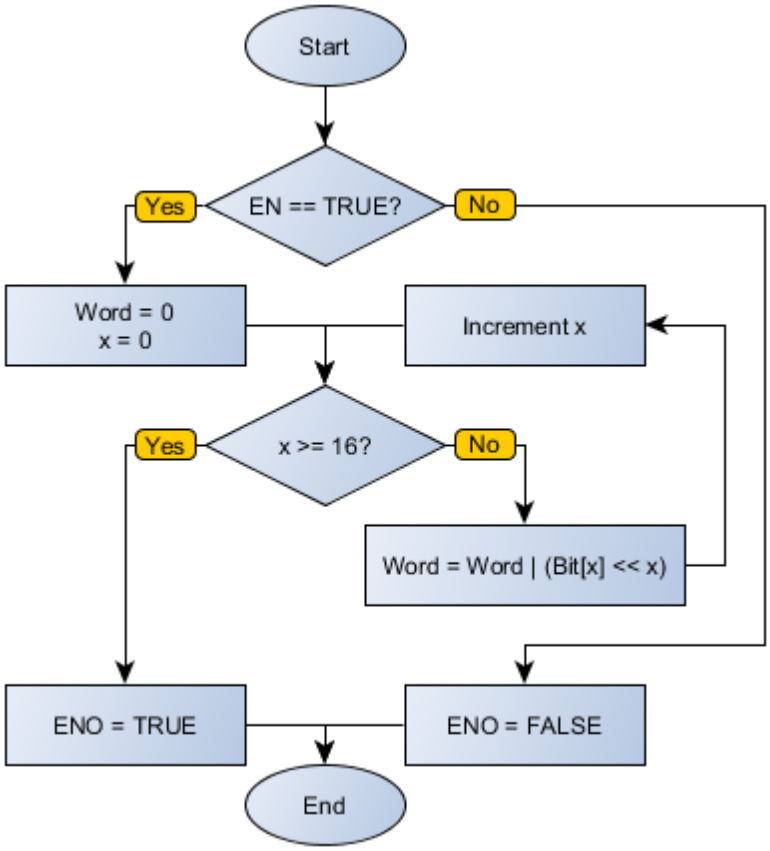
Operation

When this block has a TRUE value in EN, it concatenates Boolean values of the input variables and stores this value in the variable Word. Bit0 corresponds to LSB (least significant bit) and Bit15 corresponds to the MSB (most significant bit).

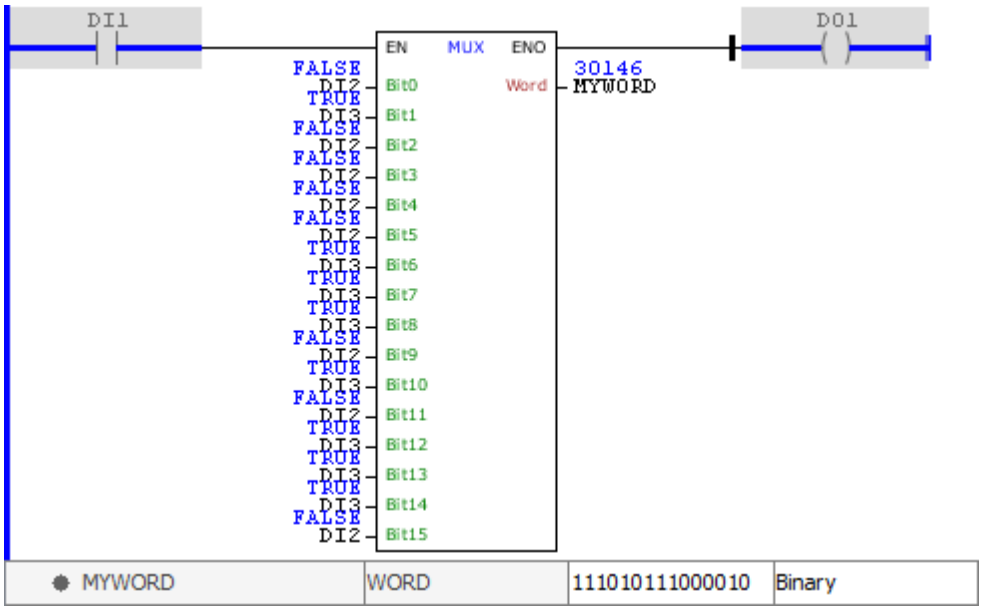
When EN has FALSE value, Word remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example

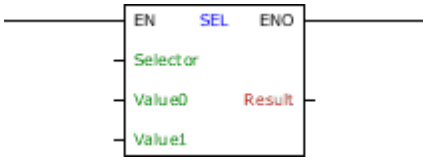


The above example concatenates the Boolean values of the input bits of the block to form the output word stored in MYWORD. The block ends with success and ENO output is activated.

11.1.6.8.7 SEL

Block that replicates to the output the value of an input variable (Value0 or Value1) according to the Selector selection.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Selector	BOOL	Variable that selects the input
	Value0	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Multiplexed input number 1
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Multiplexed input number 2
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Output value selected

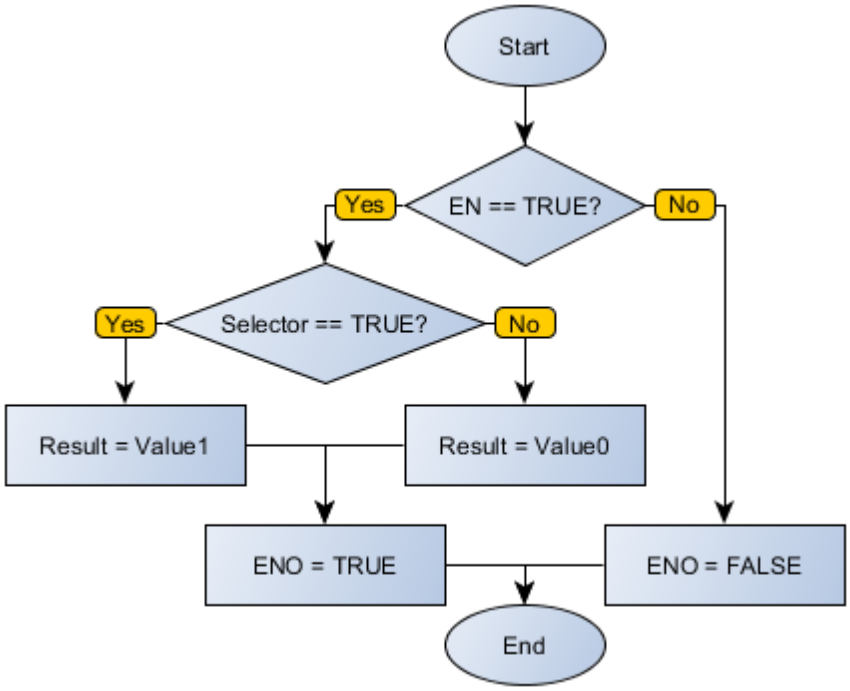
Operation

When this block has a TRUE value in EN, it replicates to the Result variable the Value0 value if selector is FALSE or the Value1 value if Selector is TRUE.

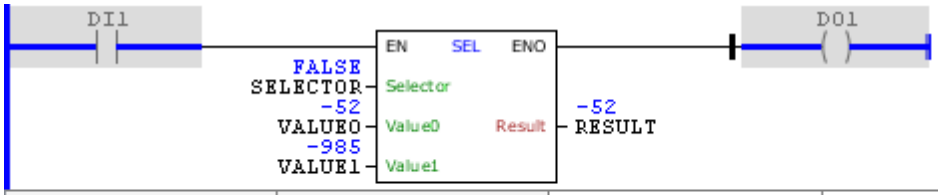
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart

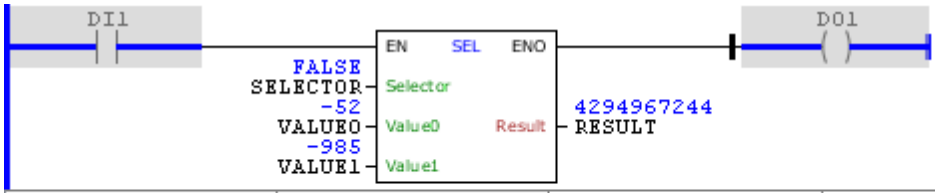


Example



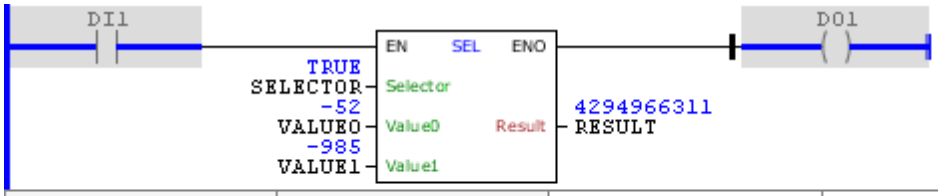
SELECTOR	BOOL	0	Binary
VALUE0	SINT	-52	Decimal
VALUE1	INT	-985	Decimal
RESULT	INT	-52	Decimal

The above example uses the SELECTOR variable as multiplexing channel selector. When it is at FALSE level, the RESULT output gets the value of VALUE0. The block ends successfully and the ENO output is activated.



SELECTOR	BOOL	0	Binary
VALUE0	SINT	-52	Decimal
VALUE1	INT	-985	Decimal
RESULT	DWORD	4294967244	Decimal

The above example uses the SELECTOR variable as multiplexing channel selector. When it is at FALSE level, the RESULT output gets the value of VALUE0. The block ends successfully and the ENO output is activated. Note that the binary pattern has been maintained even though the decimal representation is corrupted, since DWORD does not accept negative values.



SELECTOR	BOOL	1	Binary
VALUE0	SINT	-52	Decimal
VALUE1	INT	-985	Decimal
RESULT	DWORD	4294966311	Decimal

The above example uses the SELECTOR variable as multiplexing channel selector. When it is at TRUE level, the RESULT output gets the value of VALUE1. The block ends successfully and the ENO output is activated. Note that the binary pattern has been maintained even though the decimal representation is corrupted, since DWORD does not accept negative values.

11.1.6.8.8 STORE

Block that performs direct storage of data from a source to a destination.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	SRC	BYTE USINT SINT WORD UINT INT DWORD DINT DINT REAL	Data source
VAR_OUTPUT	ENO	BOOL	End of operation
	DST	BYTE USINT SINT WORD UINT INT DWORD DINT DINT REAL	Data destination

Operation

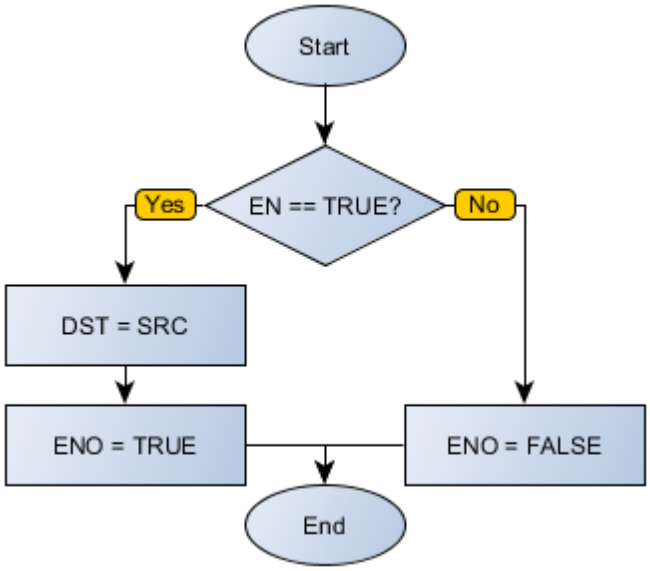
When this block has a TRUE value in EN, it stores the contents from SRC into DST.

NOTE!
SRC and DST must have data types of the same size.

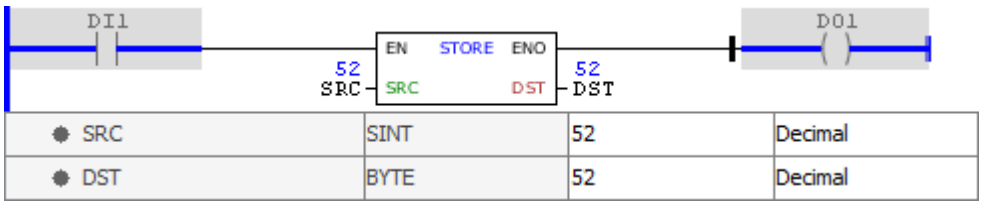
When EN has FALSE value, DST remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



The example above stores the value of the variable SRC in DST. The block ends with success and ENO output is activated.

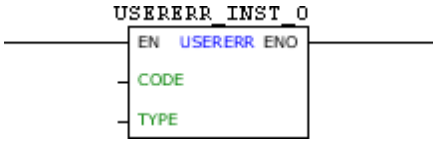
SRC	SINT	-15	Decimal
DST	BYTE	241	Decimal

The example above stores the value of the variable SRC in DST. The block ends with success and ENO output is activated. Note that the binary pattern is maintained between variables of different types.

11.1.6.8.9 USERERR

Block that generates an alarm or fault with the number programmed by the user.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	CODE	WORD UINT	Error code generated (950 - 999)
	TYPE	BYTE	Error type generated (0 - Alarm) (1 - Fault)
VAR_OUTPUT	ENO	BOOL	Success in the generation of error
VAR	USERERR_INST_0	USERERR	(*) Instance of access to block structure

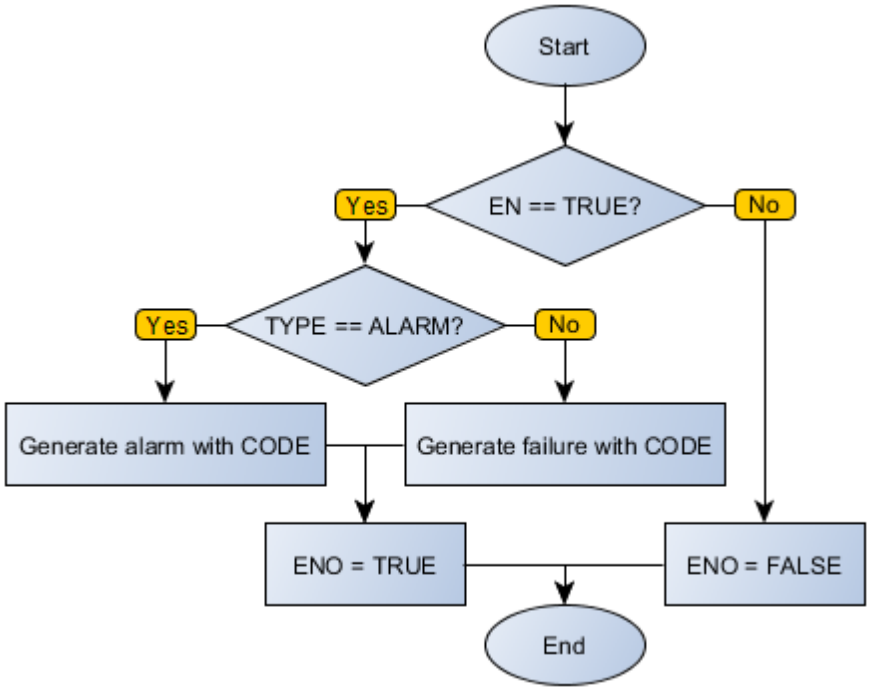
NOTE!
(*) USERERR_INST_0 instance must be configurated to SCA06 and LDW900.

Operation

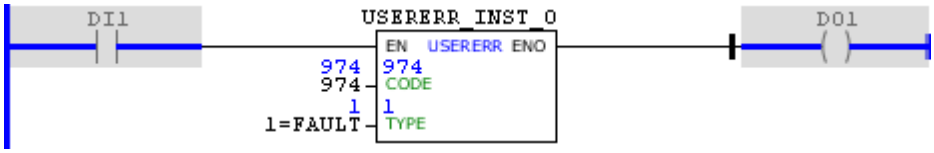
When this block has a TRUE value in EN, it generates an alarm or equipment failure, depending on the type defined in TYPE with CODE code.

The value of ENO informs if the generation of alarm or fault has been executed successfully.

Block Flowchart



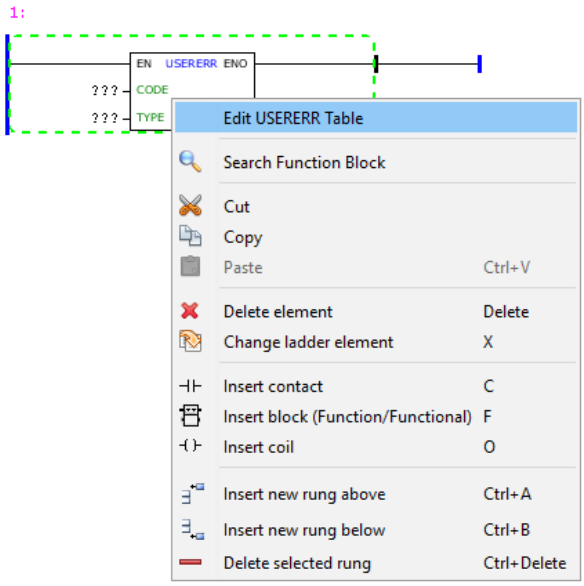
Example



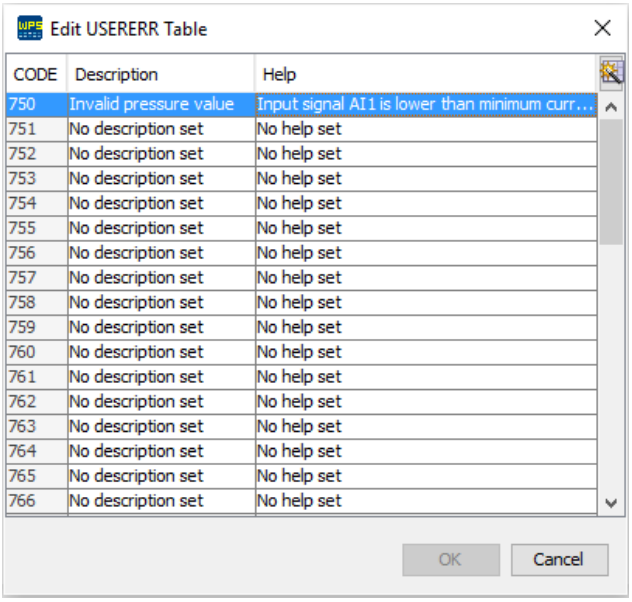
The above example, when identifying TRUE level in DI1, generates a fault with the code 974 and sets the DO1 output.

USERERR table configuration

On devices that have text-based HMI, messages can be configured through an editor. To access the editor, right click on the USERERR block and select the "Edit USERERR Table" option.



The texts configured in the table will be displayed on the HMI when the block USERERR is enabled.



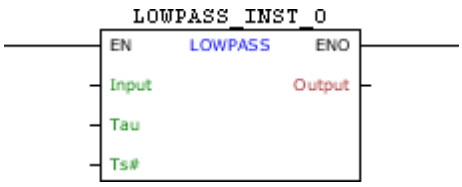
After editing the table, select the argument CODE of the block equal to the CODE column of the table.

11.1.6.9 Filter

11.1.6.9.1 LOWPASS

Block that filters the input using a low pass filter of first order and inserts the result in the output.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Input	REAL	Input signal
	Tau	REAL	Filter time constant
	Ts#	UINT	Sampling time [ms]
VAR_OUTPUT	ENO	BOOL	Output enabling
	Output	REAL	Filter output
VAR	LOWPASS_INST_0	LOWPASS	Instance of access to block structure

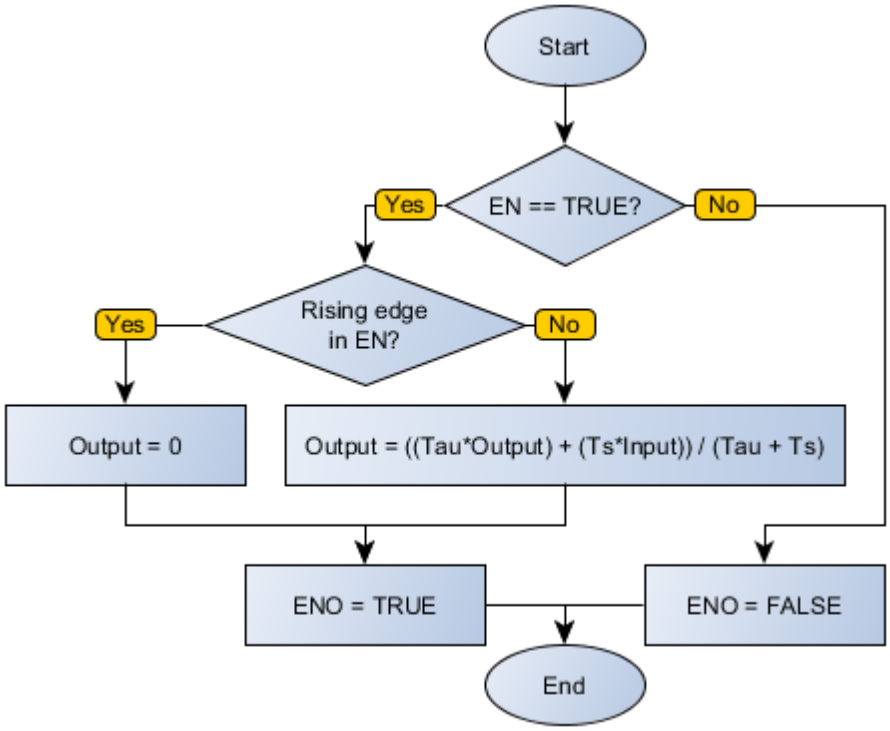
Operation

When this block has a TRUE value in EN, filters the input value of Input using a low pass first order filter described by Tau and Ts#, inserting the result in Output. On the leading edge of EN, Output receives zero.

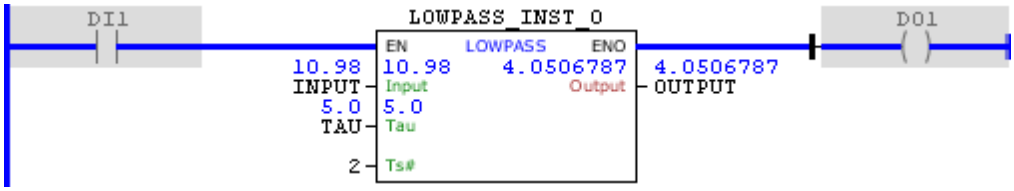
When EN has FALSE value, Output remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

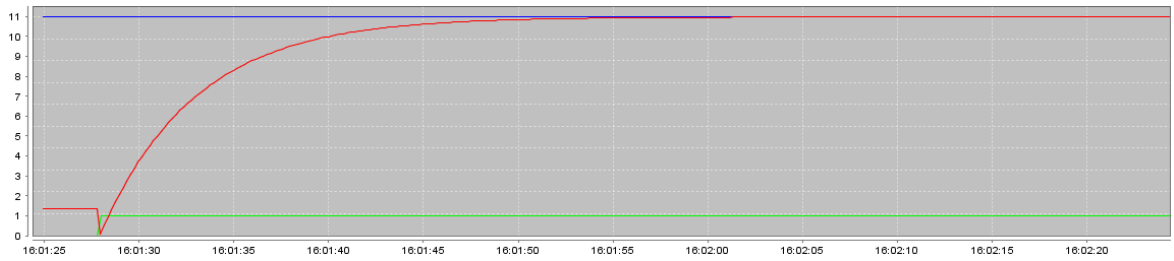
Block Flowchart



Example



The above example causes OUTPUT, by identifying a leading edge in EN, to display a behavior of first order with time constant equal to Tau and the sampling time of 2 ms, in order to achieve the reference set to INPUT. At each calculation completed successfully, the ENO output is activated.



11.1.6.10 Logic

11.1.6.10.1 Logic Bit

11.1.6.10.1.1 RESETBIT

Logical block used to perform reset of a specific bit in a field.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_IN_OUT	Data	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable w hose bit w ill be changed
VAR_INPUT	EN	BOOL	Block enabling
	Position	BYTE USINT	Position of the bit that w ill be changed
VAR_OUTPUT	DONE	BOOL	Operation successful

Operation

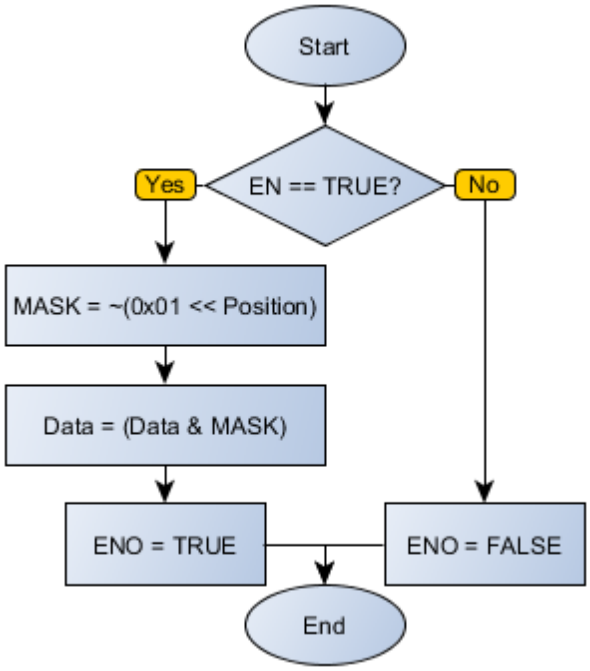
This block when it has a TRUE value in EN, resets the bit indicated in Position in the Data variable that is forwarded to the output already with its updated value.

When EN has FALSE value, Data remains unchanged.

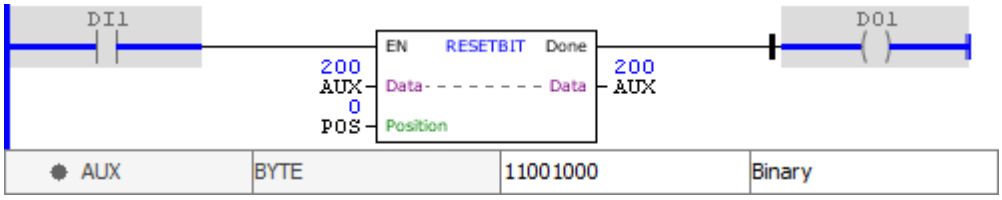
The DONE variable receives the same EN value, except when there is an error in the reset of the bit, then getting a FALSE value.

NOTE!
It is important to notice that Position is within the range of values of bits corresponding to variable type in Data. For example: if Data is a BYTE, it has 8 bits, and Position must contain a value between 0 and 7.

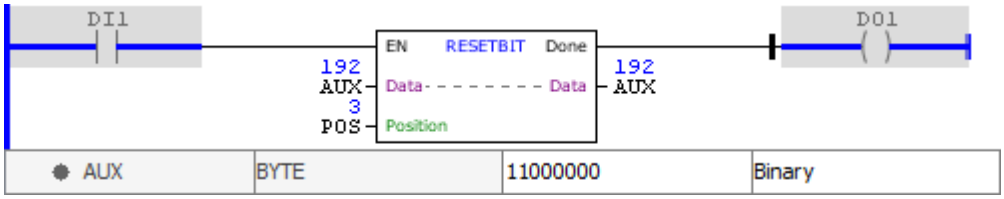
Block Flowchart



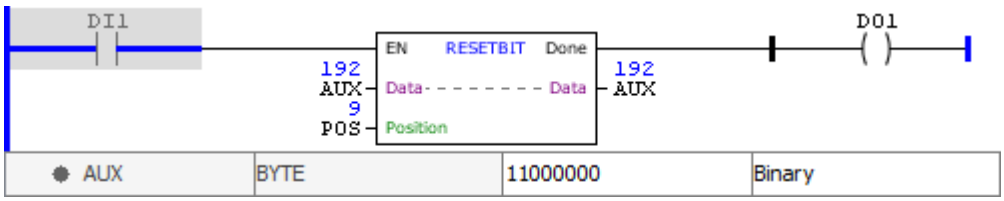
Example



The example above resets the bit of AUX zero position, whose initial value is 200 (1100 1000, in binary). Since this bit already had FALSE value, nothing has changed.



The example above resets the bit in position three of AUX by changing its binary value and, therefore, its decimal representation.



The example above resets the bit in position nine of AUX. Since AUX is a variable BYTE type, it has

only eight bits. Thus, the example above creates a runtime error in the block and therefore the output is not enabled.

11.1.6.10.1.2 SETBIT

Logical block used to perform the set of a specific bit in a field.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_IN_OUT	Data	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable w hose bit w ill be changed
VAR_INPUT	EN	BOOL	Block enabling
	Position	BYTE USINT	Position of the bit that w ill be changed
VAR_OUTPUT	DONE	BOOL	Operation successful

Operation

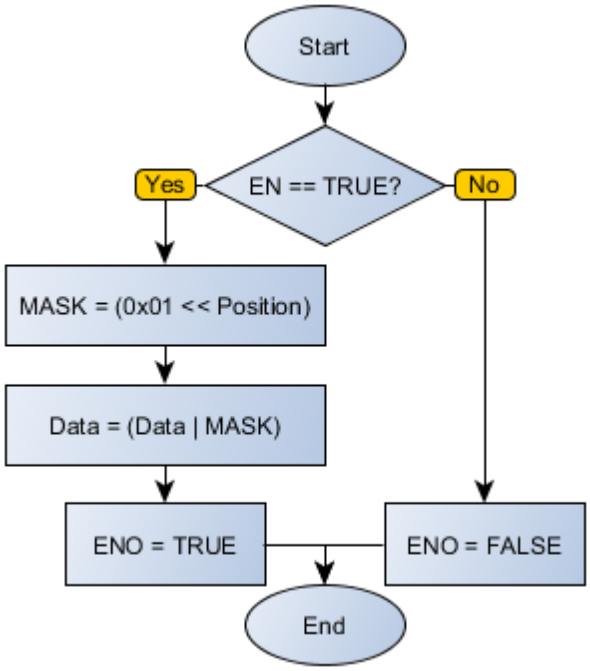
This block when it has a TRUE value in EN, sets the bit indicated in Position in the Data variable that is forwarded to the output already with its updated value.

When EN has FALSE value, Data remains unchanged.

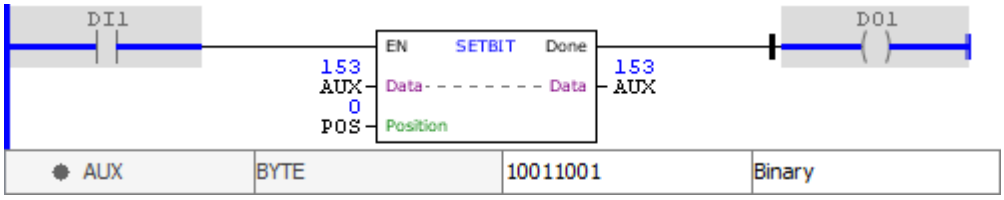
The DONE variable receives the same EN value, except when there is an error in the set of the bit, then getting a FALSE value.

NOTE!
It is important to notice that Position is within the range of values of bits corresponding to variable type in Data. For example: if Data is a BYTE, it has 8 bits, and Position must contain a value between 0 and 7.

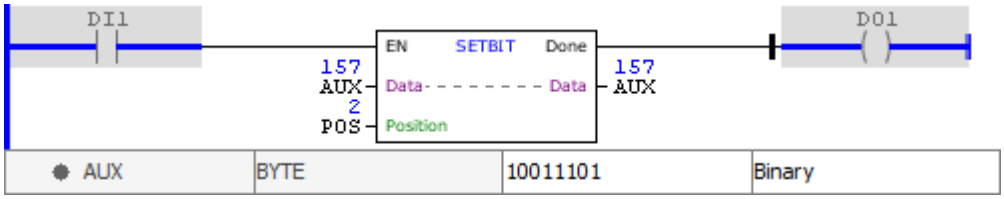
Block Flowchart



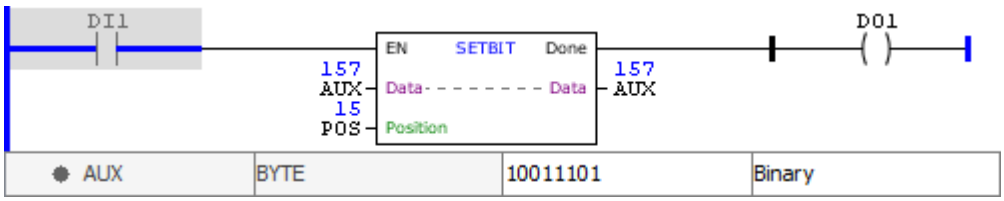
Example



The example above sets the bit of AUX zero position, whose initial value is 153 (1001 1001, in binary). Since this bit already had TRUE value, nothing has changed.



The example above sets the bit in position three of AUX by changing its binary value and, therefore, its decimal representation.



The example above sets the bit in position fifteen of AUX. Since AUX is a variable BYTE type, it has

only eight bits. Thus, the example above creates a runtime error in the block and therefore the output is not enabled.

11.1.6.10.1.3 TESTBIT

Logical block that revolutions the value of a specific bit in a field.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	Data	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable w hose bit w ill be tested
	EN	BOOL	Block enabling
	Position	BYTE USINT	Position of the bit that w ill be changed
VAR_OUTPUT	Q	BOOL	Value of the tested bit

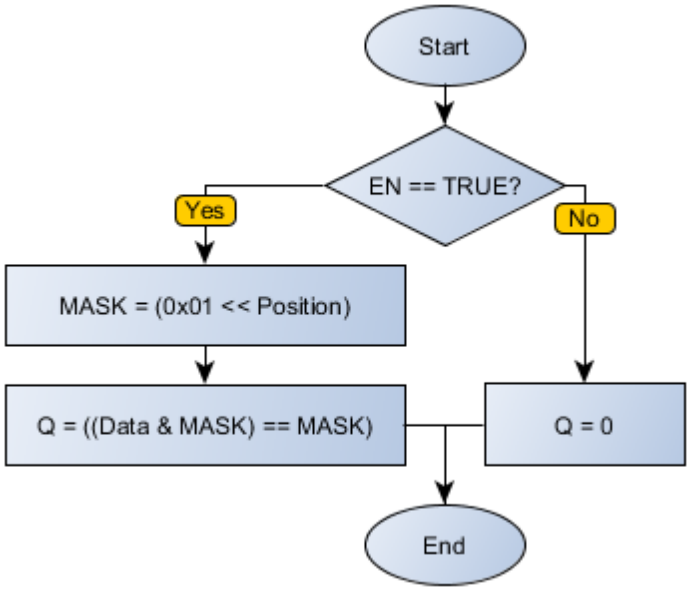
Operation

This block when it has a TRUE value in EN, sends to the output Q the bit value indicated in Position in the Data variable.

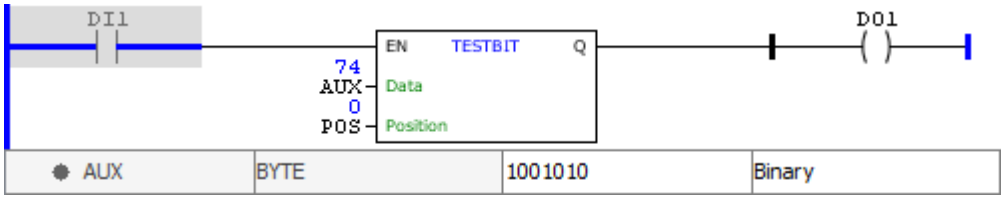
When EN has FALSE value, Q also receives FALSE.

NOTE!
It is important to notice that Position is within the range of values of bits corresponding to variable type in Data. For example: if Data is a BYTE, it has 8 bits, and Position must contain a value between 0 and 7.

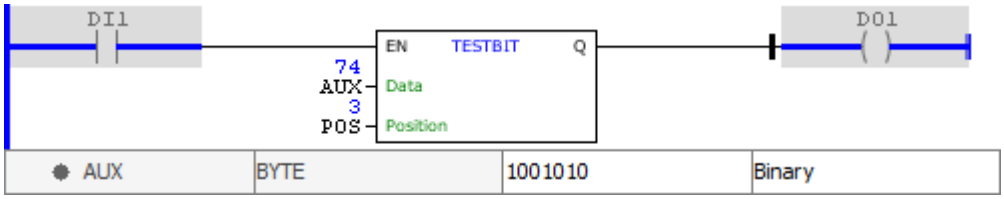
Block Flowchart



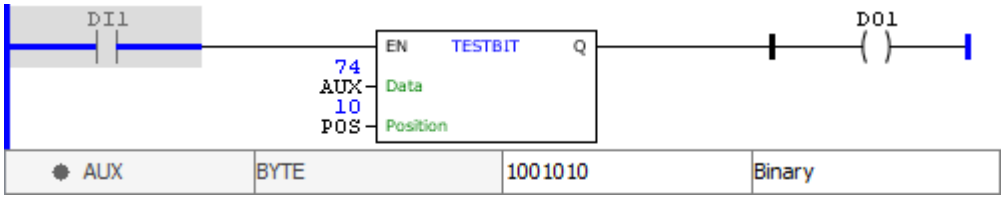
Example



The example above sets the bit value of zero position of AUX, whose initial value is 74 (0100 1010 in binary) to the output Q. Since this bit has value 0, the output is disabled.



The example above sets the value of the bit of position three of AUX to the output Q. Since this bit has value 1, the output is enabled.



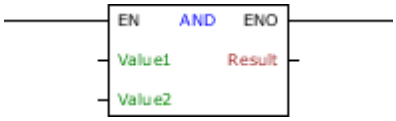
The example above sets the bit value of position ten of AUX to output Q. Since AUX is a variable of BYTE type, it has only eight bits. Thus, the example above creates a runtime error in the block and therefore the output is disabled.

11.1.6.10.2 Logic Boolean

11.1.6.10.2.1 AND

Logical block that performs an boolean "and" operation between two variables, storing the result in a third one.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable for the operation
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable for the operation
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable that stores the result of the operation

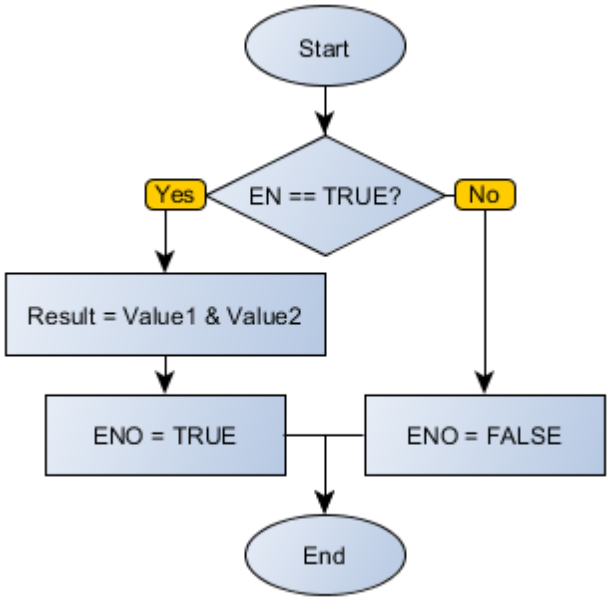
Operation

When this block has a TRUE value in EN, it sends to the Result output the “and” Boolean operation of input variables Value1 and Value2.

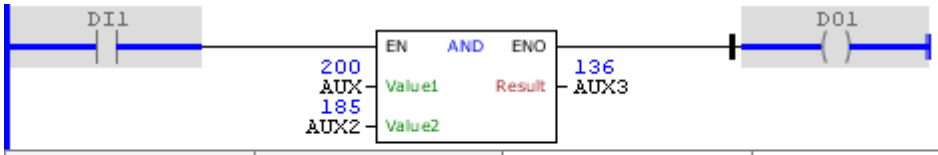
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



AUX	BYTE	11001000	Binary
AUX2	BYTE	10111001	Binary
AUX3	BYTE	10001000	Binary

The example above performs an "and" Boolean operation between AUX and AUX2, storing the result in AUX3.

11.1.6.10.2.2 NOT

Block that performs a logical operation of boolean "not" in a variable, storing the result in another.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Reference variable for the operation
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable that stores the result of the operation

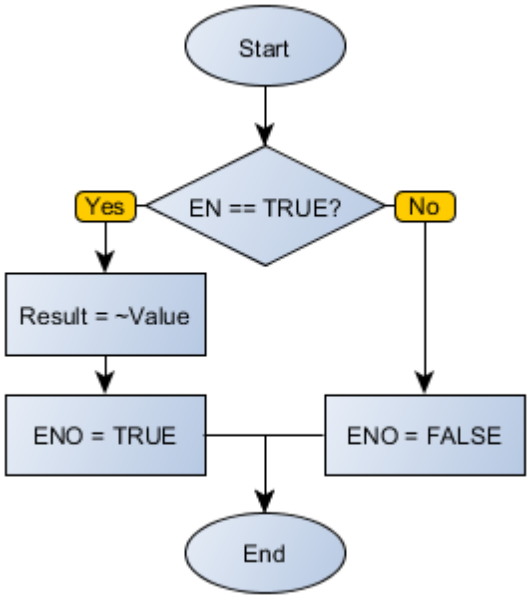
Operation

When this block has a TRUE value in EN, it sends to the Result output the denied Boolean value of the Value input variable.

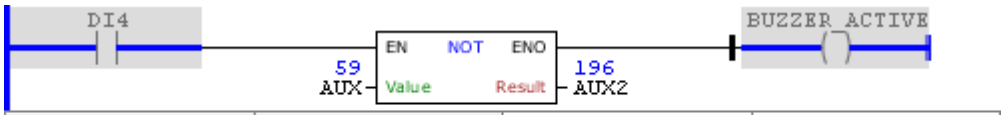
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



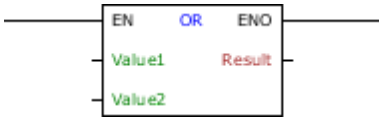
AUX	BYTE	111011	Binary
AUX2	BYTE	11000100	Binary

The example above performs a boolean "not" operation in AUX, storing the result in AUX2.

11.1.6.10.2.3 OR

Logical block that performs an Boolean "or" operation between two variables, storing the result in a third one.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable for the operation
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable for the operation
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable that stores the result of the operation

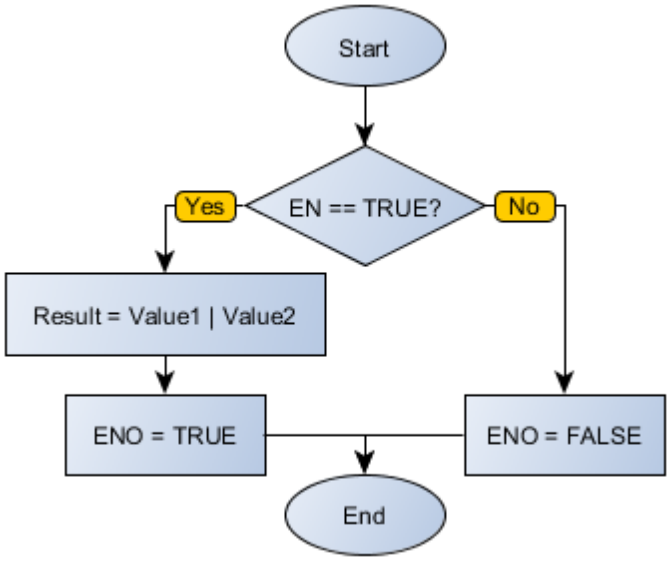
Operation

When this block has a TRUE value in EN, it sends to the Result output the “or” Boolean operation of input variables Value1 and Value2.

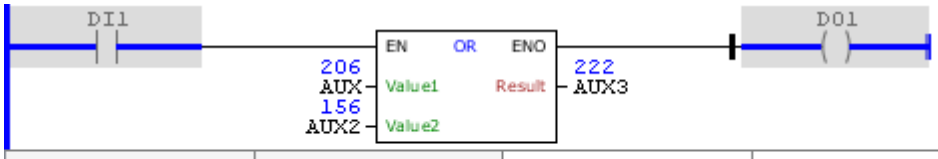
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



AUX	BYTE	11001110	Binary
AUX2	BYTE	10011100	Binary
AUX3	BYTE	11011110	Binary

The example above performs an "or" Boolean operation between AUX and AUX2, storing the result in AUX3.

11.1.6.10.2.4 XNOR

Logical block that performs an Boolean "not exclusive or" operation between two variables, storing the result in a third one.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable for the operation
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable for the operation
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable that stores the result of the operation

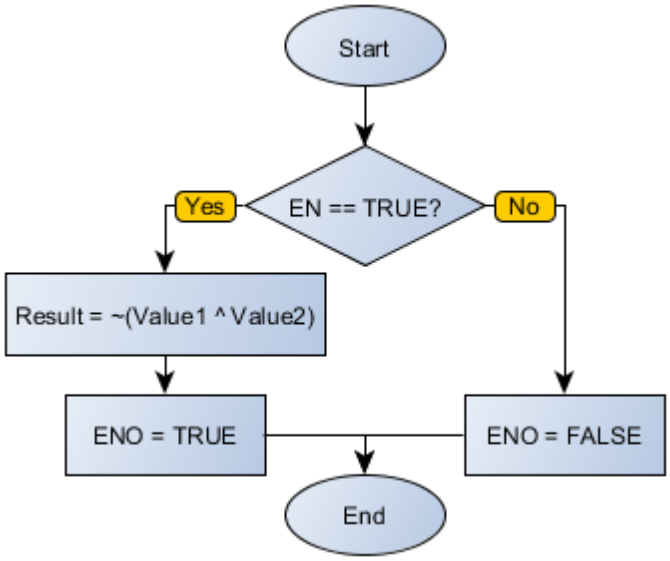
Operation

When this block has a TRUE value in EN, it sends to the Result output the “denied exclusive or” Boolean operation of input variables Value1 and Value2.

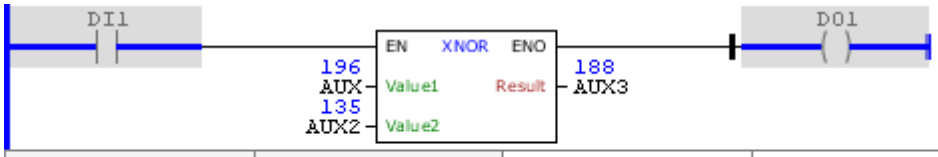
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



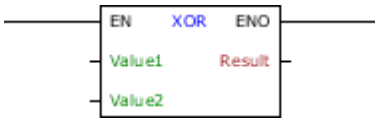
AUX	BYTE	11000100	Binary
AUX2	BYTE	10000111	Binary
AUX3	BYTE	10111100	Binary

The example above performs a "denied exclusive or" Boolean operation between AUX and AUX2, storing the result in AUX3.

11.1.6.10.2.5 XOR

Logical block that performs an Boolean "exclusive or" operation between two variables, storing the result in a third one.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable for the operation
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable for the operation
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable that stores the result of the operation

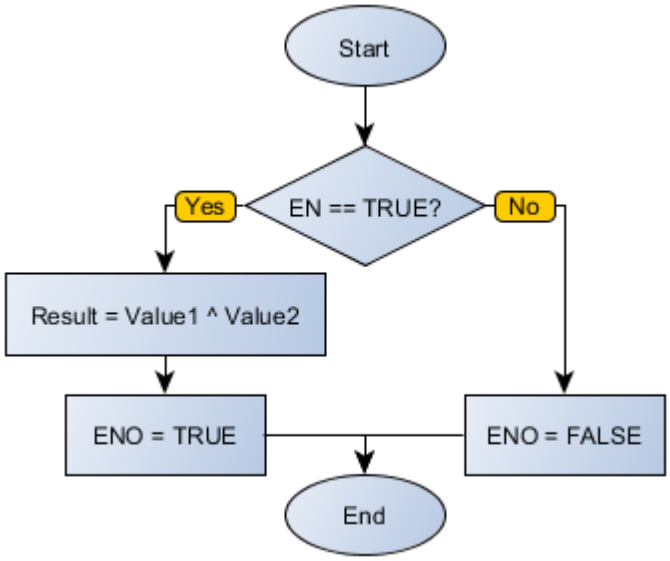
Operation

When this block has a TRUE value in EN, it sends to the Result output the “xor” Boolean operation of input variables Value1 and Value2.

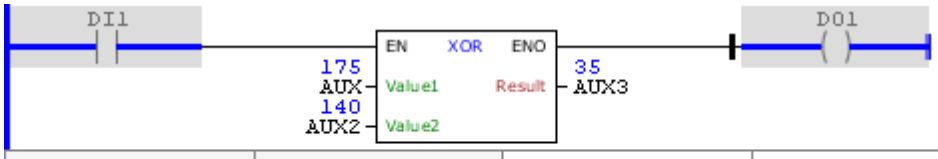
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



AUX	BYTE	10101111	Binary
AUX2	BYTE	10001100	Binary
AUX3	BYTE	100011	Binary

The example above performs a "xor" Boolean operation between AUX and AUX2, storing the result in AUX3.

11.1.6.10.3 Logic Rotate

11.1.6.10.3.1 ROL

Block that performs a logical left rotation operation in a value passed by Value, storing the result in Result.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable to undergo rotation
	Shift	BYTE USINT	Shift index
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable that stores the result of the operation

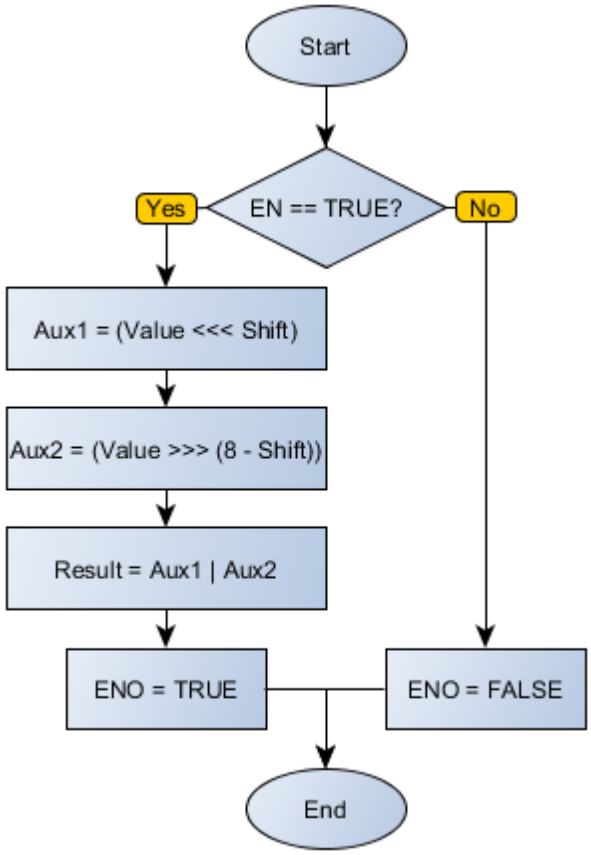
Operation

When this block has a TRUE value in EN, it sends to the Result output the value of the Value variable after performing a number of logical left shifts, according to the Shift value. The most significant bits that are being discarded are returned to the least significant bits, characterizing the rotation.

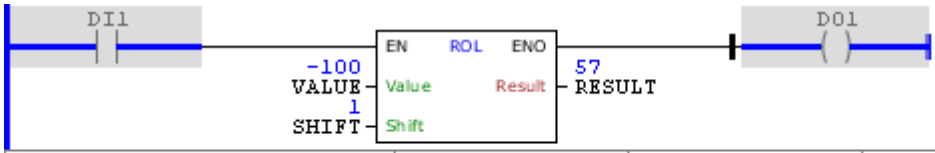
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart

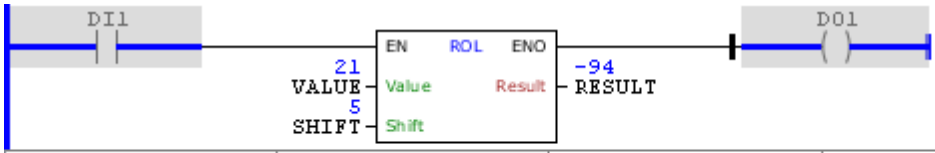


Example



VALUE	SINT	10011100	Binary
SHIFT	BYTE	1	Decimal
RESULT	SINT	111001	Binary

The above example performs a logical left shift by one position in the VALUE variable whose initial value is -100 (1001 1100 in binary). The discarded bits on the left are reinserted on the right. The final result (0011 1001 in binary) is stored in RESULT.



VALUE	SINT	10101	Binary
SHIFT	BYTE	5	Decimal
RESULT	SINT	10100010	Binary

The above example performs a logical left rotation by five positions in the VALUE variable whose initial value is 21 (0001 0101 in binary). The discarded bits on the left are reinserted on the right. The final result (1010 0010 in binary) is stored in RESULT.

11.1.6.10.3.2 ROR

Block that performs a logical right rotation operation in a value passed by Value, storing the result in Result.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable to undergo rotation
	Shift	BYTE USINT	Shift index
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable that stores the result of the operation

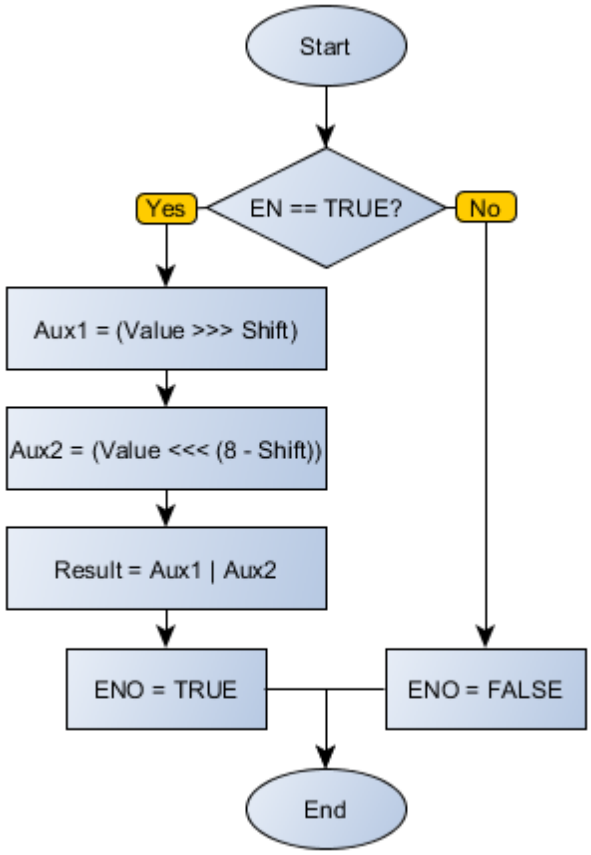
Operation

When this block has a TRUE value in EN, it sends to the Result output the value of the Value variable after performing a number of logical right shifts, according to the Shift value. The least significant bits that are being discarded are returned to the most significant bits, characterizing the rotation.

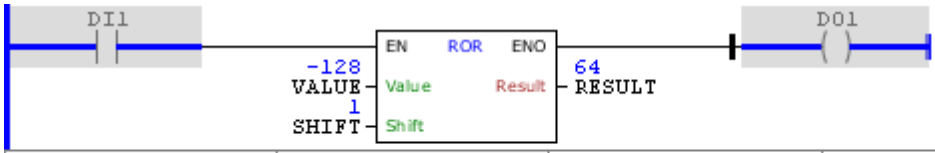
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart

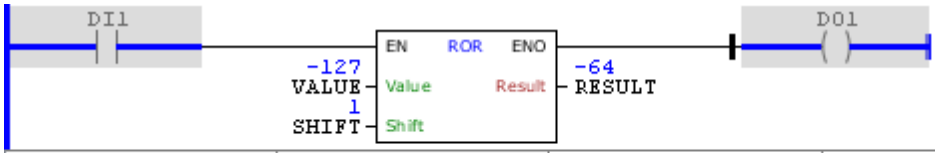


Example



VALUE	SINT	10000000	Binary
SHIFT	BYTE	1	Decimal
RESULT	SINT	1000000	Binary

The above example performs a logic right shift by one position in the VALUE variable whose initial value is -128 (1000 0000 in binary). The discarded bits on the right are reinserted on the left. The final result (0100 0000 in binary) is stored in RESULT. Notice that the sign is not preserved in this operation.



VALUE	SINT	10000001	Binary
SHIFT	BYTE	1	Decimal
RESULT	SINT	11000000	Binary

The above example performs a logical right rotation by one position in the VALUE variable whose initial value is -127 (1000 0001 in binary). The discarded bits on the right are reinserted on the left. The final result (1100 0000 in binary) is stored in RESULT.

11.1.6.10.4 Logic Shift

11.1.6.10.4.1 ASHL

Block that performs a binary left shift operation in a value passed by Value, storing the result in Result.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	SINT INT DINT	Variable to undergo shift
	Shift	BYTE USINT	Shift index
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	SINT INT DINT	Variable that stores the result of the operation

Operation

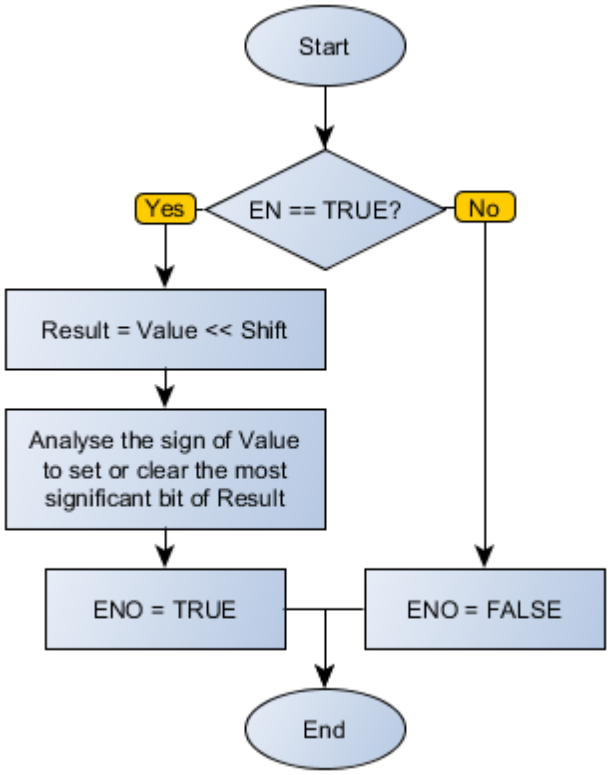
When this block has a TRUE value in EN, it sends to the Result output the value of the Value variable after performing a number of arithmetic left shifts, according to the Shift value.

NOTE!
All arithmetic shifts implemented maintain the sign of the variable.

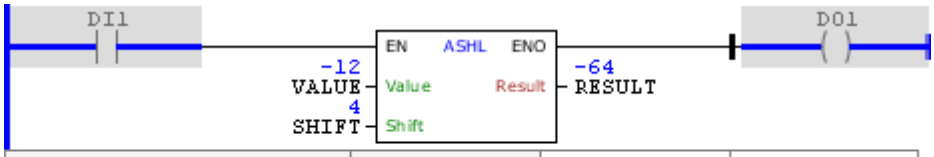
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart

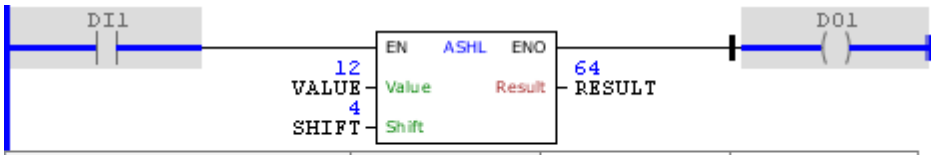


Example



VALUE	SINT	11110100	Binary
SHIFT	USINT	100	Binary
RESULT	SINT	11000000	Binary

Description of exemple.



VALUE	SINT	1100	Binary
SHIFT	USINT	100	Binary
RESULT	SINT	1000000	Binary

Description of exemple.

11.1.6.10.4.2 ASHR

Block that performs arithmetic left shift operation in a value passed by Value, storing the result in Result.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	SINT INT DINT	Variable to undergo shift
	Shift	BYTE USINT	Shift index
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	SINT INT DINT	Variable that stores the result of the operation

Operation

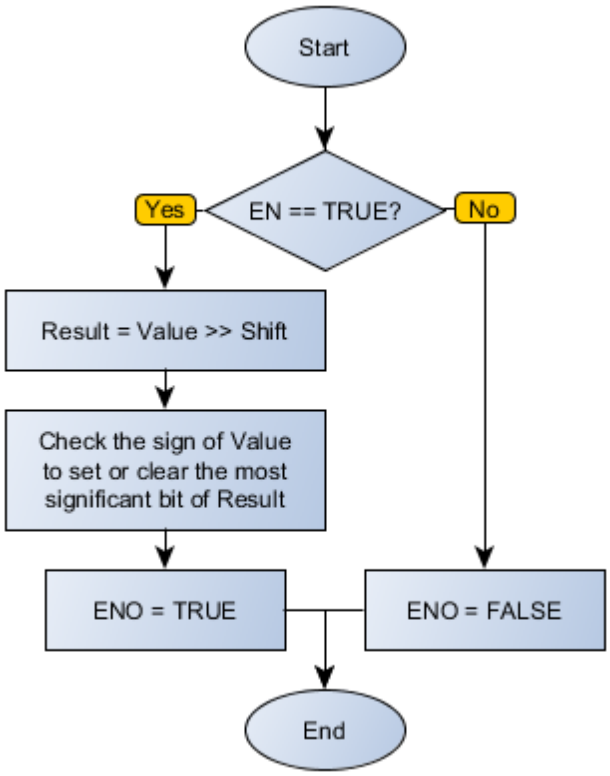
When this block has a TRUE value in EN, it sends to the Result output the value of the Value variable after performing a number of arithmetic right shifts, according to the Shift value.

NOTE!
All arithmetic shifts implemented maintain the sign of the variable.

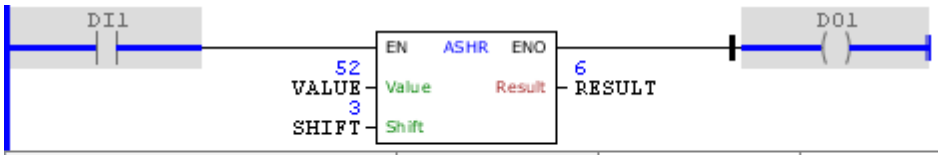
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart

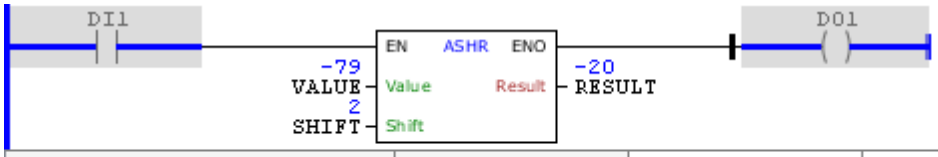


Example



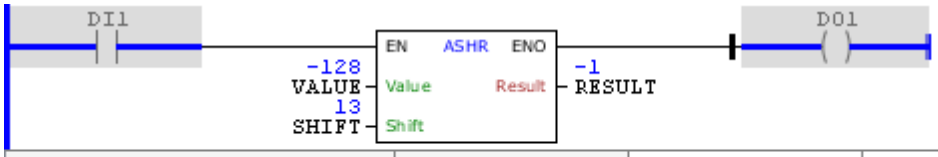
VALUE	SINT	110100	Binary
SHIFT	BYTE	3	Decimal
RESULT	INT	110	Binary

The above example performs an arithmetic right shift by three positions in the VALUE variable whose initial value is 52 (0011 0100 in binary). The bits on the right are being discarded, and on the left new zeros are inserted. The final result (0000 0110 in binary) is stored in RESULT.



VALUE	SINT	10110001	Binary
SHIFT	BYTE	2	Decimal
RESULT	INT	1111111111101100	Binary

The above example performs an arithmetic right shift by two positions in the VALUE variable whose initial value is -79 (1011 0001 in binary). The bits on the right will be discarded and new ones on the left are inserted, since the arithmetic right shifts preserve the sign of the variable. The final result (1111 0110 in binary) is stored in RESULT.



VALUE	SINT	10000000	Binary
SHIFT	BYTE	13	Decimal
RESULT	INT	1111111111111111	Binary

The above example performs an arithmetic right shift by thirteen positions in the VALUE variable whose initial value is -128 (1000 0000 in binary). The bits on the right are being discarded, and on the left new ones are inserted. The final result (1111 1111 in binary) is stored in RESULT.

11.1.6.10.4.3 SHL

Block that performs a binary logical left shift operation in a value passed by Value, storing the result in Result.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable to undergo shift
	Shift	BYTE USINT	Shift index
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable that stores the result of the operation

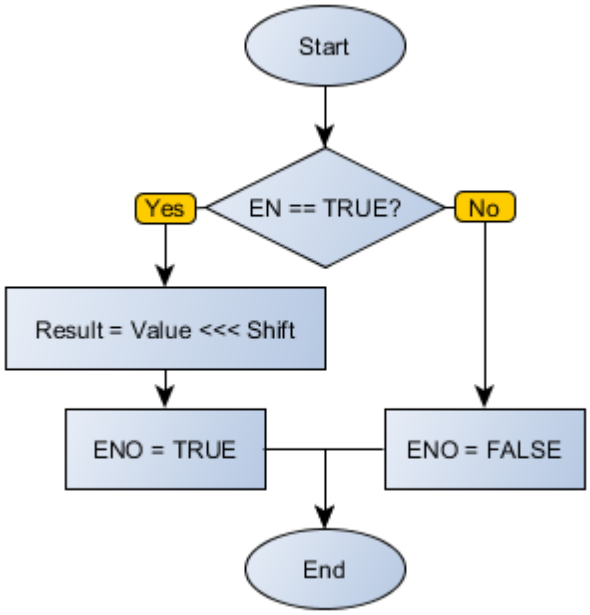
Operation

When this block has a TRUE value in EN, it sends to the Result output the value of the Value variable after performing a number of logical shifts left, according to the Shift value.

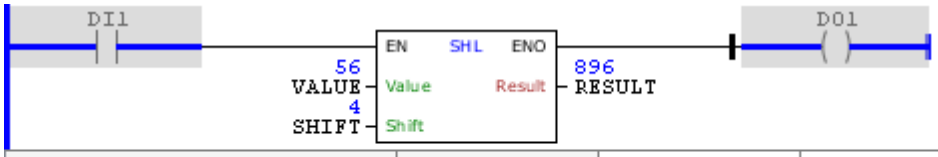
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart

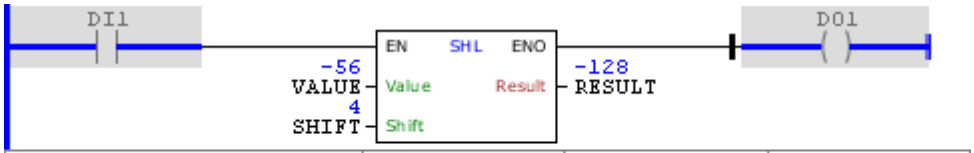


Example



VALUE	INT	111000	Binary
SHIFT	BYTE	4	Decimal
RESULT	INT	1110000000	Binary

The above example performs a logical right shift by four positions in the VALUE variable whose initial value is 56 (0011 1000 in binary). The bits on the left are being discarded, and on the left new zeros are inserted. The final result (0011 1000 0000 in binary) is stored in RESULT.



VALUE	SINT	11001000	Binary
SHIFT	USINT	4	Decimal
RESULT	SINT	10000000	Binary

The above example performs a logical right shift by four positions in the VALUE variable whose initial value is -56 (1100 1000 in binary). The bits on the left are being discarded, and on the left new zeros are inserted. The final result (1100 1000 0000 in binary) is stored in RESULT. Since RESULT is SINT type, it only accepts the first eight bits (1000 0000).

11.1.6.10.4.4 SHR

Block that performs a binary logical right shift operation in a value passed by Value, storing the result in Result.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable to undergo shift
	Shift	BYTE USINT	Shift index
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable that stores the result of the operation

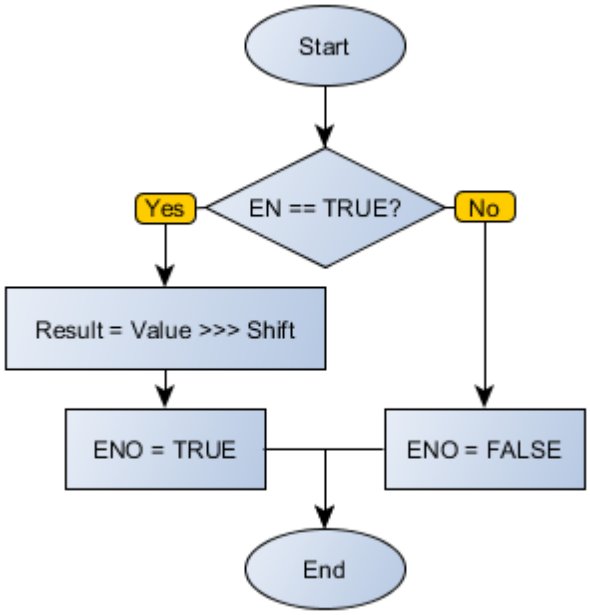
Operation

When this block has a TRUE value in EN, it sends to the Result output the value of the Value variable after performing a number of logical shifts right, according to the Shift value.

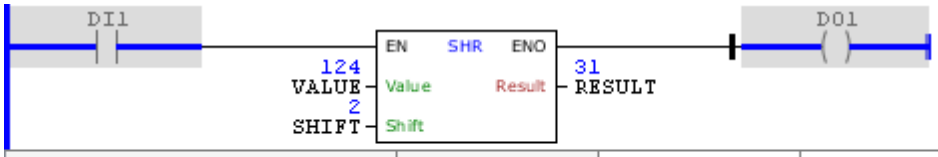
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart

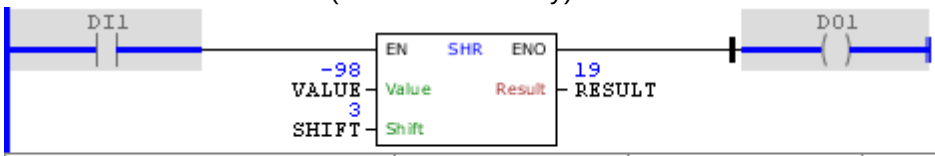


Example



VALUE	SINT	1111100	Binary
SHIFT	BYTE	2	Decimal
RESULT	SINT	11111	Binary

The above example performs a logical right shift by two positions in the VALUE variable whose initial value is 124 (0111 1100 in binary). The bits on the right are being discarded, and on the left new zeros are inserted. The final result (0001 1111 in binary) is stored in RESULT.



VALUE	SINT	10011110	Binary
SHIFT	BYTE	3	Decimal
RESULT	INT	10011	Binary

The above example performs a logical right shift by three positions in the VALUE variable whose initial value is -98 (1001 1110 in binary). The bits on the right are being discarded, and on the left new zeros are inserted. The final result (0001 0011 in binary) is stored in RESULT.

11.1.6.11 Math

11.1.6.11.1 Math Basic

11.1.6.11.1.1 ABS

Block that calculates the Value module, storing the result in Result.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Reference variable for the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Variable that stores the result of the operation

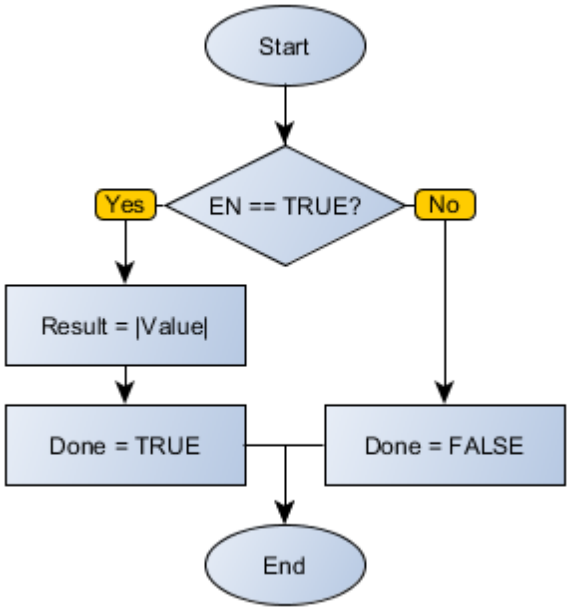
Operation

When this block has a TRUE value in EN, it sends to the Result output the absolute value of the

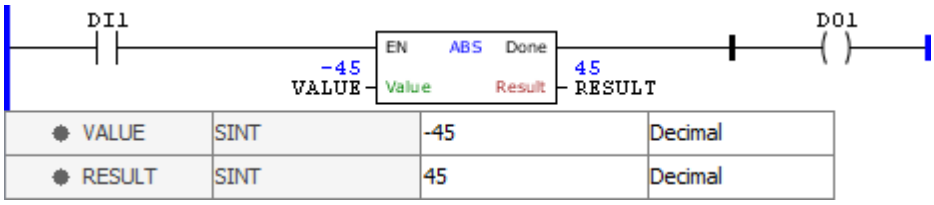
Value variable. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

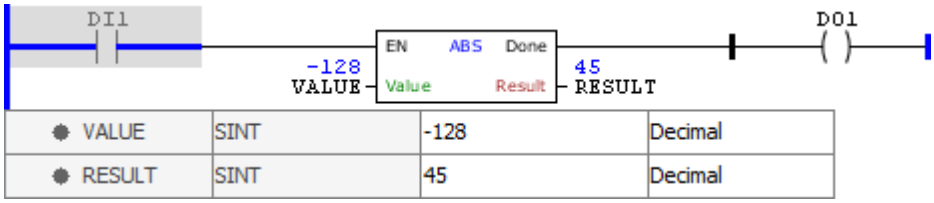


Example



VALUE	SINT	-45	Decimal
RESULT	SINT	45	Decimal

The above example calculates the absolute value of the VALUE variable whose initial value is -45, storing the final result, 45, in RESULT.



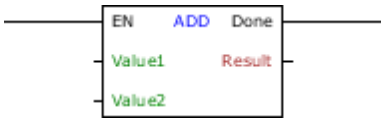
VALUE	SINT	-128	Decimal
RESULT	SINT	45	Decimal

The above example calculates the absolute value of the VALUE variable whose initial value is -45. The final result, 128, cannot be stored in RESULT, because it is outside the limits of accepted values by SINT type. Therefore, RESULT remains unchanged and the output is disabled.

11.1.6.11.1.2 ADD

Block that calculates the sum of the values of Value1 and Value2, storing the result in Result.

Ladder Representation



Block Structure

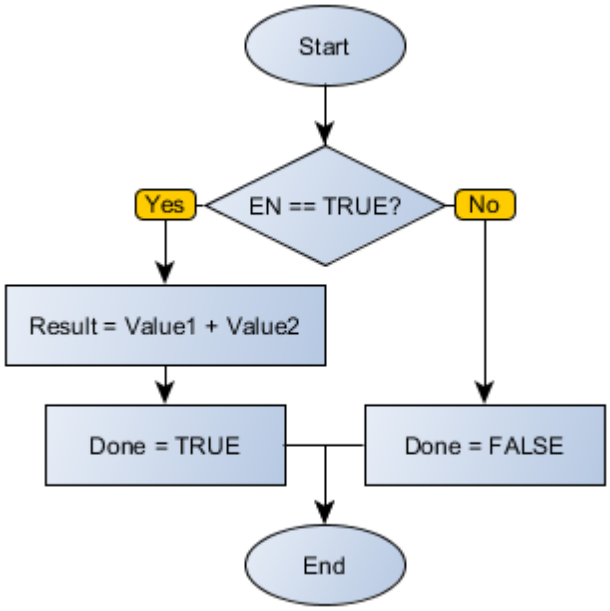
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	First addend of the operation
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Second addend of the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Variable that stores the result of the operation

Operation

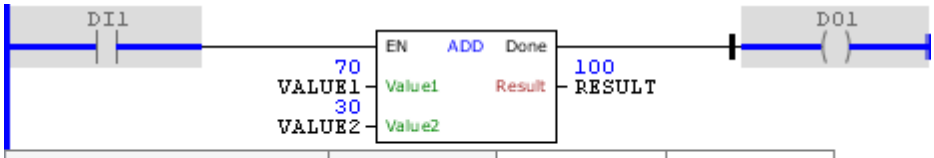
When this block has a TRUE value in EN, it sends to the Result output the sum of Value1 and Value2 variables. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

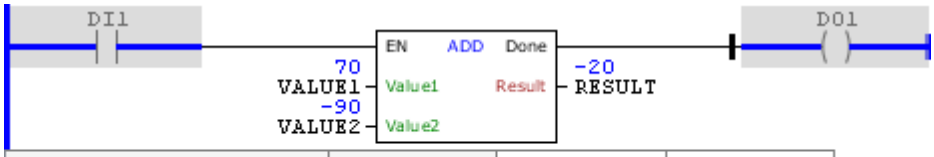


Example



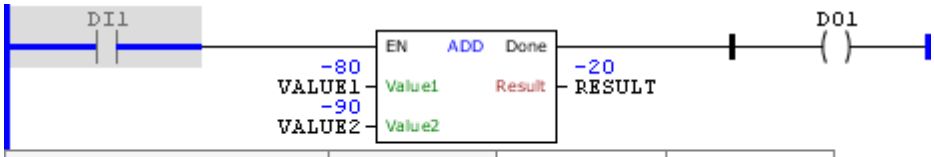
VALUE1	SINT	70	Decimal
VALUE2	SINT	30	Decimal
RESULT	SINT	100	Decimal

The above example calculates the sum of VALUE 1 and VALUE2 variables, storing the final result in RESULT.



VALUE1	SINT	70	Decimal
VALUE2	SINT	-90	Decimal
RESULT	SINT	-20	Decimal

The above example calculates the sum of VALUE 1 and VALUE2 variables, storing the final result in RESULT. Notice that the block accepts arguments of both signs.



VALUE1	SINT	-80	Decimal
VALUE2	SINT	-90	Decimal
RESULT	SINT	-20	Decimal

The above example calculates the sum of VALUE1 and VALUE2 variables. The final result -170 cannot be stored in RESULT, because it is outside the limits of accepted values by SINT type. Therefore, RESULT remains unchanged and the output is disabled.

11.1.6.11.1.3 DIV

Block that calculates the division of the values of Value1 and Value2, storing the result in Result.

Ladder Representation

Block Structure

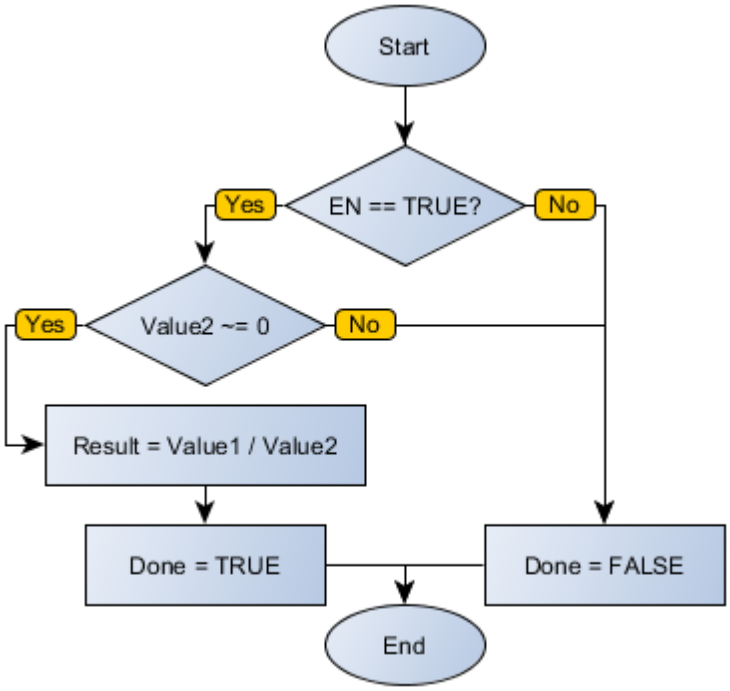
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Dividend of the operation
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Divisor of the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Variable that stores the result of the operation

Operation

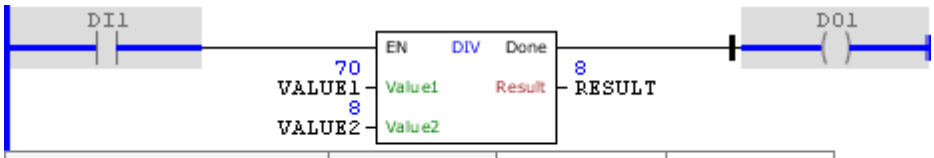
When this block has a TRUE value in EN, it sends to the Result output the division of Value1 and Value2 variables. The value stored will be the exact division if Result is REAL, or, in other cases, only the quotient. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

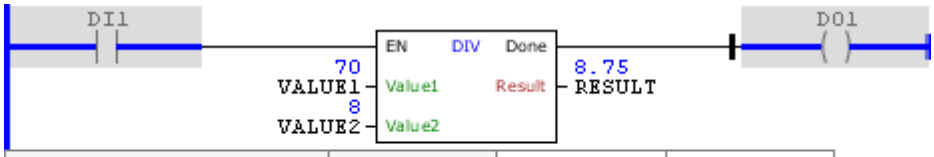


Example



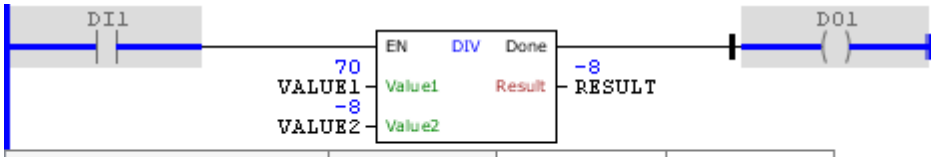
VALUE1	SINT	70	Decimal
VALUE2	SINT	8	Decimal
RESULT	SINT	8	Decimal

The above example calculates the division of VALUE 1 and VALUE2 variables, storing the final result in RESULT. Since RESULT is SINT type, only the quotient is stored in it.



VALUE1	SINT	70	Decimal
VALUE2	SINT	8	Decimal
RESULT	REAL	8.75	Float

The above example calculates the division of VALUE 1 and VALUE2 variables, storing the final result in RESULT. Since RESULT is of REAL type, the exact value of the division is stored in it.



VALUE1	SINT	70	Decimal
VALUE2	SINT	-8	Decimal
RESULT	SINT	-8	Decimal

The above example calculates the division of VALUE 1 and VALUE2 variables, storing the final result in RESULT. Since RESULT is SINT type, only the quotient is stored in it. Notice that the block accepts arguments of both signs.

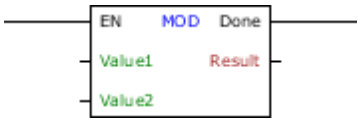
VALUE1	SINT	70	Decimal
VALUE2	SINT	0	Decimal
RESULT	SINT	-8	Decimal

The above example calculates the division of VALUE 1 and VALUE2 variables, storing the final result in RESULT. Since VALUE2 is zero, the block generates a runtime error, RESULT remains unchanged and the output is disabled.

11.1.6.11.1.4 MOD

Block that calculates the remainder of the values of Value1 and Value2, storing the result in Result.

Ladder Representation



Block Structure

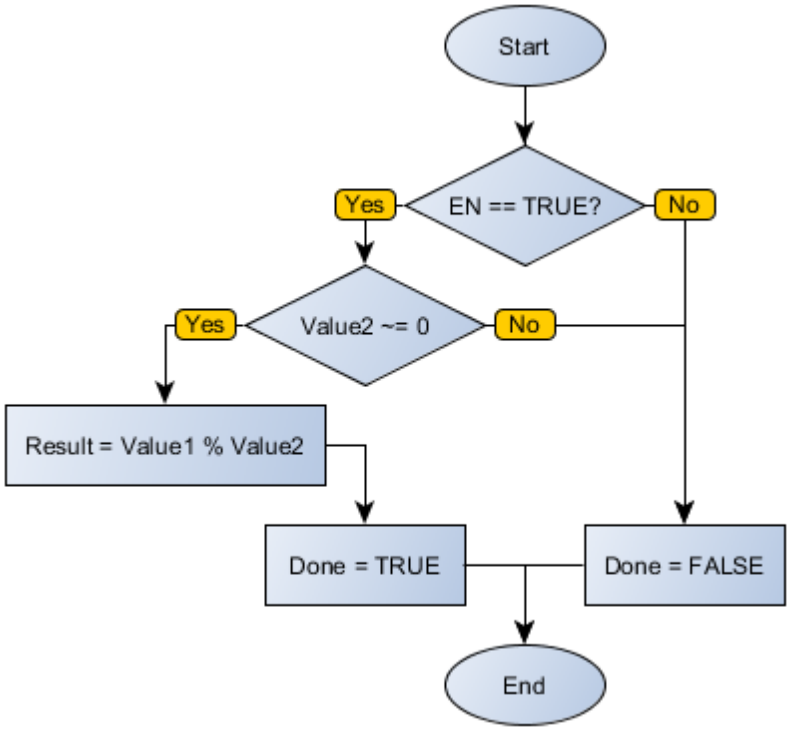
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Dividend of the operation
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Divisor of the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable that stores the result of the operation

Operation

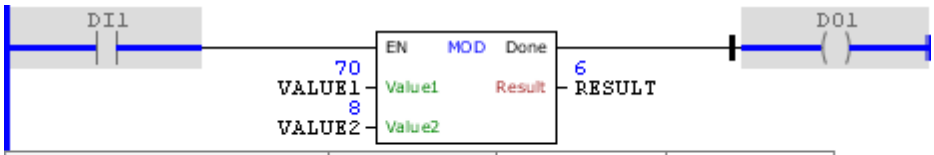
When this block has a TRUE value in EN, it sends to the Result output the remainder of Value1 and Value2 variables. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

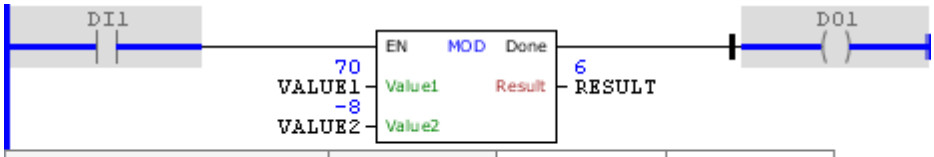


Example



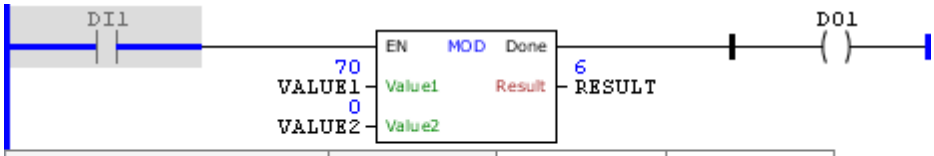
VALUE1	SINT	70	Decimal
VALUE2	SINT	8	Decimal
RESULT	SINT	6	Decimal

The above example calculates the remainder of VALUE 1 and VALUE2 variables, storing the final result in RESULT.



VALUE1	SINT	70	Decimal
VALUE2	SINT	-8	Decimal
RESULT	SINT	6	Decimal

The above example calculates the remainder of VALUE 1 and VALUE2 variables, storing the final result in RESULT. Notice that the block accepts arguments of both signs.



VALUE1	SINT	70	Decimal
VALUE2	SINT	0	Decimal
RESULT	SINT	6	Decimal

The above example calculates the remainder of VALUE 1 and VALUE2 variables, storing the final result in RESULT. Since VALUE2 is zero, the block generates a runtime error, RESULT remains unchanged and the output is disabled.

11.1.6.11.1.5 MUL

Block that calculates the multiplication of the values of Value1 and Value2, storing the result in Result.

Ladder Representation

Block Structure

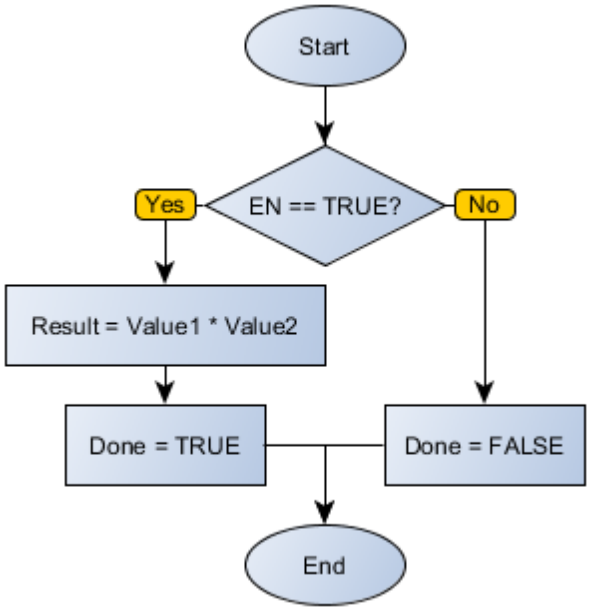
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	First factor of the operation
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Second factor of the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Variable that stores the result of the operation

Operation

When this block has a TRUE value in EN, it sends to the Result output the multiplication of Value1 and Value2 variables. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

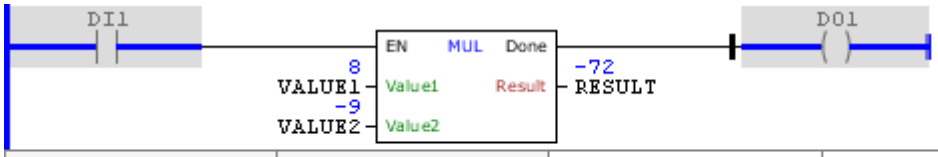
Block Flowchart



Example

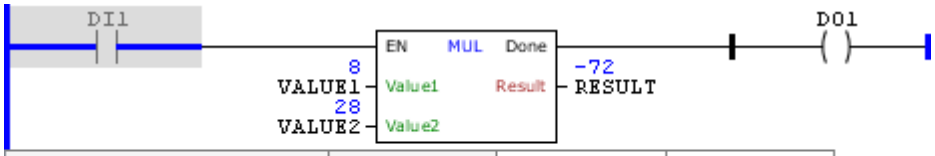
VALUE1	SINT	8	Decimal
VALUE2	SINT	12	Decimal
RESULT	SINT	96	Decimal

The above example calculates the product of VALUE 1 and VALUE2 variables, storing the final result in RESULT.



VALUE1	SINT	8	Decimal
VALUE2	SINT	-9	Decimal
RESULT	SINT	-72	Decimal

The above example calculates the product of VALUE 1 and VALUE2 variables, storing the final result in RESULT. Notice that the block accepts arguments of both signs.



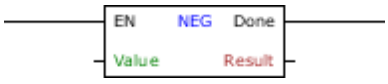
VALUE1	SINT	8	Decimal
VALUE2	SINT	28	Decimal
RESULT	SINT	-72	Decimal

The above example calculates the product of VALUE1 and VALUE2 variables. The final result 224 cannot be stored in RESULT, because it is outside the limits of accepted values by SINT type. Therefore, RESULT remains unchanged and the output is disabled.

11.1.6.11.1.6 NEG

Block that calculates the opposite (i.e., the product with -1) of a value passed by Value, storing the result in Result.

Ladder Representation



Block Structure

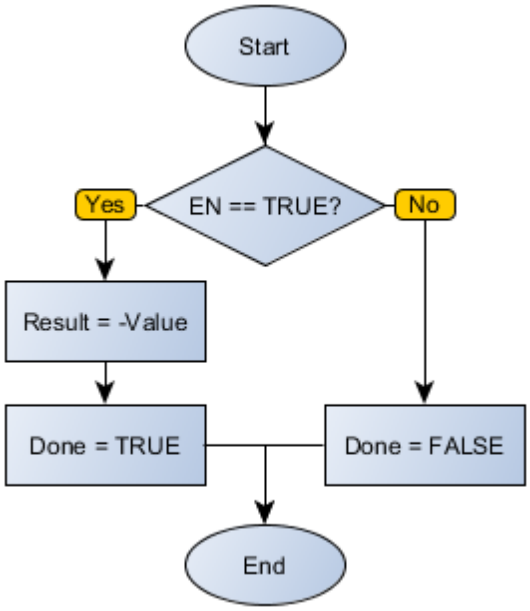
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Reference variable for the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Variable that stores the result of the operation

Operation

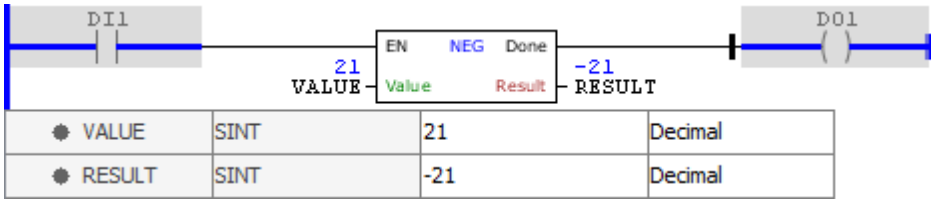
When this block has a TRUE value in EN, it sends to the Result output the opposite of the Value variable. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

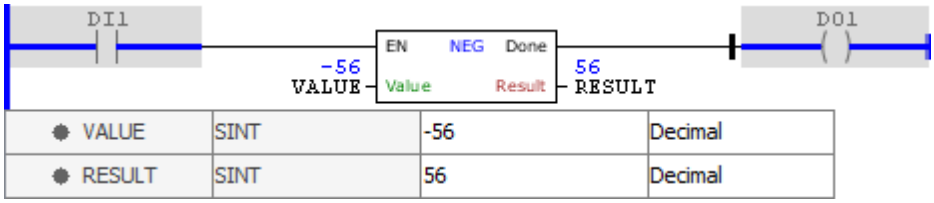


Example



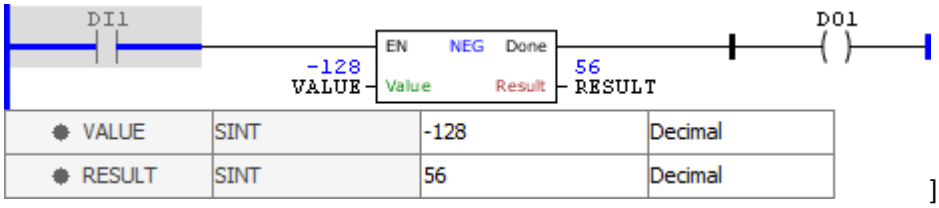
VALUE	SINT	21	Decimal
RESULT	SINT	-21	Decimal

The above example calculates the opposite of the VALUE variable whose initial value is 21, storing the final result, -21, in RESULT.



VALUE	SINT	-56	Decimal
RESULT	SINT	56	Decimal

The above example calculates the opposite of the VALUE variable whose initial value is -56, storing the final result, 56, in RESULT.



VALUE	SINT	-128	Decimal
RESULT	SINT	56	Decimal

]

The above example calculates the opposite of the VALUE variable whose initial value is -128. The final result, 128, cannot be stored in RESULT, because it is outside the limits of accepted values by SINT type. Therefore, RESULT remains unchanged and the output is disabled.

11.1.6.11.1.7 SUB

Block that calculates the subtraction between the Value1 and Value2 values, storing the result in Result.

Ladder Representation

Block Structure

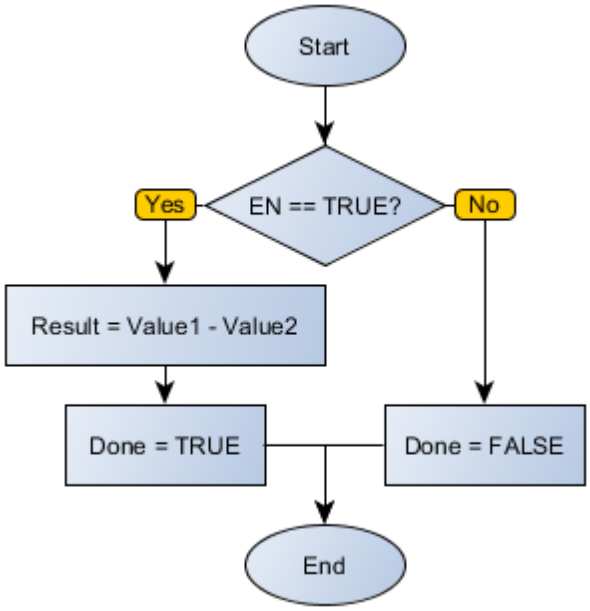
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Minuend of operation
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Subtrahend of operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Variable that stores the result of the operation

Operation

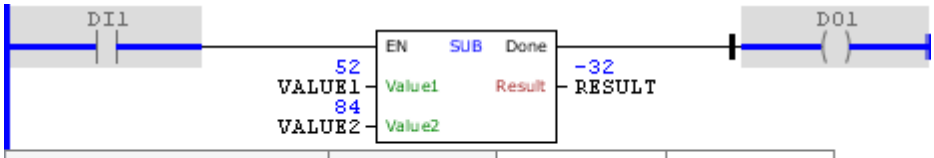
When this block has a TRUE value in EN, it sends to the Result output the subtraction of Value1 and Value2 variables. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

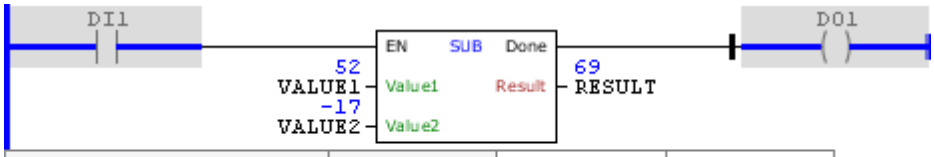


Example



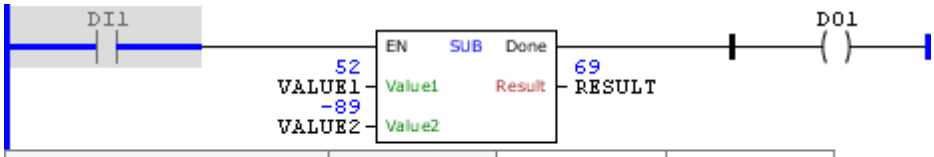
VALUE1	SINT	52	Decimal
VALUE2	SINT	84	Decimal
RESULT	SINT	-32	Decimal

The above example calculates the subtraction of VALUE 1 and VALUE2 variables, storing the final result in RESULT.



VALUE1	SINT	52	Decimal
VALUE2	SINT	-17	Decimal
RESULT	SINT	69	Decimal

The above example calculates the subtraction of VALUE 1 and VALUE2 variables, storing the final result in RESULT. Notice that the block accepts arguments of both signs.



VALUE1	SINT	52	Decimal
VALUE2	SINT	-89	Decimal
RESULT	SINT	69	Decimal

The above example calculates the subtraction of VALUE1 and VALUE2 variables. The final result 141 cannot be stored in RESULT, because it is outside the limits of accepted values by SINT type. Therefore, RESULT remains unchanged and the output is disabled.

11.1.6.11.2 Math Extended

11.1.6.11.2.1 ALOG10

Block that calculates the antilogarithm (exponent with base 10) of the Value value, storing the result in Result.

Ladder Representation

Block Structure

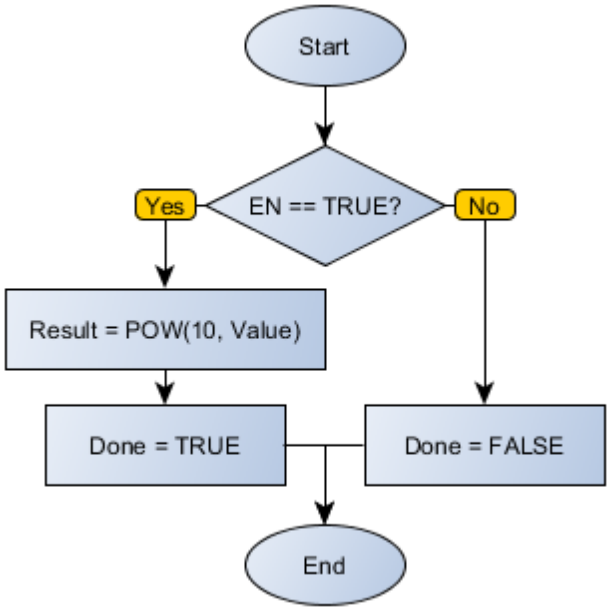
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Reference variable for the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	REAL	Variable that stores the result of the operation

Operation

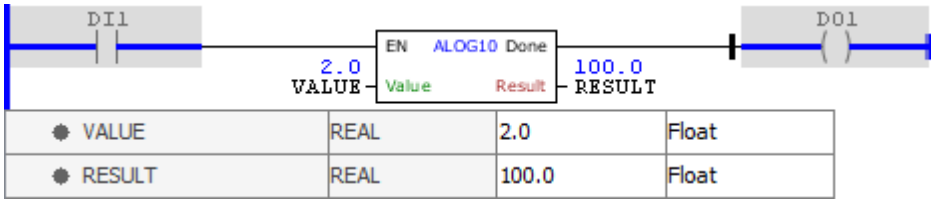
When this block has a TRUE value in EN, it sends to the Result output the antilogarithm of the Value variable. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

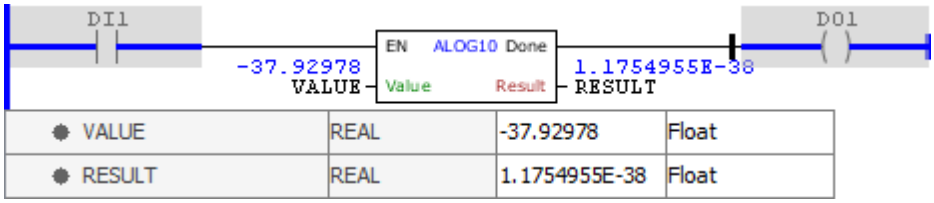


Example



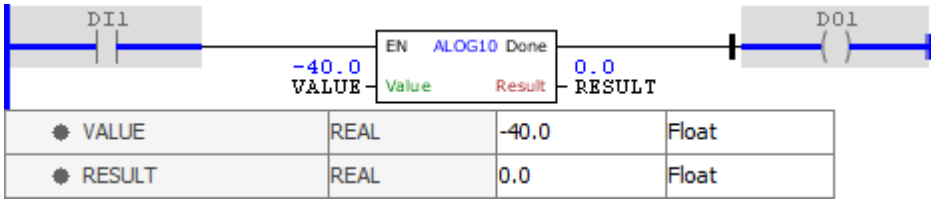
VALUE	REAL	2.0	Float
RESULT	REAL	100.0	Float

The above example calculates the antilogarithm of the VALUE variable, storing the final result in RESULT. The block ends with success and Done output is activated.



VALUE	REAL	-37.92978	Float
RESULT	REAL	1.1754955E-38	Float

The above example calculates the antilogarithm of the VALUE variable, storing the final result in RESULT. The indicated value is the minimum input value for which the block revolutions a nonzero result. The block ends with success and Done output is activated.



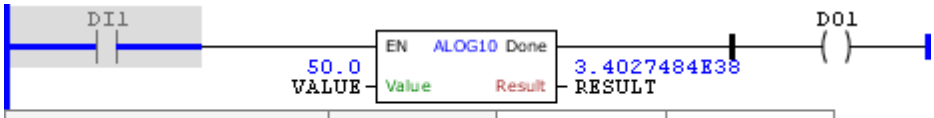
VALUE	REAL	-40.0	Float
RESULT	REAL	0.0	Float

The above example calculates the antilogarithm of the VALUE variable, storing the final result in

RESULT. Below the minimum values cause the block to return a null value. The block ends with success and Done output is activated.

VALUE	REAL	38.53183	Float
RESULT	REAL	3.4027484E38	Float

The above example calculates the antilogarithm of the VALUE variable, storing the final result in RESULT. The indicated value is the maximum input value for which the block revolutions a valid result. The block ends with success and Done output is activated.



VALUE	REAL	50.0	Float
RESULT	REAL	3.4027484E38	Float

The above example calculates the antilogarithm of the VALUE variable, storing the final result in RESULT. Values higher than the maximum cause the block to generate an error, the RESULT output remains unchanged and Done output is disabled.

11.1.6.11.2.2 EXP

Block that calculates the exponential of the Euler number "and" raised to the value of Value, storing the result in Result.

Ladder Representation

Block Structure

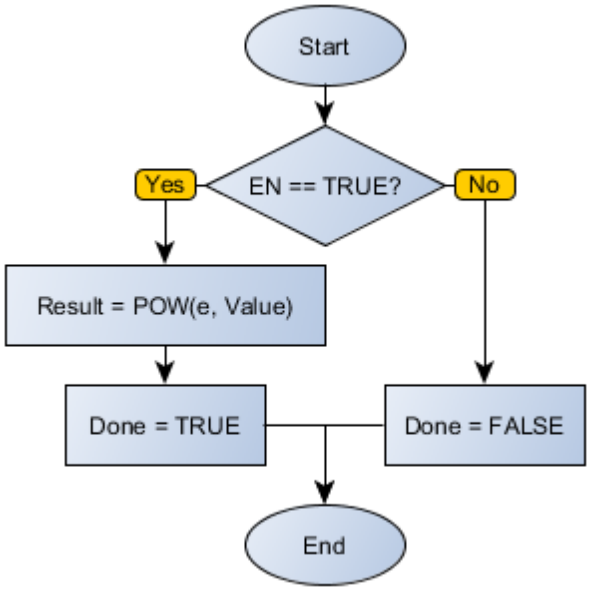
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Reference variable for the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	REAL	Variable that stores the result of the operation

Operation

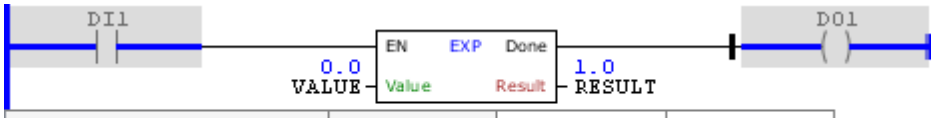
When this block has a TRUE value in EN, it sends to the Result output the exponent of the Euler number "and" raised to the Value variable. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart



Example



VALUE	REAL	0.0	Float
RESULT	REAL	1.0	Float

The above example calculates the exponent of the VALUE variable, storing the final result in RESULT. The block ends with success and Done output is activated.

VALUE	REAL	-87.33654	Float
RESULT	REAL	1.1754997E-38	Float

The above example calculates the exponent of the VALUE variable, storing the final result in RESULT. The indicated value is the minimum input value for which the block revolutions a nonzero result. The block ends with success and Done output is activated.

VALUE	REAL	-50000.0	Float
RESULT	REAL	0.0	Float

The above example calculates the exponent of the VALUE variable, storing the final result in RESULT. Values below the minimum cause the block to return to a null value. The block ends with success and Done output is activated.

VALUE	REAL	88.72283	Float
RESULT	REAL	3.4027985E38	Float

The above example calculates the exponent of the VALUE variable, storing the final result in RESULT. The indicated value is the maximum input value for which the block revolutions a valid result. The block ends with success and Done output is activated.

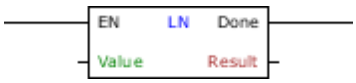
VALUE	REAL	90.0	Float
RESULT	REAL	3.4027985E38	Float

The above example calculates the exponent of the VALUE variable, storing the final result in RESULT. Values higher than the maximum cause the block to generate an error, the RESULT output remains unchanged and Done output is disabled.

11.1.6.11.2.3 LN

Block that calculates the natural logarithm of the Value value, storing the result in Result.

Ladder Representation



Block Structure

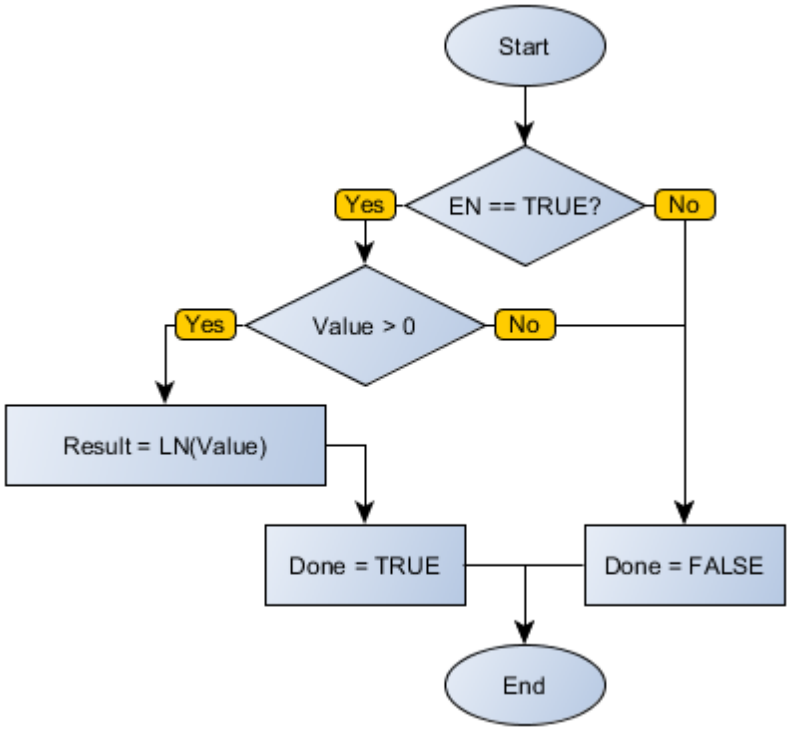
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Reference variable for the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	REAL	Variable that stores the result of the operation

Operation

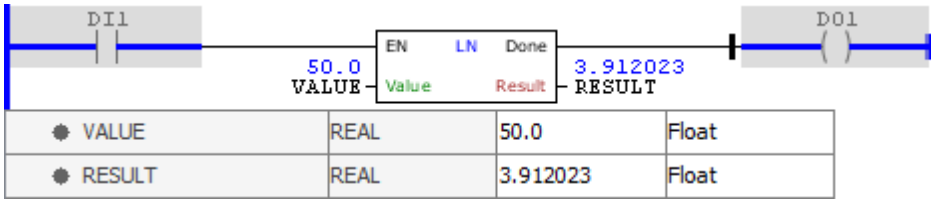
When this block has a TRUE value in EN, it sends to the Result output the natural logarithm of the Value variable. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart



Example



VALUE	REAL	50.0	Float
RESULT	REAL	3.912023	Float

The above example calculates the natural logarithm of the VALUE variable, storing the final result in RESULT. The block ends with success and Done output is activated.

VALUE	REAL	0.0	Float
RESULT	REAL	3.912023	Float

The above example calculates the natural logarithm of the VALUE variable, storing the final result in RESULT. The block generates a runtime error, since VALUE has value zero, and Done output is disabled.

11.1.6.11.2.4 LOG10

Block that calculates the common logarithm (base 10) of the Value value, storing the result in Result.

Ladder Representation

Block Structure

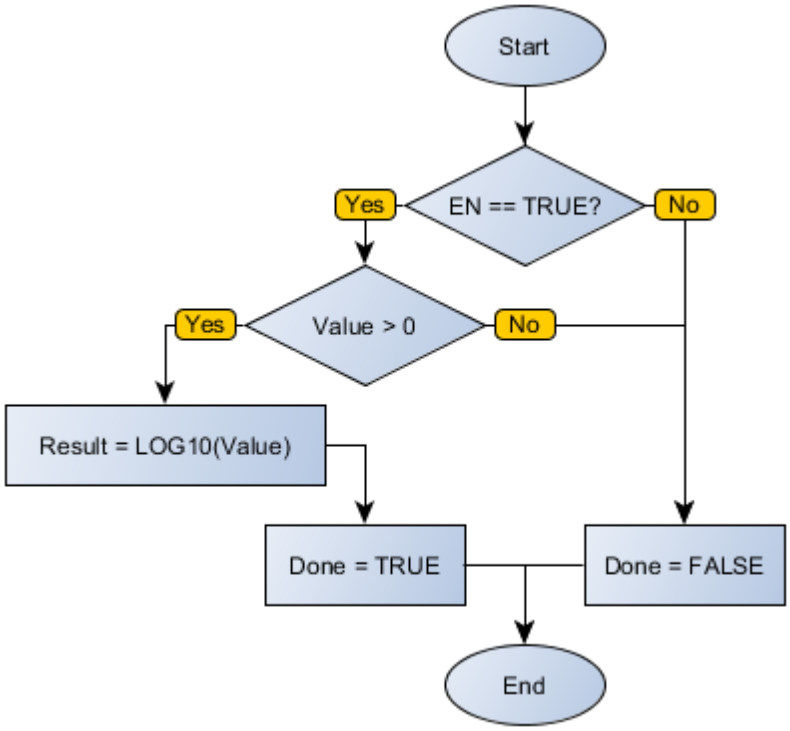
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Reference variable for the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	REAL	Variable that stores the result of the operation

Operation

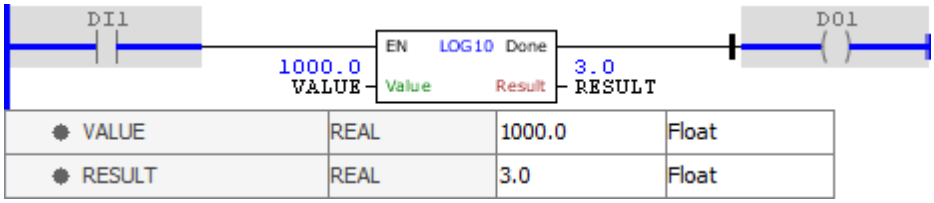
When this block has a TRUE value in EN, it sends to the Result output the common logarithm of the Value variable. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

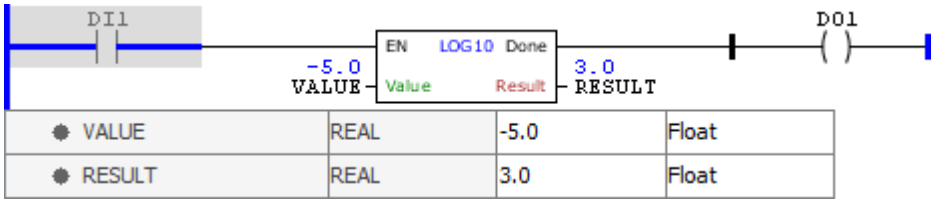


Example



VALUE	REAL	1000.0	Float
RESULT	REAL	3.0	Float

The above example calculates the common logarithm of the VALUE variable, storing the final result in RESULT. The block ends with success and Done output is activated.



VALUE	REAL	-5.0	Float
RESULT	REAL	3.0	Float

The above example calculates the common logarithm of the VALUE variable, storing the final result in RESULT. The block generates a runtime error, since VALUE has negative value, and Done output is disabled.

11.1.6.11.2.5 POW

Block that calculates the value of Value raised to the exponent Power, storing the result in Result.

Ladder Representation

Block Structure

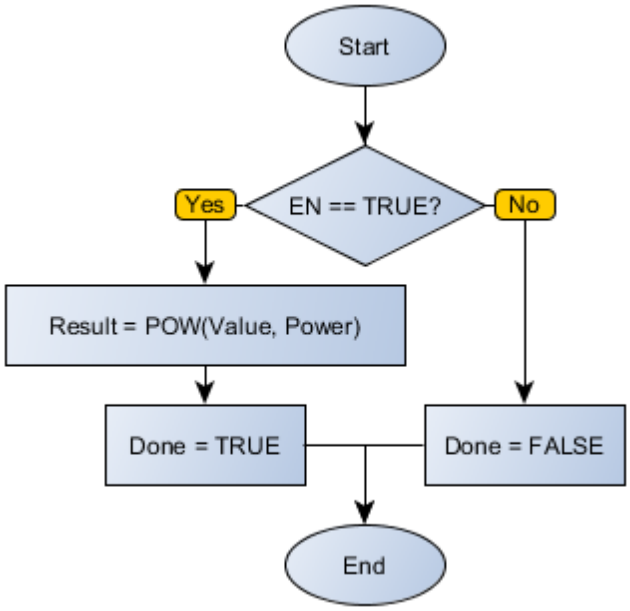
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Base of the operation
	Power	REAL	Exponent of the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	REAL	Variable that stores the result of the operation

Operation

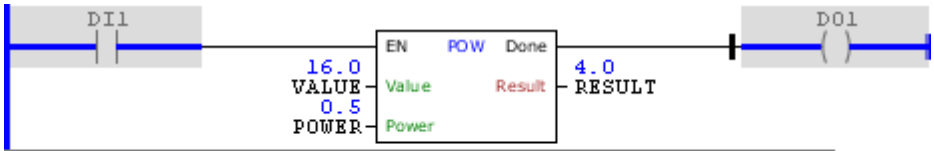
When this block has a TRUE value in EN, it sends to the Result output the value of Value raised to the exponent Power. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

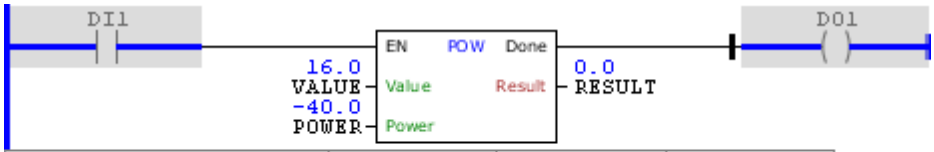


Example



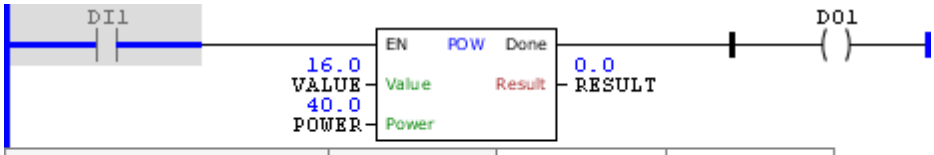
VALUE	REAL	16.0	Float
POWER	REAL	0.5	Float
RESULT	REAL	4.0	Float

The above example calculates the value of VALUE raised to the POWER variable, storing the final result in RESULT. The block ends with success and Done output is activated.



VALUE	REAL	16.0	Float
POWER	REAL	-40.0	Float
RESULT	REAL	0.0	Float

The above example calculates the value of VALUE raised to the POWER variable, storing the final result in RESULT. The block ends with success and Done output is activated.



VALUE	REAL	16.0	Float
POWER	REAL	40.0	Float
RESULT	REAL	0.0	Float

The above example calculates the value of VALUE raised to the POWER variable, storing the final result in RESULT. Since the result is higher than the maximum supported by REAL type, the block generates an error and Done output is disabled.

11.1.6.11.2.6 ROUND

Block that rounds the value of Value, storing the result in Result.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Reference variable for the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	REAL	Variable that stores the result of the operation

Operation

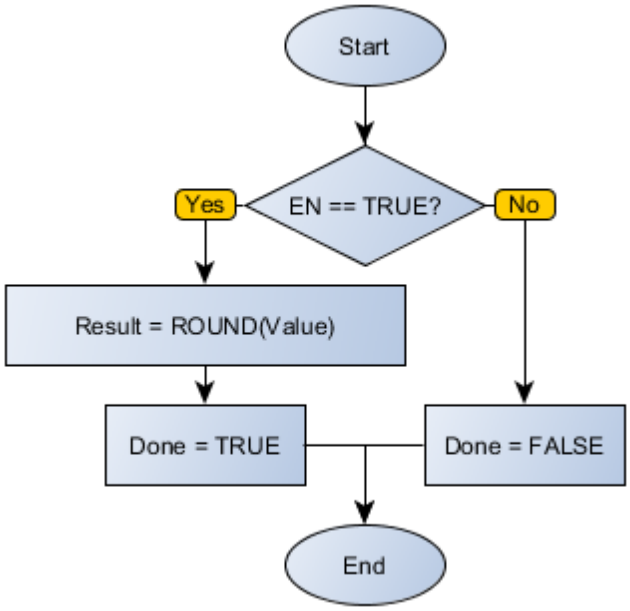
When this block has a TRUE value in EN, it sends to the Result output the rounded value of Value. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

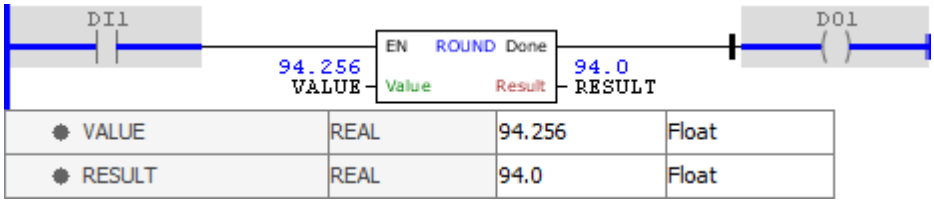
Compatibility

Device	Version
PLC300	2.10 or higher
SCA06	2.00 or higher

Block Flowchart

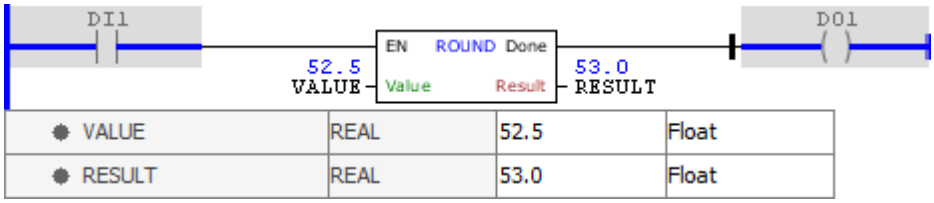


Example



VALUE	REAL	94.256	Float
RESULT	REAL	94.0	Float

The above example rounds the value of the VALUE variable, storing the final result in RESULT. Decimals less than 0.5 are discarded. The block ends with success and Done output is activated.



VALUE	REAL	52.5	Float
RESULT	REAL	53.0	Float

The above example rounds the value of the VALUE variable, storing the final result in RESULT. Decimals greater than or equal to 0.5 promote unity value immediately above. The block ends with success and Done output is activated.

11.1.6.11.2.7 SQRT

Block that calculates the square root value of Value, storing the result in Result.

Ladder Representation

Block Structure

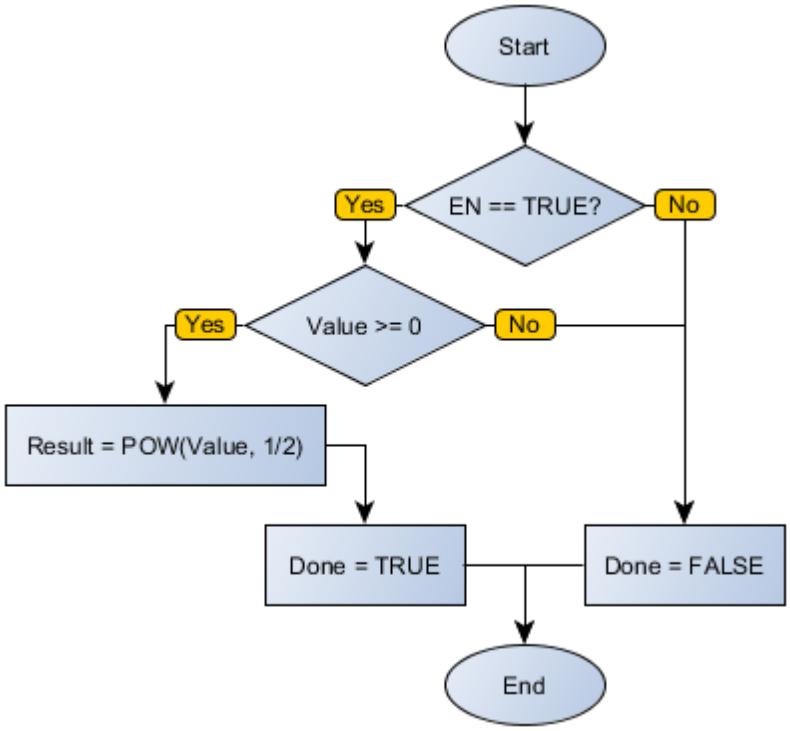
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Reference variable for the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	REAL	Variable that stores the result of the operation

Operation

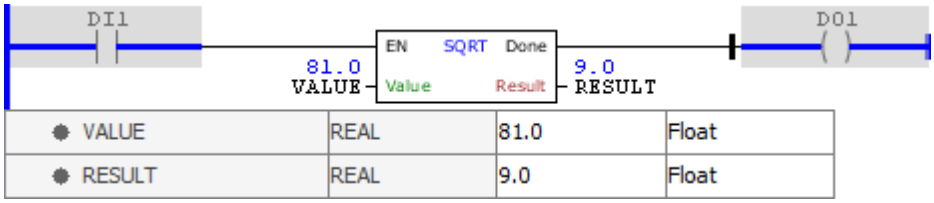
When this block has a TRUE value in EN, it sends to the Result output the square root value of Value. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

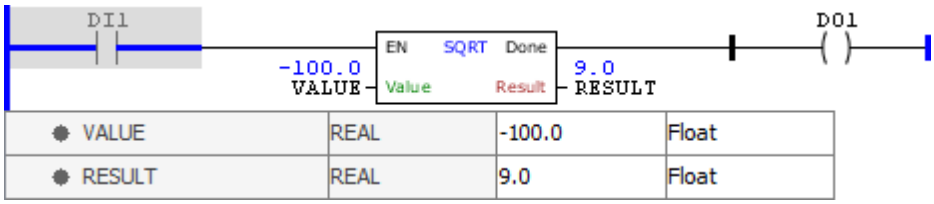


Example



VALUE	REAL	81.0	Float
RESULT	REAL	9.0	Float

The above example calculates the square root value of the VALUE variable, storing the final result in RESULT. The block ends with success and Done output is activated.



VALUE	REAL	-100.0	Float
RESULT	REAL	9.0	Float

The above example calculates the square root value of the VALUE variable, storing the final result in RESULT. The block generates a runtime error, since VALUE has negative value, and Done output is disabled.

11.1.6.11.2.8 TRUNC

Block that truncates the value of Value, storing the result in Result.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Reference variable for the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	REAL	Variable that stores the result of the operation

Operation

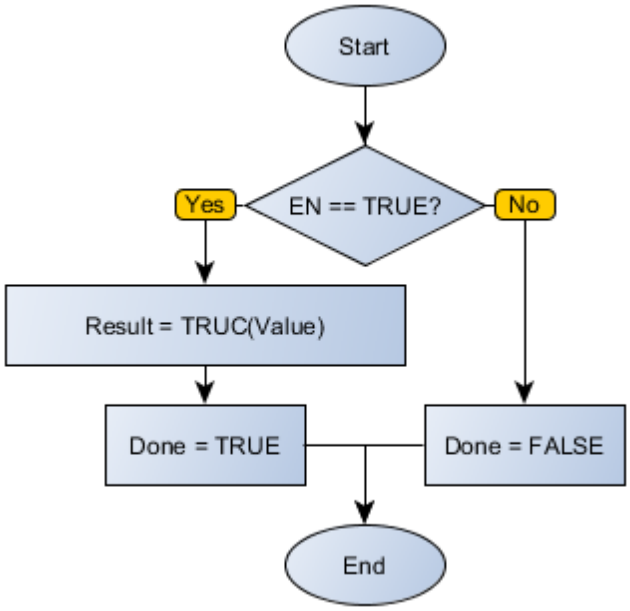
When this block has a TRUE value in EN, it sends to the Result output the truncated value of Value. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

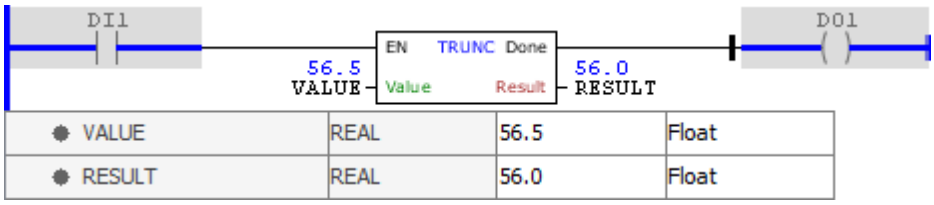
Compatibility

Device	Version
PLC300	2.10 or higher
SCA06	2.00 or higher

Block Flowchart



Example



VALUE	REAL	56.5	Float
RESULT	REAL	56.0	Float

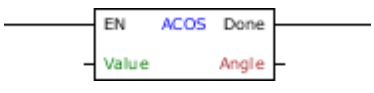
The above example truncates the value of the VALUE variable, storing the final result in RESULT. Decimals are discarded. The block ends with success and Done output is activated.

11.1.6.11.3 Math Trigonometry

11.1.6.11.3.1 ACOS

Block that calculates the arccosine of Value, storing the result in Angle.

Ladder Representation



Block Structure

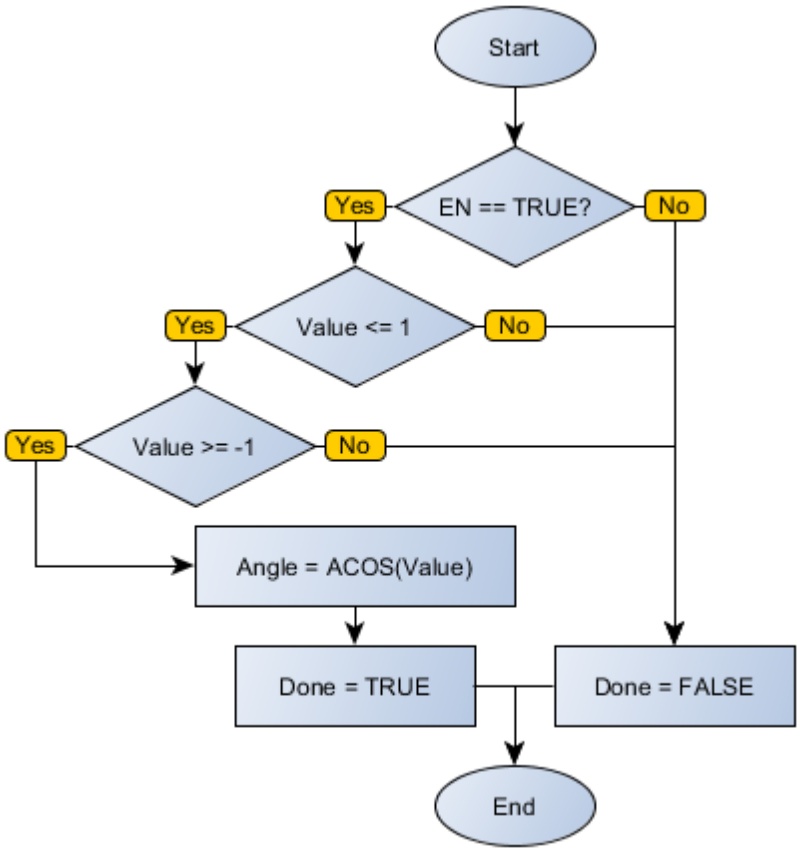
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Value of cosine
VAR_OUTPUT	Done	BOOL	End of operation
	Angle	REAL	Value of the angle w hose cosine is equal to Value (in radians)

Operation

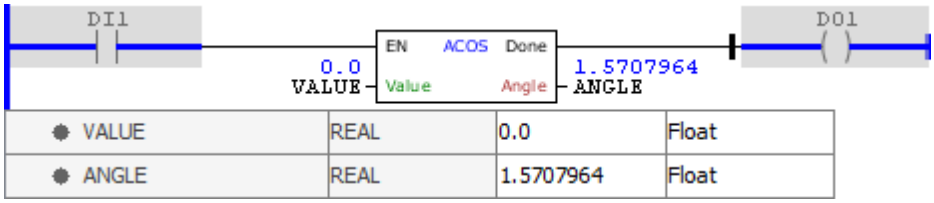
When this block has a TRUE value in EN, it sends to the Angle output the arccosine of Value. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Angle remains with its value unchanged.

When EN has FALSE value, Angle remains unchanged and Done remains in FALSE.

Block Flowchart

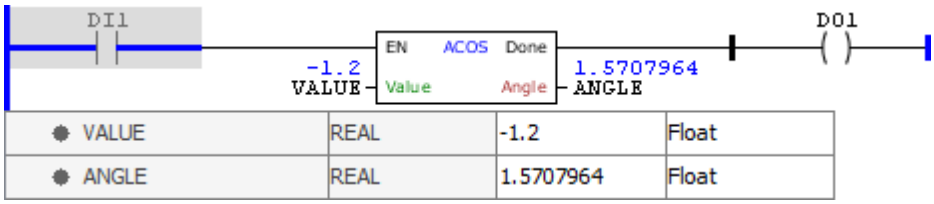


Example



VALUE	REAL	0.0	Float
ANGLE	REAL	1.5707964	Float

The above example calculates the arc, in radians, whose cosine is the VALUE variable, storing the final result in RESULT. The block ends with success and Done output is activated.



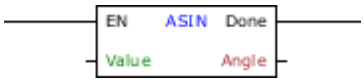
VALUE	REAL	-1.2	Float
ANGLE	REAL	1.5707964	Float

The above example calculates the arc, in radians, whose cosine is the VALUE variable, storing the final result in RESULT. The block generates a runtime error, since VALUE has value inferior to 1, and Done output is disabled.

11.1.6.11.3.2 ASIN

Block that calculates the arcsine of Value, storing the result in Angle.

Ladder Representation



Block Structure

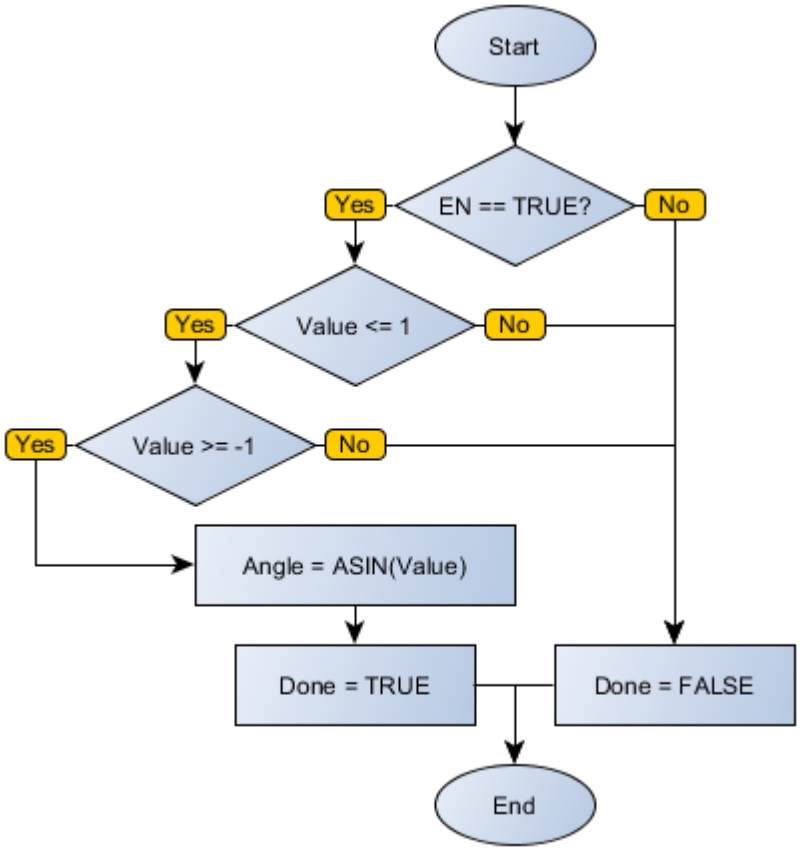
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Value of sine
VAR_OUTPUT	Done	BOOL	End of operation
	Angle	REAL	Value of the angle whose sine is equal to Value (in radians)

Operation

When this block has a TRUE value in EN, it sends to the Angle output the arcsine of Value. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Angle remains with its value unchanged.

When EN has FALSE value, Angle remains unchanged and Done remains in FALSE.

Block Flowchart



Example

VALUE	REAL	-1.0	Float
ANGLE	REAL	-1.5707964	Float

The above example calculates the arc, in radians, whose sine is the VALUE variable, storing the final result in RESULT. The block ends with success and Done output is activated.

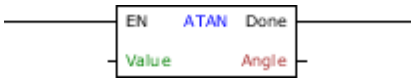
VALUE	REAL	2.0	Float
ANGLE	REAL	-1.5707964	Float

The above example calculates the arc, in radians, whose sine is the VALUE variable, storing the final result in RESULT. The block generates a runtime error, since VALUE has value superior to 1, and Done output is disabled.

11.1.6.11.3.3 ATAN

Block that calculates the arctangent of Value, storing the result in Angle.

Ladder Representation



Block Structure

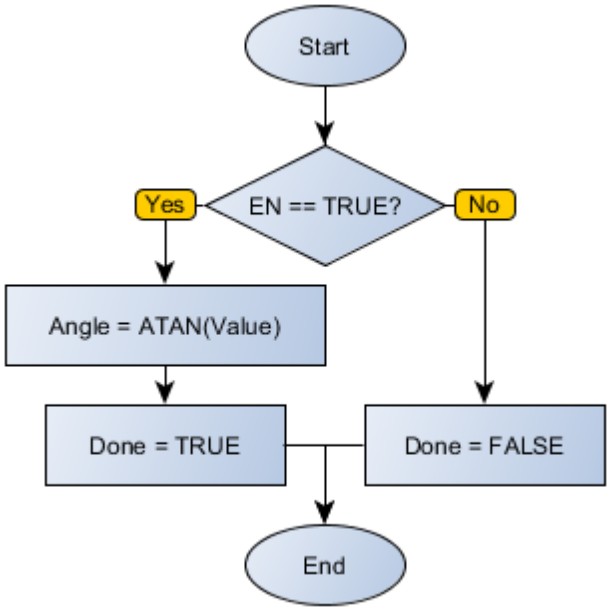
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Value of tangent
VAR_OUTPUT	Done	BOOL	End of operation
	Angle	REAL	Value of the angle whose tangent is equal to Value (in radians)

Operation

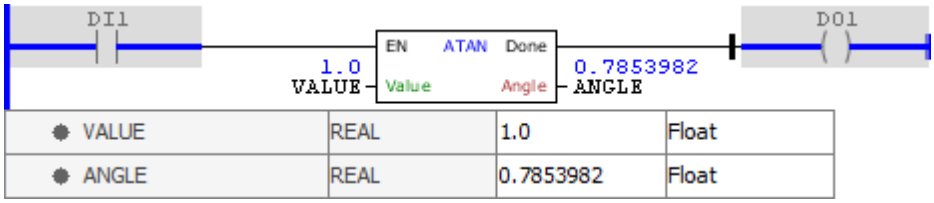
When this block has a TRUE value in EN, it sends to the Angle output the arctangent of Value. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Angle remains with its value unchanged.

When EN has FALSE value, Angle remains unchanged and Done remains in FALSE.

Block Flowchart

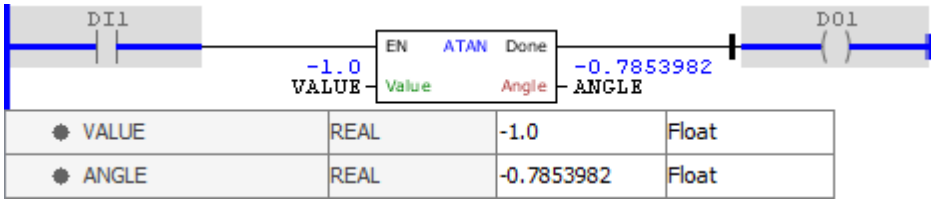


Example



VALUE	REAL	1.0	Float
ANGLE	REAL	0.7853982	Float

The above example calculates the arc, in radians, whose tangent is the VALUE variable, storing the final result in RESULT. The arc, for positive values, is always in the first quadrant. The block ends with success and Done output is activated.



VALUE	REAL	-1.0	Float
ANGLE	REAL	-0.7853982	Float

The above example calculates the arc, in radians, whose tangent is the VALUE variable, storing the final result in RESULT. The arc, for negative values, is always in the fourth quadrant. The block ends with success and Done output is activated.

11.1.6.11.3.4 ATAN2

Block that calculates the arctangent of Y/X, storing the result in Angle.

Ladder Representation

Block Structure

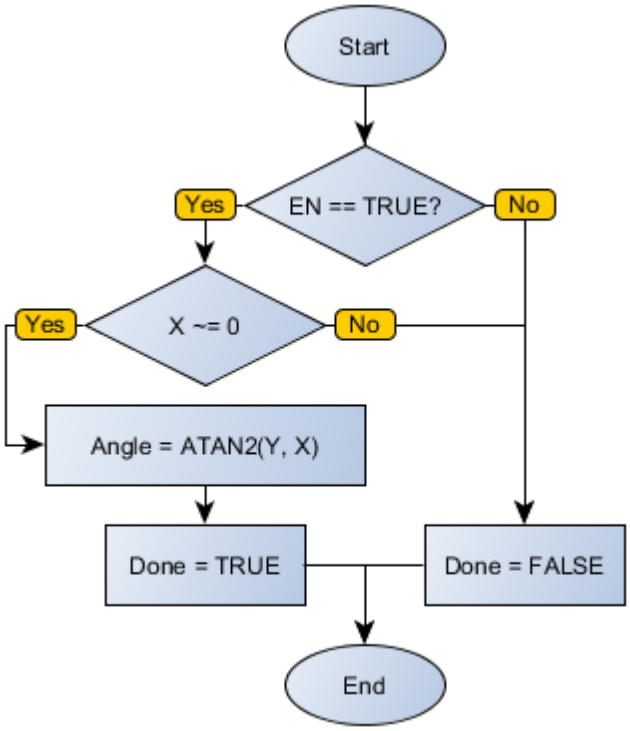
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	X	REAL	Parameter X of the function
	Y	REAL	Parameter Y of the function
VAR_OUTPUT	Done	BOOL	End of operation
	Angle	REAL	Value of the angle whose tangent is equal to (Y/X) (in radians)

Operation

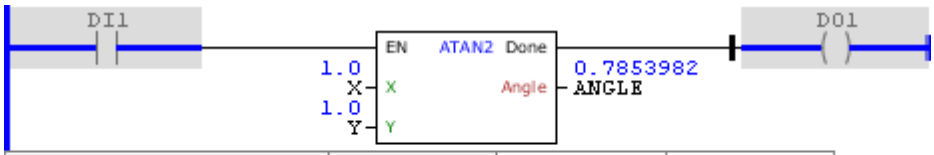
When this block has a TRUE value in EN, it sends to the Angle output the arctangent of Y/X. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Angle remains with its value unchanged.

When EN has FALSE value, Angle remains unchanged and Done remains in FALSE.

Block Flowchart



Example



X	REAL	1.0	Float
Y	REAL	1.0	Float
ANGLE	REAL	0.7853982	Float

The above example calculates the arc, in radians, whose tangent is the Y/X variable, storing the final result in RESULT. The arc, for positive values of X and Y, is always in the first quadrant. The block ends with success and Done output is activated.

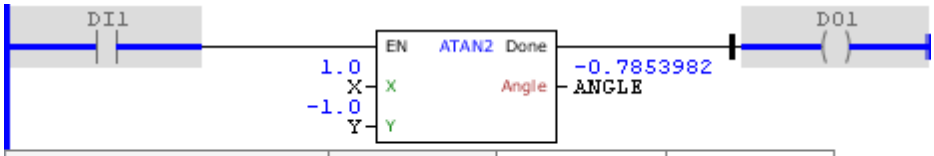
X	REAL	-1.0	Float
Y	REAL	1.0	Float
ANGLE	REAL	2.3561945	Float

The above example calculates the arc, in radians, whose tangent is the Y/X variable, storing the final

result in RESULT. The arc, for negative values of X and positive values of Y, is always in the second quadrant. The block ends with success and Done output is activated.

X	REAL	-1.0	Float
Y	REAL	-1.0	Float
ANGLE	REAL	-2.3561945	Float

The above example calculates the arc, in radians, whose tangent is the Y/X variable, storing the final result in RESULT. The arc, for negative values of X and Y, is always in the third quadrant. The block ends with success and Done output is activated.



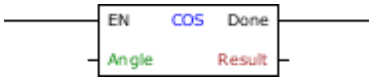
X	REAL	1.0	Float
Y	REAL	-1.0	Float
ANGLE	REAL	-0.7853982	Float

The above example calculates the arc, in radians, whose tangent is the Y/X variable, storing the final result in RESULT. The arc, for positive values of X and negative values of Y, is always in the fourth quadrant. The block ends with success and Done output is activated.

11.1.6.11.3.5 COS

Block that calculates the cosine of Angle, storing the result in Result.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Angle	REAL	Angle (in radians)
VAR_OUTPUT	Done	BOOL	End of operation
	Result	REAL	Variable that stores the result of the operation

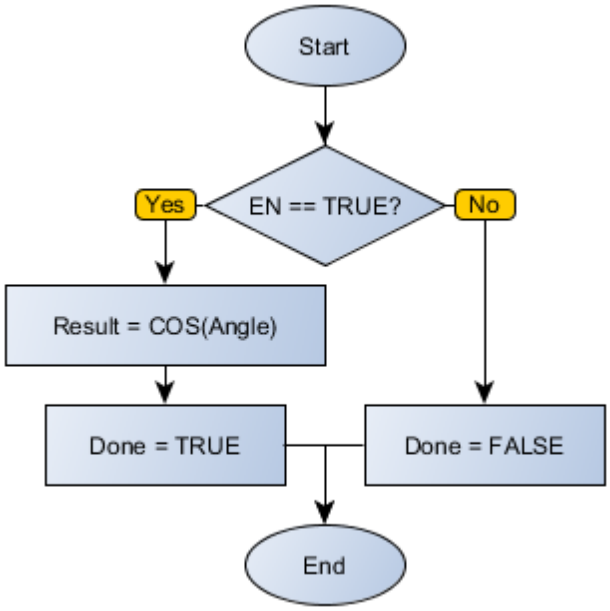
Operation

When this block has a TRUE value in EN, it sends to the Result output the cosine of Angle. If no

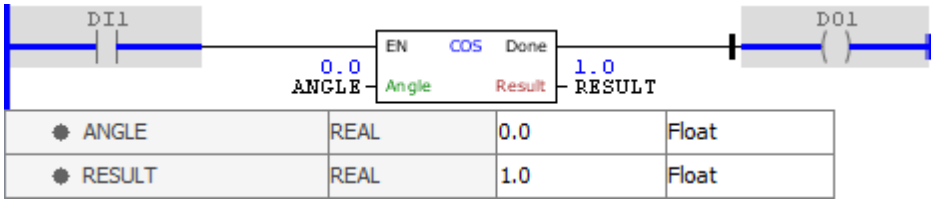
errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

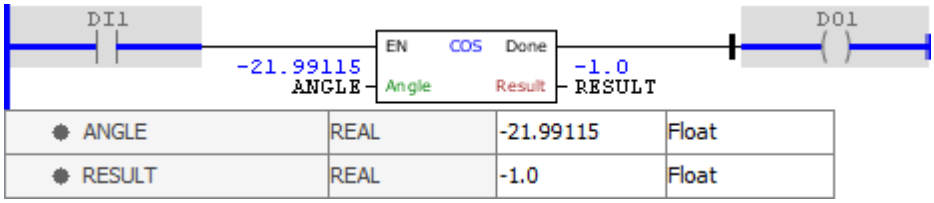


Example



ANGLE	REAL	0.0	Float
RESULT	REAL	1.0	Float

The above example calculates the cosine of the VALUE variable, interpreted in radians, storing the final result in RESULT. The block ends with success and Done output is activated.



ANGLE	REAL	-21.99115	Float
RESULT	REAL	-1.0	Float

The above example calculates the cosine of the VALUE variable, interpreted in radians, storing the final result in RESULT. The block ends with success and Done output is activated. Notice that the block accepts negative input values and greater than one turn.

11.1.6.11.3.6 SIN

Block that calculates the sine of Angle, storing the result in Result.

Ladder Representation

Block Structure

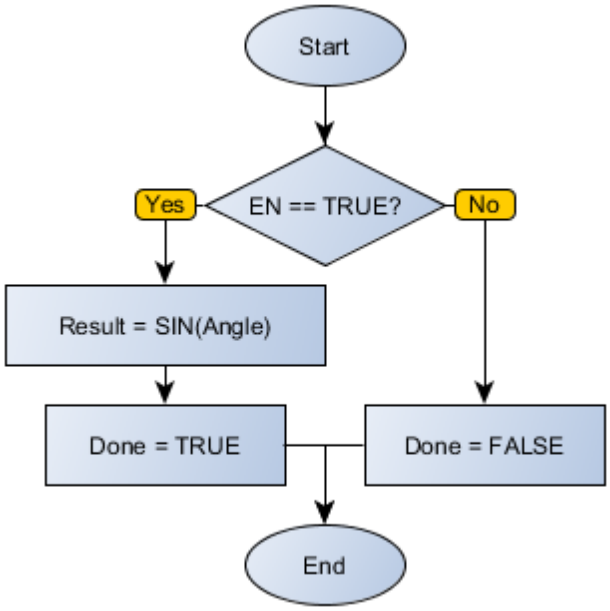
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Angle	REAL	Angle (in radians)
VAR_OUTPUT	Done	BOOL	End of operation
	Result	REAL	Variable that stores the result of the operation

Operation

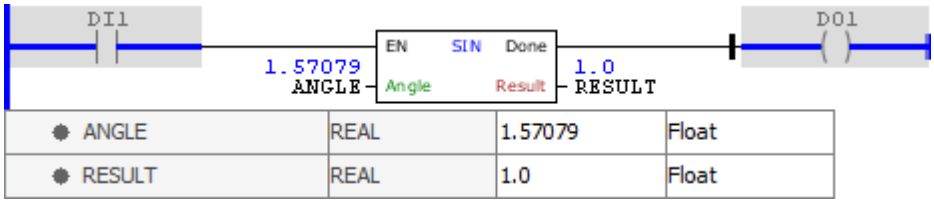
When this block has a TRUE value in EN, it sends to the Result output the sine of Angle. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

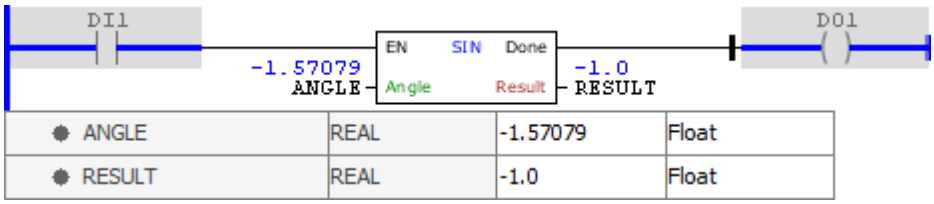


Example



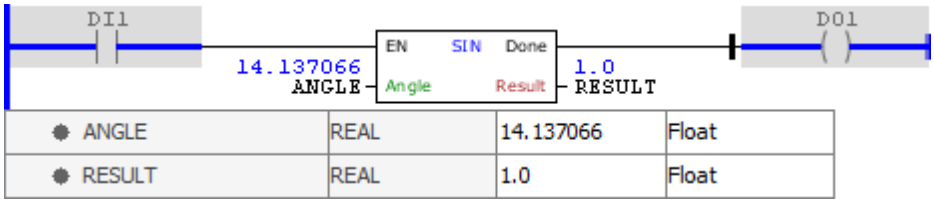
ANGLE	REAL	1.57079	Float
RESULT	REAL	1.0	Float

The above example calculates the sine of the VALUE variable, interpreted in radians, storing the final result in RESULT. The block ends with success and Done output is activated.



ANGLE	REAL	-1.57079	Float
RESULT	REAL	-1.0	Float

The above example calculates the sine of the VALUE variable, interpreted in radians, storing the final result in RESULT. The block ends with success and Done output is activated. Notice that the block accepts negative input values.



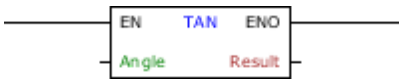
ANGLE	REAL	14.137066	Float
RESULT	REAL	1.0	Float

The above example calculates the sine of the VALUE variable, interpreted in radians, storing the final result in RESULT. The block ends with success and Done output is activated. Notice that the block accepts values greater than one full turn.

11.1.6.11.3.7 TAN

Block that calculates the tangent of Angle, storing the result in Result.

Ladder Representation



Block Structure

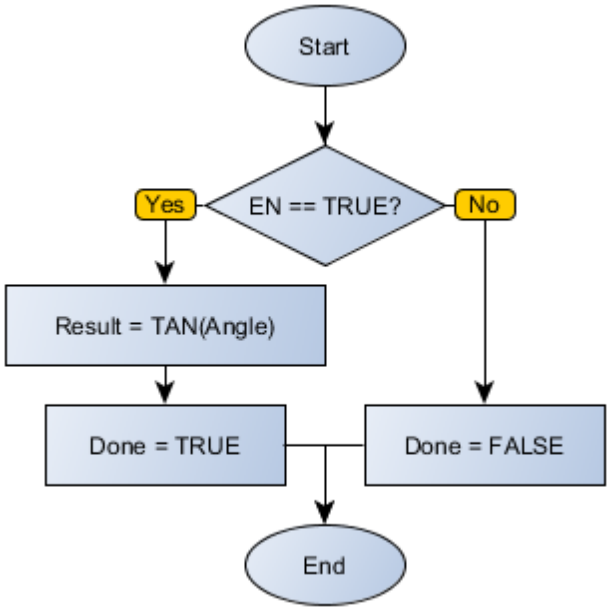
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Angle	REAL	Angle (in radians)
VAR_OUTPUT	Done	BOOL	End of operation
	Result	REAL	Variable that stores the result of the operation

Operation

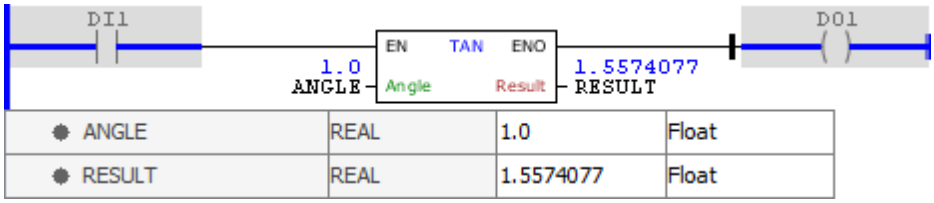
When this block has a TRUE value in EN, it sends to the Result output the tangent of Angle. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

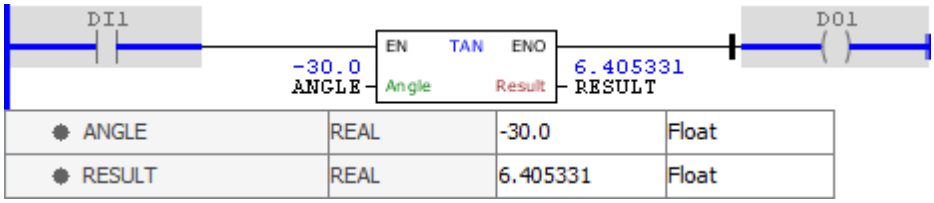


Example



ANGLE	REAL	1.0	Float
RESULT	REAL	1.5574077	Float

The above example calculates the tangent of the VALUE variable, interpreted in radians, storing the final result in RESULT. The block ends with success and Done output is activated.



ANGLE	REAL	-30.0	Float
RESULT	REAL	6.405331	Float

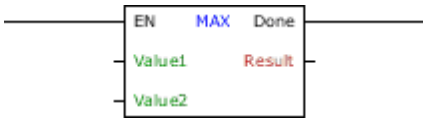
The above example calculates the tangent of the VALUE variable, interpreted in radians, storing the final result in RESULT. The block ends with success and Done output is activated. Notice that the block accepts negative input values and greater than one turn.

11.1.6.11.4 Math Util

11.1.6.11.4.1 MAX

Block that compares the values of Value1 and Value2 and stores the highest of them in Result.

Ladder Representation



Block Structure

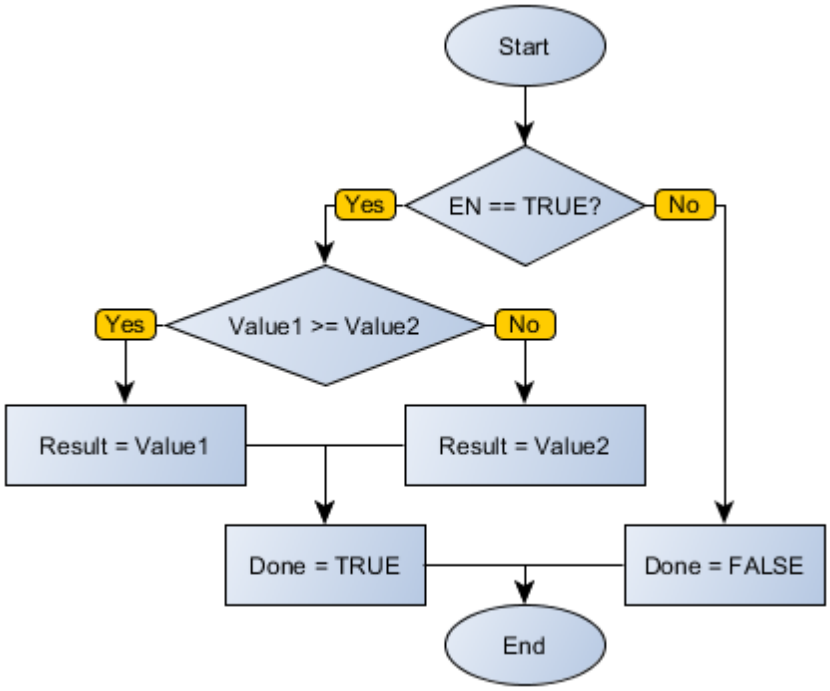
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	First value of comparison
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Second value of comparison
VAR_OUTPUT	Done	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Highest of the values compared

Operation

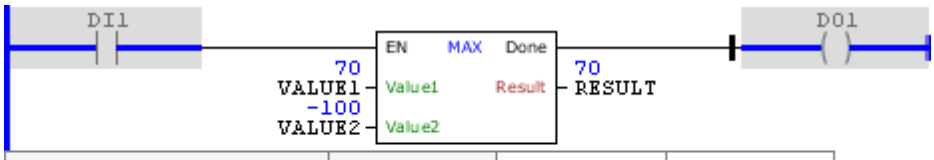
When this block has a TRUE value in EN, it sends to the Result output the highest value in the comparison between Value1 and Value2. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

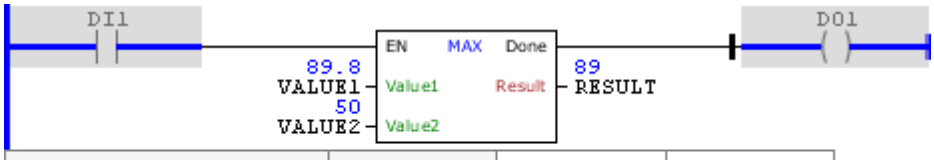


Example



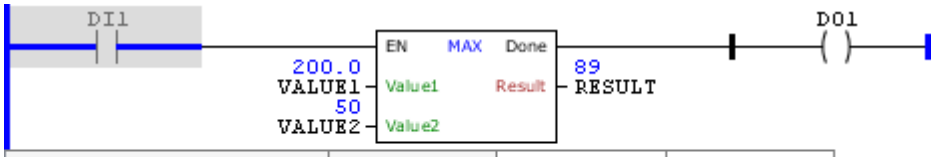
VALUE1	SINT	70	Decimal
VALUE2	SINT	-100	Decimal
RESULT	SINT	70	Decimal

The above example calculates the maximum value between VALUE 1 and VALUE2 variables, storing the final result in RESULT. The block ends with success and Done output is activated.



VALUE1	REAL	89.8	Float
VALUE2	SINT	50	Decimal
RESULT	SINT	89	Decimal

The above example calculates the maximum value between VALUE 1 and VALUE2 variables, storing the final result in RESULT. The block ends with success and Done output is activated. Notice that the types of the input variables can be different without causing execution problems.



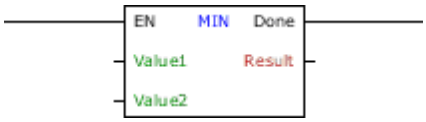
VALUE1	REAL	200.0	Float
VALUE2	SINT	50	Decimal
RESULT	SINT	89	Decimal

The above example calculates the maximum value between VALUE 1 and VALUE2 variables, storing the final result in RESULT. Since the result is higher than the maximum supported by SINT type, the block generates an error and Done output is disabled.

11.1.6.11.4.2 MIN

Block that compares the values of Value1 and Value2 and stores the lowest of them in Result.

Ladder Representation



Block Structure

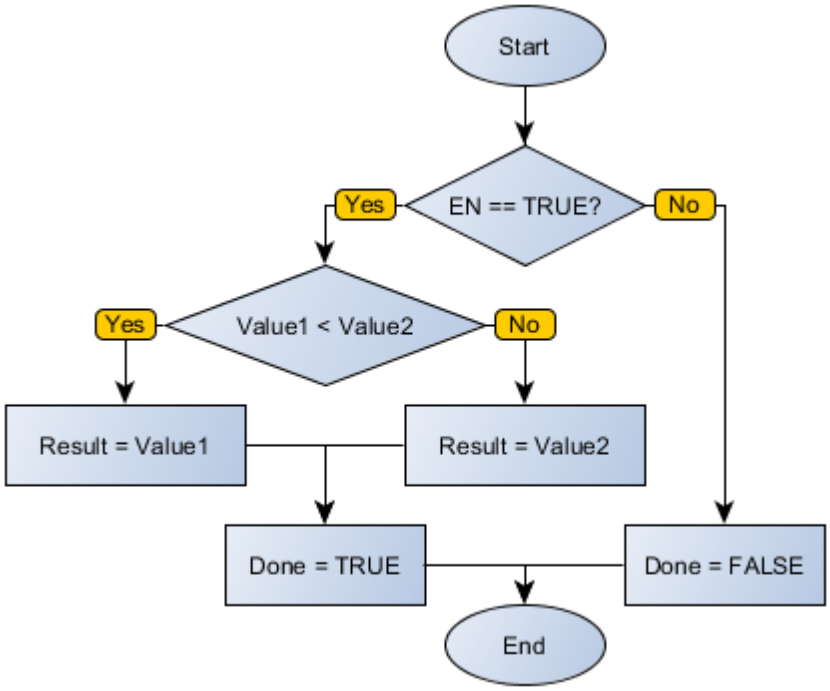
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	First value of comparison
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Second value of comparison
VAR_OUTPUT	Done	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Low est of the values compared

Operation

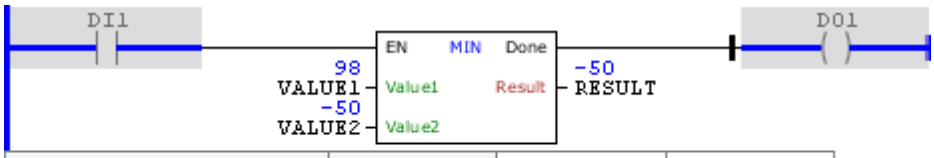
When this block has a TRUE value in EN, it sends to the Result output the lowest value in the comparison between Value1 and Value2. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

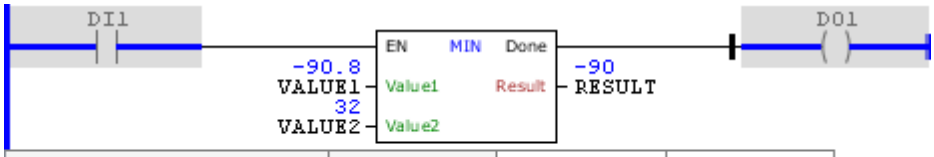


Example



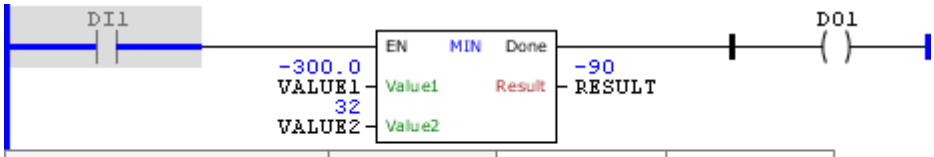
VALUE1	SINT	98	Decimal
VALUE2	SINT	-50	Decimal
RESULT	SINT	-50	Decimal

The above example calculates the minimum value between VALUE 1 and VALUE2 variables, storing the final result in RESULT. The block ends with success and Done output is activated.



VALUE1	REAL	-90.8	Float
VALUE2	SINT	32	Decimal
RESULT	SINT	-90	Decimal

The above example calculates the minimum value between VALUE 1 and VALUE2 variables, storing the final result in RESULT. The block ends with success and Done output is activated. Notice that the types of the input variables can be different without causing execution problems.



VALUE1	REAL	-300.0	Float
VALUE2	SINT	32	Decimal
RESULT	SINT	-90	Decimal

The above example calculates the minimum value between VALUE 1 and VALUE2 variables, storing the final result in RESULT. Since the result is lower than the minimum supported by SINT type, the block generates an error and Done output is disabled.

11.1.6.11.4.3 SAT

Block that performs a routine for saturation of the value found in Value in accordance with the limits for Minimum and Maximum, storing the result in Result.

Ladder Representation

Block Structure

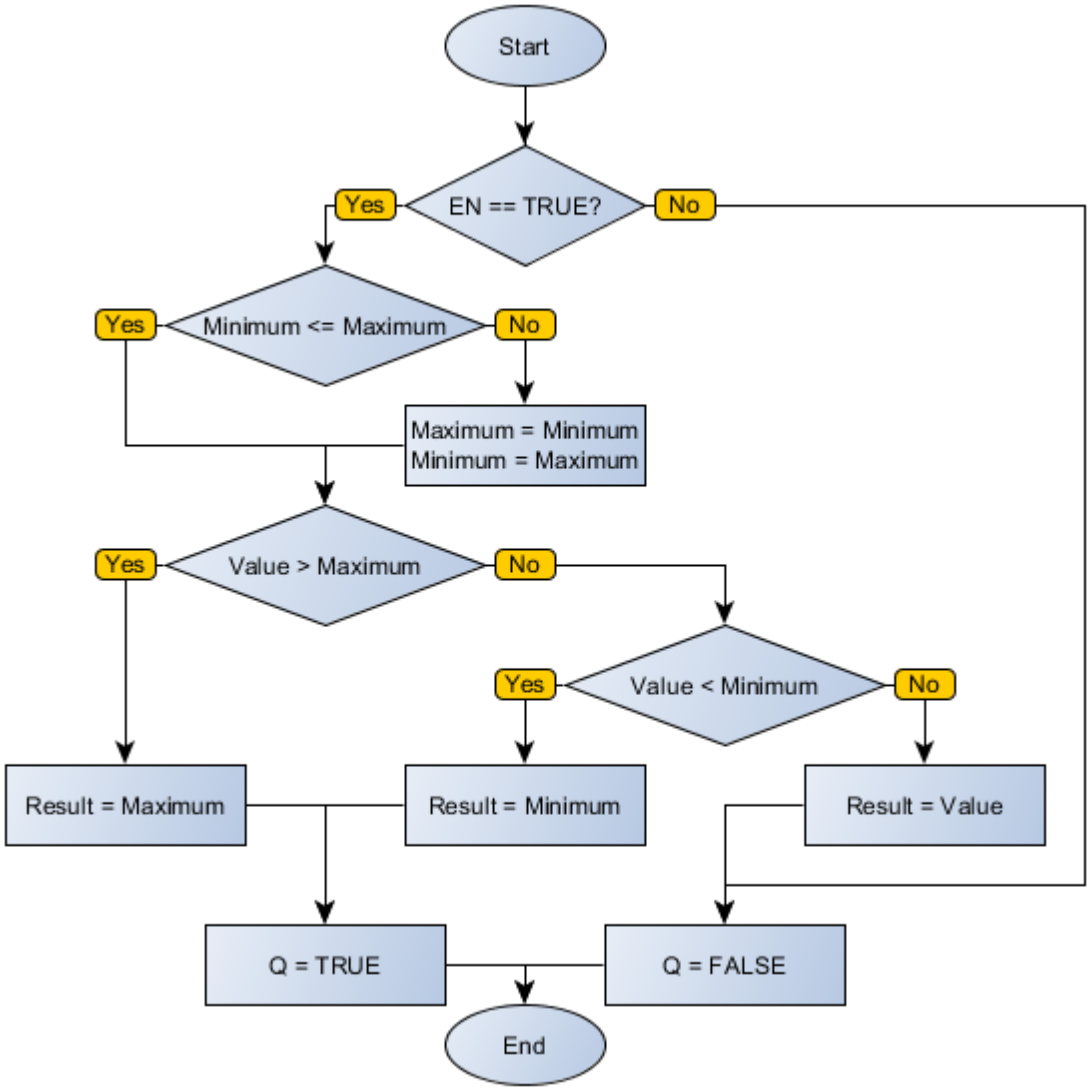
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Reference value
	Minimum	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Inferior saturation value
	Maximum	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Superior saturation value
VAR_OUTPUT	Q	BOOL	Indicator that there w as saturation in the process
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Result of operation

Operation

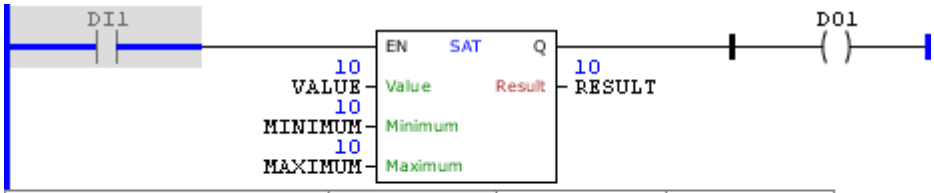
When this block has a TRUE value in EN, it performs a comparison between Value and Minimum and Maximum. If Value is in the range between Minimum and Maximum, Result receives the value of Value and Q remains FALSE. If Value is higher than Maximum, Result receives Maximum and Q receives TRUE. If Value is lower than Minimum, Result receives Minimum and Q receives TRUE. If there is any error in the operation, Q is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Q remains in FALSE.

Block Flowchart

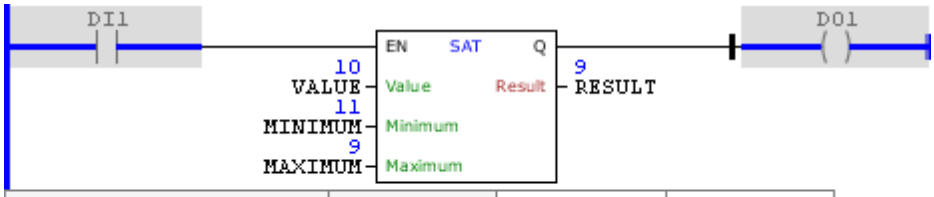


Example



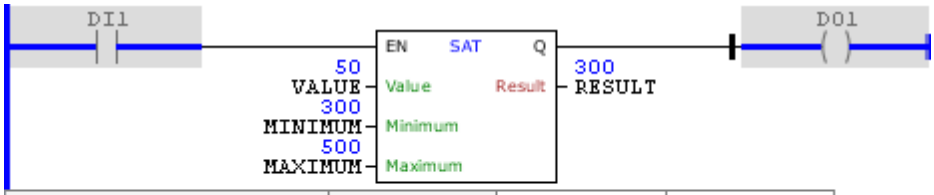
VALUE	INT	10	Decimal
MINIMUM	INT	10	Decimal
MAXIMUM	INT	10	Decimal
RESULT	INT	10	Decimal

The above example passes the VALUE value to RESULT, since it is not lower than MINIMUM or higher than MAXIMUM. The block ends successfully and the Q output is disabled, since there was no saturation.



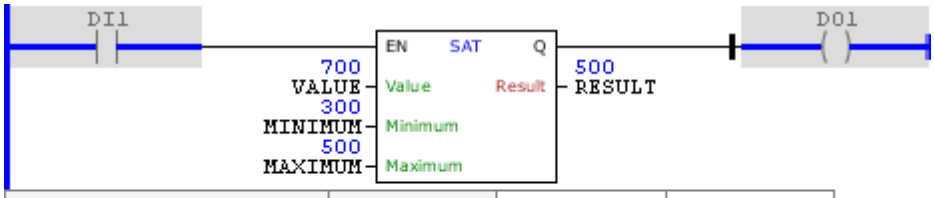
VALUE	INT	10	Decimal
MINIMUM	INT	11	Decimal
MAXIMUM	INT	9	Decimal
RESULT	INT	9	Decimal

The above example passes the MAXIMUM to RESULT, since VALUE is higher than MAXIMUM. The block ends successfully and the Q output is activated, since there was saturation.



VALUE	INT	50	Decimal
MINIMUM	INT	300	Decimal
MAXIMUM	INT	500	Decimal
RESULT	INT	300	Decimal

The above example passes the MINIMUM to RESULT, since VALUE is lower than MINIMUM. The block ends successfully and the Q output is activated, since there was saturation.



VALUE	INT	700	Decimal
MINIMUM	INT	300	Decimal
MAXIMUM	INT	500	Decimal
RESULT	INT	500	Decimal

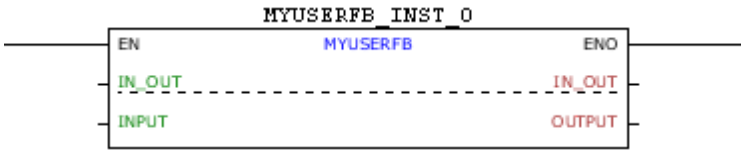
The above example passes the MAXIMUM value to RESULT, since VALUE is higher than MAXIMUM. The block ends successfully and the Q output is activated, since there was saturation.

11.1.6.12 Module

11.1.6.12.1 USERFB

Block that performs a subroutine programmed by the user.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	INPUT	According to user programming	Block inputs
VAR_OUTPUT	ENO	BOOL	End of operation
	OUTPUT	According to user programming	Block outputs
VAR_IN_OUT	IN_OUT	According to user programming	Block inputs/outputs
VAR	MYUSERFB_INST_0	MYUSERFB	Instance of access to block structure

Operation

When this block has a TRUE value in EN, it updates the values of internal fields with the input variables, performs the Ladder routine programmed by the user and updates the values of the outputs after completing routine.

When EN has FALSE value, outputs remain unchanged.

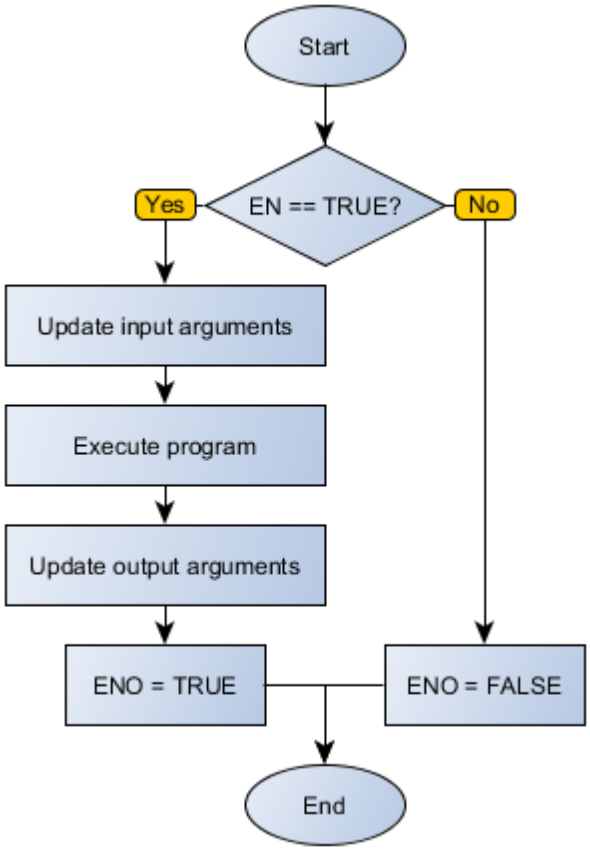
The ENO value forwards to the next Ladder block the EN value after the operation is completed.

NOTE!
Refer to section Working with USERFBs for further information.

Compatibility

Device	Version
PLC300	1.50 or higher
SCA06	2.00 or higher

Block Flowchart



11.1.6.13 Motion Control

11.1.6.13.1 MW_RefVelocity

This block sends speed reference to drive.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Velocity	DINT INT REAL	Sets speed reference to drive if block is enabled
	VelocityUnit	0 = 13Bits 1 = RPM 2 = HZ(x10)	Sets the speed unit: 13 Bits – Sends the speed value in 13 bits; RPM – Sends the speed value in RPM; HZ – Sends the speed value in Hz (x10).
	RunAutomatic	0 = FALSE 1 = TRUE	Define if block will run the Run/Stop (CFW_CMD_RUN_STOP) when it is enabled: FALSE – Do not send Run/Stop command with block enabling (it is necessary to use the marker CFW_CMD_RUN_STOP in ladder's logic to send the Run/Stop command); TRUE – Send Run command with block enabling and Stop command with block disabling.
VAR_OUTPUT	ENO	BOOL	End of operation. Conditions for ENO = 1 • Does not exist another active block MW_RefVelocity; • Drive is enabled and stop mode set "Stopping by inertia".

Operation

When this block has a "0" value in EN, it does not execute and ENO output is zero.

RunAutomatic = TRUE

When this block has a "1" value in EN input, the drive is general enabled, no other motion block is active, the Run/Stop command goes to "1", the speed reference value is send to drive and the ENO output is set to "1".
If EN input has a "0" value, and this block is active, the Run/Stop command is set to "0" and ENO output goes to "0".

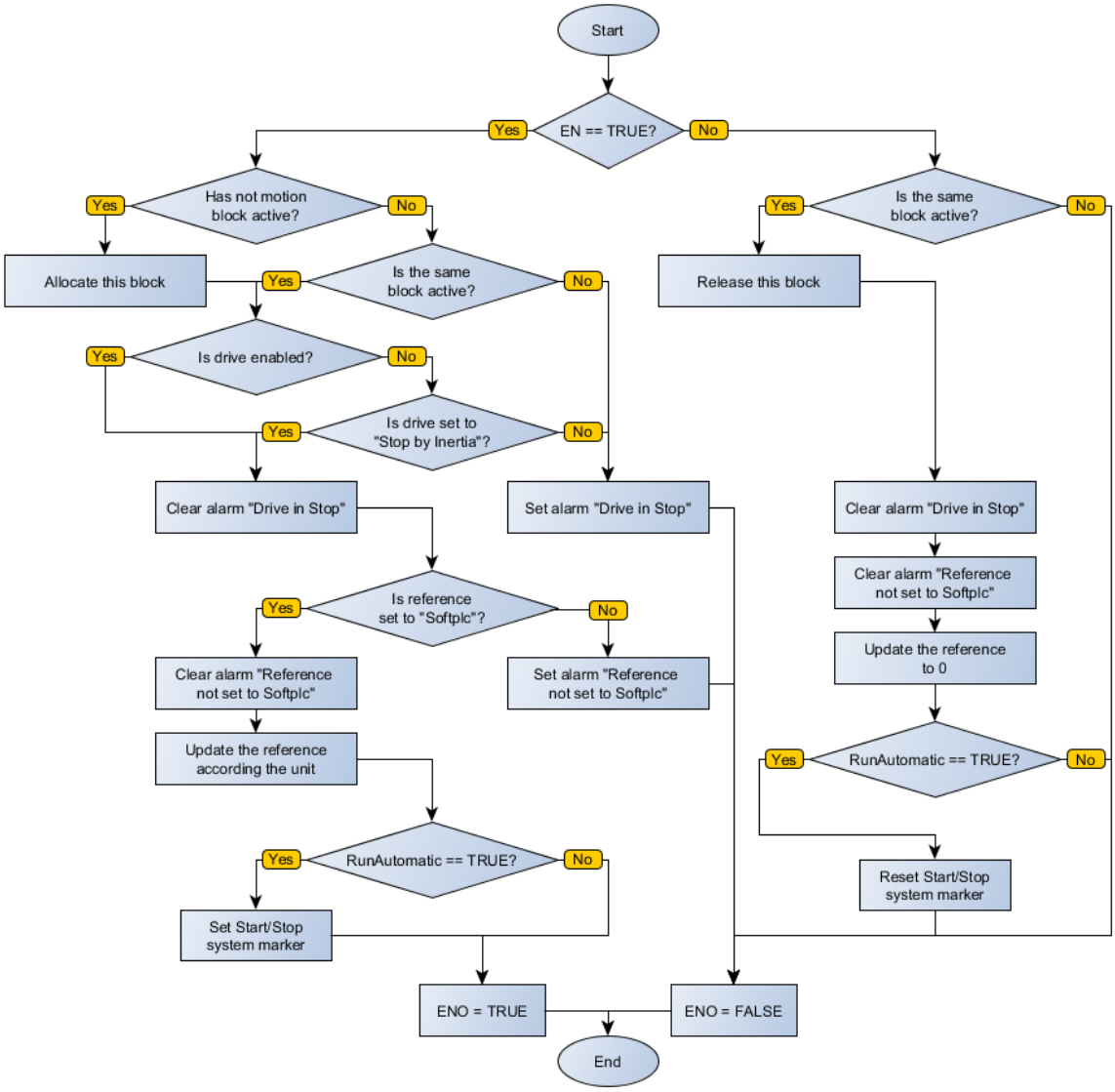
RunAutomatic = FALSE

When this block has a "1" value in EN input, the drive is general enabled, the Run/Stop command is set to "1", no other motion block is active, the speed reference value is send to drive and the ENO output is set to "1".
If EN input has a "0" value and this block is active, ENO output is set to "0".

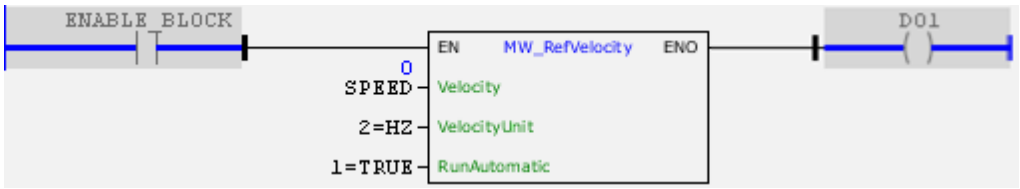
NOTE!

Check the source of speed reference and command Run/Stop for correct operation of this block

Block Flowchart

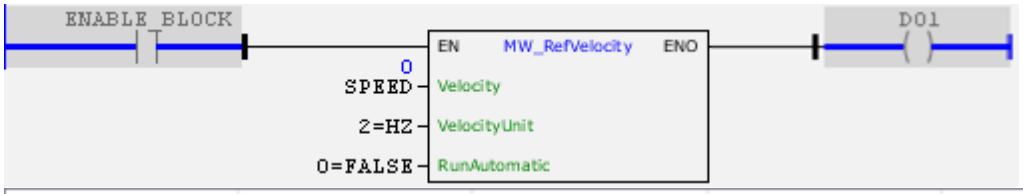


Example



Variável	Tipo	User	Monitoring	Visualização
Main Ladder				
SPEED	INT	0	500	Decimal
Global Variables				
PAR_001	UINT	0	500	Decimal

The above example shows the MW_RefVelocity block, set to Hz and the RunAutomatic command in TRUE, if drive is general enabled and the block is enabled, the speed reference is changed.



Variável	Tipo	User	Monitoring	Visualização
Main Ladder				
SPEED	INT	0	500	Decimal
Global Variables				
PAR_001	UINT	0	500	Decimal

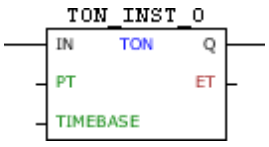
The above example shows the MW_RefVelocity block, set to Hz and RunAutomatic command in FALSE, if drive is general enabled, it is necessary the Run command. So, when the block be enabled, the speed reference would be changed.

11.1.6.14 Timer

11.1.6.14.1 TON

Timer block that, when energized, enables the output after a delay set by PT.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	IN	BOOL	Block enabling
	PT	WORD UINT DWORD UDINT	Delay of output drive
	TIMEBASE	WORD	Time base for PT and ET
VAR_OUTPUT	Q	BOOL	Block output
	ET	WORD UINT DWORD UDINT	Counter elapsed time
VAR	TON_INST_0	TON	Instance of access to block structure

NOTE!
In CFW300, the PT e ET fields can only be WORD ou UINT type.

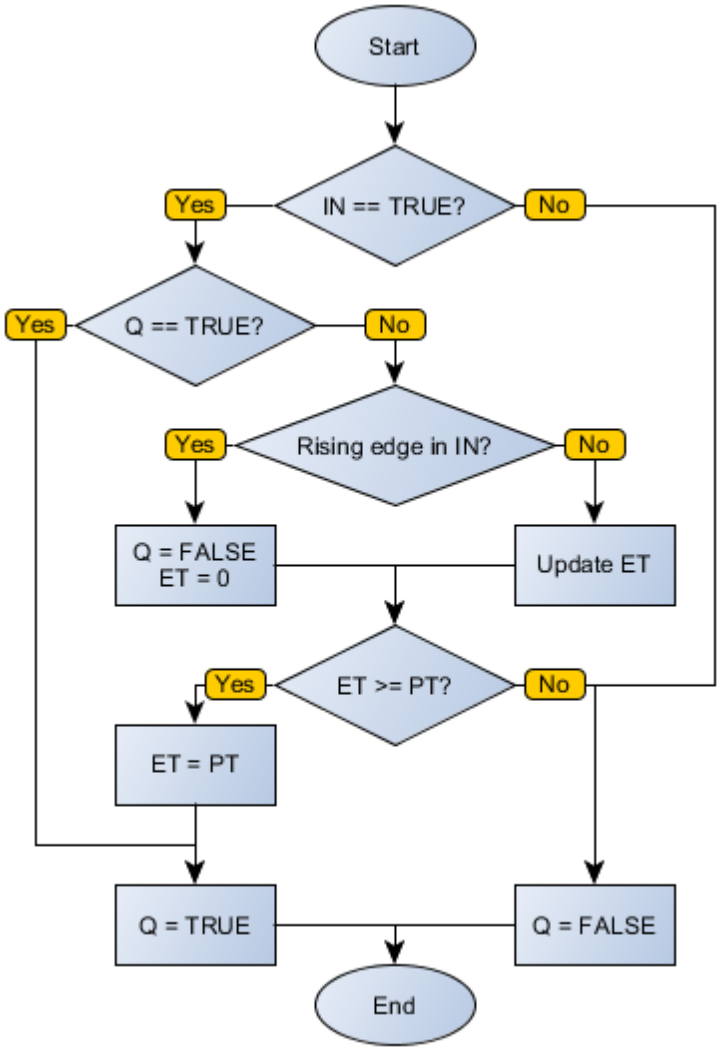
Operation

While the IN input is FALSE, the Q output is FALSE and ET also receives the value zero.
On the edge positive transition in IN, counting is triggered and ET is incremented according to TIMEBASE. When ET equals PT, the Q output goes to state TRUE until IN revolutions to FALSE.

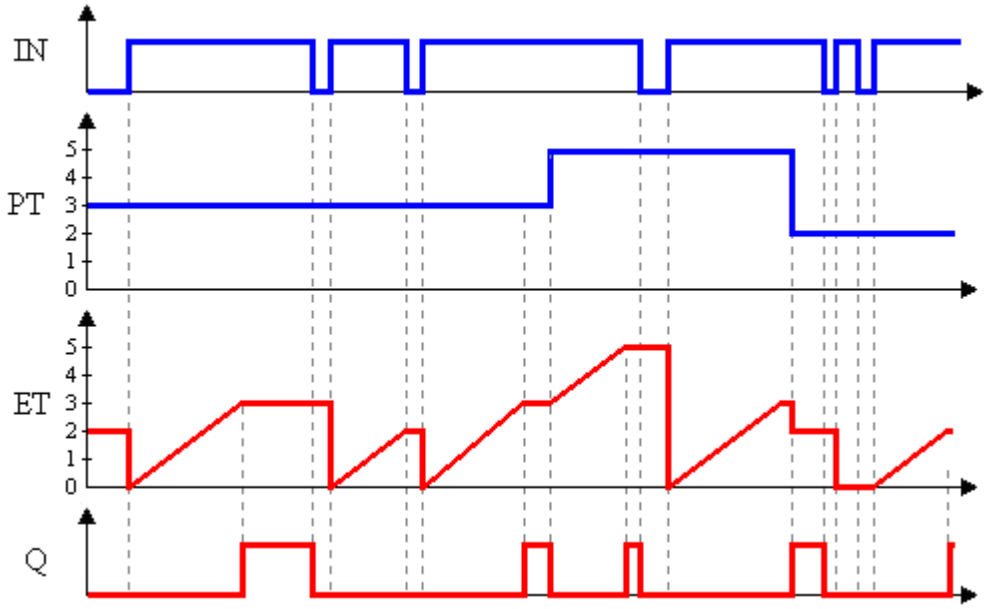
Compatibility

Device	Version
PLC300	1.50 or higher
SCA06	2.00 or higher

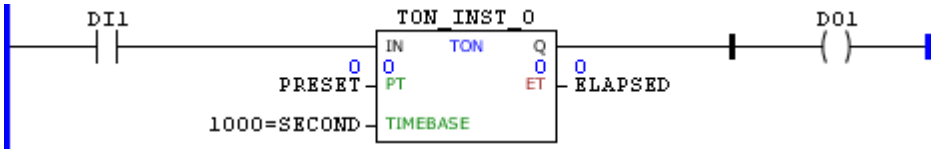
Block Flowchart



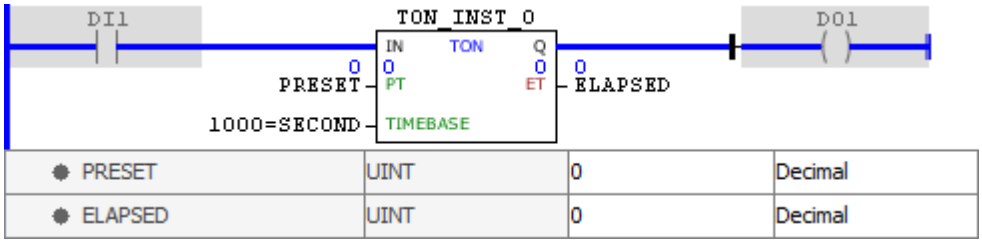
Operation Diagram



Example

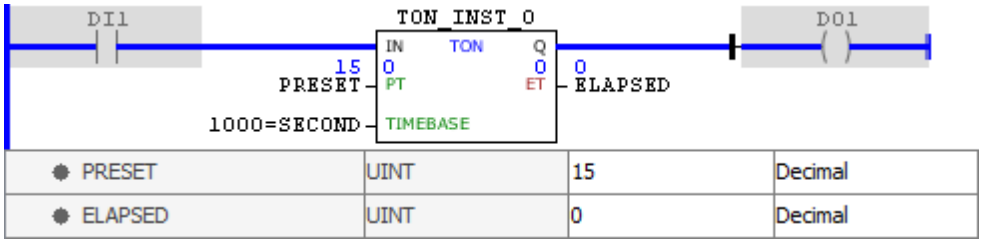


The above example shows the initial conditions of the block and of the routine variables.



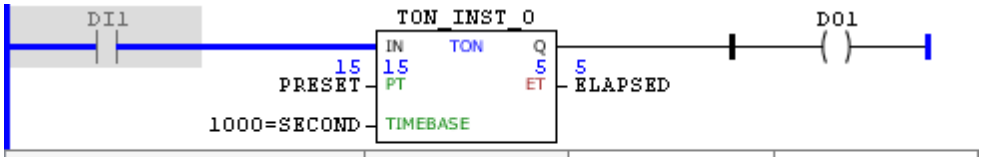
PRESET	UINT	0	Decimal
ELAPSED	UINT	0	Decimal

When activated the IN input, counting is triggered. Since ET equals PT, the Q output is enabled.



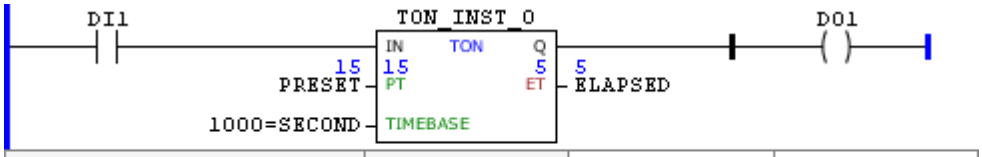
PRESET	UINT	15	Decimal
ELAPSED	UINT	0	Decimal

Note that a change in PRESET variable is not forwarded to the PT field while the IN entry remains enabled.



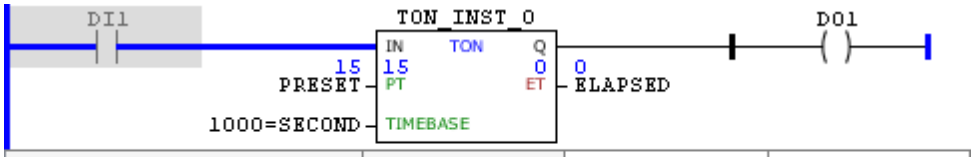
PRESET	UINT	15	Decimal
ELAPSED	UINT	5	Decimal

Disabling the IN input, the value of PT is updated and the Q output is disabled. When activating it again, counting is triggered.



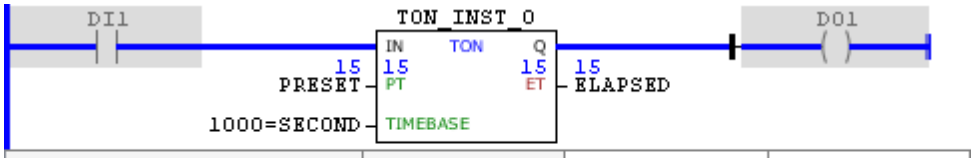
PRESET	UINT	15	Decimal
ELAPSED	UINT	5	Decimal

Disabling the IN input, the value of ET remains saved.



PRESET	UINT	15	Decimal
ELAPSED	UINT	0	Decimal

Enabling the IN input, the value of ET is reset and counting is triggered.



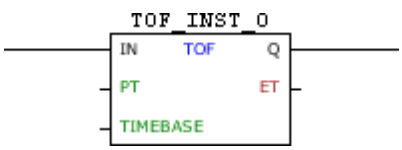
PRESET	UINT	15	Decimal
ELAPSED	UINT	15	Decimal

When ET reaches the value PT, the Q is output enabled and remains so while IN is at TRUE level.

11.1.6.14.2 TOF

Timer block that, when energized, disables the output after a delay set by PT.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	IN	BOOL	Block enabling
	PT	WORD UINT DWORD UDINT	Delay of output deactivating
	TIMEBASE	WORD	Time base for PT and ET
VAR_OUTPUT	Q	BOOL	Block output
	ET	WORD UINT DWORD UDINT	Counter elapsed time
VAR	TOF_INST_0	TOF	Instance of access to block structure

NOTE!
In CFW300, the PT e ET fields can only be WORD ou UINT type.

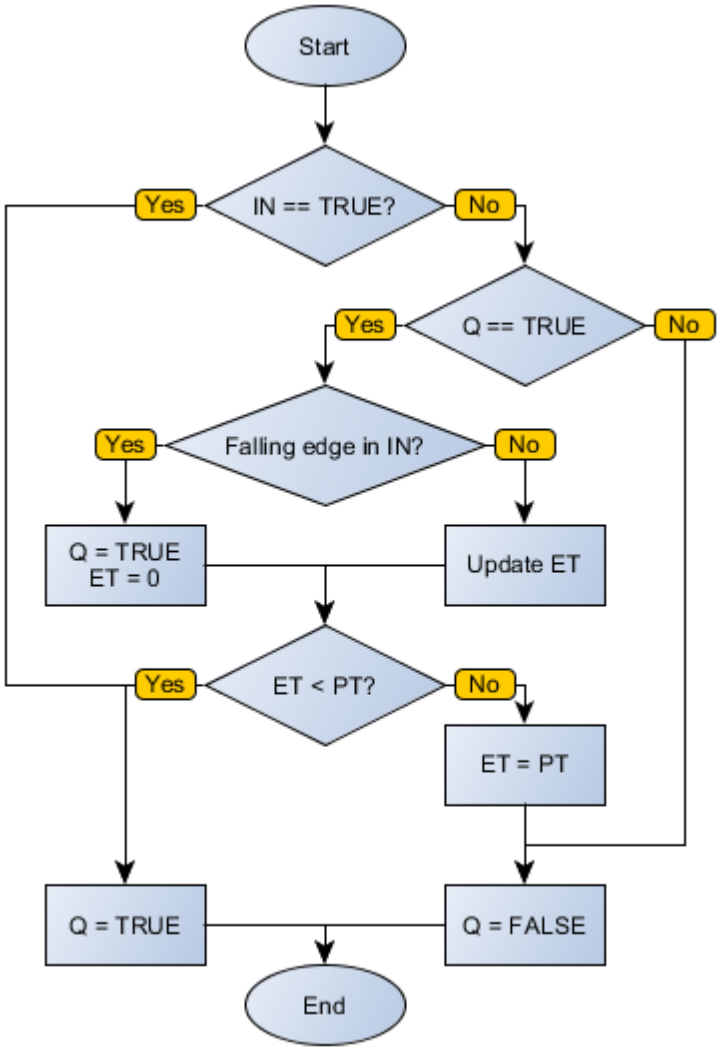
Operation

While the IN input is TRUE, the Q output is also TRUE and ET also receives the value zero.
On the negative transition edge in IN, counting is triggered and ET is incremented according to TIMEBASE. When ET equals PT, the Q output goes to state FALSE until IN revolutions to FALSE.

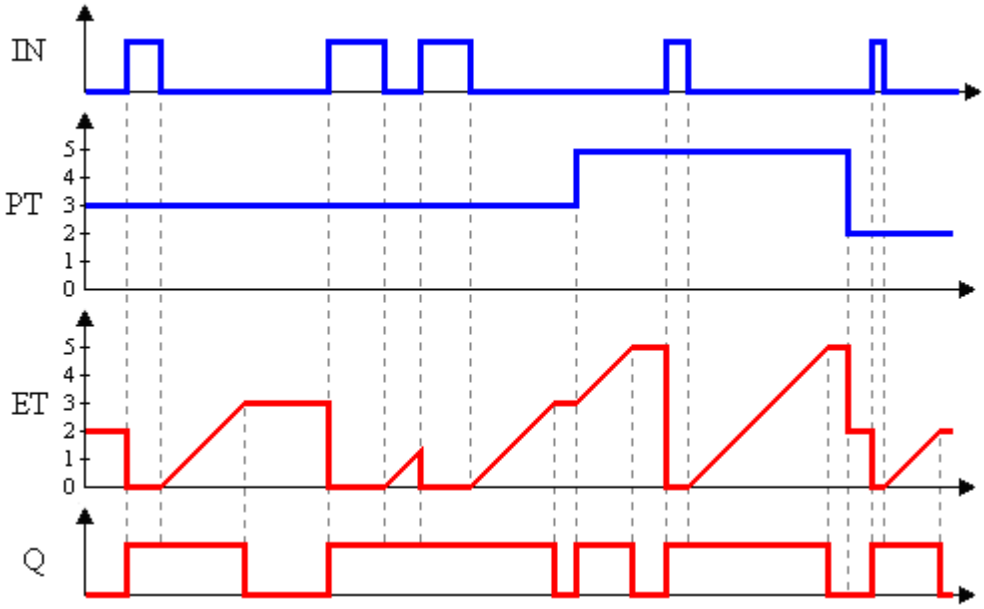
Compatibility

Device	Version
PLC300	1.50 or higher
SCA06	2.00 or higher

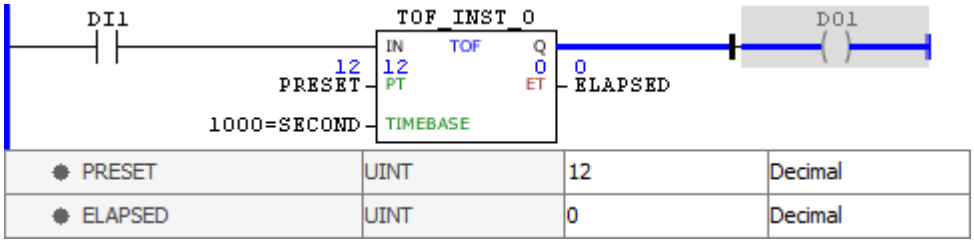
Block Flowchart



Operation Diagram



Example



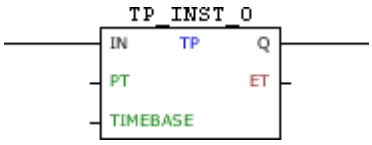
PRESET	UINT	12	Decimal
ELAPSED	UINT	0	Decimal

The above example disables the DO1 output to identify a low level in DI1 for 12 seconds, remaining disabled until DI1 again be TRUE.

11.1.6.14.3 TP

Timer block that, when identifies it is energized, enables the output after a delay set by PT.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	IN	BOOL	Block enabling
	PT	WORD UINT DWORD UDINT	Time w hile the output is enabled
	TIMEBASE	WORD	Time base for PT and ET
VAR_OUTPUT	Q	BOOL	Block output
	ET	WORD UINT DWORD UDINT	Counter elapsed time
VAR	TP_INST_0	TP	Instance of access to block structure

NOTE!
In CFW300, the PT e ET fields can only be WORD ou UINT type.

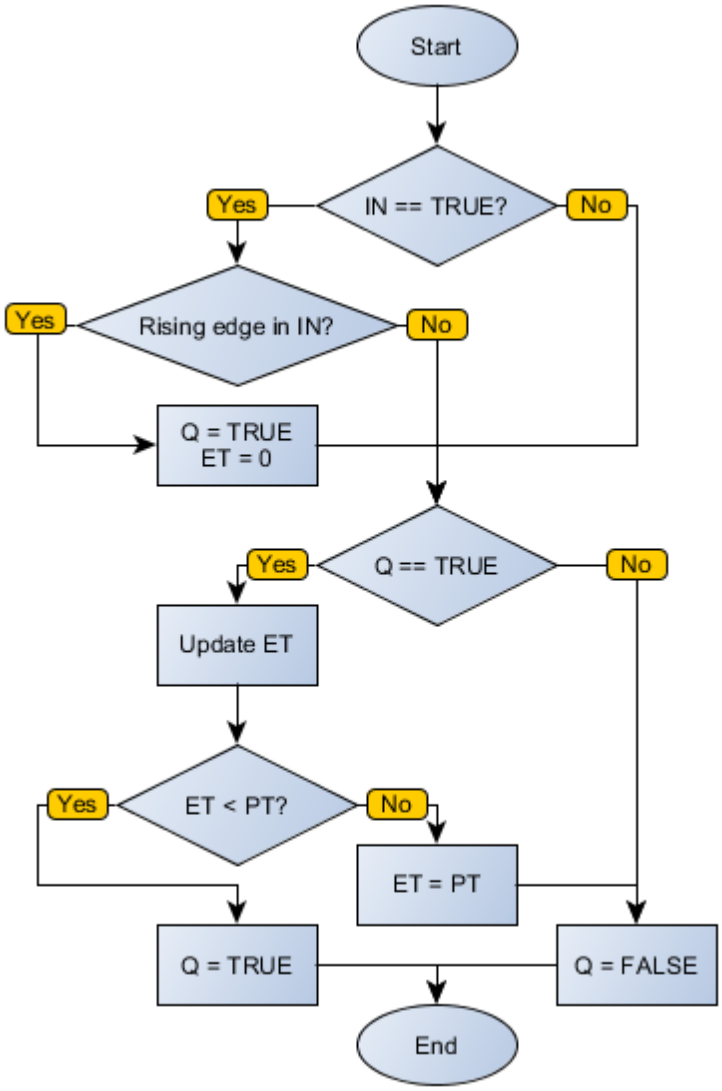
Operation

On the edge positive transition in IN, Q receives TRUE value, counting is triggered and ET is incremented according to TIMEBASE. When ET equals PT, the Q output goes to state FALSE until IN revolutions to FALSE. At that moment, if IN is at TRUE level, nothing happens. On the edge positive transition in IN, ET is automatically reset.

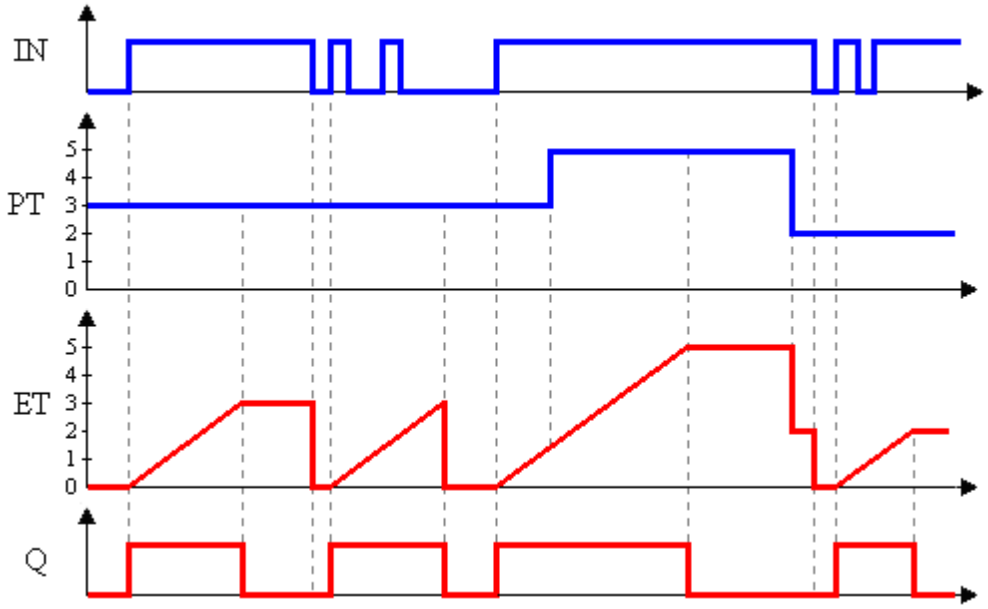
Compatibility

Device	Version
PLC300	1.50 or higher
SCA06	2.00 or higher

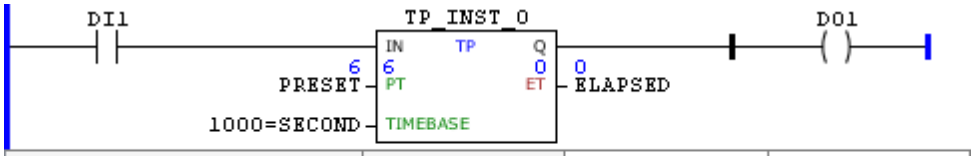
Block Flowchart



Operation Diagram



Example



PRESET	UINT	6	Decimal
ELAPSED	UINT	0	Decimal

The above example enables the DO1 output for six seconds at each DI1 positive transition.

11.1.6.15 Structures

Structure is a data grouping used to define a recipe or an object.

In the Ladder program, it is possible to create variables of the structure type and use them in the blocks. To access the internal members of the structure, the '.' is used followed by its respective member.

Creating a structure

1. With the right button of the mouse on the folder **Structure**, click on **New file**.

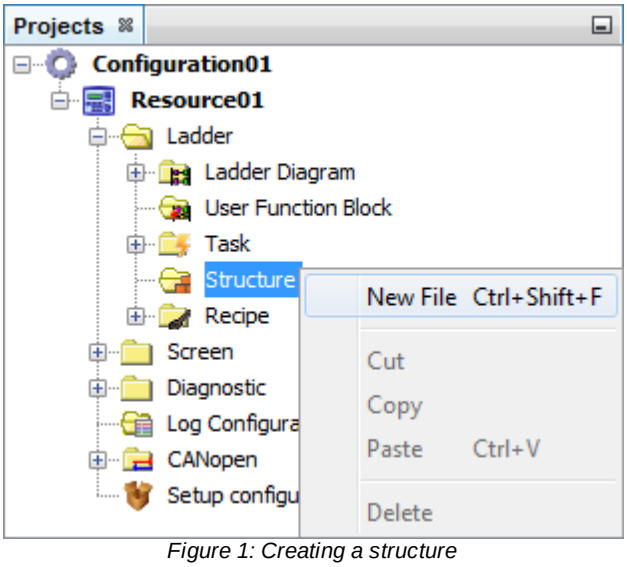


Figure 1: Creating a structure

2. Define the file name and press the **Next** button.

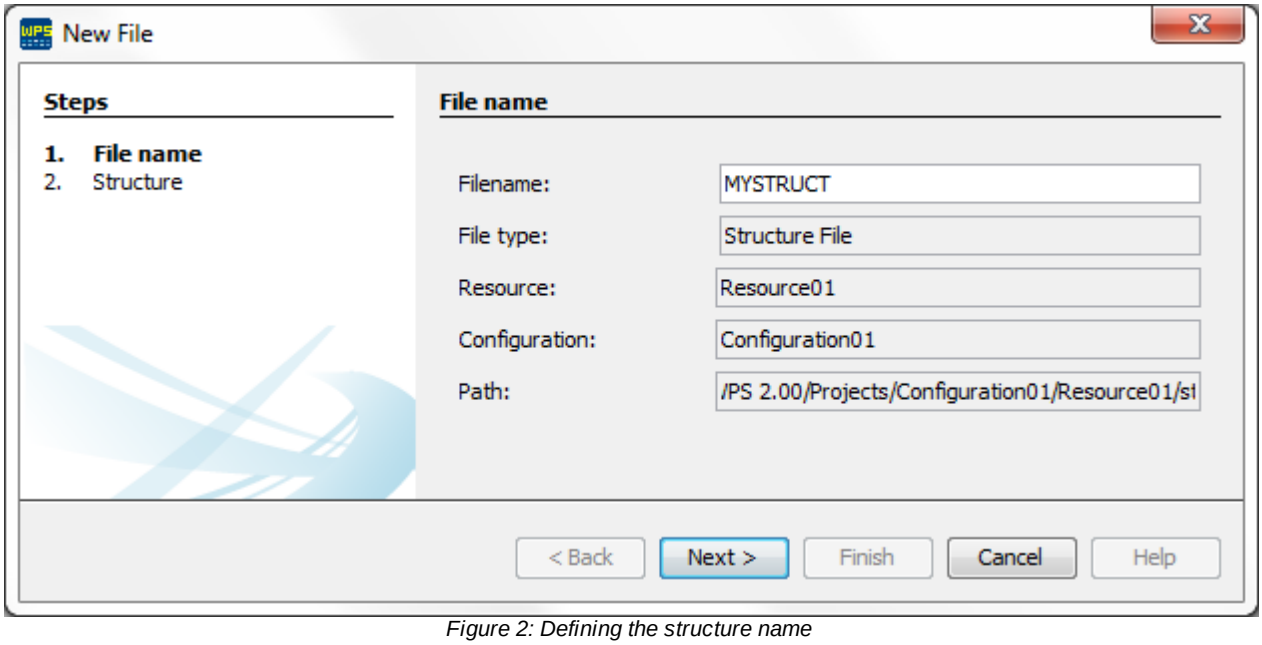


Figure 2: Defining the structure name

3. Configure the structure using the buttons presented in the figure below.

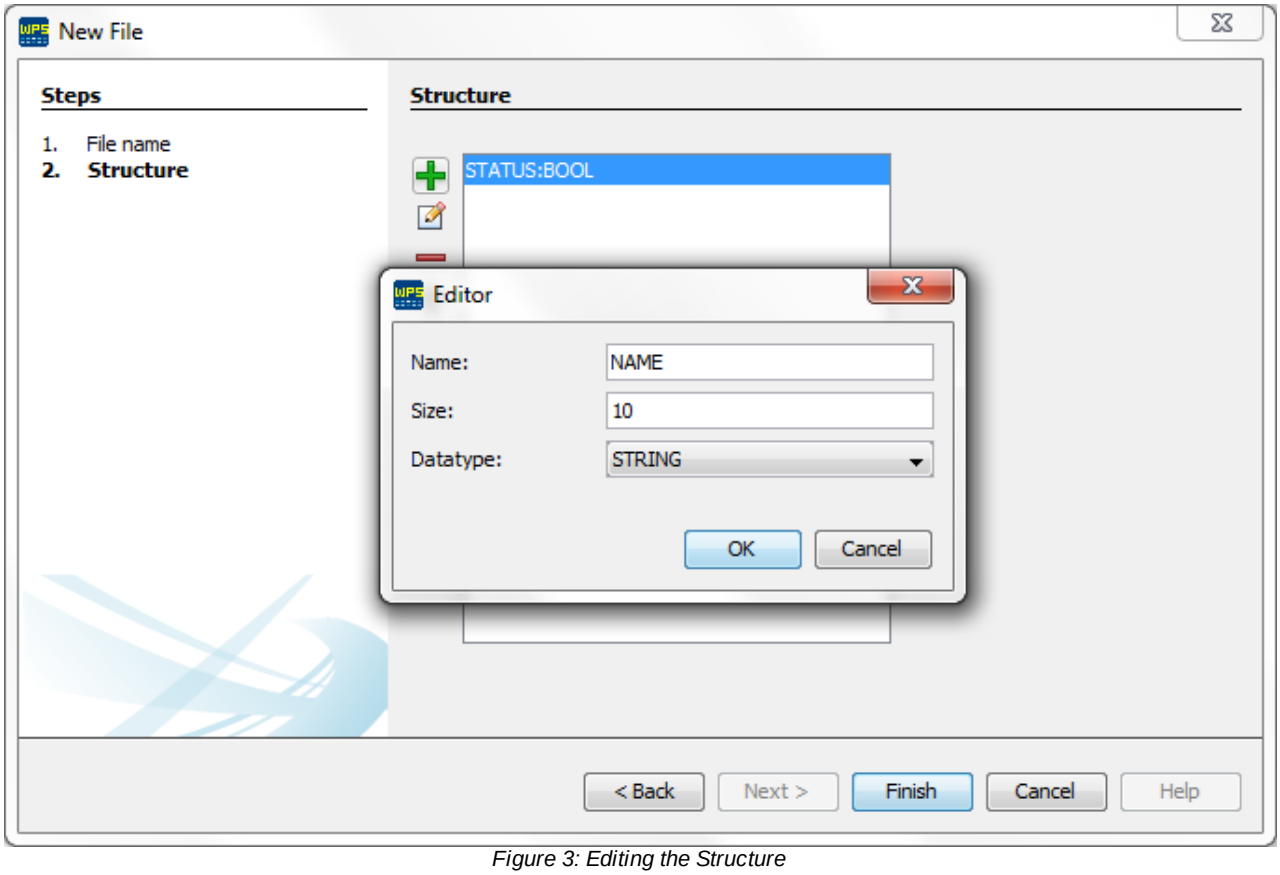


Figure 3: Editing the Structure

4. After finishing the edition of the structure, click on the button **Finish**.

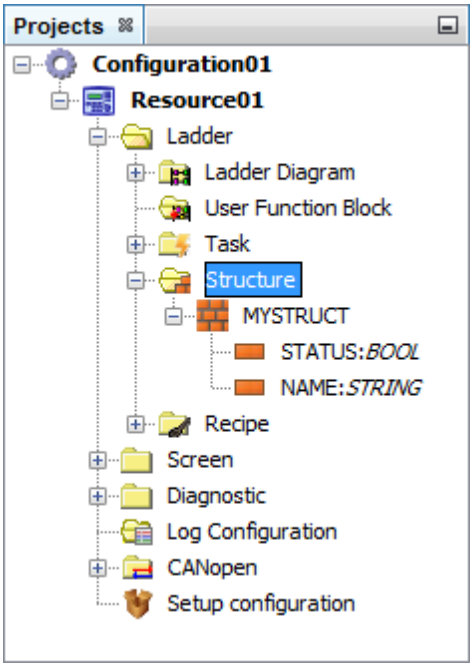


Figure 4: Structure created in the project

Editing a structure

Just double click on the desired structure, as shown in figure 4, and a window will open as shown in figure 3, allowing to insert new data, erase or move the position of the data.

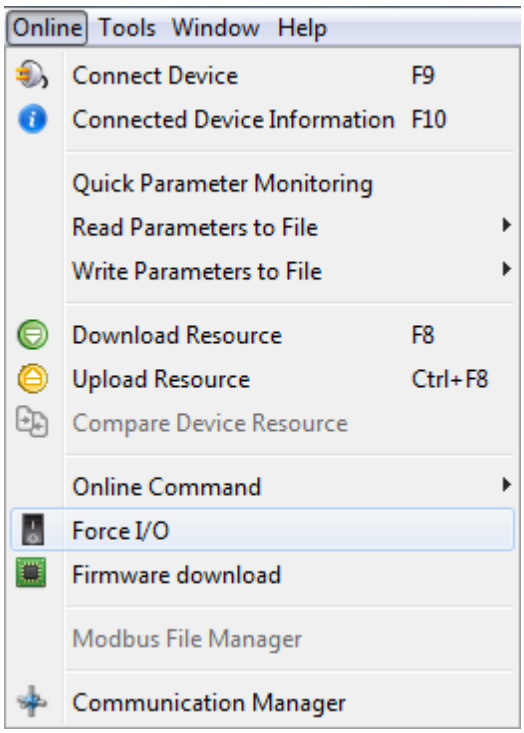
11.1.7 Communication

11.1.7.1 Force I/O

Overview

The force inputs and outputs window is used for the values of the digital and analog inputs to be read by the program, by values manipulated by the user, regardless their physical state. It also allows the manipulation of the physical states of the digital and analog outputs by the user independently of the values calculated by the program.
In order to force the device inputs and outputs, it is necessary that the online monitoring be active and the option **Run cyclically** be enabled. The data are sent to the device every 2 seconds.
The values can be edited with the device disconnected. The configurations are stored in the resources and recorded whenever the main resource selection is changed.
The data displayed on the force I/O window contain the values belonging to the resource (and configuration) selected as main.

The force I/O window is open trough the menu **Online** > **Force I/O**:



Toolbar

The toolbar of the force window has the options to run cyclically, upload the device force configuration, enable

all and disable all:

Run cyclically: Sends the user's configurations to the device and updates the state of the inputs and outputs in a cyclic way.

Upload configuration: Allows the current configuration of the device to be read. For this option to be enabled, it is necessary that the online monitoring be active and the option run cyclically be disabled.

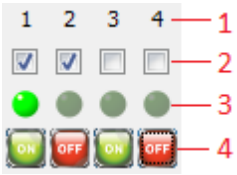
Enable all: Enables the force I/O of all of the inputs and outputs of the device.

Disable all: Disables the force I/O of all of the inputs and outputs of the device.

Input and Output commands

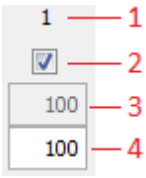
For each digital and analog input and output there is a selection box linked to enable the force, a status field and an edition field.

Digital:



1. Number of the digital inputs/output
2. Enable/disable Force I/O
3. Current status of the I/O: It has three statuses: 1. light green LED: activated; 2. dark green LED: deactivated; 3. gray LED: the value is not being read.
4. Enable/disable the input/output

Analog:



1. Number of the analog input/output
2. Enable/disable Force I/O
3. Current value of the input/output
4. Value of the input/output configured by the user

NOTE!
The analog signal scale has 15 bits plus 1 bit for signal, except for SSW900 which it has only 10 unsigned bits.

11.2 CFW300

11.2.1 Description

The CFW300 frequency inverter is a high-performance product which allows speed control of three-phase induction motors. This product provides the user with the options of vector (VVW) or scalar (V/f) control, both

programmable according to the application.

The scalar mode (V/f) is recommended for simpler applications, such as the activation of most pumps and fans. In such cases it is possible to reduce the losses in the motor and the inverter using the "V/f Quadratic", which results in energy savings. The V/f mode is used when more than a motor is activated by an inverter simultaneously (multimotor applications). In the vector mode (VVW), the operation is optimized for the motor in use, obtaining a better performance in terms of speed regulation.

The frequency inverter CFW300 also has functions of PLC (Programmable Logic Controller) by means of the SoftPLC (integrated) feature. It has two slots for simultaneous connection of the accessories: Slot 1 - Communication accessory or external HMI and Slot 2 - Input and output (I/O) expansion accessory.

Refer to the user's manual of the CFW300 for further details about the product.

11.2.2 System Markers

The following variables contained in the **GLOBAL_SYSTEM** group of the variables table, have the fixed tag. The tag of system markers were divided into groups and subgroups, where:

Groups:
- CFW: reading and writing variables of the CFW300 frequency inverter.

Subgroups:
- STS: reading variable (status);
- CMD: writing variable (command).

Reading System Markers (Status)

Reading - Function Modbus 02 "Read Discrete Inputs"

Address	Bit	Modbus	Tag	Description
Ladder				
%SB6000	0	0	SYS_FREQ_2HZ	Oscillator with frequency of 2 Hz
%SB6000	1	1	SYS_PULSE_1SCAN	Pulse during the first scan cycle
%SB6000	2	2	SYS_FALSE	Always in 0
%SB6000	3	3	SYS_TRUE	Always in 1
Logical Status				
%SB6002	1	17	CFW_STS_RUN_COMMAND	The run motor command is active in the inverter
%SB6002	2	18	CFW_STS_FIRE_MODE_ACTIVE	Fire Mode Function is active
%SB6002	5	21	CFW_STS_SEC_RAMP_ACTIVE	The inverter is configured to use the first or second ramp values (0-First, 1-Second)
%SB6002	6	22	CFW_STS_CONFIG_MODE	The inverter is in the configuration mode
%SB6002	7	23	CFW_STS_ALARM_ACTIVE	The inverter is in alarm condition
%SB6003	0	24	CFW_STS_MOTOR_RUNNING	The inverter is running the motor at the

				speed reference, or executing either the acceleration or the deceleration ramp
%SB6003	1	25	CFW_STS_GENERAL_ENABLED	General Enable is active and the inverter is ready to run the motor
%SB6003	2	26	CFW_STS_FWD_REV_DIRECTION	The motor is running in the reverse or forward direction (0-Reverse, 1-Forward)
%SB6003	3	27	CFW_STS_JOG_ACTIVE	The JOG function is active
%SB6003	4	28	CFW_STS_LOC_REM_MODE	The inverter is in local or remote mode (0-Local, 1-Remote)
%SB6003	5	29	CFW_STS_UNDERVOLTAGE	The inverter is in undervoltage
%SB6003	7	31	CFW_STS_FAULT_ACTIVE	The inverter has detected a fault
%SB6004	0	32	CFW_STS_AI1_BROKEN_CABLE	It indicates that the signal of analog input AI1 in 4 to 20 mA or 20 to 4 mA is below 2 mA
%SB6004	1	33	CFW_STS_AI2_BROKEN_CABLE	It indicates that the signal of analog input AI2 in 4 to 20 mA or 20 to 4 mA is below 2 mA
HMI keys				
%SB6006	0	48	CFW_STS_KEY_START_STOP	START/STOP key (I)/(0) pressed
%SB6006	2	50	CFW_STS_KEY_UP	UP key pressed
%SB6006	3	51	CFW_STS_KEY_DOWN	DOWN key pressed
Infrared Remote Control (IRC 1)				
%SB6010	0	80	CFW_STS_IRC_1_KEY_ON	Start/Stop Motor key pressed
%SB6010	1	81	CFW_STS_IRC_1_KEY_DOWN	Browse Downwards key pressed
%SB6010	2	82	CFW_STS_IRC_1_KEY_UP	Browse Upwards key pressed
%SB6010	3	83	CFW_STS_IRC_1_KEY_CHANGE	Commute view key pressed. This key allows commute view between two parameters (values) defined by parameters P842 and P843
%SB6010	4	84	CFW_STS_IRC_1_KEY_P	Confirm/Program key pressed
%SB6010	5	85	CFW_STS_IRC_1_KEY_SFK1	Special Function key 1 pressed
%SB6010	6	86	CFW_STS_IRC_1_KEY_SFK2	Special Function key 2 pressed
%SB6010	7	87	CFW_STS_IRC_1_KEY_SFK3	Special Function key 3 pressed
Infrared Remote Control (IRC 2)				
%SB6012	0	96	CFW_STS_IRC_2_DRY	Dry key pressed
%SB6012	1	97	CFW_STS_IRC_2_CLEAN	Clean key pressed
%SB6012	2	98	CFW_STS_IRC_2_TIMER	Timer key pressed
%SB6012	3	99	CFW_STS_IRC_2_FUNC	Func function active
%SB6012	4	100	CFW_STS_IRC_2_SWING	Swing function active

%SB6012	5	101	CFW_STS_IRC_2_COOL	Cool key pressed
%SB6012	6	102	CFW_STS_IRC_2_MODE	Mode key pressed
%SB6012	7	103	CFW_STS_IRC_2_POWER	Pow er key pressed
%SB6013	0	104	CFW_STS_IRC_2_TEMP_UNIT	Temp unit key pressed
%SB6013	1	105	CFW_STS_IRC_2_UVC	UVC function active

Reading - Function Modbus 04 "Read Input Registers"

Speed				
% SW6200	--	3100	CFW_STS_MOTOR_SPEED_13BITS	Motor speed in 13 bits (8192)
% SW6202	--	3101	CFW_STS_MOTOR_SYNC_SPEED	Motor synchronous speed in rpm
% SW6204	--	3102	CFW_STS_MOTOR_SPEED_RPM	Motor speed in rpm
% SW6206	--	3103	CFW_STS_SPEED_REFERENCE	Speed reference after ramp in rpm
Alarm and Fault				
% SW6208	--	3104	CFW_STS_PRES_ALARM	Alarm number that may be present in the inverter
% SW6210	--	3105	CFW_STS_PRES_FAULT	Fault number that may be present in the inverter
Current and Torque				
% SW6212	--	3106	CFW_STS_RATED_CURRENT	Inverter rated current (HD) in A (x10)
% SW6214	--	3107	CFW_STS_MOTOR_CURRENT	Motor current w ithout filter in A (x10)
% SW6216	--	3108	CFW_STS_MOTOR_TORQUE	Motor torque w ithout filter in % (x10)
Infrared Remote Control				
% SW6218	--	3109	CFW_STS_IRC_2_INFO	IRC_2 info

Writing / Reading System Markers (Command)

Reading - Function Modbus 01 "Read Coils"
Writing - Function Modbus 05 "Write Single Coil" and 15 "Write Multiple Coils"

Address	Bit	Modbus	Tag	Description
Logical Command				
%CB6008	0	16	CFW_CMD_RUN_STOP	Run the motor according to the speed reference value (0-Stop, 1-Run)
%CB6008	1	17	CFW_CMD_GENERAL_ENABLE	Enables the inverter allowing the motor operation (0-Disable, 1-Enable)
%CB6008	2	18	CFW_CMD_SPEED_DIRECTION	The motor runs in the direction indicated by the speed reference (0-Reverse, 1-Forward)
%CB6008	3	19	CFW_CMD_JOG	Enables the JOG function (0-Disable, 1-Enable)
%CB6008	4	20	CFW_CMD_LOC_REM	Selects the inverter operation mode (0-Local, 1-Remote)
%CB6008	5	21	CFW_CMD_SECOND_RAMP	Selects the ramp to accelerate and decelerate the motor (0-First, 1-Second)
%CB6008	6	22	CFW_CMD_FORCE_RUN_STOP_SPLC	It allows that the SoftPLC command CFW_CMD_RUN_STOP change the inverter command Run/Stop regardless of source programmed for Start/Stop via P224 or P227
%CB6008	7	23	CFW_CMD_FAULT_RESET	Executes the fault reset command

11.2.3 I/O's

Hardware information can be found in the Manual of the CFW300 at the website www.weg.net.

Digital Inputs

Endereço	Bit	Modbus	Tag	Descrição
%IB0	0	16000	DI1	Digital input 1
%IB0	1	16001	DI2	Digital input 2
%IB0	2	16002	DI3	Digital input 3
%IB0	3	16003	DI4	Digital input 4
%IB0	4	16004	DI5	Digital input 5 - I/O expansion module
%IB0	5	16005	DI6	Digital input 6 - I/O expansion module
%IB0	6	16006	DI7	Digital input 7 - I/O expansion module
%IB0	7	16007	DI8	Digital input 8 - I/O expansion module

Analog Inputs

Endereço	Bit	Modbus	Tag	Descrição
%IW2	--	5001	AI1	Analog input 1
%IW4	--	5002	AI2	Analog input 2 - I/O expansion module

Added from firmware version V2.00

Endereço	Bit	Modbus	Tag	Descrição
%IW6	--	5003	AIP	Analog input (Potentiometer) - I/O expansion module
%IW8	--	5004	FI1	Frequency input 1
%IW10	--	5005	FI2	Frequency input 2 - I/O expansion module
%IW12	--	5006	FI3	Frequency input 3 - I/O expansion module
%IW14	--	5007	FI4	Frequency input 4 - I/O expansion module

Digital Outputs

Endereço	Bit	Modbus	Tag	Descrição
%QB0	0	16000	DO1	Digital output 1
%QB0	1	16001	DO2	Digital output 2 - I/O expansion module
%QB0	2	16002	DO3	Digital output 3 - I/O expansion module
%QB0	3	16003	DO4	Digital output 4 - I/O expansion module

Analog Outputs

Endereço	Bit	Modbus	Tag	Descrição
%QW2	--	5001	AO1	Analog output 1
%QW4	--	5002	AO2	Analog output 1 - I/O expansion module

Added from firmware version V2.00

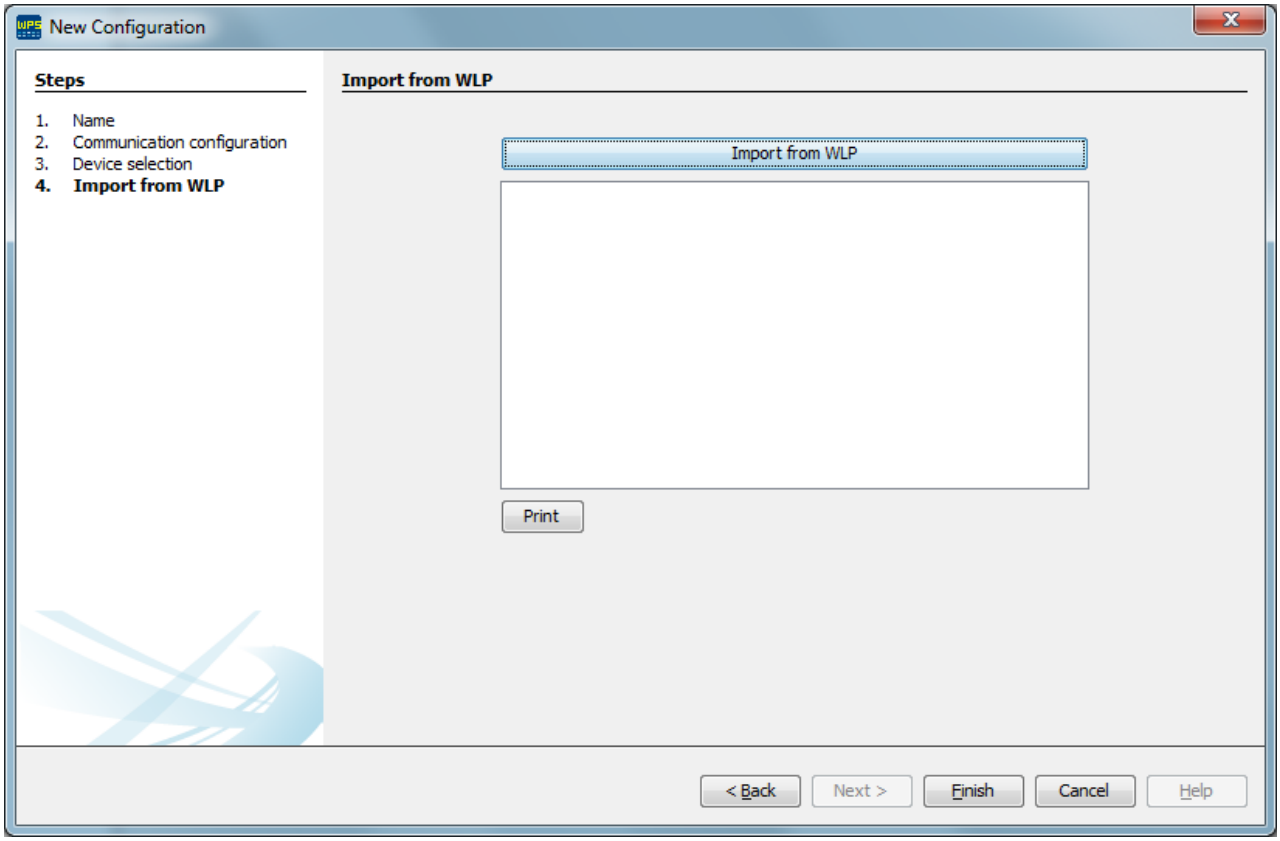
Endereço	Bit	Modbus	Tag	Descrição
%QW6	--	5003	FO1	Frequency output 1 - I/O expansion module
%QW8	--	5004	FO2	Frequency output 2 - I/O expansion module
%QW10	--	5005	FO3	Frequency output 3 - I/O expansion module

NOTE!
The addresses of the digital and analog outputs were changed from firmware version 1.20. To convert the variables it is necessary to change the addresses:
%QB6 => %QB0
%QW8 => %QW2
%QW10 => %QW4

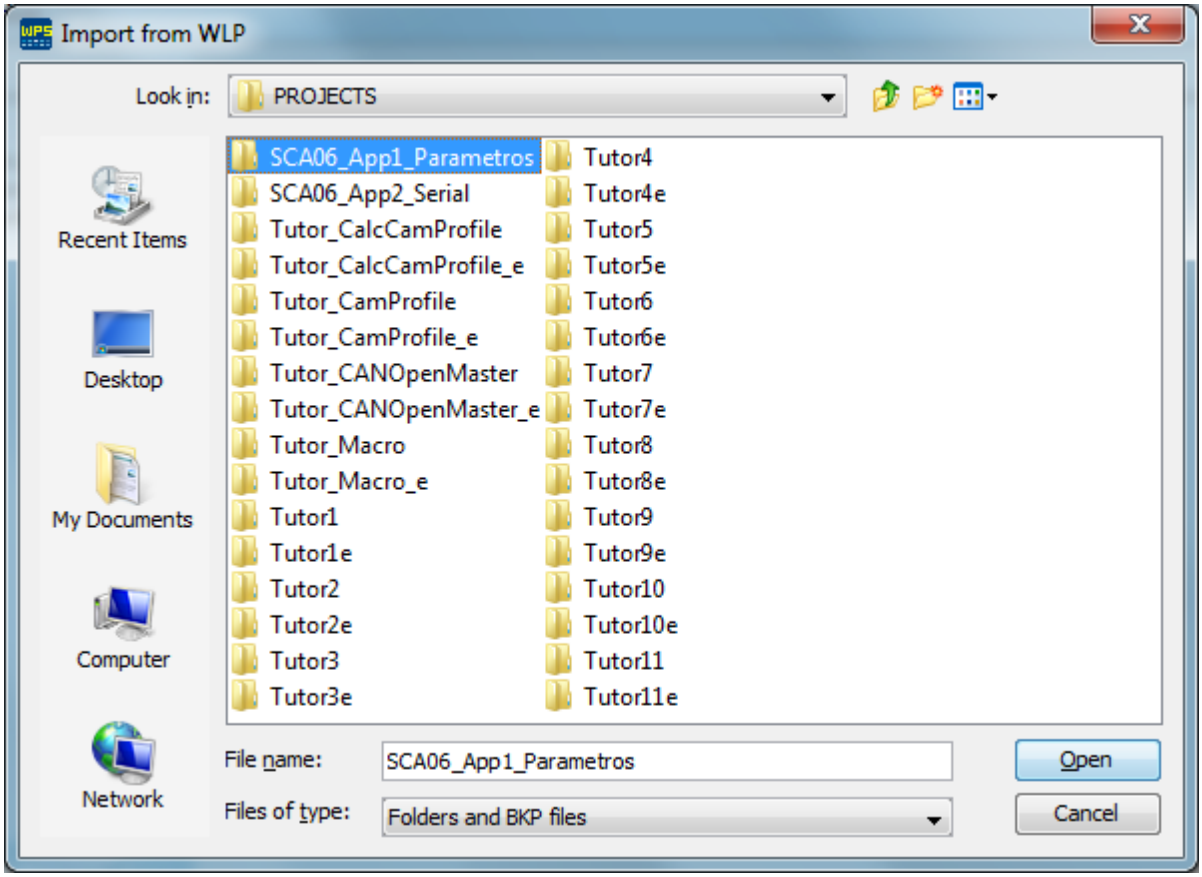
11.2.4 Import from WLP

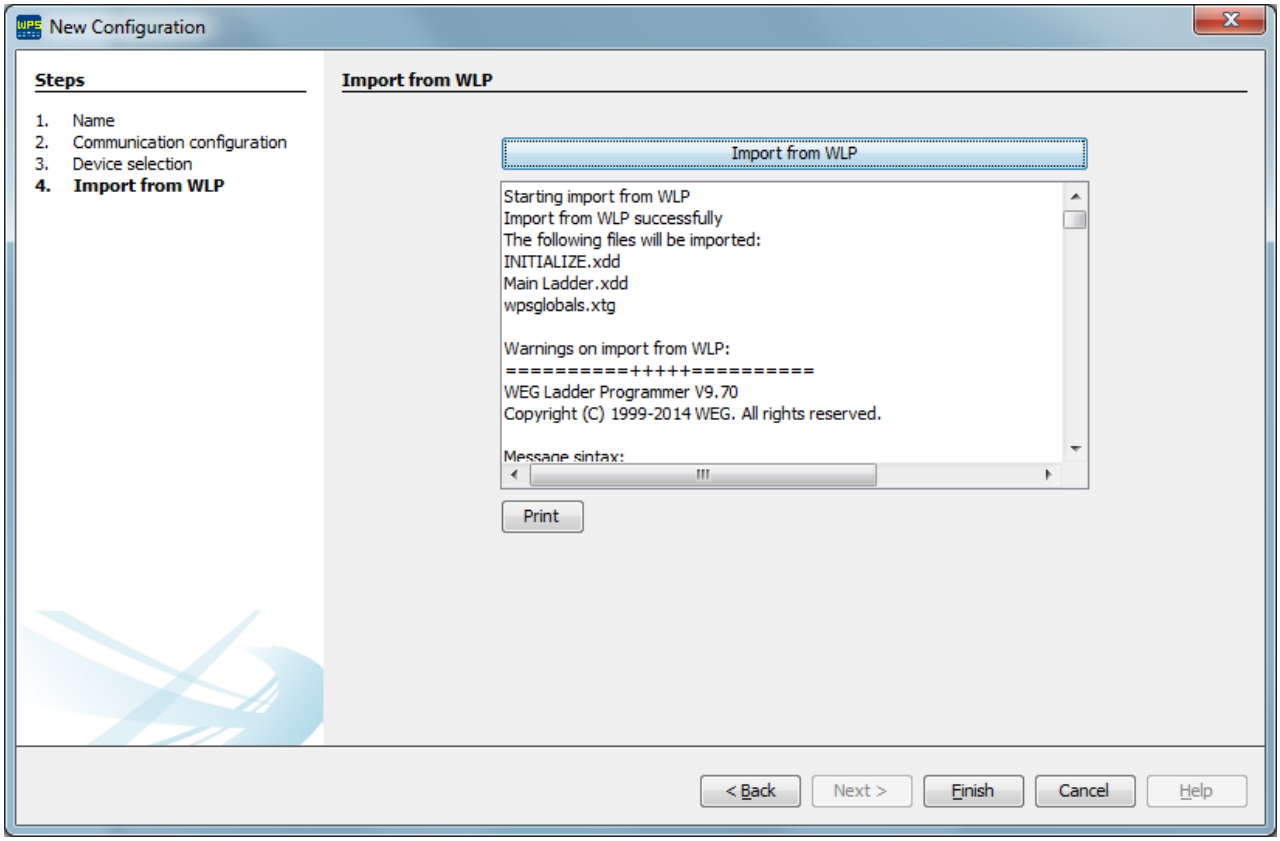
The function import from WLP is utilized to import Ladder developed on WLP software to equipment (device).

The import from WLP can be executed during the resource creation.



1. To execute the import WLP function click the Import from WLP button and select the WLP project folder or the WLP BKP file.


Import from WLP
Look in:
PROJECTS
SCA06_App1_Parametros
SCA06_App2_Serial
Tutor_CalcCamProfile
Tutor_CalcCamProfile_e
Tutor_CamProfile
Tutor_CamProfile_e
Tutor_CANOpenMaster
Tutor_CANOpenMaster_e
Tutor_Macro
Tutor_Macro_e
Tutor1
Tutor1e
Tutor2
Tutor2e
Tutor3
Tutor3e
Tutor4
Tutor4e
Tutor5
Tutor5e
Tutor6
Tutor6e
Tutor7
Tutor7e
Tutor8
Tutor8e
Tutor9
Tutor9e
Tutor10
Tutor10e
Tutor11
Tutor11e
Recent Items
Desktop
My Documents
Computer
Network
File name:
SCA06_App1_Parametros
Files of type:
Folders and BKP files
Open
Cancel



2. After import from WLP completed successfully click the Finish button to copy the imported files to new resource.

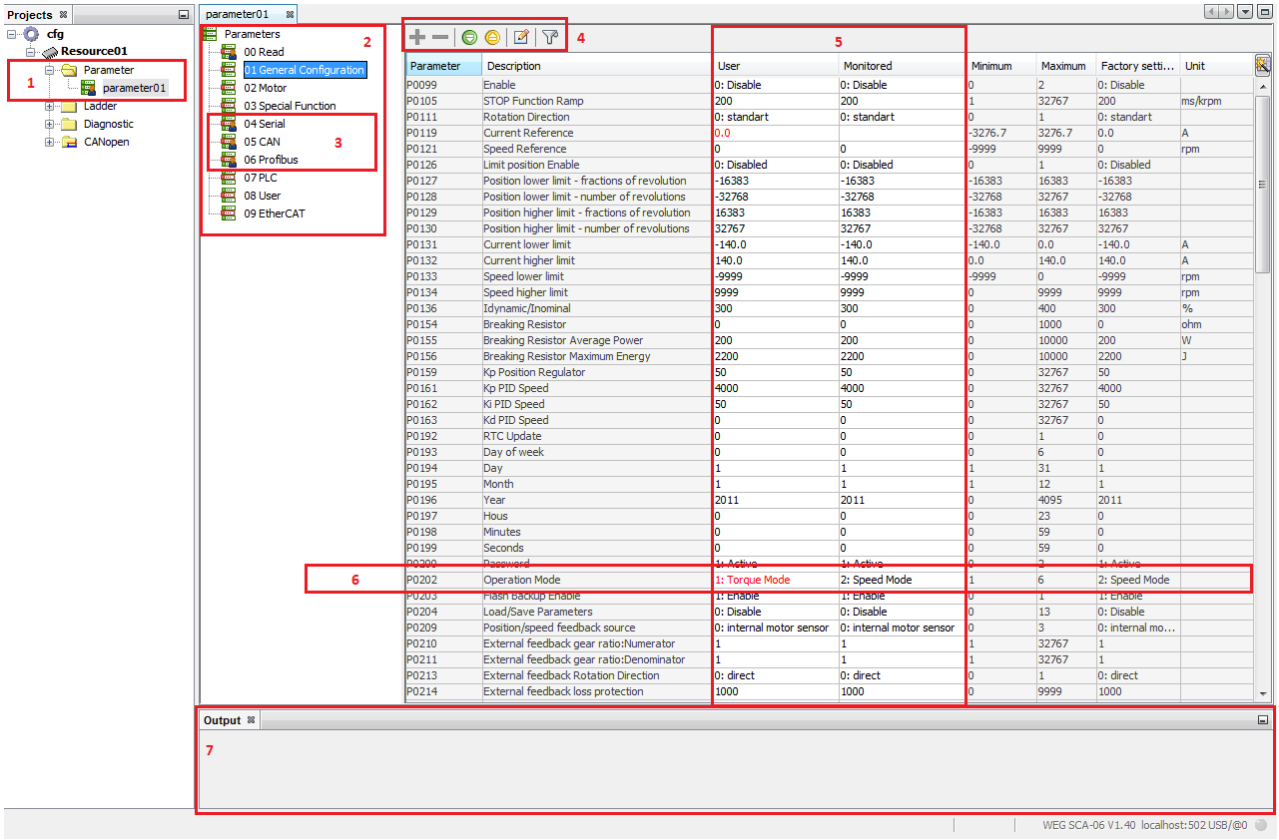
11.2.5 Parameters

11.2.5.1 Overview

The parameter configuration screen is used to configure and monitor all the parameters of the equipment, including the user parameters.

NOTE!
The reading and writing of such parameters is done on this screen; only the user parameter configuration must be sent the first time or whenever modified by means of the resource download routine.

Below is an overview of the parameter configuration screen.



1. **Parameter files**. In this part are all the parameter configuration files created by the user. Notice that when the file features a person figure on the table, it means this parameter table contains hidden parameters/ group of parameters.
2. **Group of parameters**. This tree shows all the group of parameters. Notice that the same parameter can be in more than one group, and when its value is modified, it will be modified in all the groups to which it belongs.
3. **Modified group of parameters**. Group of parameters which contain the figure of a person on the table means they have hidden parameters.
4. **Commands**. The commands are described below in the order they appear:
   4.1. **Unhide parameter**: In case some parameter has been hidden, this button allows making it visible again.
   4.2. **Hide parameter**: Just select one or more parameters on the table and trigger this command to hide them.
   4.3. **Save table**: It saves the values of the parameters shown on the equipment screen; the sent values are the ones in the User column. The flow is User -> Monitored (equipment)
   4.4. **Read table**: It reads the parameters of the equipment shown in the Monitored column and saves them in the parameter file in the User column. The flow is Monitored (equipment) -> User
   4.5. **User parameters**: It opens a screen to edit the user parameters.
   4.6. **Filter**: It opens a parameter filter option, and it can filter by parameter number or description.
   4.7. **User Parameters and Monitored Parameters**. These two columns show the off-line and on-line parameters, so to speak. The **User** column shows the values contained in the file located on the computer and the **Monitored** column shows the values that are effectively saved on the equipment. Whenever you use the **Save Parameter** option, the sent values will be from the **User** column to the **Monitored** column, that is, File -> Equipment. In case of reading, the flow is the opposite, from the **Monitored** column to the **User** column, that is, Equipment -> File. In case you wish to change the values directly on the equipment without changing it in the file, just click on the monitored column,

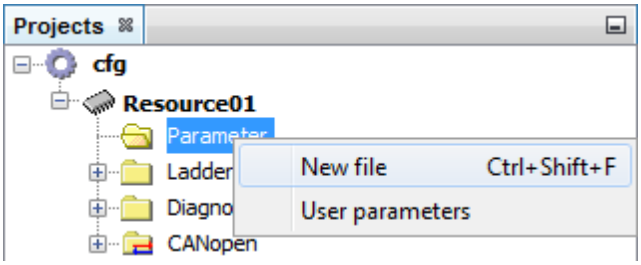
change the values and the modification will occur on-line.

5. **Modified parameters**: Whenever a parameter value in the **User** column is different from the **Monitored** column, it will be shown in red.
6. **Output**. This screen shows error information in case they occur during the writing or reading of the parameters.

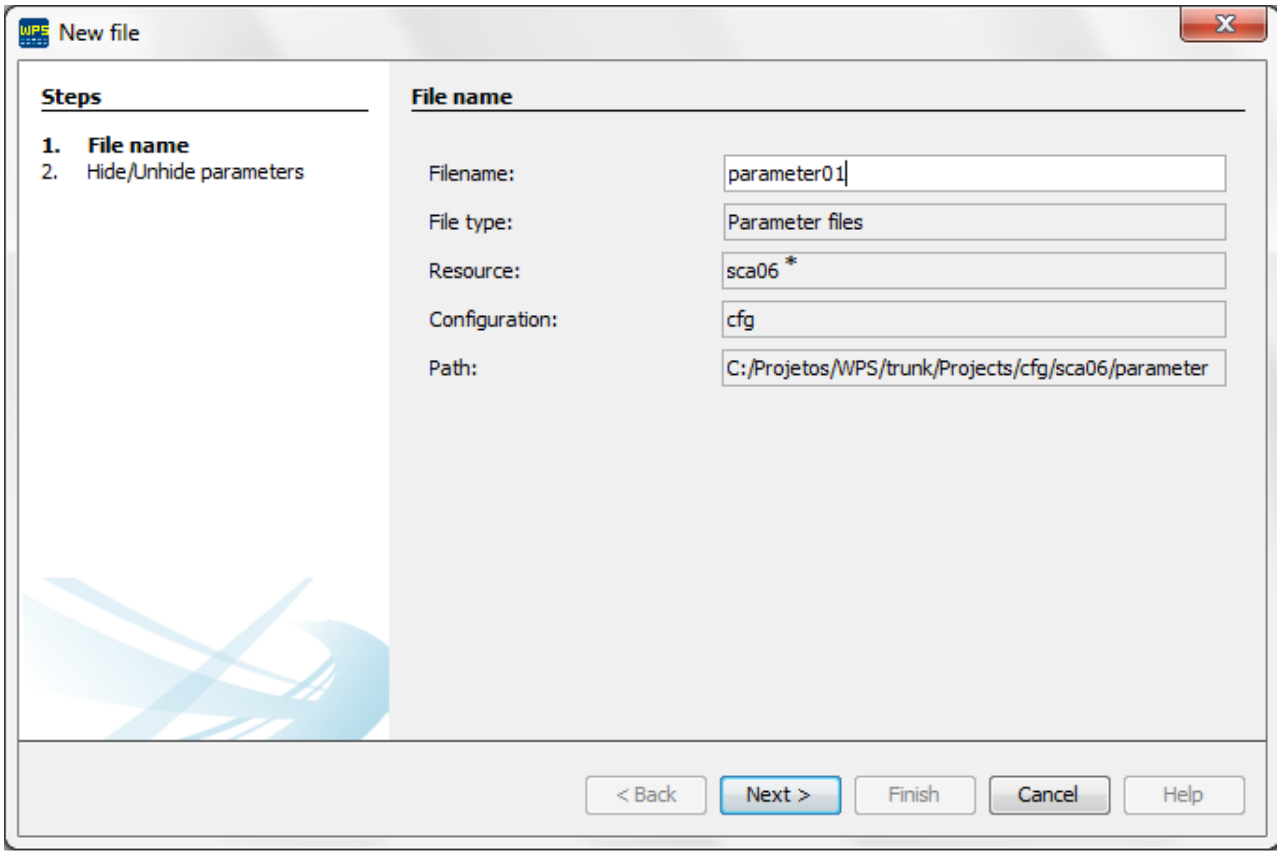
11.2.5.2 Configuration

Below is the list of the required steps to create a parameter file.

1. Create a new parameter file.

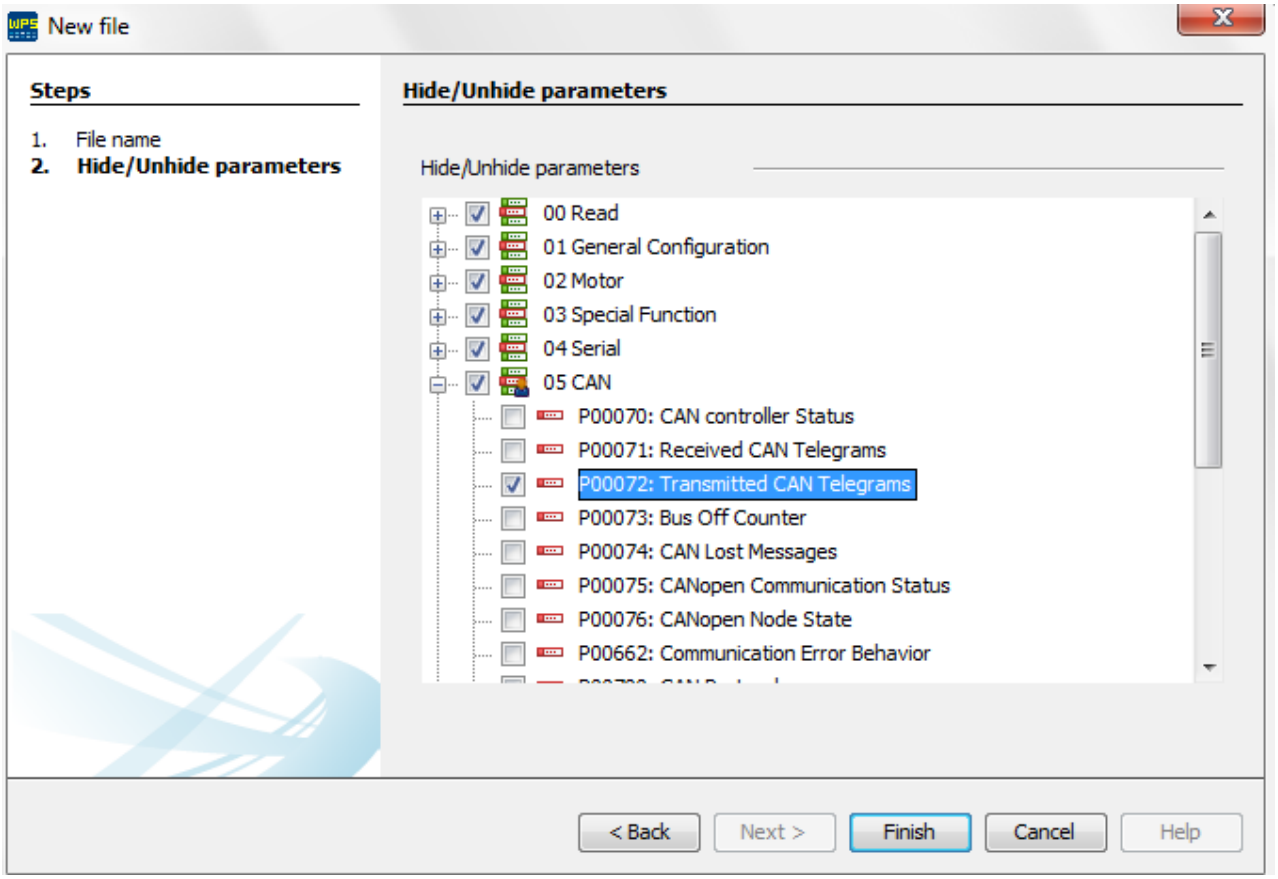


2. Define a name for the parameter file

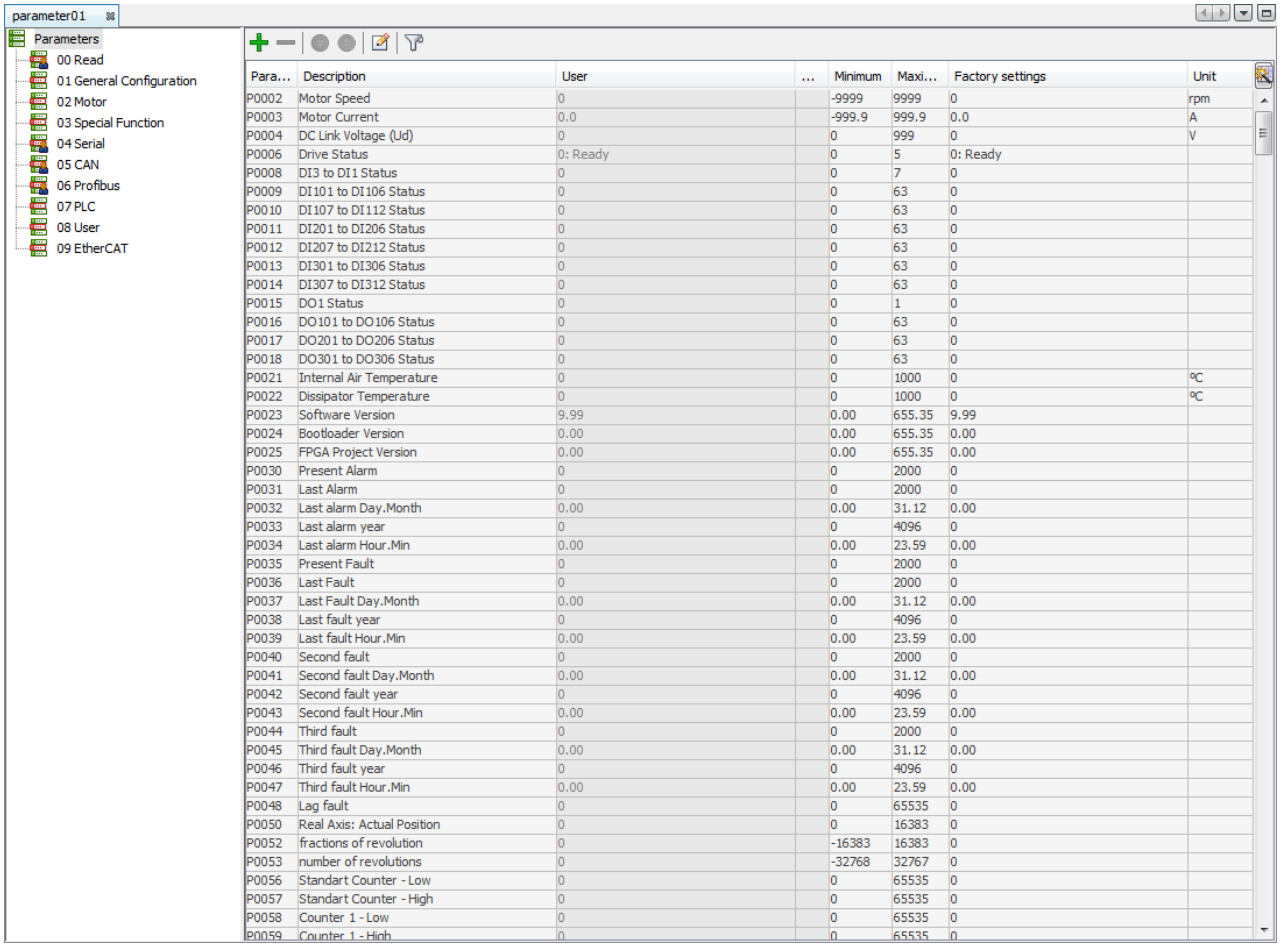


*** Resource: Resource01, SCA06, CFW300, etc.**

3. Configure which parameters you wish to view in your parameter table



4. After performing the steps above, the parameter file will be created and the equipment can be parameterized.

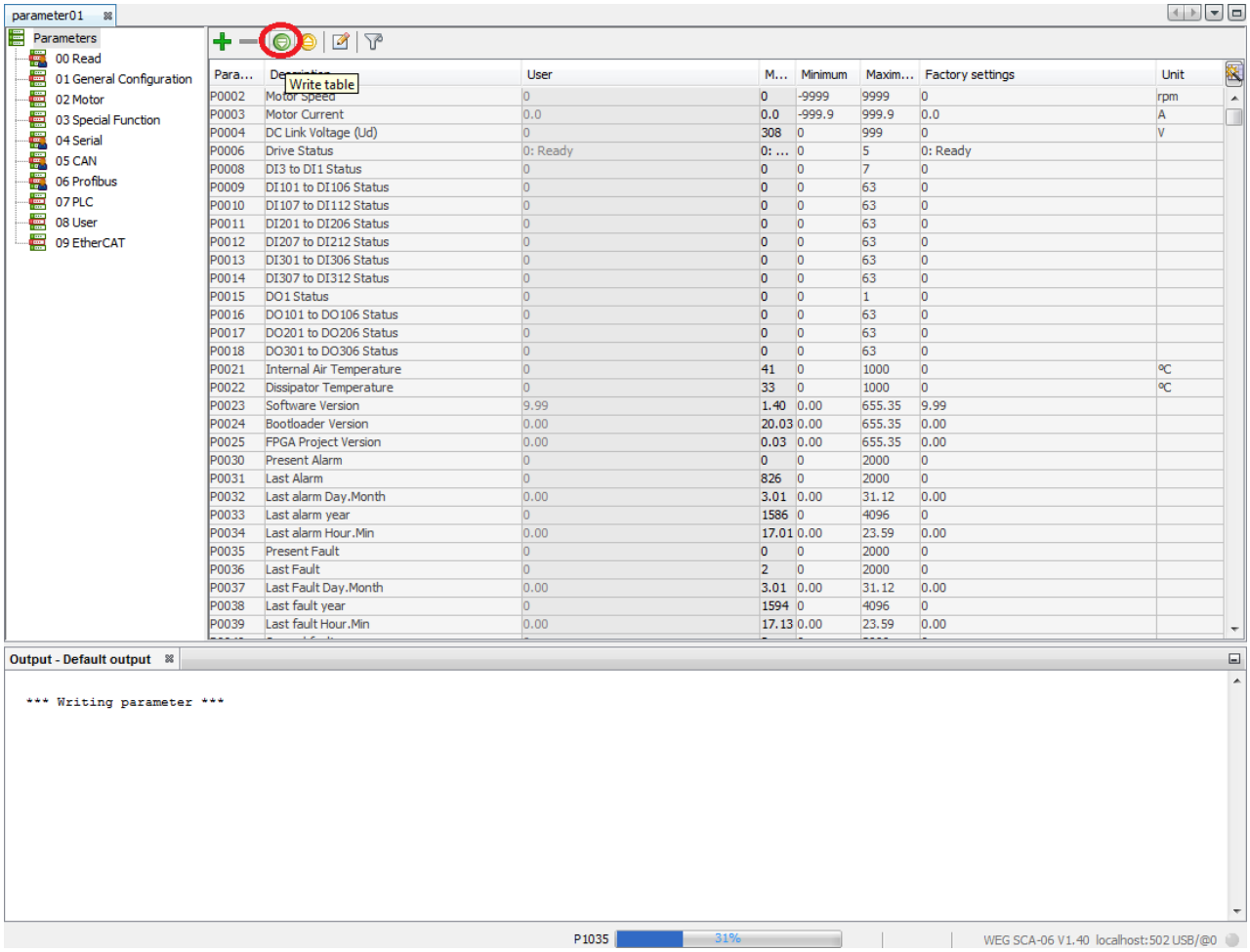


Para...	Description	User	...	Minimum	Maxi...	Factory settings	Unit
P0002	Motor Speed	0		-9999	9999	0	rpm
P0003	Motor Current	0.0		-999.9	999.9	0.0	A
P0004	DC Link Voltage (Ud)	0		0	999	0	V
P0006	Drive Status	0: Ready		0	5	0: Ready	
P0008	DI3 to DI1 Status	0		0	7	0	
P0009	DI101 to DI106 Status	0		0	63	0	
P0010	DI107 to DI112 Status	0		0	63	0	
P0011	DI201 to DI206 Status	0		0	63	0	
P0012	DI207 to DI212 Status	0		0	63	0	
P0013	DI301 to DI306 Status	0		0	63	0	
P0014	DI307 to DI312 Status	0		0	63	0	
P0015	DO1 Status	0		0	1	0	
P0016	DO101 to DO106 Status	0		0	63	0	
P0017	DO201 to DO206 Status	0		0	63	0	
P0018	DO301 to DO306 Status	0		0	63	0	
P0021	Internal Air Temperature	0		0	1000	0	ºC
P0022	Dissipator Temperature	0		0	1000	0	ºC
P0023	Software Version	9.99		0.00	655.35	9.99	
P0024	Bootloader Version	0.00		0.00	655.35	0.00	
P0025	FPGA Project Version	0.00		0.00	655.35	0.00	
P0030	Present Alarm	0		0	2000	0	
P0031	Last Alarm	0		0	2000	0	
P0032	Last alarm Day.Month	0.00		0.00	31.12	0.00	
P0033	Last alarm year	0		0	4096	0	
P0034	Last alarm Hour.Min	0.00		0.00	23.59	0.00	
P0035	Present Fault	0		0	2000	0	
P0036	Last Fault	0		0	2000	0	
P0037	Last Fault Day.Month	0.00		0.00	31.12	0.00	
P0038	Last fault year	0		0	4096	0	
P0039	Last fault Hour.Min	0.00		0.00	23.59	0.00	
P0040	Second fault	0		0	2000	0	
P0041	Second fault Day.Month	0.00		0.00	31.12	0.00	
P0042	Second fault year	0		0	4096	0	
P0043	Second fault Hour.Min	0.00		0.00	23.59	0.00	
P0044	Third fault	0		0	2000	0	
P0045	Third fault Day.Month	0.00		0.00	31.12	0.00	
P0046	Third fault year	0		0	4096	0	
P0047	Third fault Hour.Min	0.00		0.00	23.59	0.00	
P0048	Lag fault	0		0	65535	0	
P0050	Real Axis: Actual Position	0		0	16383	0	
P0052	fractions of revolution	0		-16383	16383	0	
P0053	number of revolutions	0		-32768	32767	0	
P0056	Standart Counter - Low	0		0	65535	0	
P0057	Standart Counter - High	0		0	65535	0	
P0058	Counter 1 - Low	0		0	65535	0	
P0059	Counter 1 - High	0		0	65535	0	

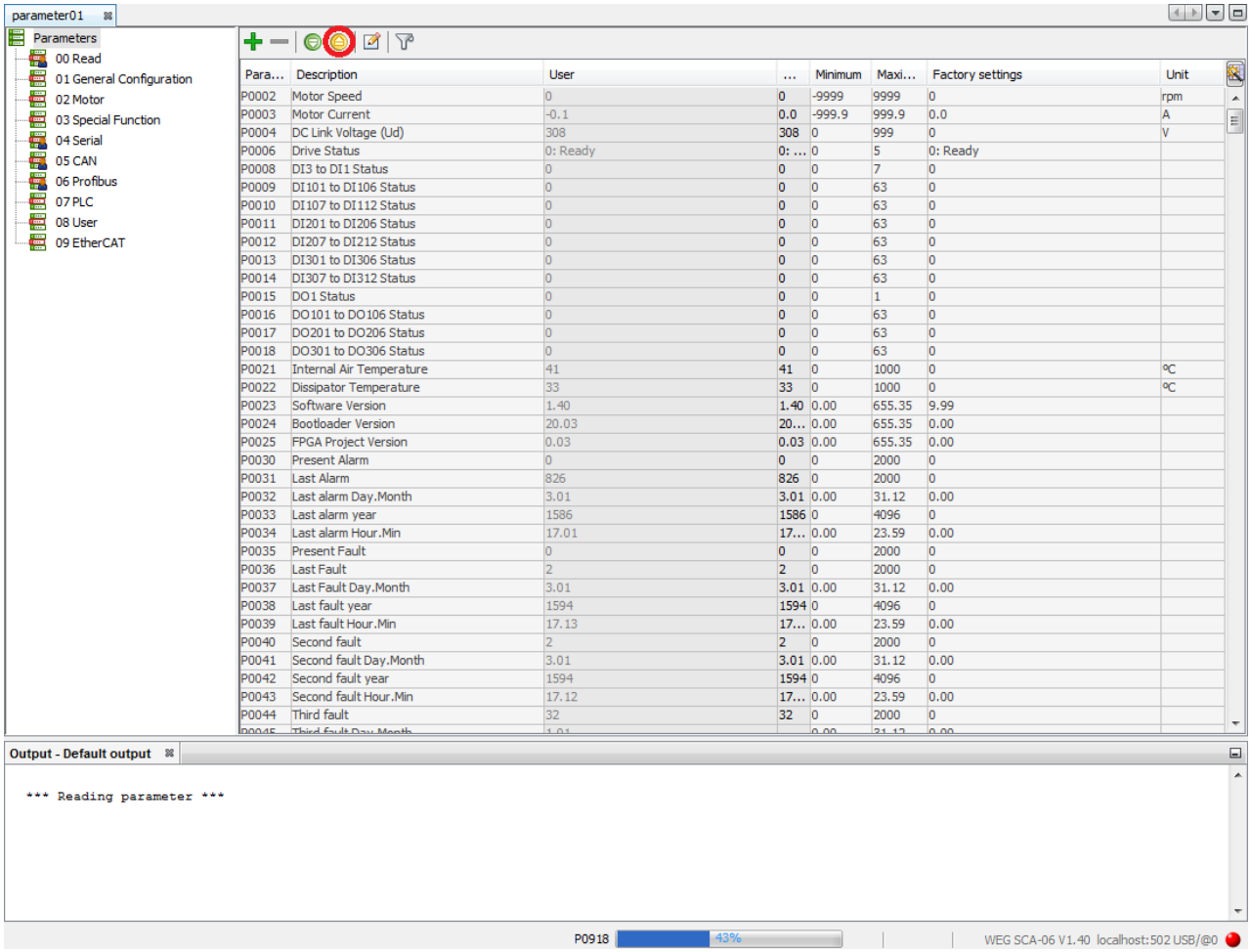
11.2.5.3 Read and Write of Parameters

There are 3 (three) ways to do the reading and writing of the parameters: by means of table, selection and group.

1. **Table writing**. The table writing command will send all visible parameters on the equipment screen. If and error occurs during the sending of some specific parameter, a message will be shown on the output window informing the error. It is important to notice that only visible parameters will be sent; therefore, it is necessary attention to which node of the group of parameters tree you are viewing. Example: If you wish to write all of them without filtering per group, just select the tree root.

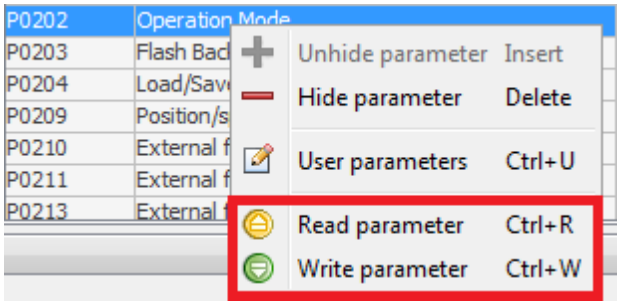


2. **Table reading**. The table reading command will read all the parameters of the equipment. If a error occurs during the reading of some specific parameter, a message will be shown on the output window informing the error. It is important to notice that only visible parameters will be read; therefore, it is necessary attention to which node of the group of parameters tree you are viewing. Example: If you wish to read all of them without filtering per group, just select the tree root.

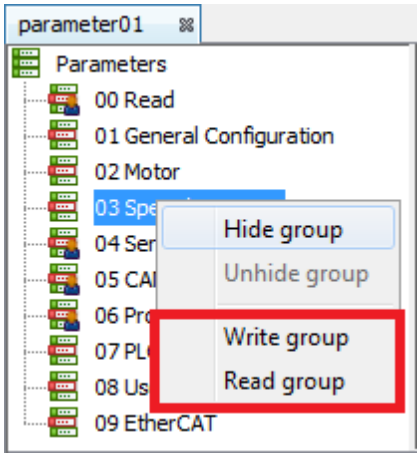


Para...	Description	User	...	Minimum	Maxi...	Factory settings	Unit
P0002	Motor Speed	0	0	-9999	9999	0	rpm
P0003	Motor Current	-0.1	0.0	-999.9	999.9	0.0	A
P0004	DC Link Voltage (Ud)	308	308	0	999	0	V
P0006	Drive Status	0: Ready	0: ...	0	5	0: Ready	
P0008	DI3 to DI1 Status	0	0	0	7	0	
P0009	DI101 to DI106 Status	0	0	0	63	0	
P0010	DI107 to DI112 Status	0	0	0	63	0	
P0011	DI201 to DI206 Status	0	0	0	63	0	
P0012	DI207 to DI212 Status	0	0	0	63	0	
P0013	DI301 to DI306 Status	0	0	0	63	0	
P0014	DI307 to DI312 Status	0	0	0	63	0	
P0015	DO1 Status	0	0	0	1	0	
P0016	DO101 to DO106 Status	0	0	0	63	0	
P0017	DO201 to DO206 Status	0	0	0	63	0	
P0018	DO301 to DO306 Status	0	0	0	63	0	
P0021	Internal Air Temperature	41	41	0	1000	0	°C
P0022	Dissipator Temperature	33	33	0	1000	0	°C
P0023	Software Version	1.40	1.40	0.00	655.35	9.99	
P0024	Bootloader Version	20.03	20...	0.00	655.35	0.00	
P0025	FPGA Project Version	0.03	0.03	0.00	655.35	0.00	
P0030	Present Alarm	0	0	0	2000	0	
P0031	Last Alarm	826	826	0	2000	0	
P0032	Last alarm Day.Month	3.01	3.01	0.00	31.12	0.00	
P0033	Last alarm year	1586	1586	0	4096	0	
P0034	Last alarm Hour.Min	17.01	17...	0.00	23.59	0.00	
P0035	Present Fault	0	0	0	2000	0	
P0036	Last Fault	2	2	0	2000	0	
P0037	Last Fault Day.Month	3.01	3.01	0.00	31.12	0.00	
P0038	Last fault year	1594	1594	0	4096	0	
P0039	Last fault Hour.Min	17.13	17...	0.00	23.59	0.00	
P0040	Second fault	2	2	0	2000	0	
P0041	Second fault Day.Month	3.01	3.01	0.00	31.12	0.00	
P0042	Second fault year	1594	1594	0	4096	0	
P0043	Second fault Hour.Min	17.12	17...	0.00	23.59	0.00	
P0044	Third fault	32	32	0	2000	0	

3. **Reading/writing of specific parameters**. In order to read/write one or more specific parameters, just select them on the table, right click and choose the desired option: read or write parameter.



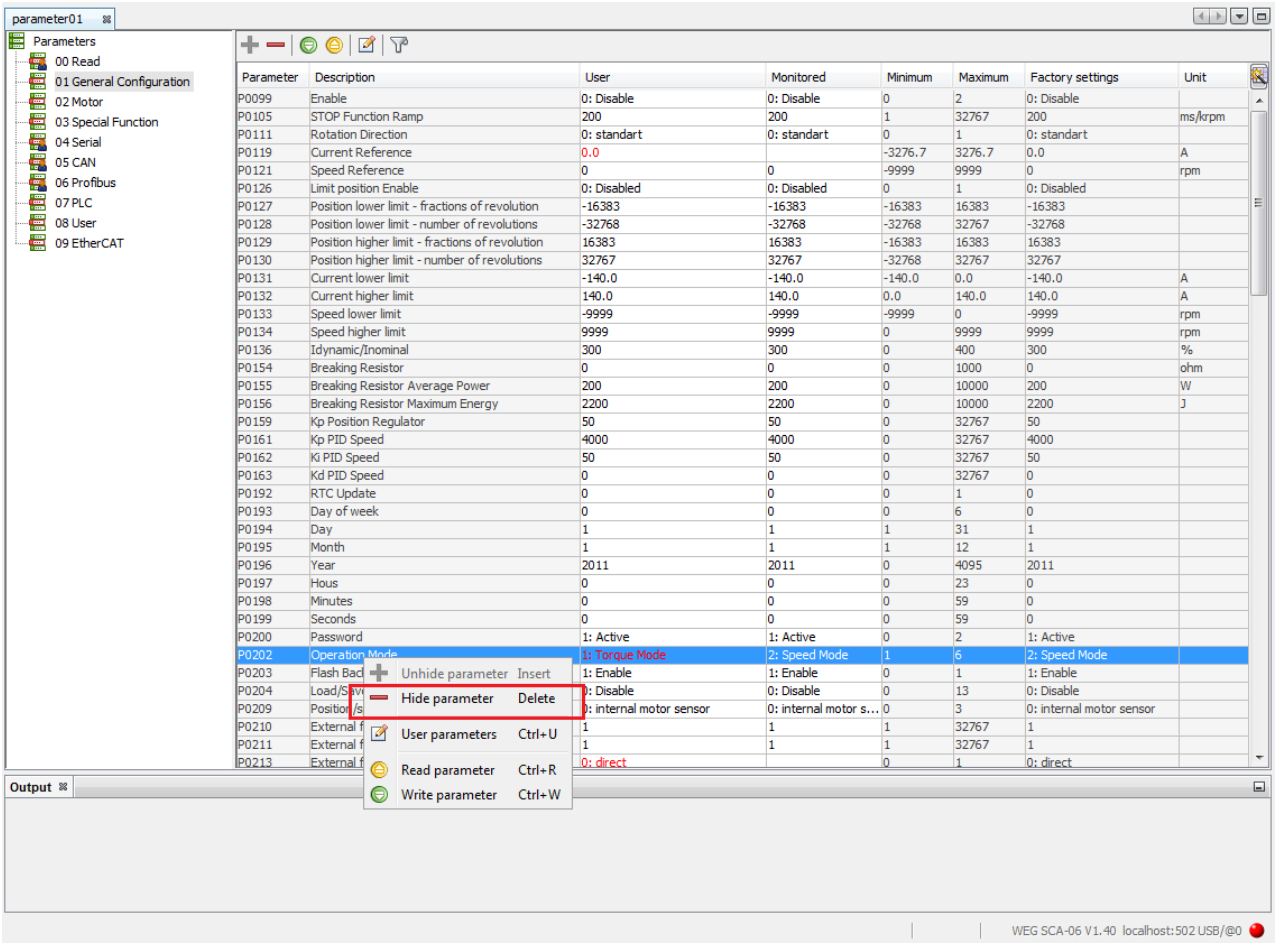
4. **Reading/writing of group of parameters**. In order to read/write only one group of parameters, just select it on the group tree, right click and choose the desired option: read or write group.



11.2.5.4 Hide/Unhide Parameters and Group of Parameters

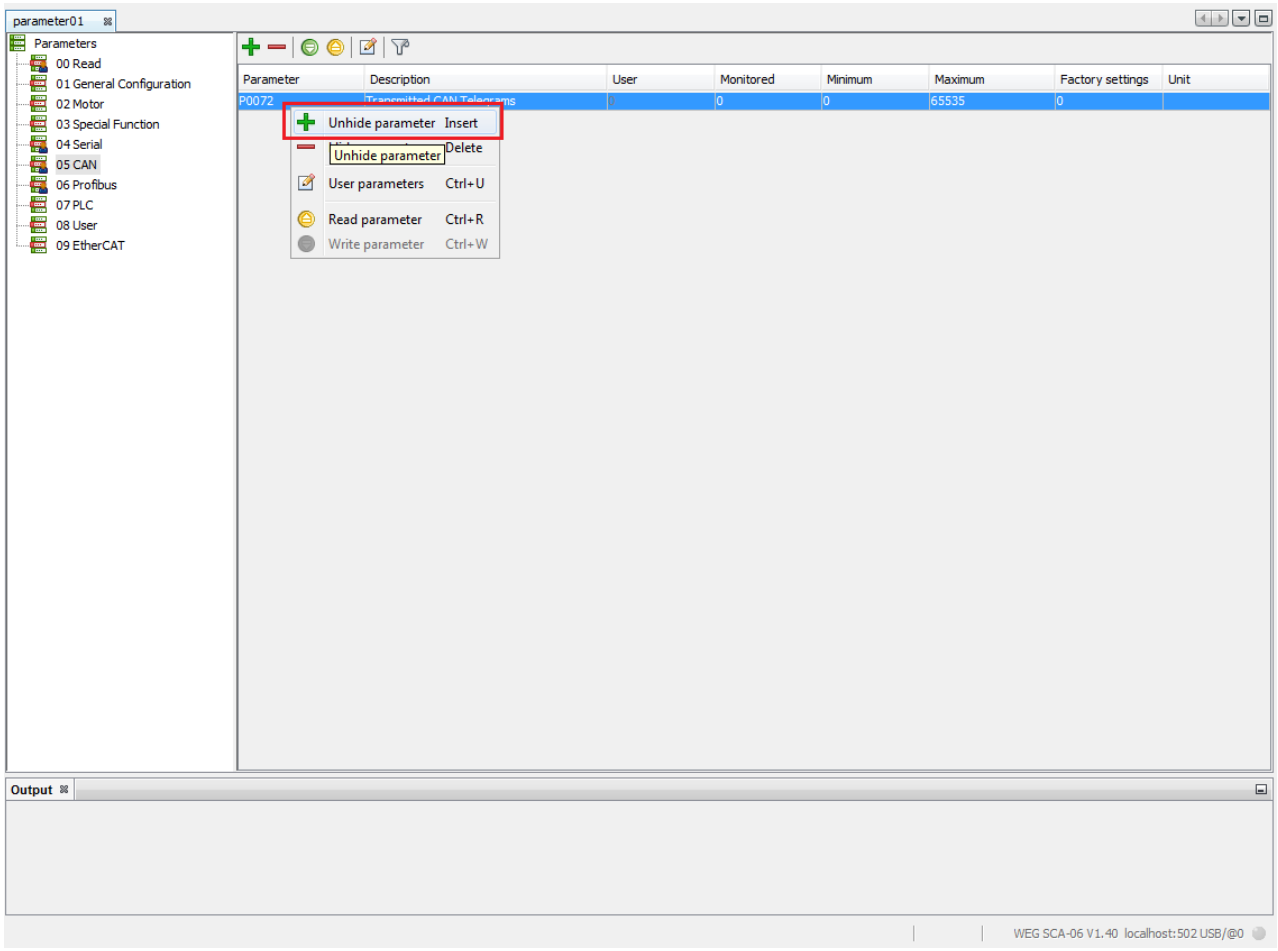
The parameter can be hidden/unhidden in two ways: individually or in group.

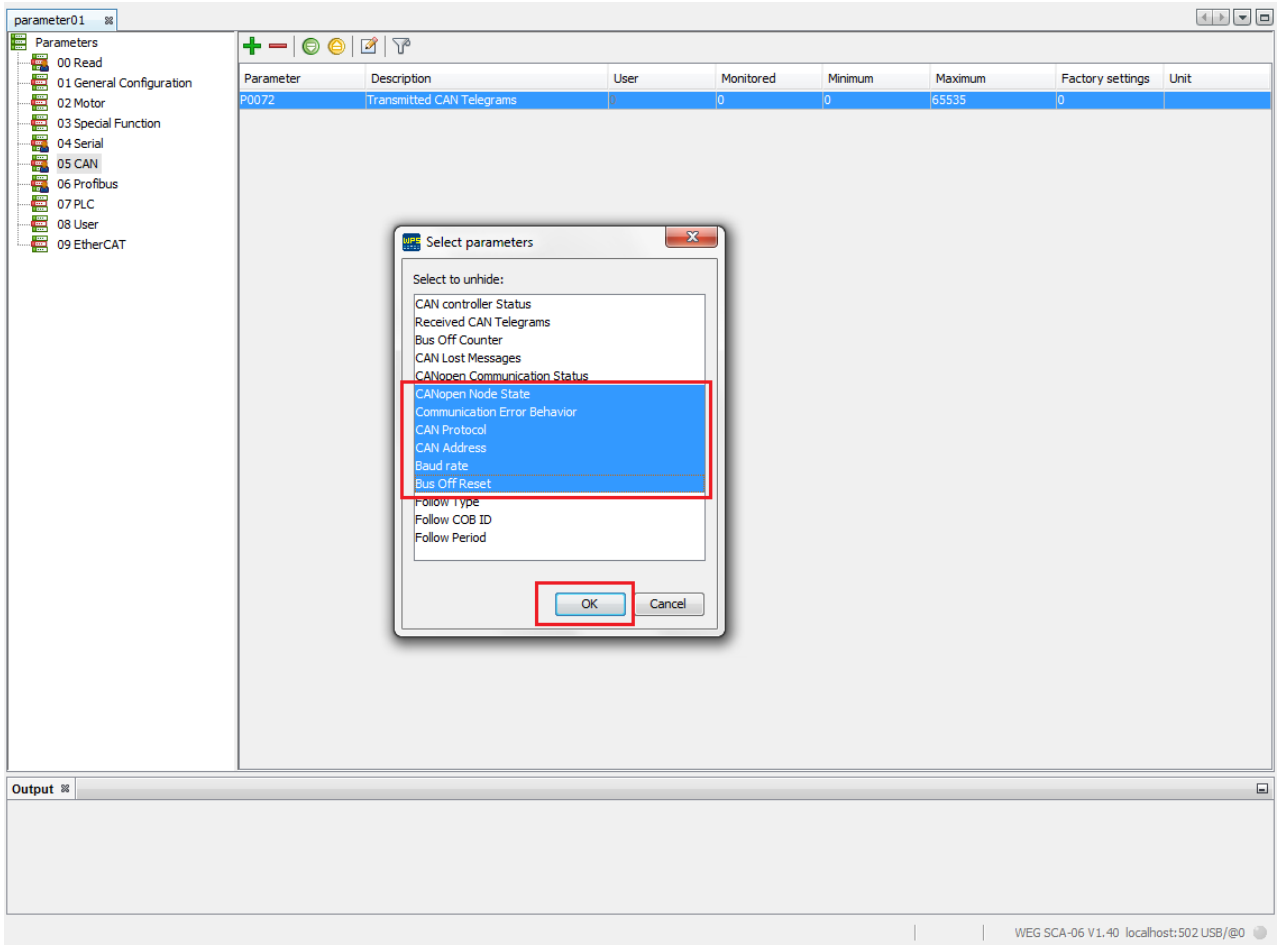
1. **Hide parameters**. In order to hide a parameter individually, just right click on the desired parameters and select the **Hide Parameter** option. You can also press the **Delete** key.

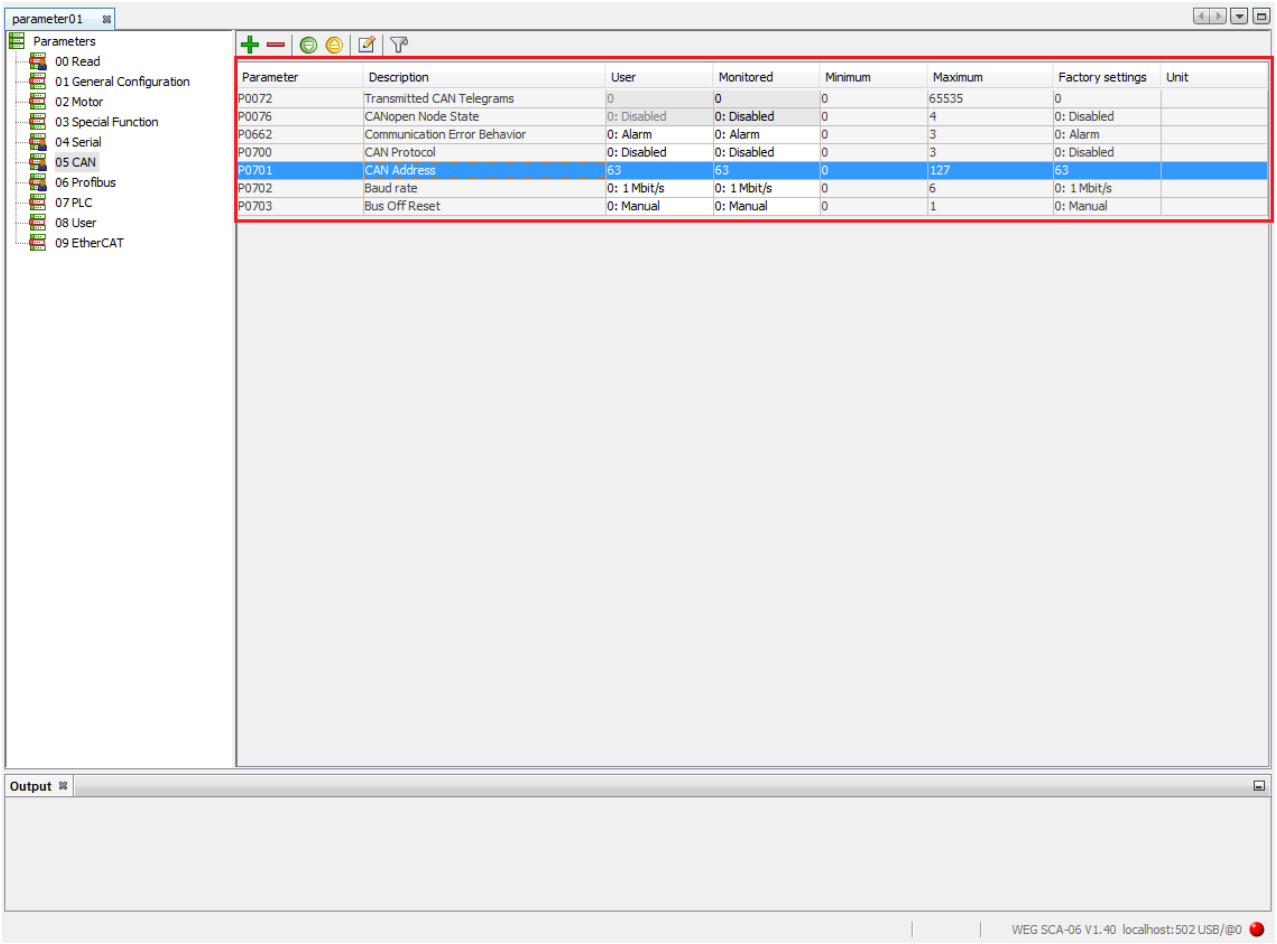


2. **Unhide Parameters**. In order to show hidden parameters, right click and choose the **Unhide Parameters**

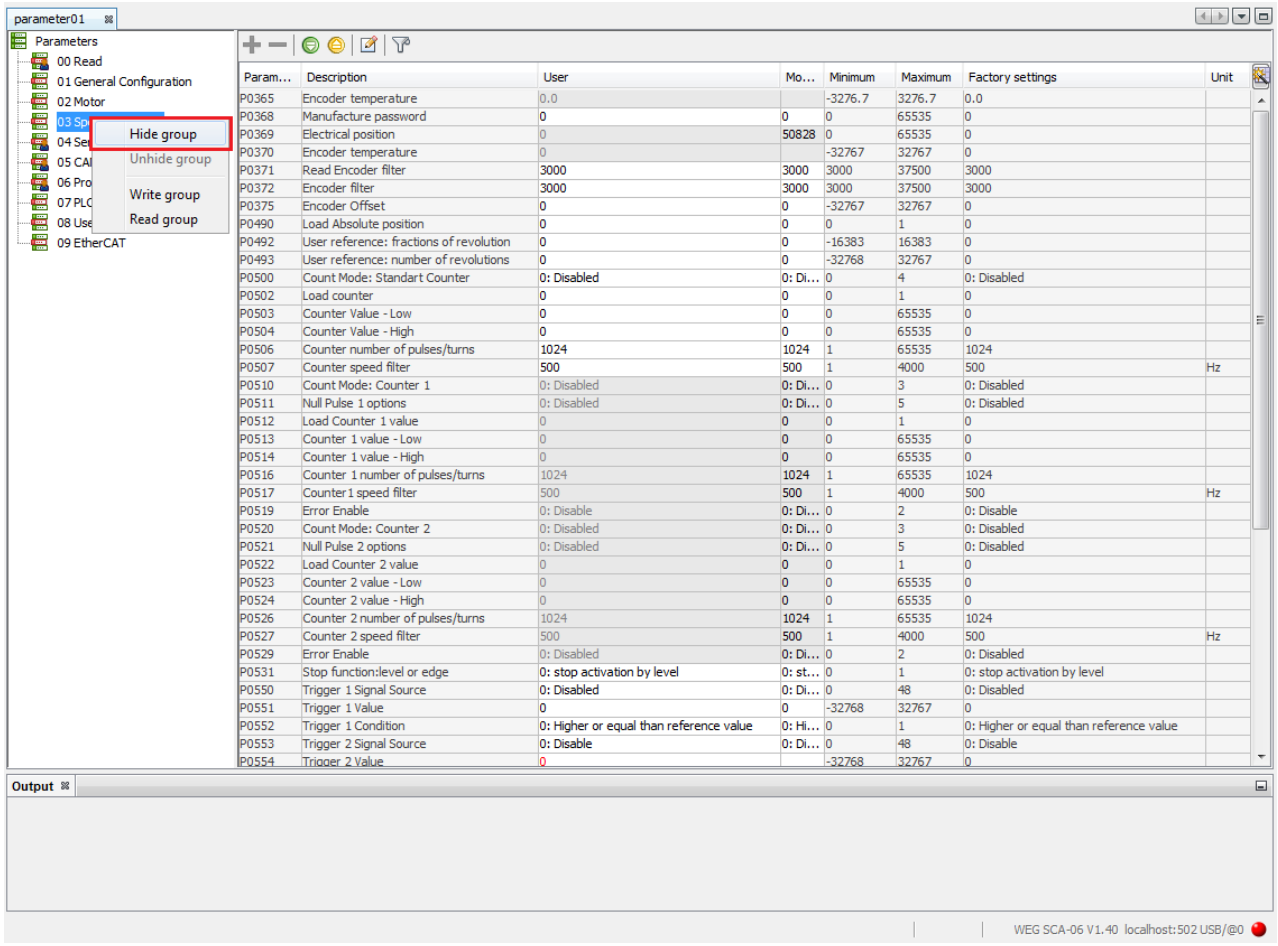
or press the **Insert** key. Then, a window will open and show the hidden parameters. Now, you just have to select the desired parameters and confirm.
Note: The parameters shown on this new window are only those which belong to the current filter according to the selection on the parameter group tree. In the figures below, the CAN group is selected; that means that only the hidden parameters of this group will be shown.

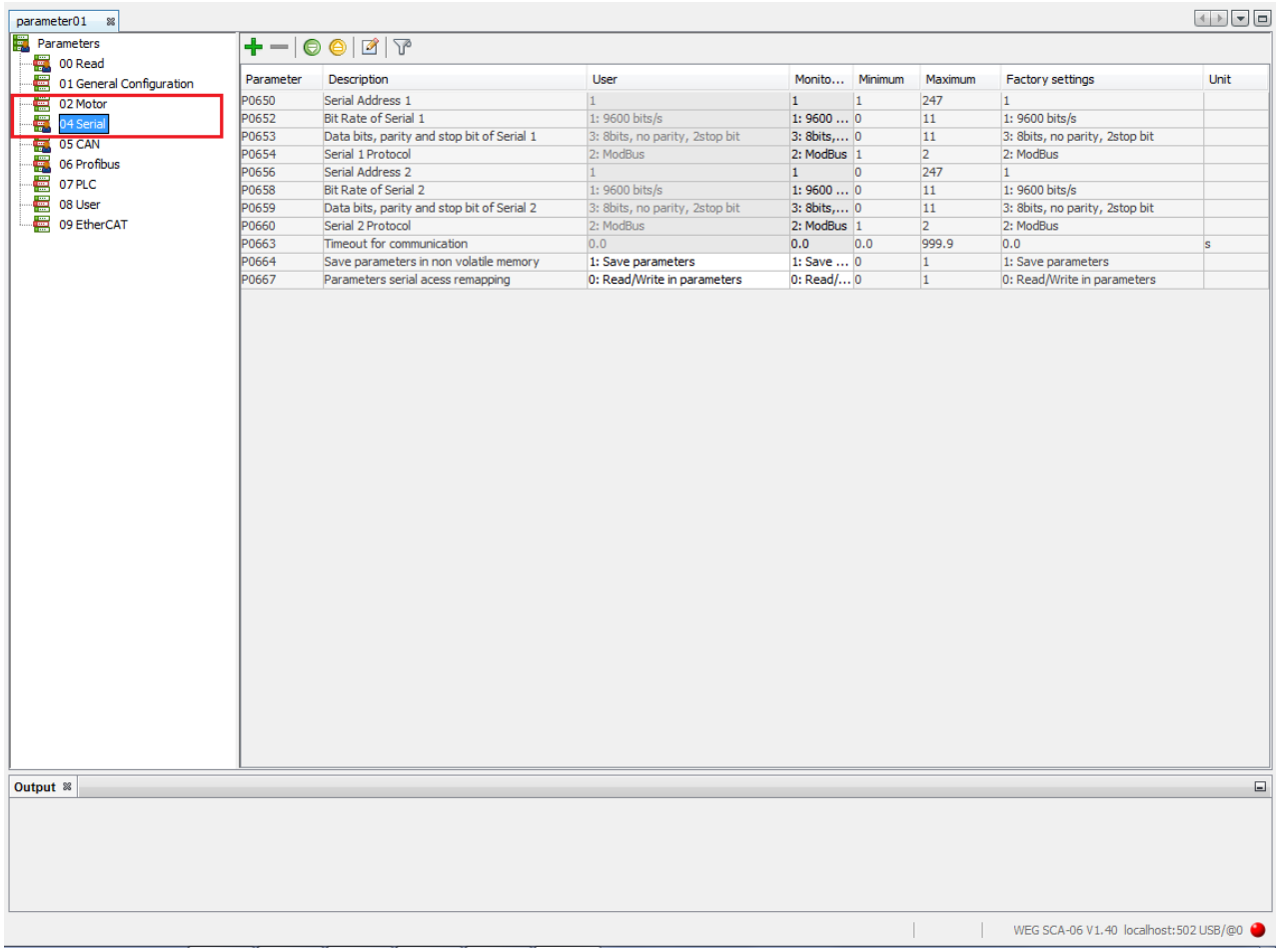



parameter01
Parameters
00 Read
01 General Configuration
02 Motor
03 Special Function
04 Serial
05 CAN
06 Profibus
07 PLC
08 User
09 EtherCAT
Parameter
Description
User
Monitored
Minimum
Maximum
Factory settings
Unit
P0072
Transmitted CAN Telegrams
0
0
65535
0
Select parameters
Select to unhide:
CAN controller Status
Received CAN Telegrams
Bus Off Counter
CAN Lost Messages
CANopen Communication Status
CANopen Node State
Communication Error Behavior
CAN Protocol
CAN Address
Baud rate
Bus Off Reset
Follow Type
Follow COB ID
Follow Period
OK
Cancel
Output
WEG SCA-06 V1.40 localhost:502 USB/@0

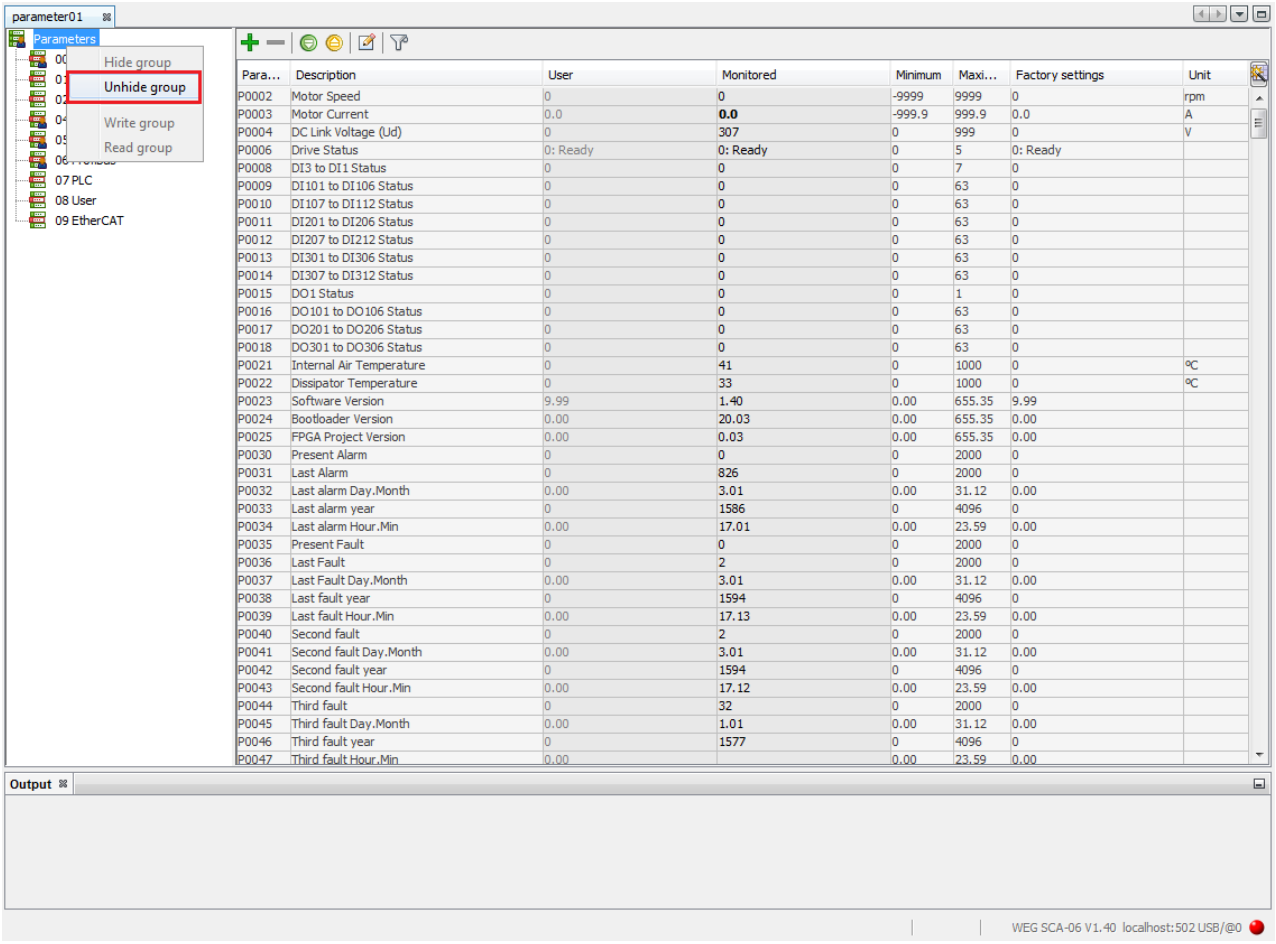


3. **Hide Group of Parameters**. In order to hide a group of parameters, just select the group on the tree and use the **Hide Group** option.

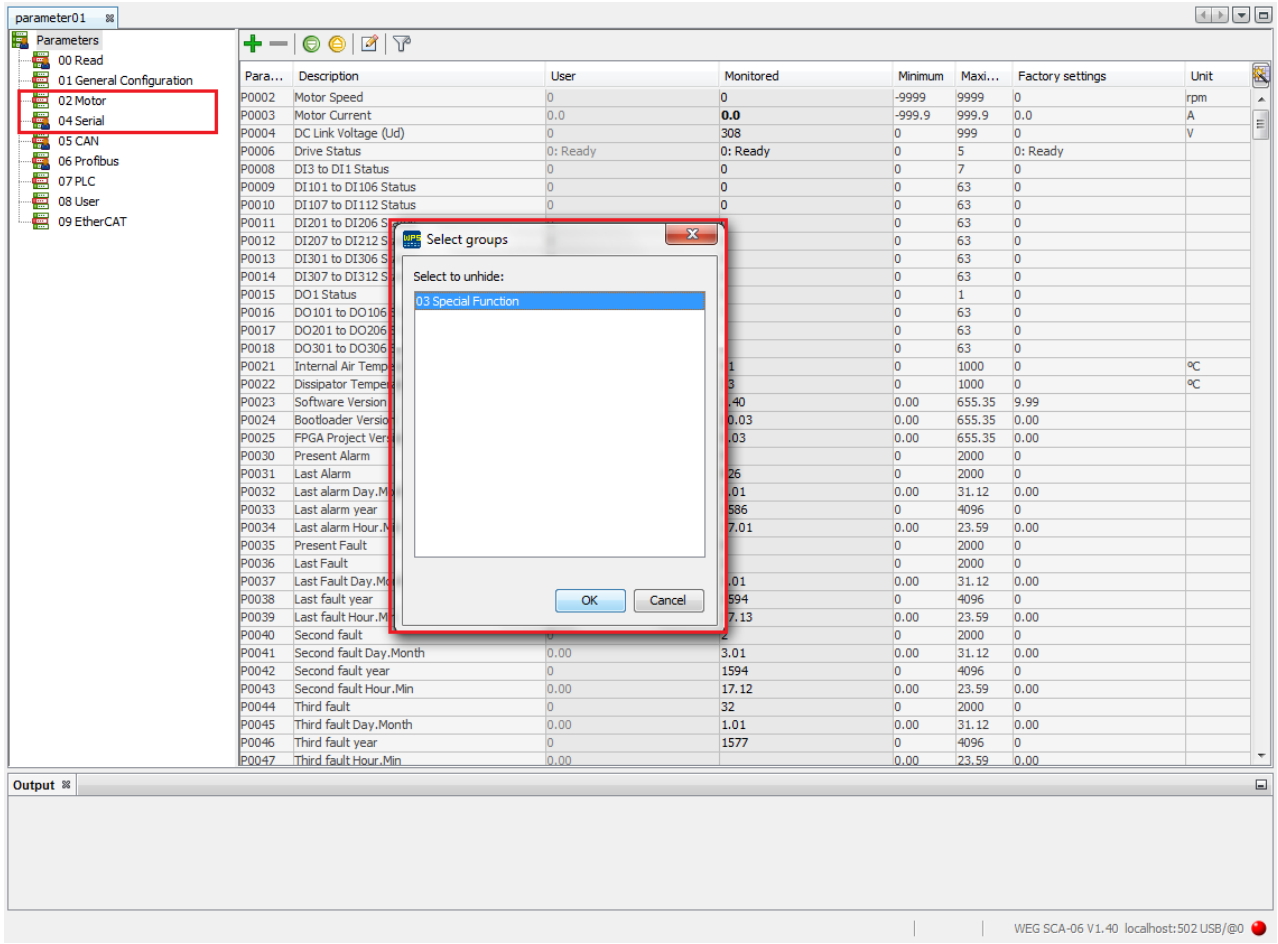

parameter01
Parameters
00 Read
01 General Configuration
02 Motor
03 Sp
04 Se
05 CA
06 Pro
07 PLC
08 Use
09 EtherCAT
Hide group
Unhide group
Write group
Read group
Param...
Description
User
Mo...
Minimum
Maximum
Factory settings
Unit
P0365 Encoder temperature 0.0 -3276.7 3276.7 0.0
P0368 Manufacture password 0 0 0 65535 0
P0369 Electrical position 0 50828 0 65535 0
P0370 Encoder temperature 0 -32767 32767 0
P0371 Read Encoder filter 3000 3000 3000 37500 3000
P0372 Encoder filter 3000 3000 3000 37500 3000
P0375 Encoder Offset 0 0 -32767 32767 0
P0490 Load Absolute position 0 0 0 1 0
P0492 User reference: fractions of revolution 0 0 -16383 16383 0
P0493 User reference: number of revolutions 0 0 -32768 32767 0
P0500 Count Mode: Standart Counter 0: Disabled 0: Di... 0 4 0: Disabled
P0502 Load counter 0 0 0 1 0
P0503 Counter Value - Low 0 0 0 65535 0
P0504 Counter Value - High 0 0 0 65535 0
P0506 Counter number of pulses/turns 1024 1024 1 65535 1024
P0507 Counter speed filter 500 500 1 4000 500 Hz
P0510 Count Mode: Counter 1 0: Disabled 0: Di... 0 3 0: Disabled
P0511 Null Pulse 1 options 0: Disabled 0: Di... 0 5 0: Disabled
P0512 Load Counter 1 value 0 0 0 1 0
P0513 Counter 1 value - Low 0 0 0 65535 0
P0514 Counter 1 value - High 0 0 0 65535 0
P0516 Counter 1 number of pulses/turns 1024 1024 1 65535 1024
P0517 Counter 1 speed filter 500 500 1 4000 500 Hz
P0519 Error Enable 0: Disable 0: Di... 0 2 0: Disable
P0520 Count Mode: Counter 2 0: Disabled 0: Di... 0 3 0: Disabled
P0521 Null Pulse 2 options 0: Disabled 0: Di... 0 5 0: Disabled
P0522 Load Counter 2 value 0 0 0 1 0
P0523 Counter 2 value - Low 0 0 0 65535 0
P0524 Counter 2 value - High 0 0 0 65535 0
P0526 Counter 2 number of pulses/turns 1024 1024 1 65535 1024
P0527 Counter 2 speed filter 500 500 1 4000 500 Hz
P0529 Error Enable 0: Disabled 0: Di... 0 2 0: Disabled
P0531 Stop function:level or edge 0: stop activation by level 0: st... 0 1 0: stop activation by level
P0550 Trigger 1 Signal Source 0: Disabled 0: Di... 0 48 0: Disabled
P0551 Trigger 1 Value 0 0 -32768 32767 0
P0552 Trigger 1 Condition 0: Higher or equal than reference value 0: Hi... 0 1 0: Higher or equal than reference value
P0553 Trigger 2 Signal Source 0: Disable 0: Di... 0 48 0: Disable
P0554 Trigger 2 Value 0 -32768 32767 0
Output
WEG SCA-06 V1.40 localhost:502 USB/@0

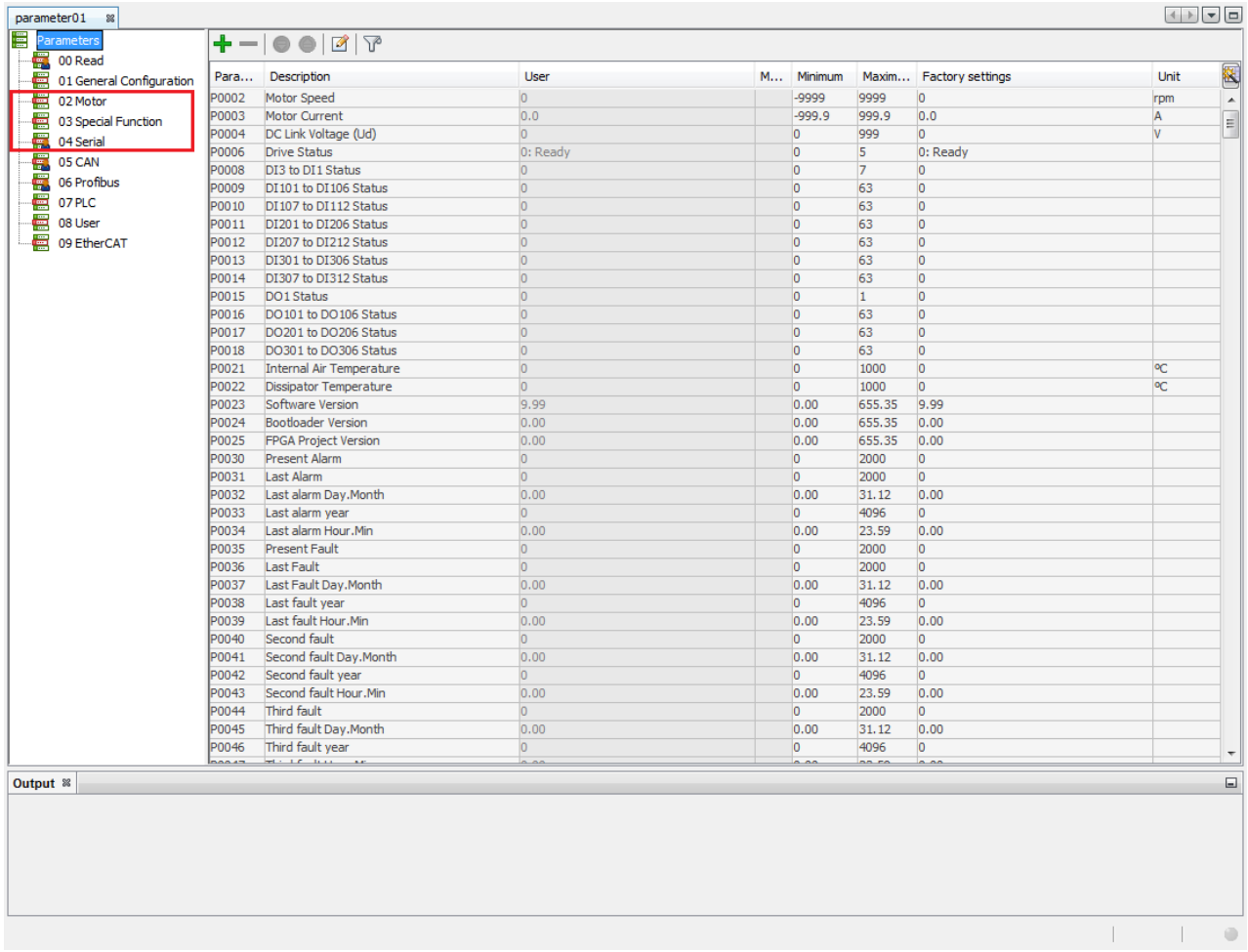


4. **Unhide Group of Parameters**. In order to show a hidden group of parameters, just select the root of the group tree and select the **Unhide Group** option. A window will open showing the groups that are hidden; then just select the group you wish to unhide.



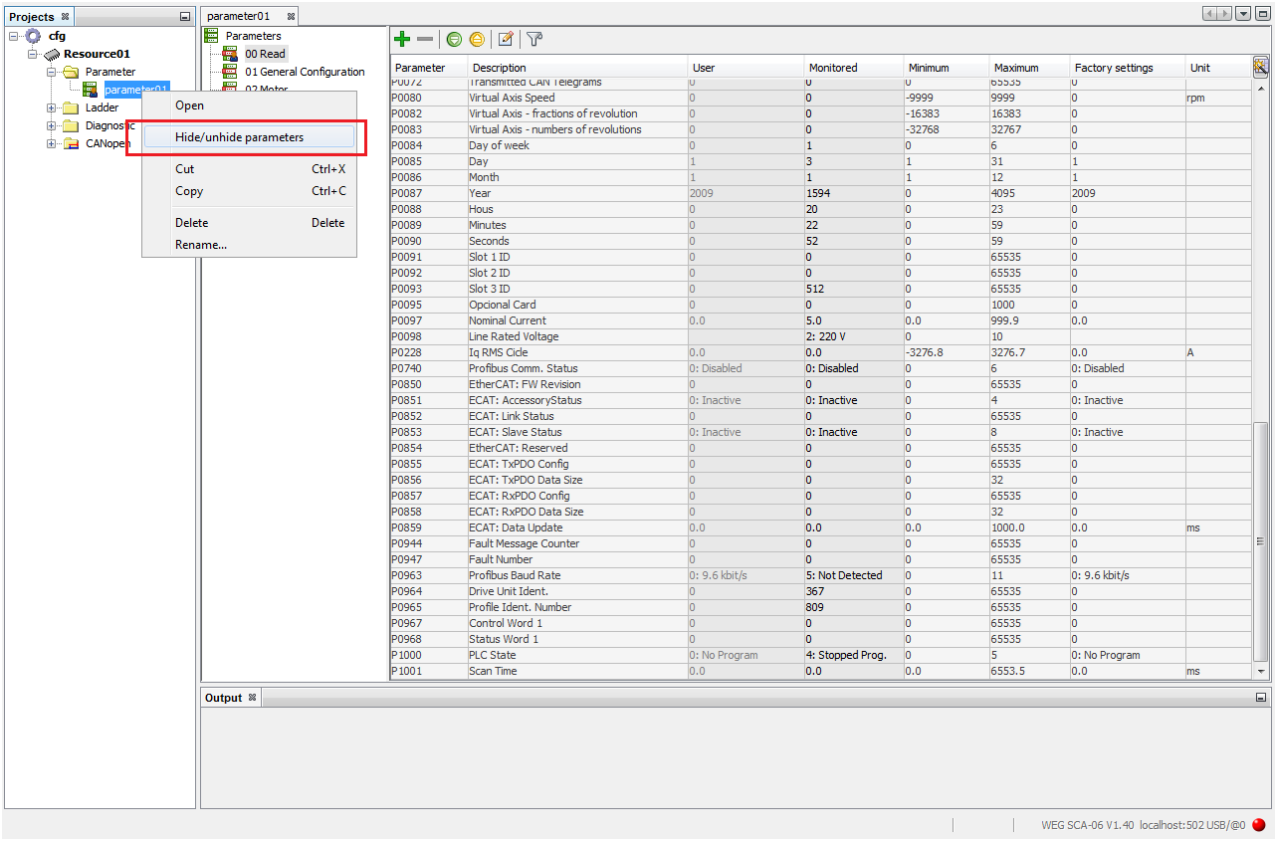
Para...	Description	User	Monitored	Minimum	Maxi...	Factory settings	Unit
P0002	Motor Speed	0	0	-9999	9999	0	rpm
P0003	Motor Current	0.0	**0.0**	-999.9	999.9	0.0	A
P0004	DC Link Voltage (Ud)	0	307	0	999	0	V
P0006	Drive Status	0: Ready	0: Ready	0	5	0: Ready	
P0008	DI3 to DI1 Status	0	0	0	7	0	
P0009	DI101 to DI106 Status	0	0	0	63	0	
P0010	DI107 to DI112 Status	0	0	0	63	0	
P0011	DI201 to DI206 Status	0	0	0	63	0	
P0012	DI207 to DI212 Status	0	0	0	63	0	
P0013	DI301 to DI306 Status	0	0	0	63	0	
P0014	DI307 to DI312 Status	0	0	0	63	0	
P0015	DO1 Status	0	0	0	1	0	
P0016	DO101 to DO106 Status	0	0	0	63	0	
P0017	DO201 to DO206 Status	0	0	0	63	0	
P0018	DO301 to DO306 Status	0	0	0	63	0	
P0021	Internal Air Temperature	0	41	0	1000	0	°C
P0022	Dissipator Temperature	0	33	0	1000	0	°C
P0023	Software Version	9.99	1.40	0.00	655.35	9.99	
P0024	Bootloader Version	0.00	20.03	0.00	655.35	0.00	
P0025	FPGA Project Version	0.00	0.03	0.00	655.35	0.00	
P0030	Present Alarm	0	0	0	2000	0	
P0031	Last Alarm	0	826	0	2000	0	
P0032	Last alarm Day.Month	0.00	3.01	0.00	31.12	0.00	
P0033	Last alarm year	0	1586	0	4096	0	
P0034	Last alarm Hour.Min	0.00	17.01	0.00	23.59	0.00	
P0035	Present Fault	0	0	0	2000	0	
P0036	Last Fault	0	2	0	2000	0	
P0037	Last Fault Day.Month	0.00	3.01	0.00	31.12	0.00	
P0038	Last fault year	0	1594	0	4096	0	
P0039	Last fault Hour.Min	0.00	17.13	0.00	23.59	0.00	
P0040	Second fault	0	2	0	2000	0	
P0041	Second fault Day.Month	0.00	3.01	0.00	31.12	0.00	
P0042	Second fault year	0	1594	0	4096	0	
P0043	Second fault Hour.Min	0.00	17.12	0.00	23.59	0.00	
P0044	Third fault	0	32	0	2000	0	
P0045	Third fault Day.Month	0.00	1.01	0.00	31.12	0.00	
P0046	Third fault year	0	1577	0	4096	0	
P0047	Third fault Hour.Min	0.00		0.00	23.59	0.00	

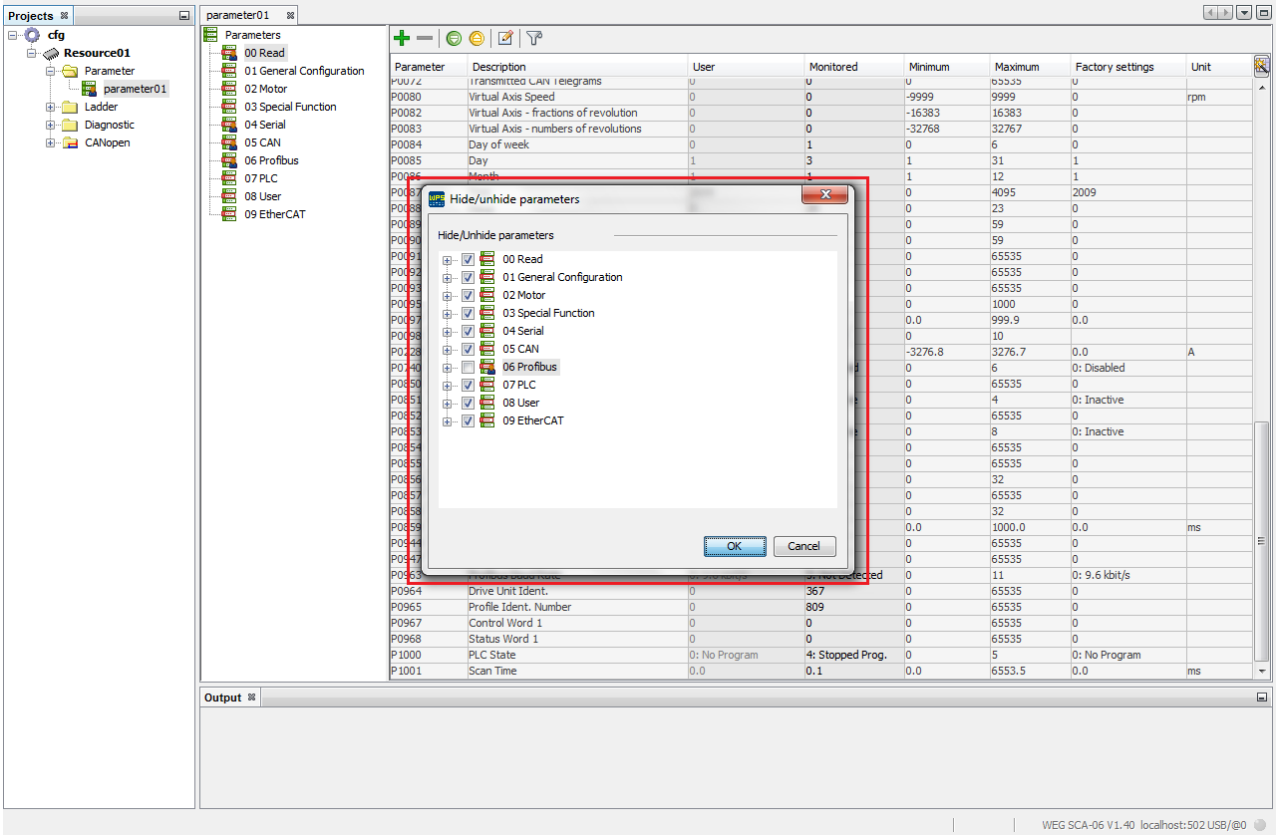

parameter01
Parameters
00 Read
01 General Configuration
02 Motor
04 Serial
05 CAN
06 Profibus
07 PLC
08 User
09 EtherCAT
Select groups
Select to unhide:
03 Special Function
OK
Cancel
Output
WEG SCA-06 V1.40 localhost:502 USB/@0



Para...	Description	User	M...	Minimum	Maxim...	Factory settings	Unit
P0002	Motor Speed	0		-9999	9999	0	rpm
P0003	Motor Current	0.0		-999.9	999.9	0.0	A
P0004	DC Link Voltage (Ud)	0		0	999	0	V
P0006	Drive Status	0: Ready		0	5	0: Ready	
P0008	DI3 to DI1 Status	0		0	7	0	
P0009	DI101 to DI106 Status	0		0	63	0	
P0010	DI107 to DI112 Status	0		0	63	0	
P0011	DI201 to DI206 Status	0		0	63	0	
P0012	DI207 to DI212 Status	0		0	63	0	
P0013	DI301 to DI306 Status	0		0	63	0	
P0014	DI307 to DI312 Status	0		0	63	0	
P0015	DO1 Status	0		0	1	0	
P0016	DO101 to DO106 Status	0		0	63	0	
P0017	DO201 to DO206 Status	0		0	63	0	
P0018	DO301 to DO306 Status	0		0	63	0	
P0021	Internal Air Temperature	0		0	1000	0	°C
P0022	Dissipator Temperature	0		0	1000	0	°C
P0023	Software Version	9.99		0.00	655.35	9.99	
P0024	Bootloader Version	0.00		0.00	655.35	0.00	
P0025	FPGA Project Version	0.00		0.00	655.35	0.00	
P0030	Present Alarm	0		0	2000	0	
P0031	Last Alarm	0		0	2000	0	
P0032	Last alarm Day.Month	0.00		0.00	31.12	0.00	
P0033	Last alarm year	0		0	4096	0	
P0034	Last alarm Hour.Min	0.00		0.00	23.59	0.00	
P0035	Present Fault	0		0	2000	0	
P0036	Last Fault	0		0	2000	0	
P0037	Last Fault Day.Month	0.00		0.00	31.12	0.00	
P0038	Last fault year	0		0	4096	0	
P0039	Last fault Hour.Min	0.00		0.00	23.59	0.00	
P0040	Second fault	0		0	2000	0	
P0041	Second fault Day.Month	0.00		0.00	31.12	0.00	
P0042	Second fault year	0		0	4096	0	
P0043	Second fault Hour.Min	0.00		0.00	23.59	0.00	
P0044	Third fault	0		0	2000	0	
P0045	Third fault Day.Month	0.00		0.00	31.12	0.00	
P0046	Third fault year	0		0	4096	0	

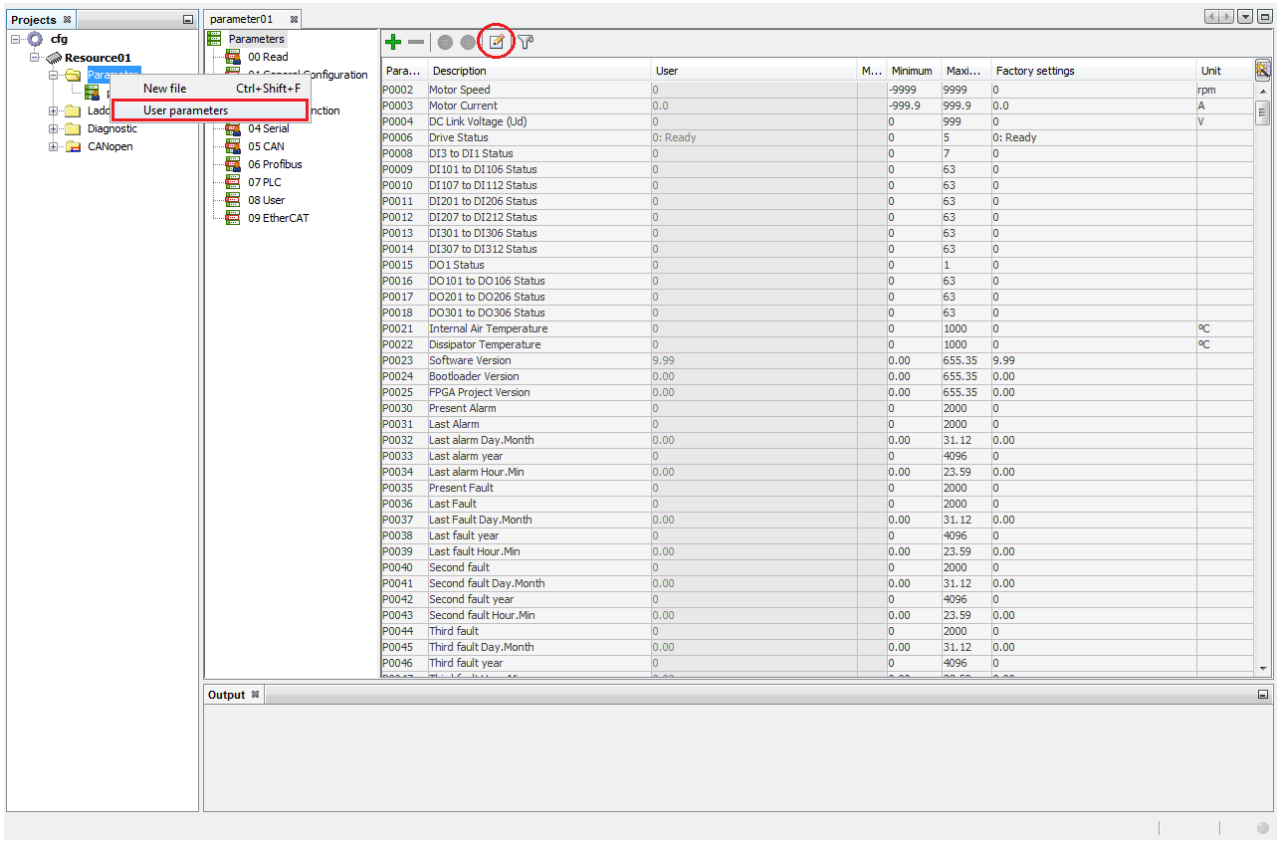
5. **Hide and Show Parameters and Groups of Parameters**. By means of this option, you have full control of the parameters and groups of parameters. It is possible to hide and unhide individual parameters, multiple parameters, individual groups and multiple groups in the same action.


Projects
cfg
Resource01
Parameter
parameter01
Ladder
Diagnostic
CANopen
Open
Hide/unhide parameters
Cut Ctrl+X
Copy Ctrl+C
Delete Delete
Rename...
parameter01
Parameters
00 Read
01 General Configuration
Parameter
Description
User
Monitored
Minimum
Maximum
Factory settings
Unit
P0080 Virtual Axis Speed
P0082 Virtual Axis - fractions of revolution
P0083 Virtual Axis - numbers of revolutions
P0084 Day of week
P0085 Day
P0086 Month
P0087 Year
P0088 Hous
P0089 Minutes
P0090 Seconds
P0091 Slot 1 ID
P0092 Slot 2 ID
P0093 Slot 3 ID
P0095 Opcional Card
P0097 Nominal Current
P0098 Line Rated Voltage
P0228 Iq RMS Cicle
P0740 Profibus Comm. Status
P0850 EtherCAT: FW Revision
P0851 ECAT: AccessoryStatus
P0852 ECAT: Link Status
P0853 ECAT: Slave Status
P0854 EtherCAT: Reserved
P0855 ECAT: TxPDO Config
P0856 ECAT: TxPDO Data Size
P0857 ECAT: RxPDO Config
P0858 ECAT: RxPDO Data Size
P0859 ECAT: Data Update
P0944 Fault Message Counter
P0947 Fault Number
P0963 Profibus Baud Rate
P0964 Drive Unit Ident.
P0965 Profile Ident. Number
P0967 Control Word 1
P0968 Status Word 1
P1000 PLC State
P1001 Scan Time
Output
WEG SCA-06 V1.40 localhost:502 USB/@0


Projects
cfg
Resource01
Parameter
parameter01
Ladder
Diagnostic
CANopen
parameter01
Parameters
00 Read
01 General Configuration
02 Motor
03 Special Function
04 Serial
05 CAN
06 Profibus
07 PLC
08 User
09 EtherCAT
Hide/unhide parameters
Hide/Unhide parameters
OK
Cancel
Output
WEG SCA-06 V1.40 localhost:502 USB/@0

11.2.5.5 User Parameters

In order to open the configuration screen of the user parameters, just click on the **User Parameters** option on the Parameter node of the project tree or click on the icon indicated on the tool bar of the parameter file.



Para...	Description	User	M...	Minimum	Maxi...	Factory settings	Unit
P0002	Motor Speed	0		-9999	9999	0	rpm
P0003	Motor Current	0.0		-999.9	999.9	0.0	A
P0004	DC Link Voltage (Ud)	0		0	999	0	V
P0006	Drive Status	0: Ready		0	5	0: Ready	
P0008	DI3 to DI1 Status	0		0	7	0	
P0009	DI101 to DI106 Status	0		0	63	0	
P0010	DI107 to DI112 Status	0		0	63	0	
P0011	DI201 to DI206 Status	0		0	63	0	
P0012	DI207 to DI212 Status	0		0	63	0	
P0013	DI301 to DI306 Status	0		0	63	0	
P0014	DI307 to DI312 Status	0		0	63	0	
P0015	DO1 Status	0		0	1	0	
P0016	DO101 to DO106 Status	0		0	63	0	
P0017	DO201 to DO206 Status	0		0	63	0	
P0018	DO301 to DO306 Status	0		0	63	0	
P0021	Internal Air Temperature	0		0	1000	0	°C
P0022	Dissipator Temperature	0		0	1000	0	°C
P0023	Software Version	9.99		0.00	655.35	9.99	
P0024	Bootloader Version	0.00		0.00	655.35	0.00	
P0025	FPGA Project Version	0.00		0.00	655.35	0.00	
P0030	Present Alarm	0		0	2000	0	
P0031	Last Alarm	0		0	2000	0	
P0032	Last alarm Day.Month	0.00		0.00	31.12	0.00	
P0033	Last alarm year	0		0	4096	0	
P0034	Last alarm Hour.Min	0.00		0.00	23.59	0.00	
P0035	Present Fault	0		0	2000	0	
P0036	Last Fault	0		0	2000	0	
P0037	Last Fault Day.Month	0.00		0.00	31.12	0.00	
P0038	Last fault year	0		0	4096	0	
P0039	Last fault Hour.Min	0.00		0.00	23.59	0.00	
P0040	Second fault	0		0	2000	0	
P0041	Second fault Day.Month	0.00		0.00	31.12	0.00	
P0042	Second fault year	0		0	4096	0	
P0043	Second fault Hour.Min	0.00		0.00	23.59	0.00	
P0044	Third fault	0		0	2000	0	
P0045	Third fault Day.Month	0.00		0.00	31.12	0.00	
P0046	Third fault year	0		0	4096	0	

Configuration Table.

On the user parameter configuration table, it is possible to define several attributes to the parameters, such as description, minimum and maximum values, unit, digits, data type, etc.

NOTE!
These settings will be automatically displayed in the parameter table. However, to be sent to the device, you need to *download* the resource.

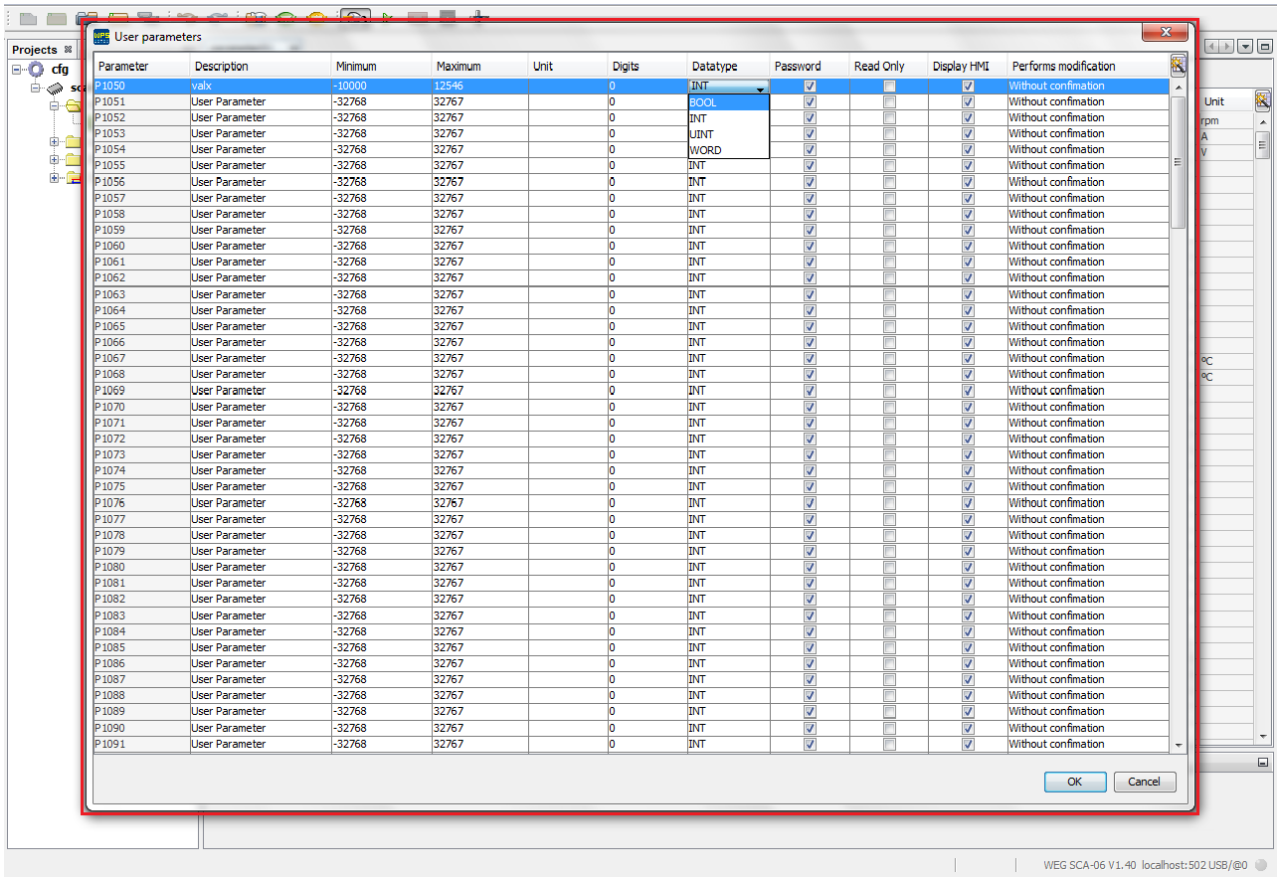


Table fields:

- Parameter: User parameter identification.
- Description: Description of the user parameter in the parameter table. On devices that have text-based HMIs, the description is sent to the machine and displayed on the HMI.
- Minimum: Minimum input value for parameter.
- Maximum: Maximum input value for parameter.
- Unit: Unit displayed on the device's HMI.
- Default: Value loaded when restore factory default is selected.
- Retentive: Retain value after rebooting devices.
- Hexadecimal: Displays the value in hexadecimal.
- Digits: Number of decimal digits for displaying value.
- Datatype: Parameter datatype used by the ladder application.
- Password: Enables password request by changing parameter value.

- Read only: It does not allow the writing of values in the parameter by the communication network or the HMI. Writing is done only by the ladder application.

- Display HMI: Displays the parameter in the HMI.

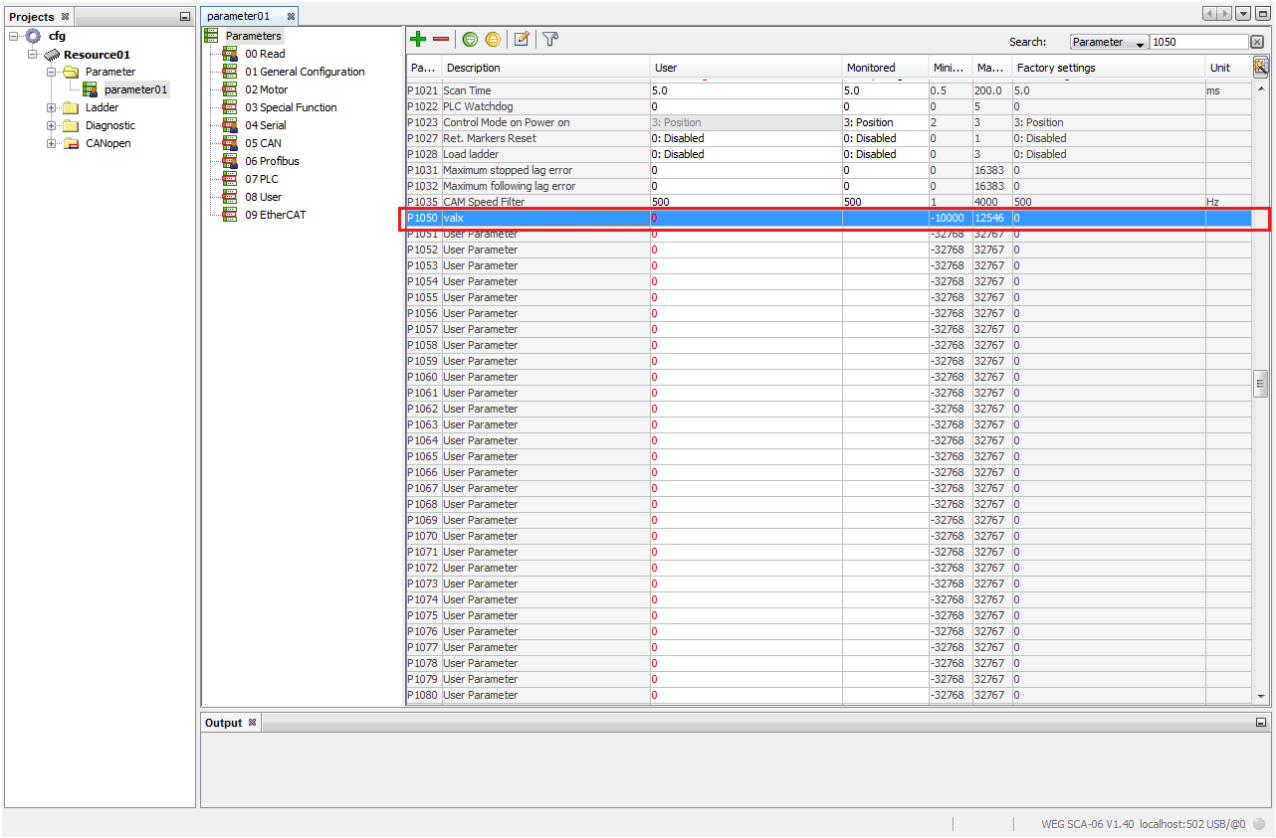
- Performs modification: Confirmation options when changing the parameter:
o No confirmation: Does not prompt for confirmation when changing parameter.
o With confirmation and engine stopped: Request confirmation and allow change only with engine stopped.
o With confirmation: Prompt for confirmation when changing parameter.

- Stopped motor: Perform change only with motor stopped.

- Help: On devices that have text-based HMI, you can edit a help text for the parameter.

View the user parameter

In the parameter table, the user parameters will be shown as they are configured on the configuration screen.



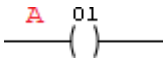
11.2.6 Ladder

11.2.6.1 Coil

11.2.6.1.1 DIRECTCOIL

Logical block used to assign direct values of the output variables.

Ladder Representation



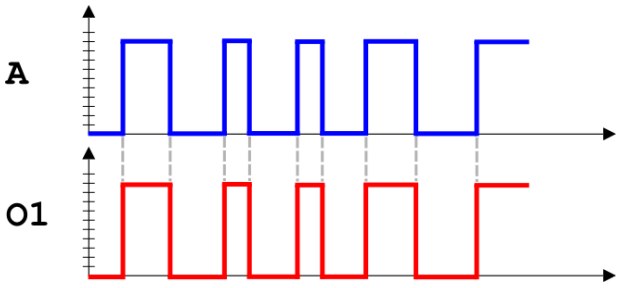
Block Structure

Variable Type	Name	Data Type	Description
VAR_OUTPUT	O1	BOOL	Block log output

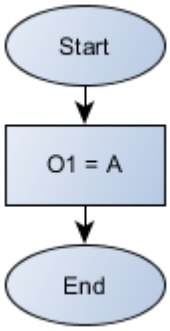
Operation

The block transfers the value of A for the memory address corresponding to O1.

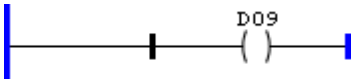
Diagram



Block Flowchart



Example

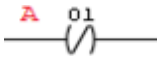


The above example keeps the digital output DO9 permanently connected, because the value of A in this case is the value of the left bus which is always considered high logic level (TRUE).

11.2.6.1.2 INVERTEDCOIL

Logical block used for assigning values denied to output variables.

Ladder Representation



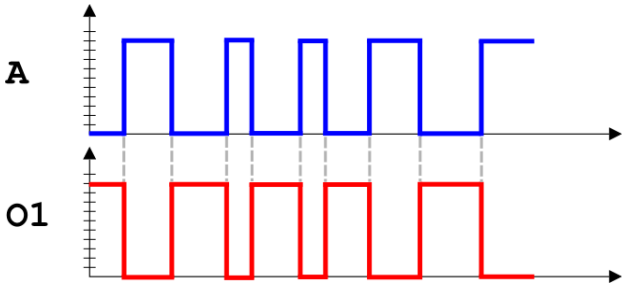
Block Structure

Variable Type	Name	Data Type	Description
VAR_OUTPUT	O1	BOOL	Block log output

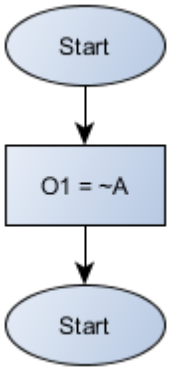
Operation

The block transfers the denied value of A for the memory address corresponding to O1.

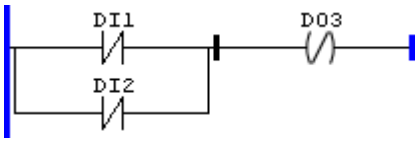
Diagram



Block Flowchart



Example

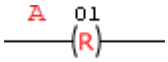


The above example disables the digital output DO3 when some of the digital inputs DI1 and DI2 are with FALSE value. When both inputs are with a TRUE value, DO3 activates.

11.2.6.1.3 RESETCOIL

Logical block used for indefinite disabling of output variables.

Ladder Representation



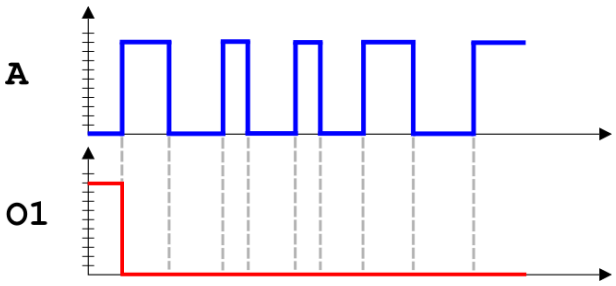
Block Structure

Variable Type	Name	Data Type	Description
VAR_OUTPUT	O1	BOOL	Block log output

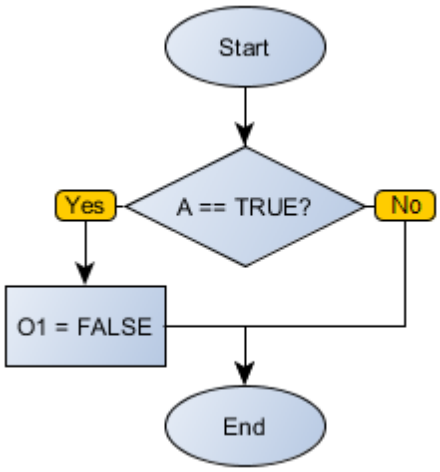
Operation

When identifying a TRUE value in A, this block transfers a FALSE value to the memory address corresponding to O1.
When identifying a FALSE value in A, this block performs no operation.

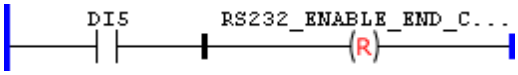
Diagram



Block Flowchart



Example

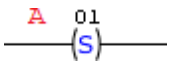


The example above activates permanently the system control marker that enables end-of-message character in RS232 communication to identify a TRUE level at the digital input DI5.

11.2.6.1.4 SETCOIL

Logical block used for indefinite enabling of output variables.

Ladder Representation



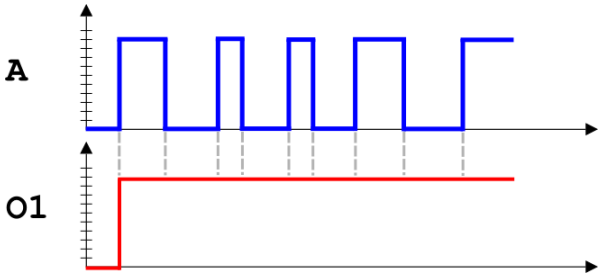
Block Structure

Variable Type	Name	Data Type	Description
VAR_OUTPUT	O1	BOOL	Block log output

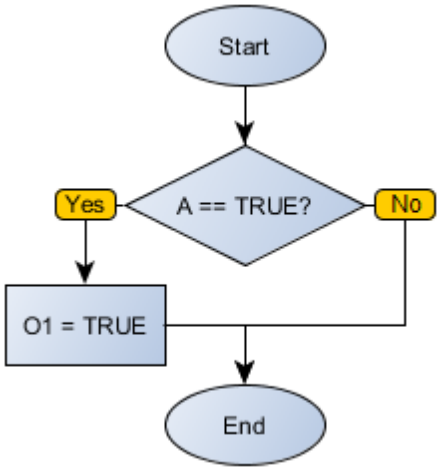
Operation

When identifying a TRUE value in A, this block transfers the value of A for the memory address corresponding to O1.
When identifying a FALSE value in A, this block performs no operation.

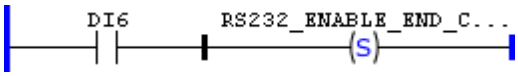
Diagram



Block Flowchart



Example

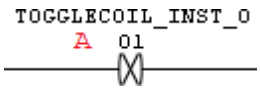


The example above activates permanently the system control marker that enables end-of-message character in RS232 communication to identify a TRUE level at the digital input DI6.

11.2.6.1.5 TOGGLECOIL

Logical block used for output variables alternance.

Ladder Representation



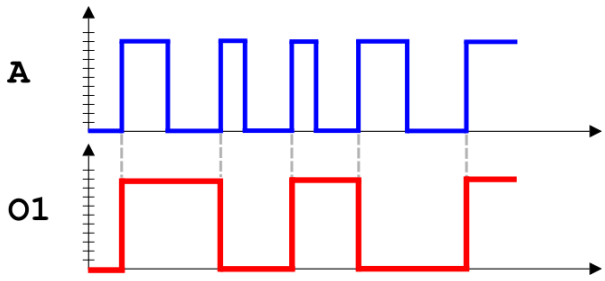
Block Structure

Variable Type	Name	Data Type	Description
VAR_OUTPUT	O1	BOOL	Block log output
VAR	TOGGLECOIL_INST_0	TOGGLECOIL	Instance of access to block structure

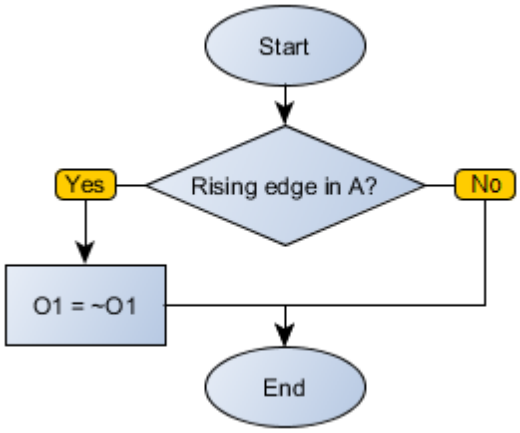
Operation

When identifying a transition from FALSE to TRUE (leading edge) on A, the block reverses the status of O1.

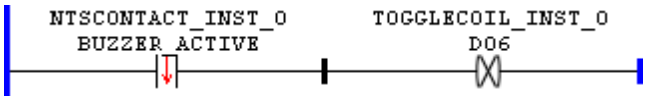
Diagram



Block Flowchart



Example



The above example inverts the state of the digital output DO6 to each disabling the internal buzzer.

11.2.6.2 Communication Network

11.2.6.2.1 Modbus RTU

11.2.6.2.1.1 Modbus RTU Overview

Operation in the Modbus RTU Network - Master Mode

The CFW300 allows operation as a master for the Modbus RTU network. For this operation, it is necessary to observe the following points:

- Only interface RS485 allows operation as a network master.
- It is necessary to program, in product configurations, the operation mode as "Master", besides the communication rate, parity, and stop bits, which must be the same for the whole equipment in the network.
- The Modbus RTU network master does not have an address, so the address configured in the CFW300 is not used.
- Sending and receiving telegrams via RS485 interface using the Modbus RTU is programmed by using blocks in Ladder programming language. It is necessary to know the available blocks and the Ladder programming software in order to be able to program the network master.

- The following functions are available for the sending of requisitions by the Modbus master:
  - Function 01: Read Coils
  - Function 02: Read Discrete Inputs
  - Function 03: Read Holding Registers
  - Function 04: Read Input Registers
  - Function 05: Write Single Coil
  - Function 06: Write Single Register
  - Function 15: Write Multiple Coils
  - Function 16: Write Multiple Registers

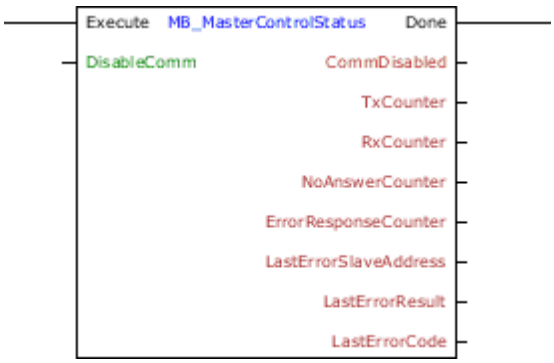
Blocks to program the master

In order to control and monitor the Modbus RTU communication using the CFW300, the following blocks were developed, and they must be used when programming in Ladder.

11.2.6.2.1.2 MB_MasterControlStatus

Block that allows monitoring various statuses of the Modbus RTU network master.

Ladder Representation



Block Structure

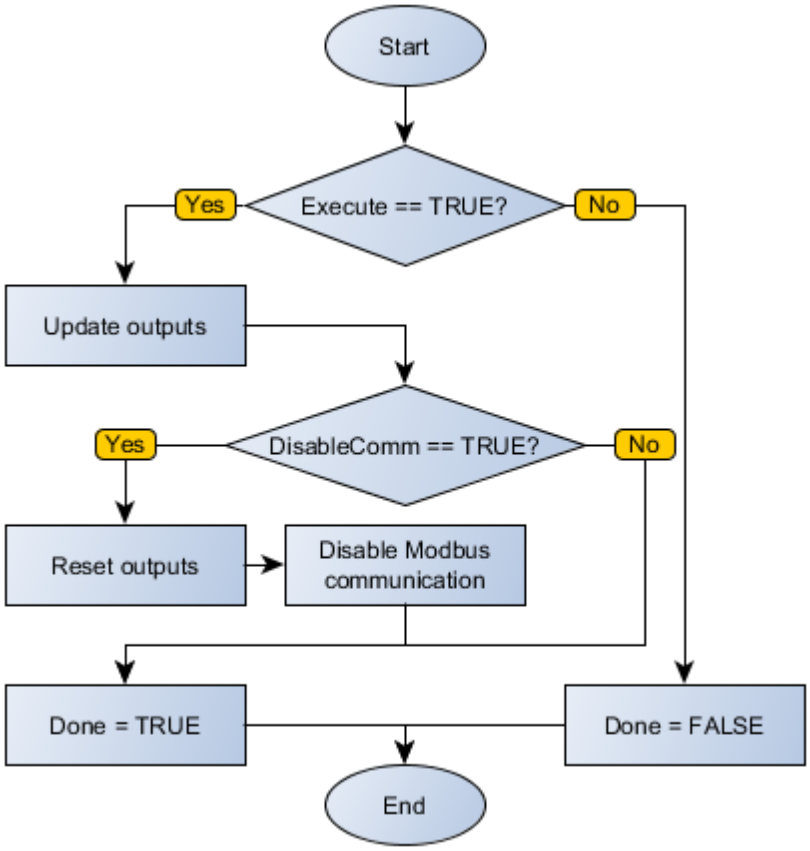
Variable Type	Name	Data Type	Description
VAR_INPUT	Execute	BOOL	Block enabling
	DisableComm	BOOL	Disables Modbus RTU communication
VAR_OUTPUT	Done	BOOL	Output enabling
	CommDisabled	BOOL	Disabled communication flag
	TxCounter	WORD UINT	Counter of requests sent
	RxCounter	WORD UINT	Counter of telegrams received
	NoAnswerCounter	WORD UINT	Counter of requests not answered
	ErrorResponseCounter	WORD UINT	Counter of responses received with error information
	LastErrorSlaveAddress	BYTE USINT	Slave address in which the last communication error was detected
	LastErrorResult	BYTE USINT	Operation result of the last communication error received (0 = No error) (4 – Response Timeout) (5 = Slave returned error)
	LastErrorCode	BYTE USINT	Code of the last communication error received

Operation

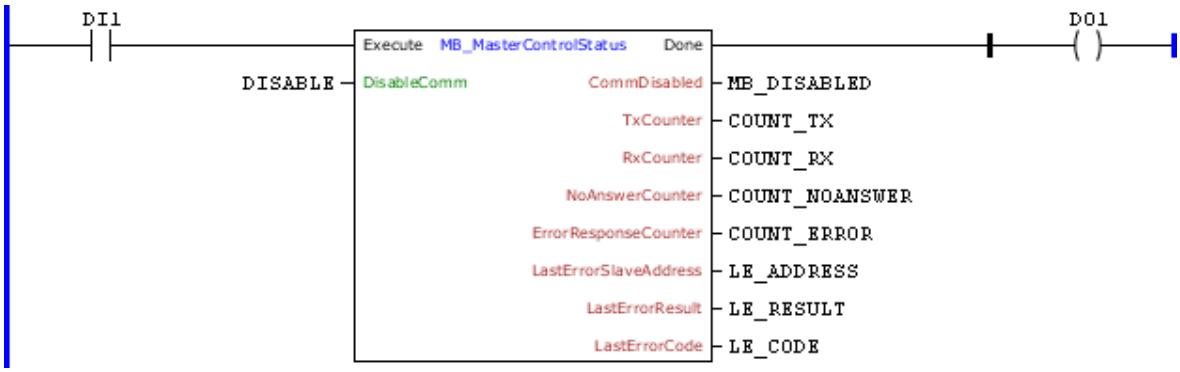
This block remains active while Execute is at TRUE level, updating its outputs according to the monitoring of the master and input requests. When Execute receives FALSE level, the inputs are ignored and the outputs are zeroed. The Done output receives TRUE level when Execute has TRUE level and block finished its execution.

A TRUE level DisableComm disables the Modbus RTU communication and resets the status counters and markers of the master. These markers and counters are displayed in the output block each having some data corresponding to its description. Their values are also cleared at shutdown of the master.

Block Flowchart



Example

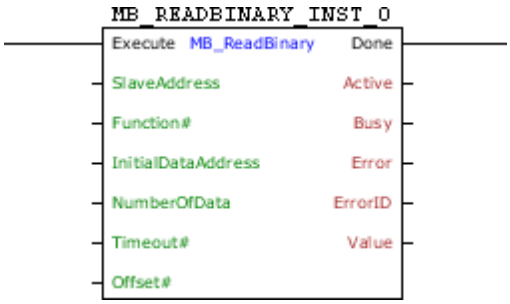


The example above requests status data of the Modbus RTU network master, and allows disabling communication through DISABLE. The block ends successfully, Done output is activated.

11.2.6.2.1.3 MB_ReadBinary

Block that performs a reading of up to 128 binary data (via Read Coils or Read Discrete Inputs) of a slave on the Modbus RTU network.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	Execute	BOOL	Block enabling
	SlaveAddress	BYTE	Slave address
	Function#	BYTE	Reading function code
	InitialDataAddress	WORD	Initial bit address of the data to be read
	NumberOfData	BYTE	Number of bits to be read (1 to 128)
	Timeout#	WORD	Maximum w aiting time for the slave response [ms]
	Offset#	BOOL	Offset Indication in InitialDataAddress, i.e., need to subtract 1 from this number
VAR_OUTPUT	Done	BOOL	Output enabling
	Active	BOOL	Aw aiting response flag
	Busy	BOOL	Flag indicating the RS485 interface is busy w ith another request
	Error	BOOL	Error in the execution flag
	ErrorID	BYTE	Identifier of the occurred error
	Value	BOOL	Variable that stores the received data
VAR	MB_READBINARY_INST_0	MB_READBINARY	Instance of access to block structure

Operation

When this block detects a leading edge on Execute, it checks whether the Modbus slave RTU in specified address in SlaveAddress is free to send data (Busy variable at FALSE level). If so, it sends the reading request of a number of bits indicated by NumberOfData in InitialDataAddress address using chosen function in Function# and sets the Active output, resetting it when receiving the response from the slave. The received data is stored in the Value variable. If the slave is not free, the block waits Busy go to FALSE level to resubmit the request.

NOTE!
If Execute goes to FALSE level and Busy is still at TRUE level, the request is canceled.

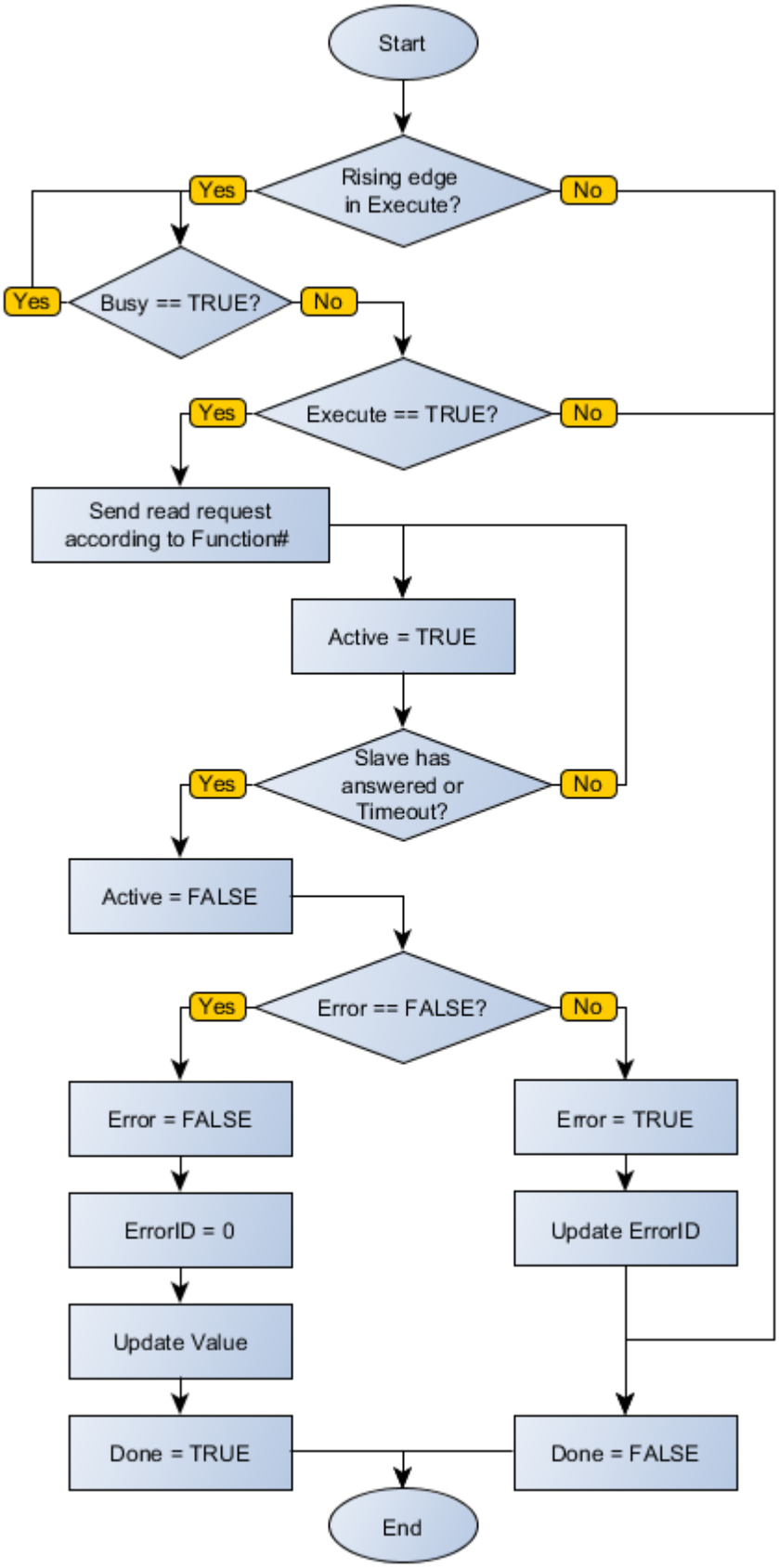
NOTE!
Value is an array of size equal to NumberOfData. It is important to check this compatibility not to generate errors in the block.

When Execute has FALSE value, Done remains FALSE. The Done output is only activated when the block finishes executing successfully, remaining at TRUE level until Execute receives FALSE.

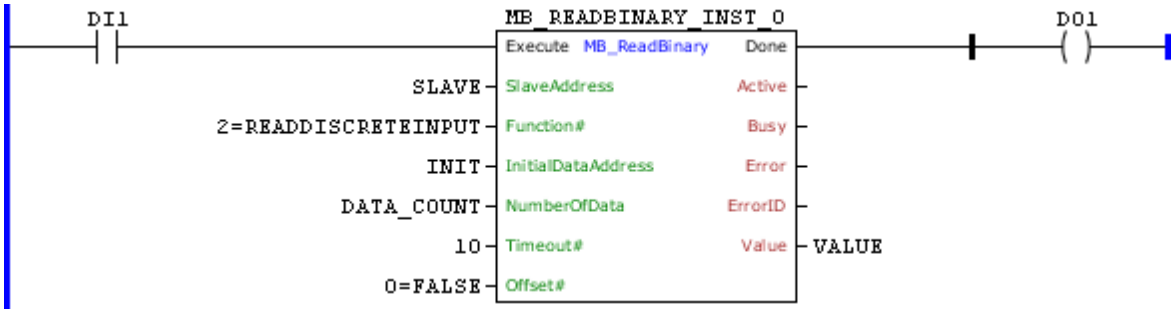
If there is any error in the execution, the Error output is enabled and ErrorID displays an error code according to the table below.

Code	Description
0	Executed successfully
1	Invalid input data
2	Master not enabled
4	Timeout in slave response
5	Slave returned error

Block Flowchart


Start
Rising edge in Execute?
Yes
No
Busy == TRUE?
Yes
No
Execute == TRUE?
Yes
No
Send read request according to Function#
Active = TRUE
Slave has answered or Timeout?
Yes
No
Active = FALSE
Error == FALSE?
Yes
No
Error = FALSE
ErrorID = 0
Update Value
Done = TRUE
Error = TRUE
Update ErrorID
Done = FALSE
End

Example

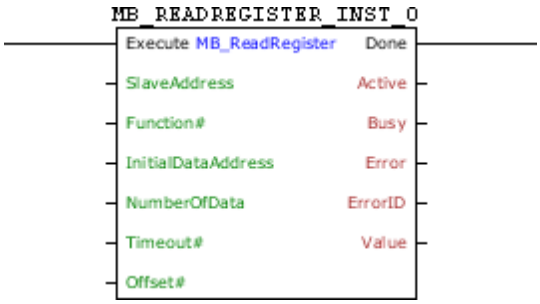


The above example requests reading of a number of binary data described by DATA_COUNT positioned in the INIT Modbus RTU slave of SLAVE address through the Read Discrete Input function. These data are forwarded to VALUE. The block ends successfully, Done output is activated.

11.2.6.2.1.4 MB_ReadRegister

Block that performs a reading of up to 64 16-bit registers (via Read Holding Registers or Read Input Registers) of a slave on the Modbus RTU network.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	Execute	BOOL	Block enabling
	SlaveAddress	BYTE	Slave address
	Function#	BYTE	Reading function code
	InitialDataAddress	WORD	Initial register address to be read
	NumberOfData	BYTE	Number of registers to be read (1 to 64)
	Timeout#	WORD	Maximum waiting time for the slave response [ms]
	Offset#	BOOL	Offset Indication in InitialDataAddress, i.e., need to subtract 1 from this number
VAR_OUTPUT	Done	BOOL	Output enabling
	Active	BOOL	Awaiting response flag
	Busy	BOOL	Flag indicating the RS485 interface is busy with another request
	Error	BOOL	Error in the execution flag
	ErrorID	BYTE	Identifier of the occurred error
	Value	BYTE SINT USINT WORD UINT INT DWORD UDINT DINT REAL	Variable that stores the received data
VAR	MB_READREGISTER_INST_0	MB_READREGISTER	Instance of access to block structure

Operation

When this block detects a leading edge on Execute, it checks whether the Modbus RTU slave in specified address in SlaveAddress is free to send data (Busy variable at FALSE level). If so, it sends the reading request of a number of registers indicated by NumberOfData in InitialDataAddress address using chosen function in Function# and sets the Active output, resetting them when receiving the response from the slave. The received data is stored in the Value variable. If the slave is not free, the block waits Busy go to FALSE level to resubmit the request.

NOTE!
If Execute goes to FALSE level and Busy is still at TRUE level, the request is canceled.

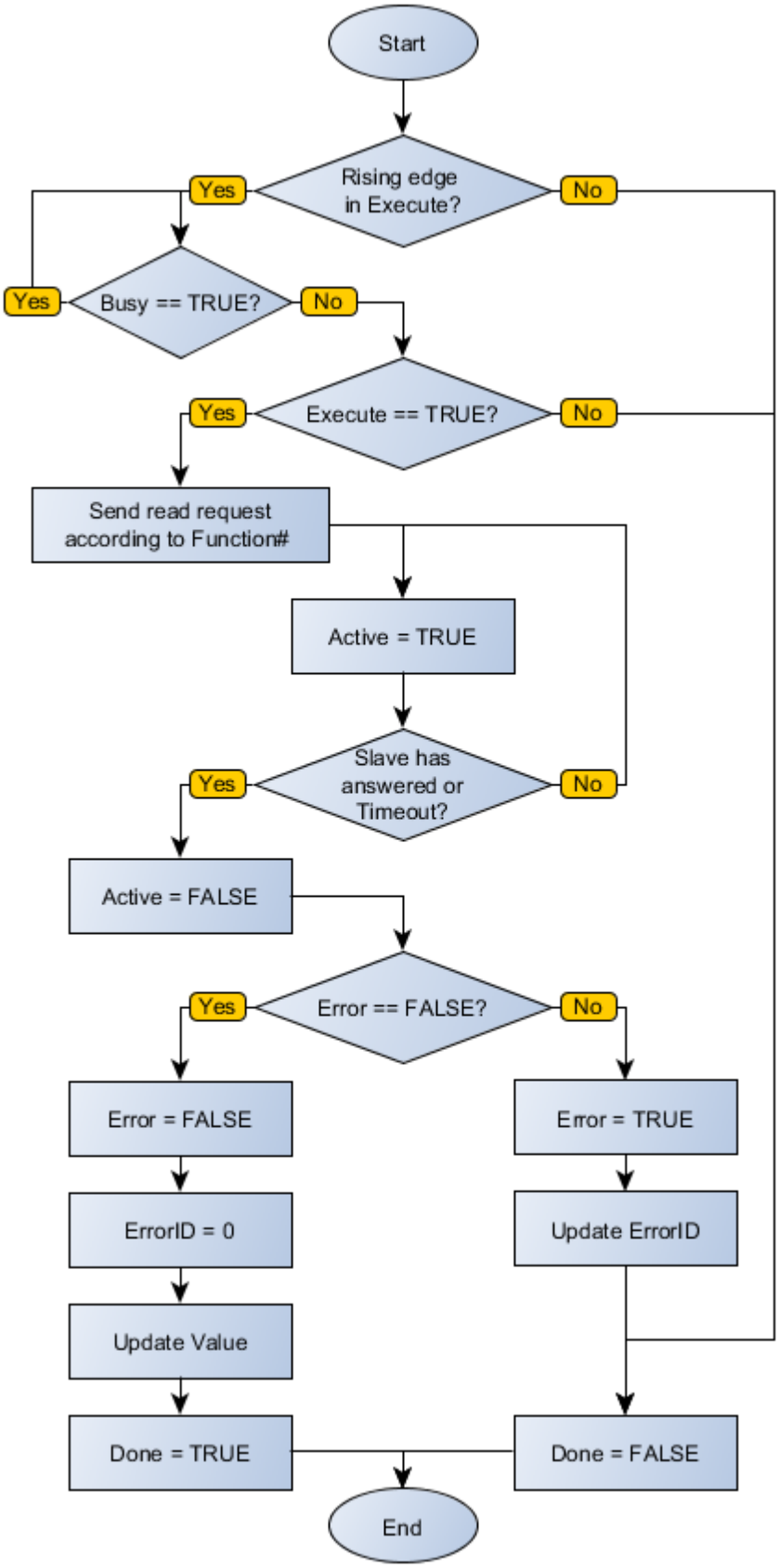
NOTE!
Value is an array of number of bits NumberOfData multiplied by 16. That is, if NumberOfData is 16, Value can be an array of 32 BYTE positions, 16 WORD positions or 8 DWORD positions. It is important to check this compatibility not to generate errors in the block.

When Execute has FALSE value, Done remains FALSE. The Done output is only activated when the block finishes executing successfully, remaining at TRUE level until Execute receives FALSE.

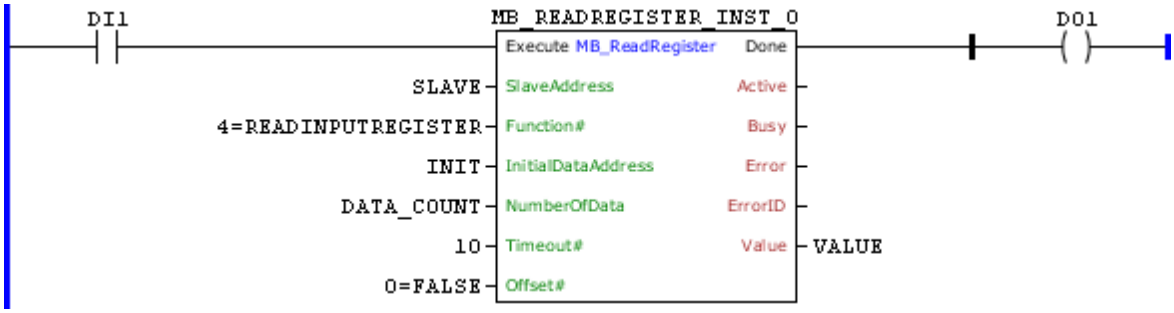
If there is any error in the execution, the Error output is enabled and ErrorID displays an error code according to the table below.

Code	Description
0	Executed successfully
1	Invalid input data
2	Master not enabled
4	Timeout in slave response
5	Slave returned error

Block Flowchart


Start
Rising edge in Execute?
Yes
No
Busy == TRUE?
Yes
No
Execute == TRUE?
Yes
No
Send read request according to Function#
Active = TRUE
Slave has answered or Timeout?
Yes
No
Active = FALSE
Error == FALSE?
Yes
No
Error = FALSE
ErrorID = 0
Update Value
Done = TRUE
Error = TRUE
Update ErrorID
Done = FALSE
End

Example



The above example requests reading of a number of binary data described by DATA_COUNT positioned in the INIT in the Modbus RTU slave of SLAVE address through the Read Input Register function. These data are forwarded to VALUE. The block ends successfully, Done output is activated.

11.2.6.2.1.5 MB_SlaveStatus

Block that allows monitoring the status of 4 slaves of the Modbus RTU network.

Ladder Representation

Block Structure

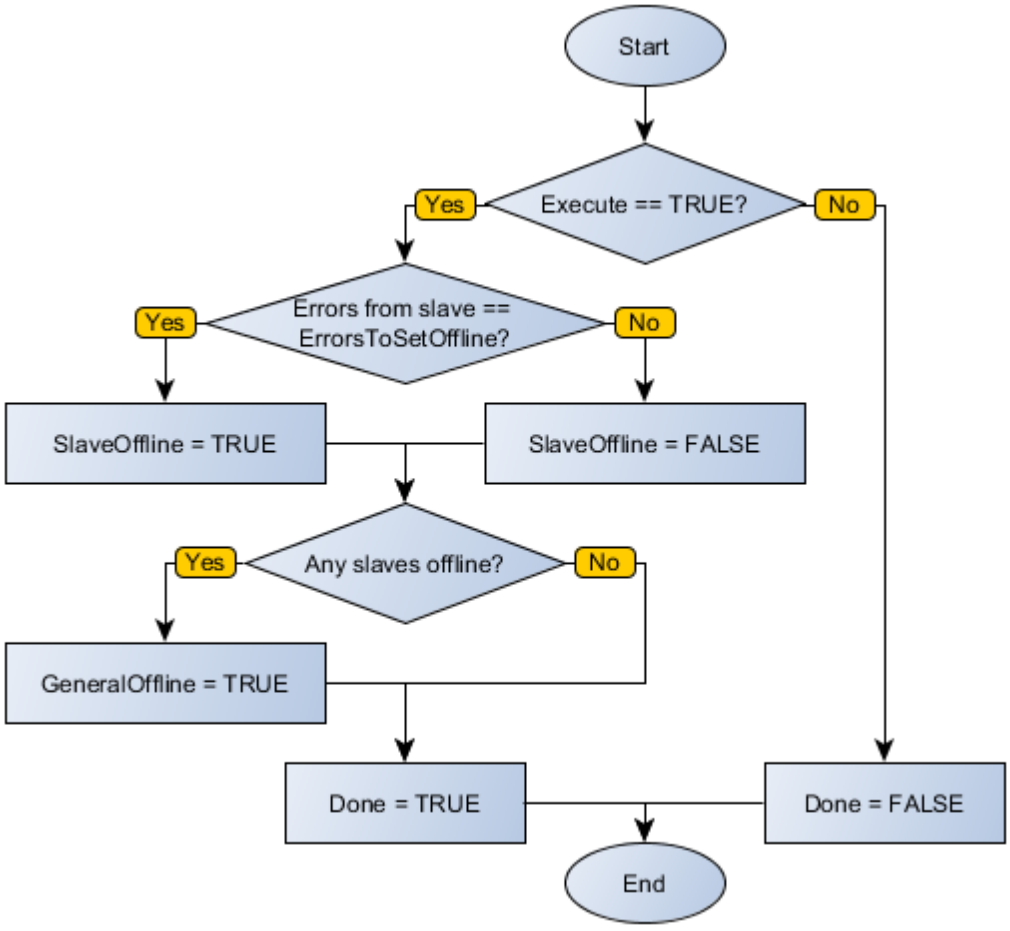
Variable Type	Name	Data Type	Description
VAR_INPUT	Execute	BOOL	Block enabling
	ErrorsToSetOffline#	BYTE	Amount of errors that the master must identify until it considers communication w ith an offline slave
	AddressSlave1#	BYTE	Slave address 1 to be monitored
	AddressSlave2#	BYTE	Slave address 2 to be monitored
	AddressSlave3#	BYTE	Slave address 3 to be monitored
	AddressSlave4#	BYTE	Slave address 4 to be monitored
VAR_OUTPUT	Done	BOOL	Output enabling
	GeneralOffline	BOOL	Flag indicating any one of the monitored communication is offline
	Slave1Offline	BOOL	Flag of offline status slave 1
	Slave2Offline	BOOL	Flag of offline status slave 2
	Slave3Offline	BOOL	Flag of offline status slave 3
	Slave4Offline	BOOL	Flag of offline status slave 4

Operation

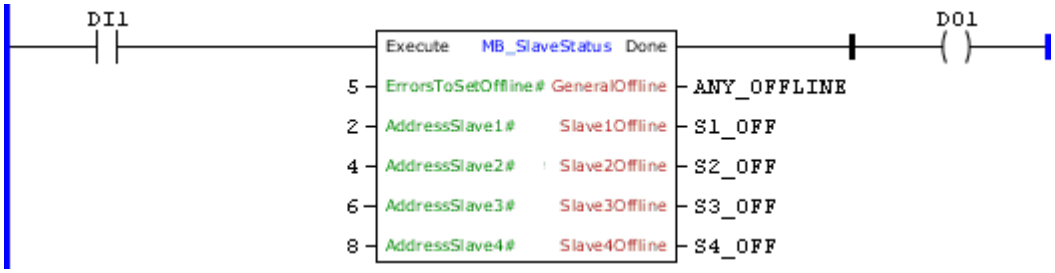
This block remains active while Execute is at TRUE level, updating its outputs according to the number of errors recorded for each slave. When Execute receives FALSE level, the inputs are ignored and the outputs are zeroed. The Done output receives TRUE level when Execute has TRUE level and block finished its execution.

The ErrorsToSetOffline # input allows registering the number of errors identified in a slave that will feature an offline communication. AddressSlave inputs allow inserting four slave addresses to be monitored. When this monitored slave reports the programmed number of errors, its corresponding SlaveOffline output is set to TRUE level. If any of SlaveOffline outputs is at TRUE level, GeneralOffline also receives TRUE level.

Block Flowchart



Example

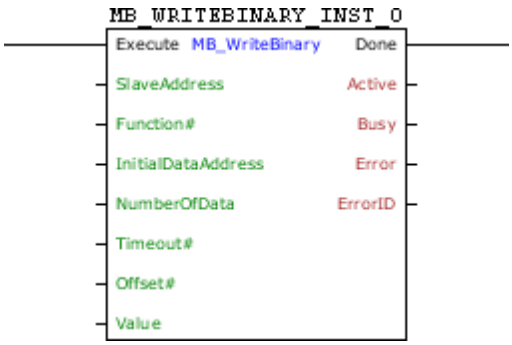


The above example checks the number of error responses sent by the slaves 2, 4, 6 and 8 of the Modbus RTU. If any of them is greater than 5, its SX_OFF status is led to TRUE level. The block ends successfully, Done output is activated.

11.2.6.2.1.6 MB_WriteBinary

Block that performs a writing of up to 128 binary data (via Write Single Coil or Write Multiple Coils) in a slave on the Modbus RTU network.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	Execute	BOOL	Block enabling
	SlaveAddress	BYTE	Slave address
	Function#	BYTE	Writing function code
	InitialDataAddress	WORD	Initial bit address where the data will be written
	NumberOfData	BYTE	Number of bits to be written (1 to 128)
	Timeout#	WORD	Maximum waiting time for the slave response [ms]
	Offset#	BOOL	Offset Indication in InitialDataAddress, i.e., need to subtract 1 from this number
	Value	BOOL	Variable that stores the data to be written
VAR_OUTPUT	Done	BOOL	Output enabling
	Active	BOOL	Awaiting response flag
	Busy	BOOL	Flag indicating the RS485 interface is busy with another request
	Error	BOOL	Error in the execution flag
	ErrorID	BYTE	Identifier of the occurred error
VAR	MB_WRITEBINARY_INST_0	MB_WRITEBINARY	Instance of access to block structure

Operation

When this block detects a leading edge on Execute, it checks whether the Modbus RTU slave in specified address in SlaveAddress is free to send data (Busy variable at FALSE level). If so, it sends the writing request of a number of bits indicated by NumberOfData in InitialDataAddress address using chosen function in Function# and sets the Active output, resetting it when receiving the response from the slave. If the slave is not free, the block waits Busy go to FALSE level to resubmit the request.

NOTE!
If Execute goes to FALSE level and Busy is still at TRUE level, the request is canceled.

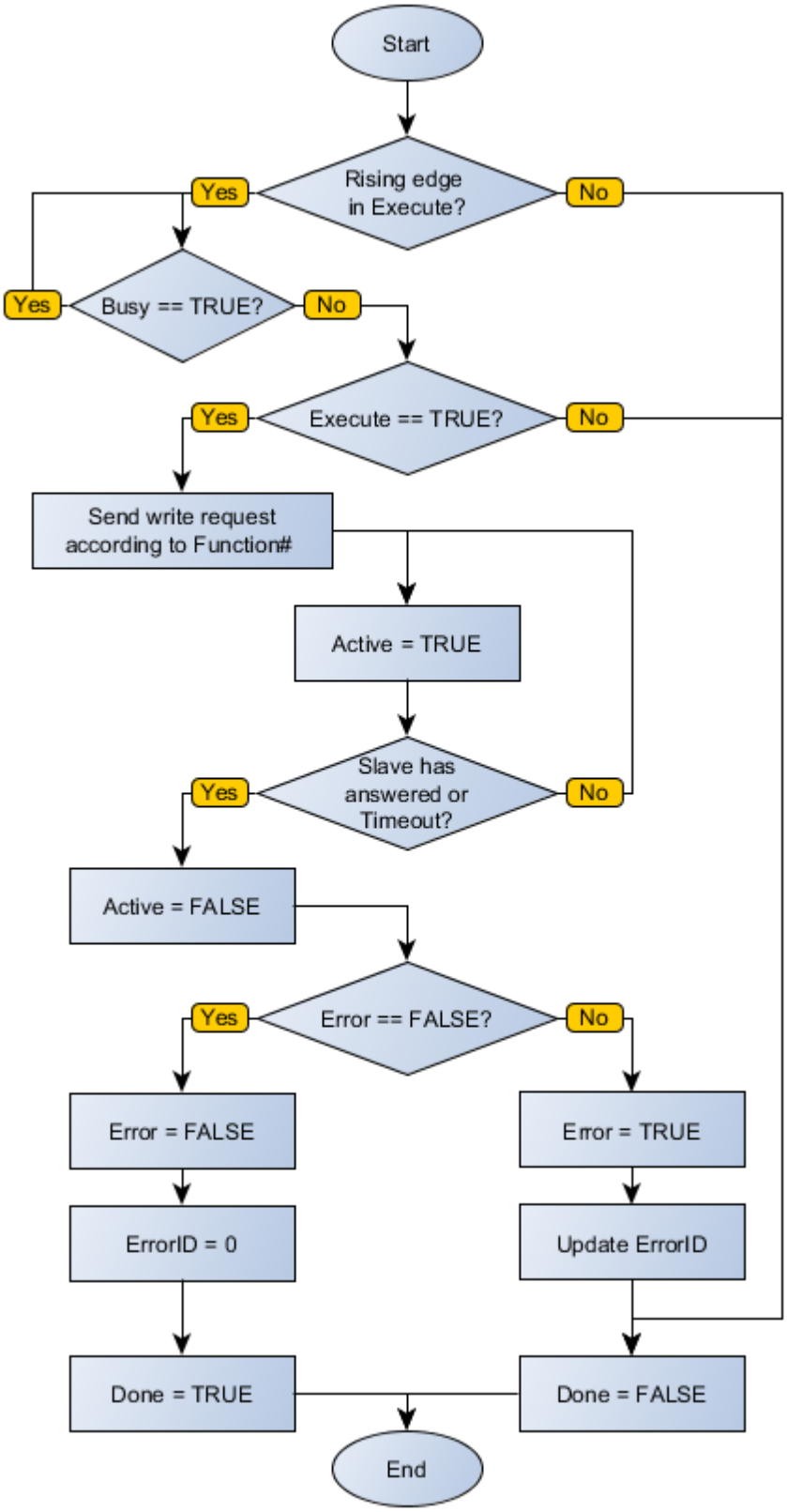
NOTE!
Value is an array of size equal to NumberOfData. It is important to check this compatibility not to generate errors in the block.

When Execute has FALSE value, Done remains FALSE. The Done output is only activated when the block finishes executing successfully, remaining at TRUE level until Execute receives FALSE.

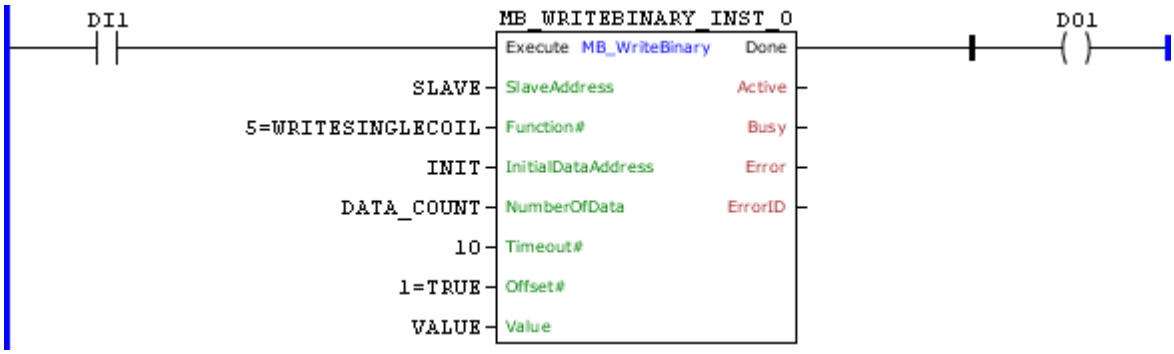
If there is any error in the execution, the Error output is enabled and ErrorID displays an error code according to the table below.

Code	Description
0	Executed successfully
1	Invalid input data
2	Master not enabled
4	Timeout in slave response
5	Slave returned error

Block Flowchart


Start
Rising edge in Execute?
Yes
No
Busy == TRUE?
Yes
No
Execute == TRUE?
Yes
No
Send write request according to Function#
Active = TRUE
Slave has answered or Timeout?
Yes
No
Active = FALSE
Error == FALSE?
Yes
No
Error = FALSE
Error = TRUE
ErrorID = 0
Update ErrorID
Done = TRUE
Done = FALSE
End

Example

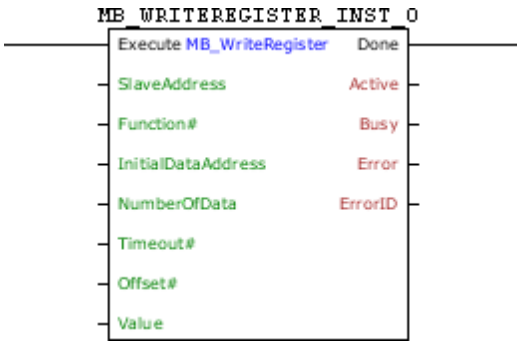


The example above requests written data contained in VALUE, with size described by DATA_COUNT, at addresses positioned from INIT on Modbus RTU slave at address SLAVE using the function Write Single Coil. The block ends successfully, Done output is activated.

11.2.6.2.1.7 MB_WriteRegister

Block that performs a reading of up to sixteen 16-bit registers (via Write Single Register or Write Multiple Registers) of a slave on the Modbus RTU network.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	Execute	BOOL	Block enabling
	SlaveAddress	BYTE	Slave address
	Function#	BYTE	Writing function code
	InitialDataAddress	WORD	Initial register address to be written
	NumberOfData	BYTE	Number of registers to be written (1 to 16)
	Timeout#	WORD	Maximum waiting time for the slave response [ms]
	Offset#	BOOL	Offset Indication in InitialDataAddress, i.e., need to subtract 1 from this number
	Value	BYTE SINT USINT WORD UINT INT DWORD UDINT DINT REAL	Variable that stores the data to be written
VAR_OUTPUT	Done	BOOL	Output enabling
	Active	BOOL	Awaiting response flag
	Busy	BOOL	Flag indicating the RS485 interface is busy with another request
	Error	BOOL	Error in the execution flag
	ErrorID	BYTE	Identifier of the occurred error
VAR	MB_WRITEREGISTER_INST_0	MB_WRITEREGISTER	Instance of access to block structure

Operation

When this block detects a leading edge on Execute, it checks whether the Modbus RTU slave in specified address in SlaveAddress is free to send data (Busy variable at FALSE level). If so, it sends the writing request of Value values in a number of registers indicated by NumberOfData in InitialDataAddress address using chosen function in Function# and sets the Active output, resetting it when receiving the response from the slave. If the slave is not free, the block waits Busy go to FALSE level to resubmit the request.

NOTE!
If Execute goes to FALSE level and Busy is still at TRUE level, the request is canceled.

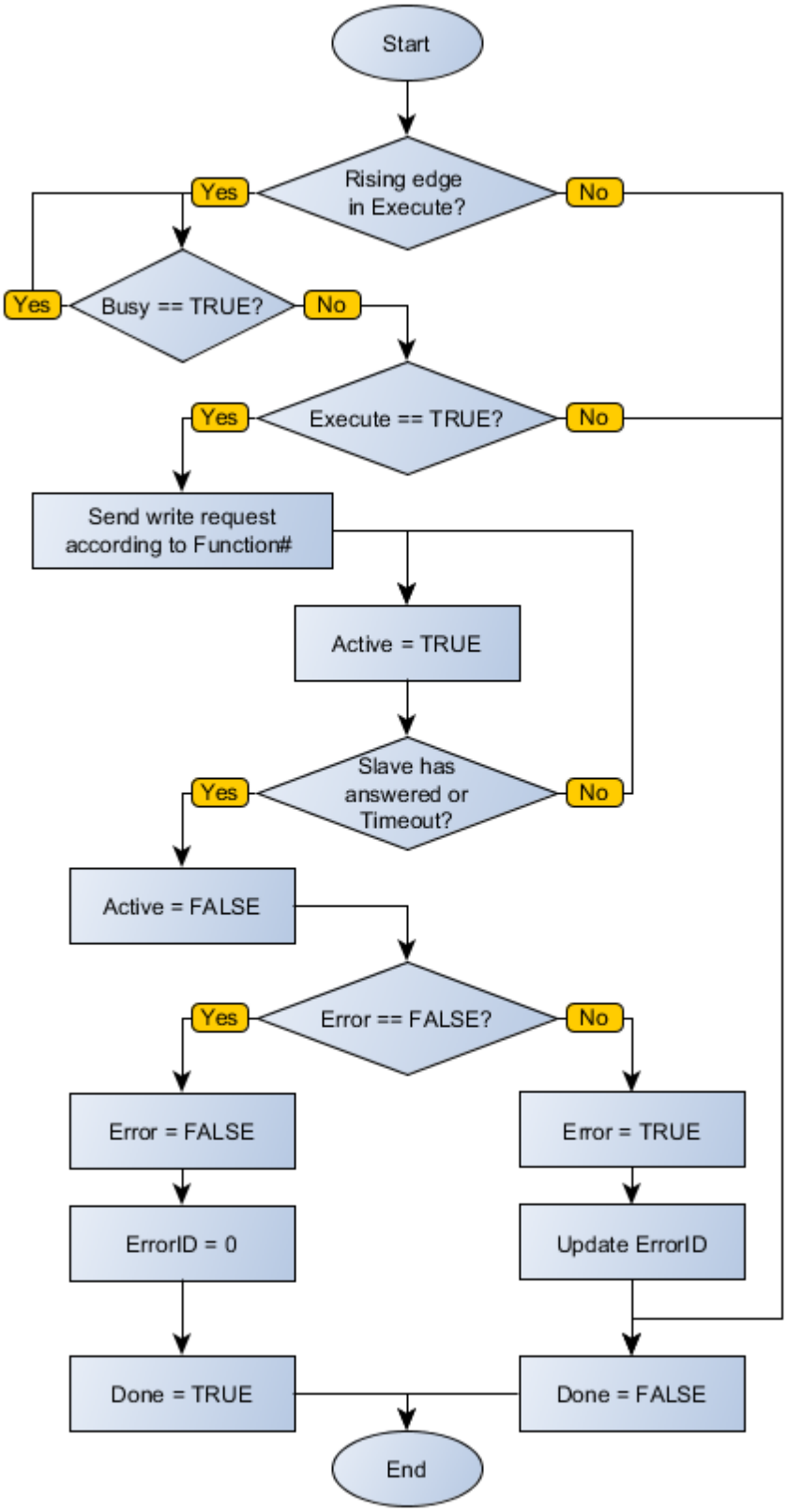
NOTE!
Value is an array of number of bits NumberOfData multiplied by 16. That is, if NumberOfData is 16, Value can be an array of 32 BYTE positions, 16 WORD positions or 8 DWORD positions. It is important to check this compatibility not to generate errors in the block.

When Execute has FALSE value, Done remains FALSE. The Done output is only activated when the block finishes executing successfully, remaining at TRUE level until Execute receives FALSE.

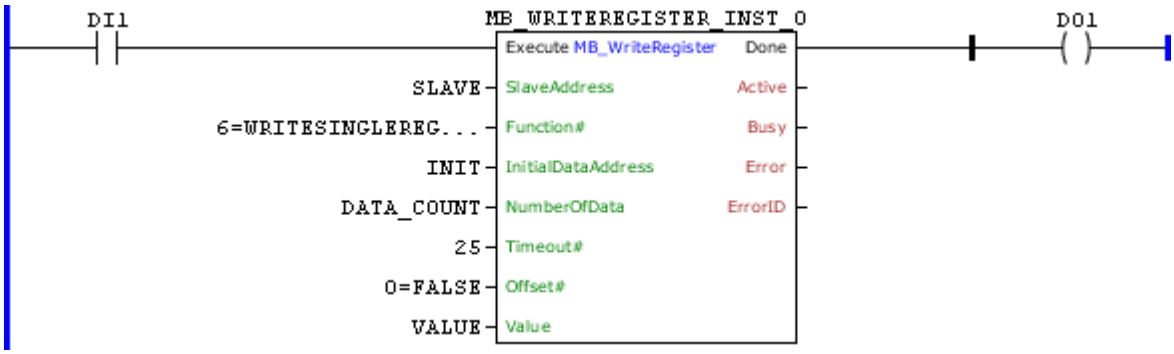
If there is any error in the execution, the Error output is enabled and ErrorID displays an error code according to the table below.

Code	Description
0	Executed successfully
1	Invalid input data
2	Master not enabled
4	Timeout in slave response
5	Slave returned error

Block Flowchart


Start
Rising edge in Execute?
Yes
No
Busy == TRUE?
Yes
No
Execute == TRUE?
Yes
No
Send write request according to Function#
Active = TRUE
Slave has answered or Timeout?
Yes
No
Active = FALSE
Error == FALSE?
Yes
No
Error = FALSE
Error = TRUE
ErrorID = 0
Update ErrorID
Done = TRUE
Done = FALSE
End

Example



The example above requests written data contained in VALUE, with size described by DATA_COUNT, at addresses positioned from INIT on Modbus RTU slave at address SLAVE using the function Write Single Register. The block ends successfully, Done output is activated.

11.2.6.3 Compare

11.2.6.3.1 COMP_EQ

Block that compares the values of Value1 and Value2, enabling the output Q if both are equal.

Ladder Representation

Block Structure

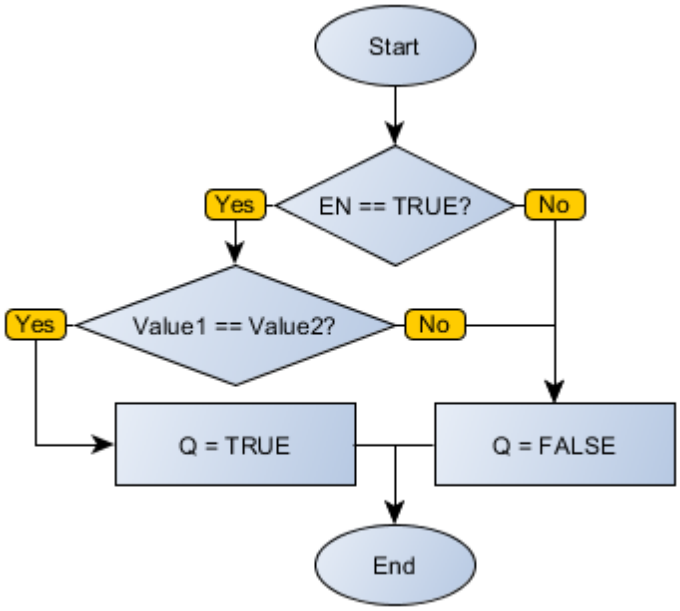
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	First value of comparison
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Second value of comparison
VAR_OUTPUT	Q	BOOL	Indicator of equality

Operation

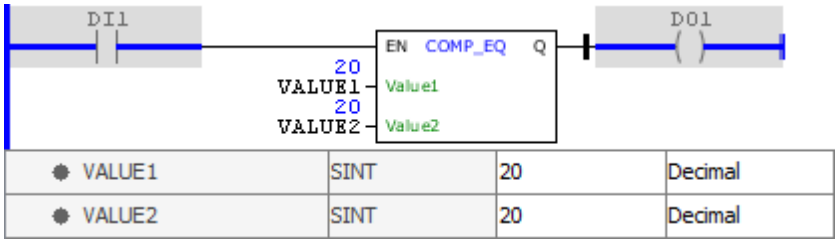
When this block has a TRUE value in EN, it sends to the output Q the TRUE value if Value1 and Value2 are the same. Otherwise, Q receives FALSE.

When EN has FALSE value, Q remains in FALSE.

Block Flowchart

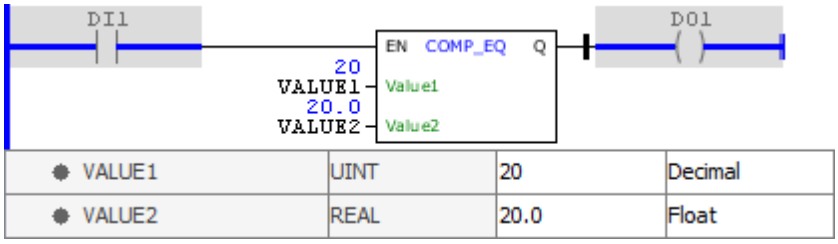


Example



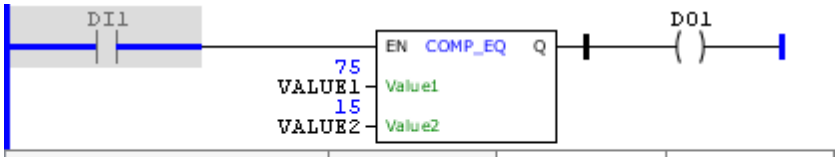
VALUE1	SINT	20	Decimal
VALUE2	SINT	20	Decimal

The example above checks equality between VALUE1 and VALUE2. Since both variables have the same value, the Q output is activated.



VALUE1	UINT	20	Decimal
VALUE2	REAL	20.0	Float

The example above checks equality between VALUE1 and VALUE2. Since both variables have the same value, the Q output is activated. Notice that the types of the input variables can be different without causing execution problems.



VALUE1	UINT	75	Decimal
VALUE2	UINT	15	Decimal

The example above checks equality between VALUE1 and VALUE2. Since both variables have different values, the Q output is disabled.

11.2.6.3.2 COMP_GE

Block that compares the values of Value1 and Value2, enabling the output Q if Value1 is higher than or equal to Value2.

Ladder Representation

Block Structure

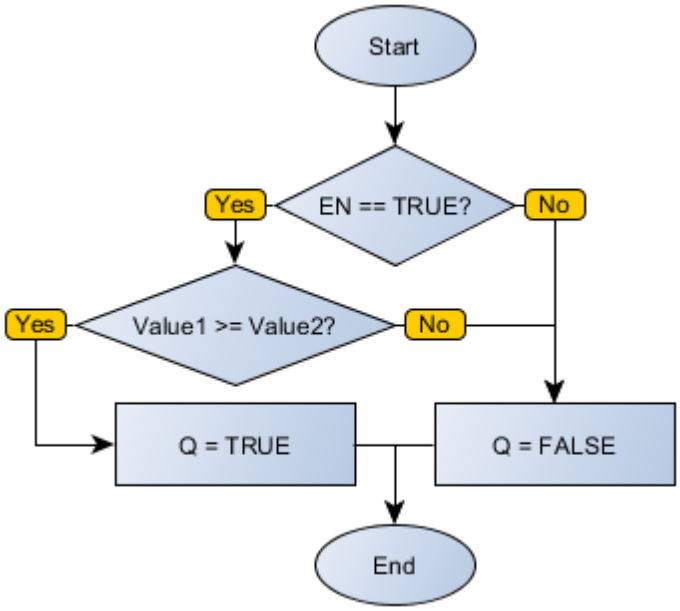
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	First value of comparison
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Second value of comparison
VAR_OUTPUT	Q	BOOL	Indicator of equality or majority of Value1

Operation

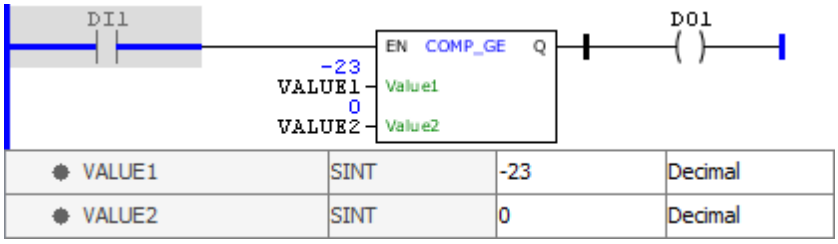
When this block has a TRUE value in EN it sends the Q output to the TRUE value if Value1 is higher than or equal to Value2. Otherwise, Q receives FALSE.

When EN has FALSE value, Q remains in FALSE.

Block Flowchart

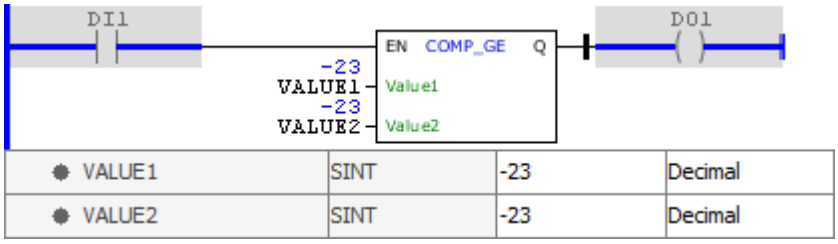


Example



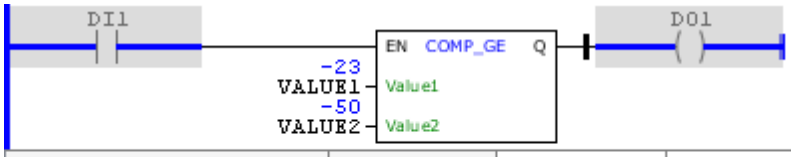
VALUE1	SINT	-23	Decimal
VALUE2	SINT	0	Decimal

The example above checks equality or majority of VALUE1 in relation to VALUE2. Since VALUE1 has lower value than VALUE2, the Q output is disabled.



VALUE1	SINT	-23	Decimal
VALUE2	SINT	-23	Decimal

The example above checks equality or majority of VALUE1 in relation to VALUE2. Since both variables have the same value, the Q output is activated.



VALUE1	SINT	-23	Decimal
VALUE2	SINT	-50	Decimal

The example above checks equality or majority of VALUE1 in relation to VALUE2. Since VALUE1 has higher value than VALUE2, the Q output is activated.

11.2.6.3.3 COMP_GT

Block that compares the values of Value1 and Value2, enabling the output Q if Value1 is higher than Value2.

Ladder Representation

Block Structure

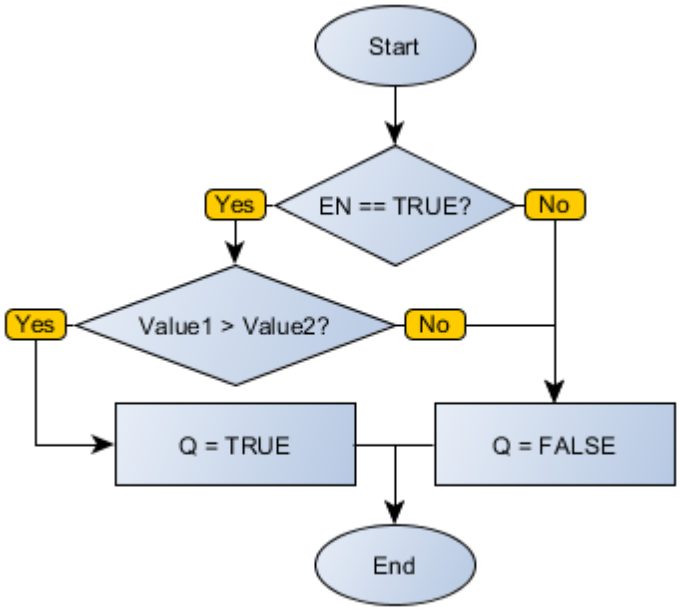
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	First value of comparison
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Second value of comparison
VAR_OUTPUT	Q	BOOL	Indicator of majority of Value1

Operation

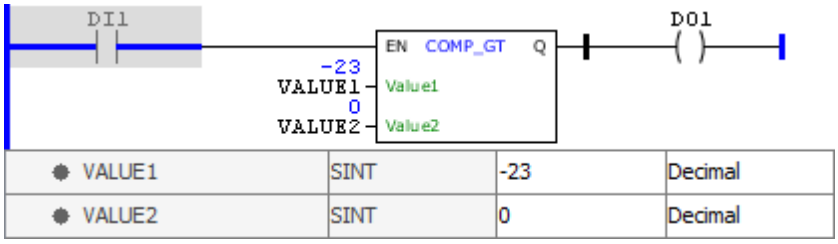
When this block has a TRUE value in EN, it sends to the Q output the TRUE value if Value1 is higher than Value2. Otherwise, Q receives FALSE.

When EN has FALSE value, Q remains in FALSE.

Block Flowchart

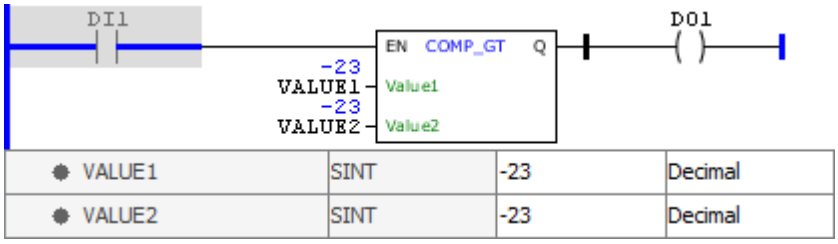


Example



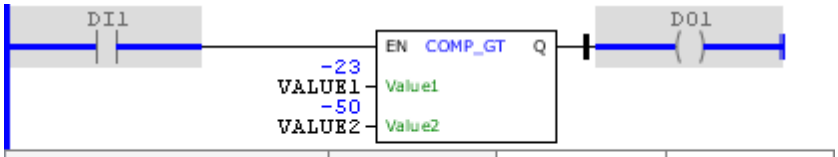
VALUE1	SINT	-23	Decimal
VALUE2	SINT	0	Decimal

The example above checks the majority of VALUE1 in relation to VALUE2. Since VALUE1 has lower value than VALUE2, the Q output is disabled.



VALUE1	SINT	-23	Decimal
VALUE2	SINT	-23	Decimal

The example above checks the majority of VALUE1 in relation to VALUE2. Since both variables have the same value, the Q output is disabled.



VALUE1	SINT	-23	Decimal
VALUE2	SINT	-50	Decimal

The example above checks the majority of VALUE1 in relation to VALUE2. Since VALUE1 has higher value than VALUE2, the Q output is activated.

11.2.6.3.4 COMP_LE

Block that compares the values of Value1 and Value2, enabling the output Q if Value1 is lower than or equal to Value2.

Ladder Representation

Block Structure

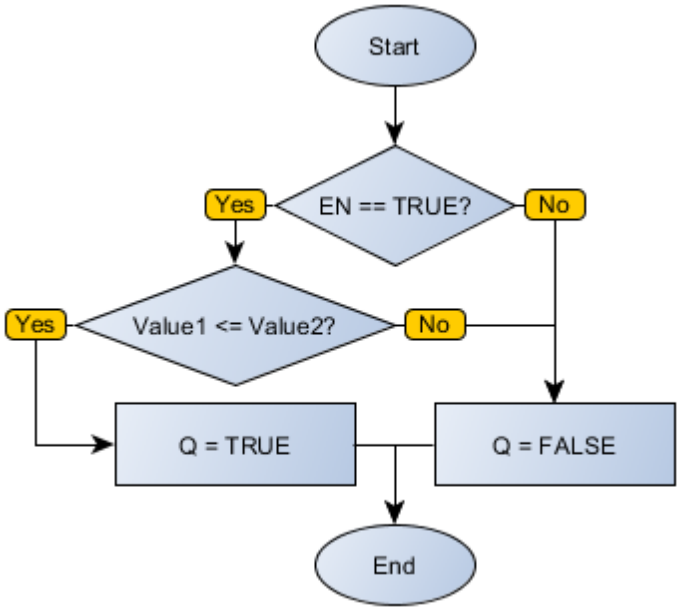
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	First value of comparison
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Second value of comparison
VAR_OUTPUT	Q	BOOL	Indicator of equality or minority of Value1

Operation

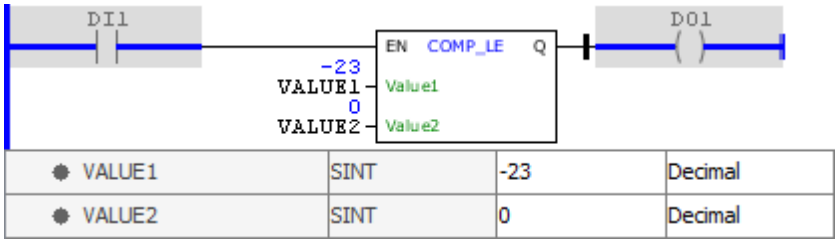
When this block has a TRUE value in EN, it sends to the Q output the TRUE value if Value1 is lower than or equal to Value2. Otherwise, Q receives FALSE.

When EN has FALSE value, Q remains in FALSE.

Block Flowchart

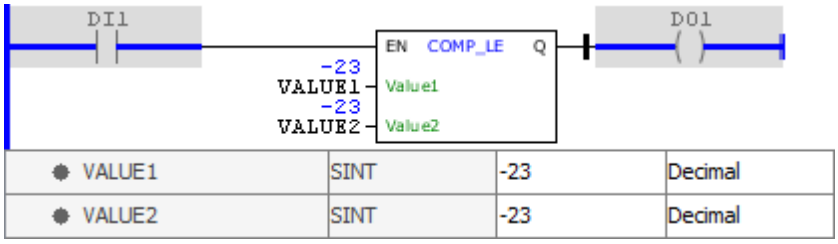


Example



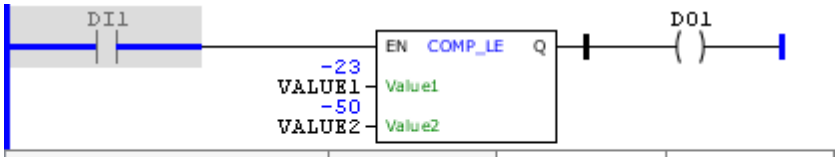
VALUE1	SINT	-23	Decimal
VALUE2	SINT	0	Decimal

The example above checks equality or minority of VALUE1 in relation to VALUE2. Since VALUE1 has lower value than VALUE2, the Q output is activated.



VALUE1	SINT	-23	Decimal
VALUE2	SINT	-23	Decimal

The example above checks equality or minority of VALUE1 in relation to VALUE2. Since both variables have the same value, the Q output is activated.



VALUE1	SINT	-23	Decimal
VALUE2	SINT	-50	Decimal

The example above checks equality or minority of VALUE1 in relation to VALUE2. Since VALUE1 has higher value than VALUE2, the Q output is disabled.

11.2.6.3.5 COMP_LT

Block that compares the values of Value1 and Value2, enabling the output Q if Value1 is lower than Value2.

Ladder Representation

Block Structure

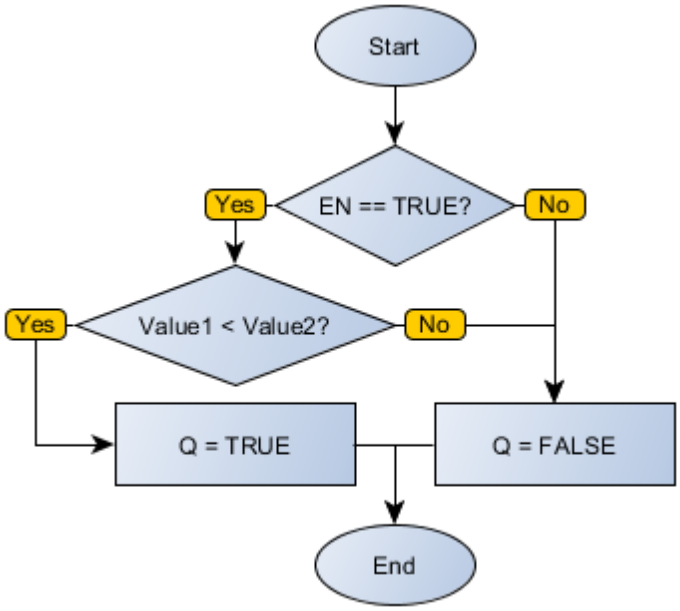
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	First value of comparison
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Second value of comparison
VAR_OUTPUT	Q	BOOL	Indicator of minority of Value1

Operation

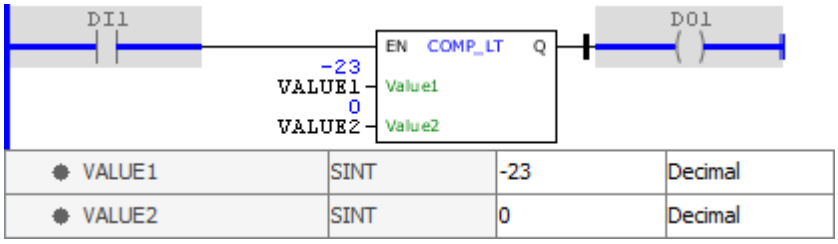
When this block has a TRUE value in EN, it sends to the Q output the TRUE value if Value1 is lower than or equal to Value2. Otherwise, Q receives FALSE.

When EN has FALSE value, Q remains in FALSE.

Block Flowchart

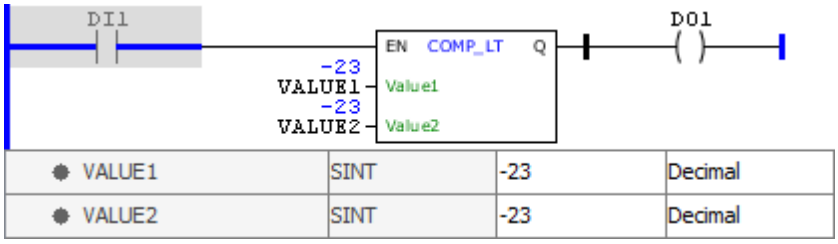


Example



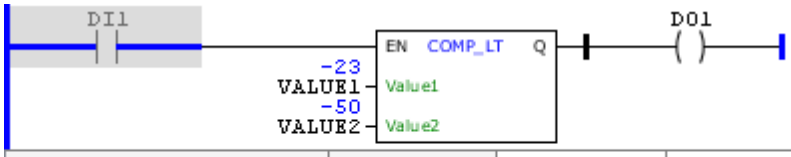
VALUE1	SINT	-23	Decimal
VALUE2	SINT	0	Decimal

The example above checks minority of VALUE1 in relation to VALUE2. Since VALUE1 has lower value than VALUE2, the Q output is activated.



VALUE1	SINT	-23	Decimal
VALUE2	SINT	-23	Decimal

The example above checks the minority of VALUE1 in relation to VALUE2. Since both variables have the same value, the Q output is disabled.



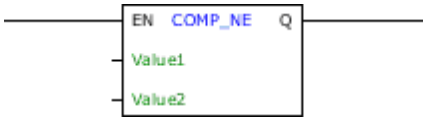
VALUE1	SINT	-23	Decimal
VALUE2	SINT	-50	Decimal

The example above checks the minority of VALUE1 in relation to VALUE2. Since VALUE1 has higher value than VALUE2, the Q output is disabled.

11.2.6.3.6 COMP_NE

Block that compares the values of Value1 and Value2, enabling the Q output if Value1 is different from Value2.

Ladder Representation



Block Structure

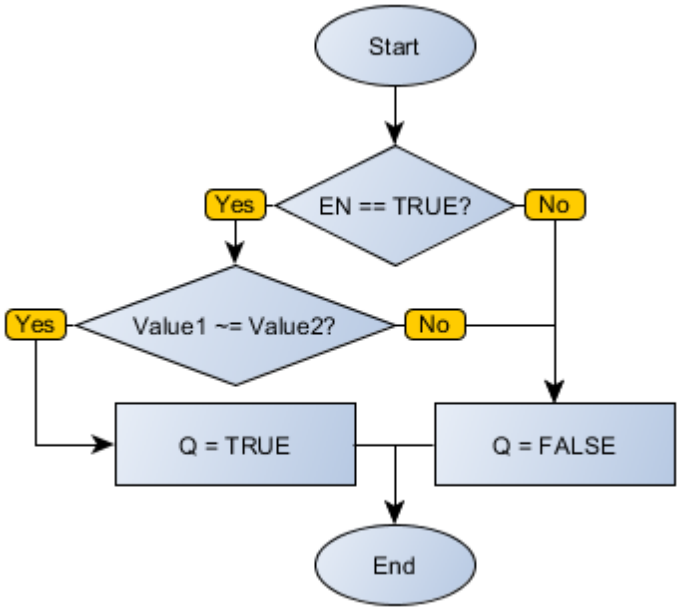
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	First value of comparison
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Second value of comparison
VAR_OUTPUT	Q	BOOL	Indicator of inequality

Operation

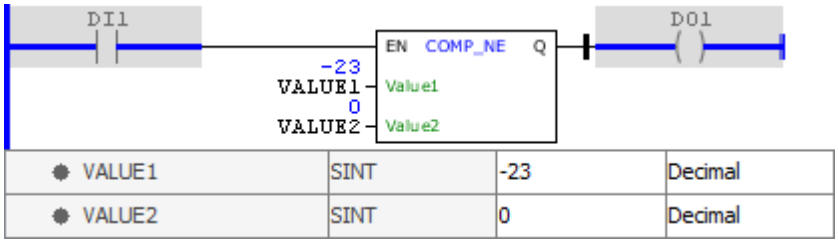
When this block has a TRUE value in EN, it sends to the Q output the TRUE value if Value1 is different from Value2. Otherwise, Q receives FALSE.

When EN has FALSE value, Q remains in FALSE.

Block Flowchart

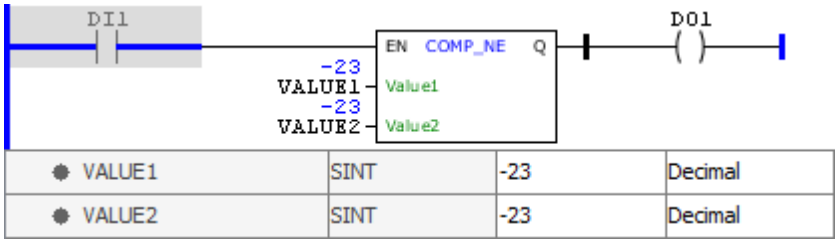


Example



VALUE1	SINT	-23	Decimal
VALUE2	SINT	0	Decimal

The example above checks inequality between VALUE1 and VALUE2. Since both variables have different values, the Q output is activated.



VALUE1	SINT	-23	Decimal
VALUE2	SINT	-23	Decimal

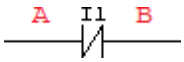
The example above checks equality between VALUE1 and VALUE2. Since both variables have the same value, the Q output is disabled.

11.2.6.4 Contact

11.2.6.4.1 NCCONTACT

Normally closed contact.

Ladder Representation



Block Structure

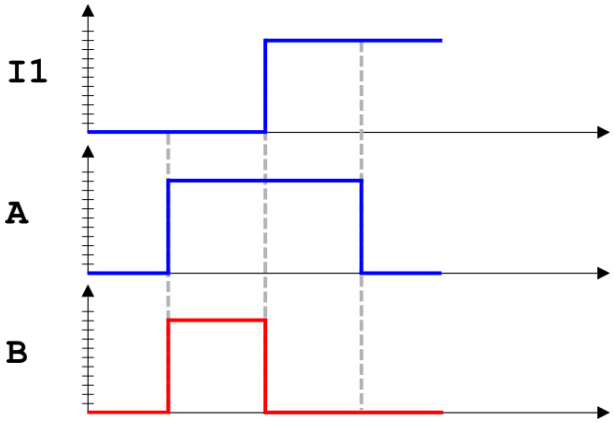
Variable Type	Name	Data Type	Description
VAR_OUTPUT	I1	BOOL	Block control input

Operation

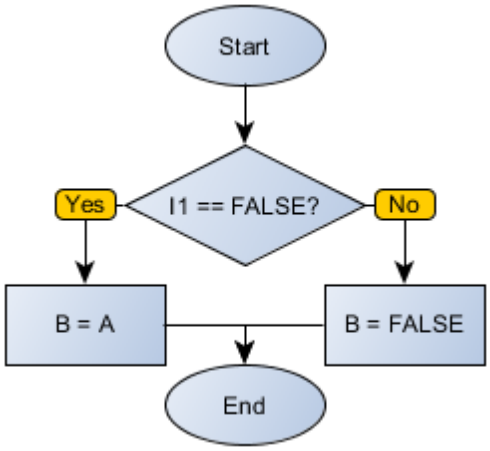
When variable I1 is with TRUE value, B receives FALSE.
When variable I1 is with FALSE value, B receives the value of A.

NOTE!
Watch out for series and parallel associations of contacts. Refer to section Contact Logic for further information.

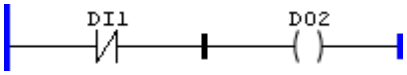
Diagram



Block Flowchart



Example

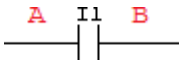


The above example performs the transfer of the opposite value of digital input DI1 to the digital output DO2.

11.2.6.4.2 NOCONTACT

Normally open contact.

Ladder Representation



Block Structure

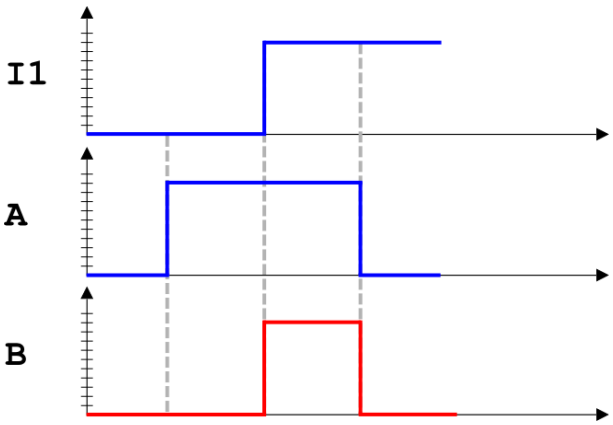
Variable Type	Name	Data Type	Description
VAR_OUTPUT	I1	BOOL	Block control input

Operation

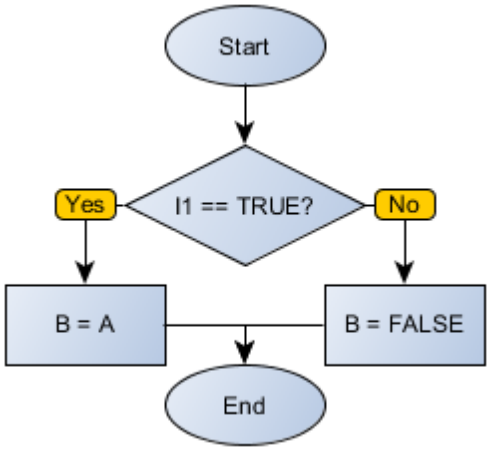
When variable I1 is with FALSE value, B receives FALSE.
When variable I1 is with TRUE value, B receives the value of A.

NOTE!
Watch out for series and parallel associations of contacts. Refer to section Contact Logic for further information.

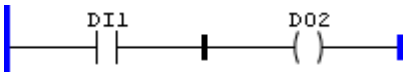
Diagram



Block Flowchart



Example



The above example performs the transfer of the value of digital input DI1 to the digital output DO2.

11.2.6.4.3 NTSCONTACT

Falling edge transition contact.

Ladder Representation

Block Structure

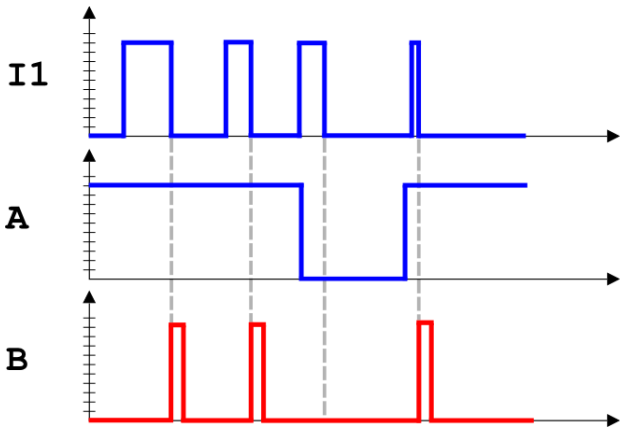
Variable Type	Name	Data Type	Description
VAR_INPUT	I1	BOOL	Block control input
VAR	NTSCONTACT_INST_0	NTSCONTACT	Instance of access to block structure

Operation

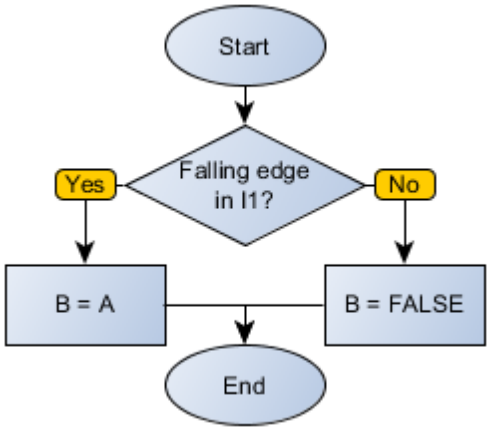
At the instant the variable I1 transitions from TRUE to FALSE (falling edge or negative edge transition), B receives the value of A for a scan cycle.
At all other times, B receives the FALSE value.

NOTE!
Watch out for series and parallel associations of contacts. Refer to section Contact Logic for further information.

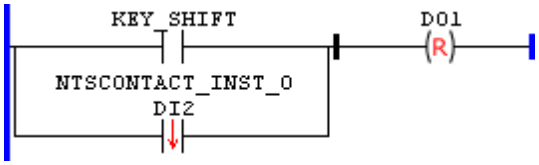
Diagram



Block Flowchart



Example

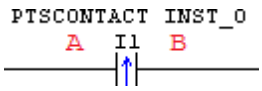


The above example resets the digital output DO1 if the SHIFT key is pressed or a positive pulse on the digital input DI2 is given.

11.2.6.4.4 PTSCONTACT

Leading edge transition contact.

Ladder Representation



Block Structure

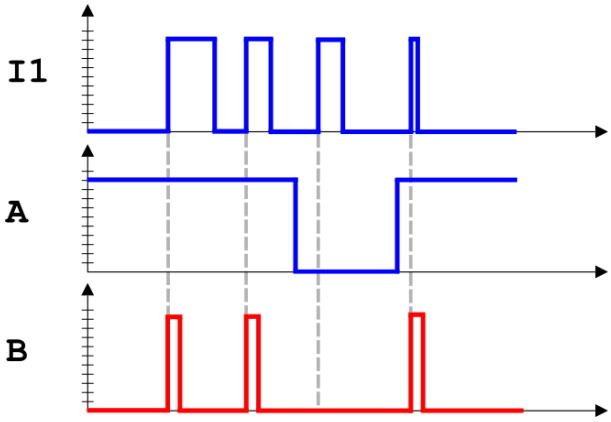
Variable Type	Name	Data Type	Description
VAR_INPUT	I1	BOOL	Block control input
VAR	PTSCONTACT_INST_0	PTSCONTACT	Instance of access to block structure

Operation

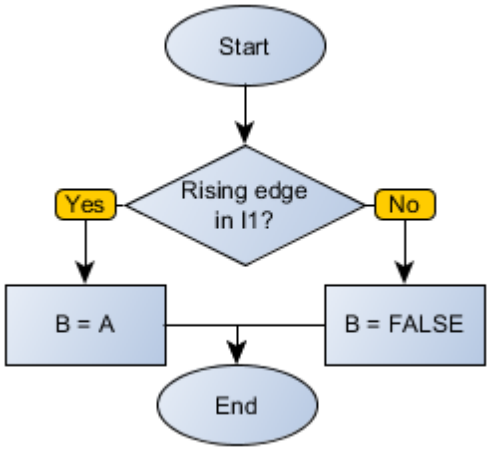
At the instant the variable I1 transitions from FALSE to TRUE (leading edge or positive edge transition), B receives the value of A for a scan cycle.
At all other times, B receives the FALSE value.

NOTE!
Watch out for series and parallel associations of contacts. Refer to section Contact Logic for further information.

Diagram



Block Flowchart



Example

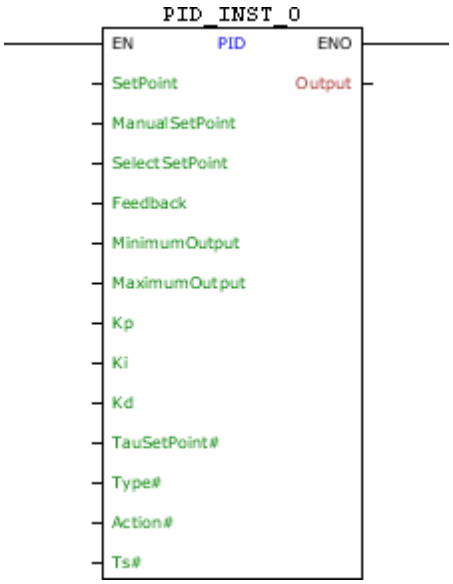
The above example resets the digital output DO1 if the SHIFT key is pressed and a positive pulse on the digital input DI2 is given.

11.2.6.5 Control

11.2.6.5.1 PID

Block that performs the function of a discrete PID controller. From the input variables, it calculates the corresponding controller output.

Ladder Representation



Block Structure

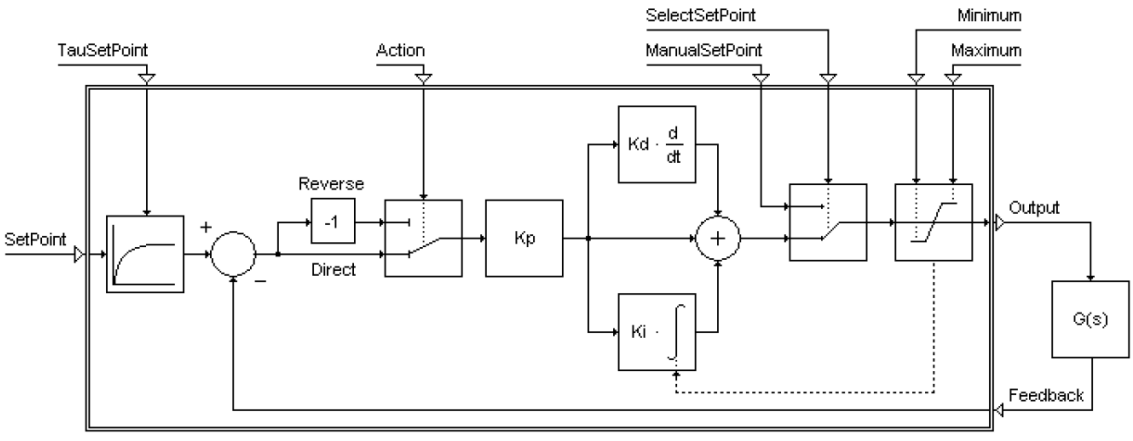
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	SetPoint	REAL	Automatic reference (pre-control)
	ManualSetPoint	REAL	Forced reference (post control)
	SelectSetPoint	BOOL	Selects w hich reference to use
	Feedback	REAL	Feedback loop variable
	MinimumOutput	REAL	Minimum value of the controller output
	MaximumOutput	REAL	Maximum value of the controller output
	Kp	REAL	Proportional gain
	Ki	REAL	Integral gain
	Kd	REAL	Derivative gain
	TauSetPoint#	REAL	Time constant of the automatic reference in put filter
	Type#	BYTE	Controller type
	Action#	BYTE	Control action
	Ts#	UINT	Sampling time [ms]
VAR_OUTPUT	ENO	BOOL	Output enabling
	Output	REAL	Controller output
VAR	PID_INST_0	PID	Instance of access to block structure

Operation

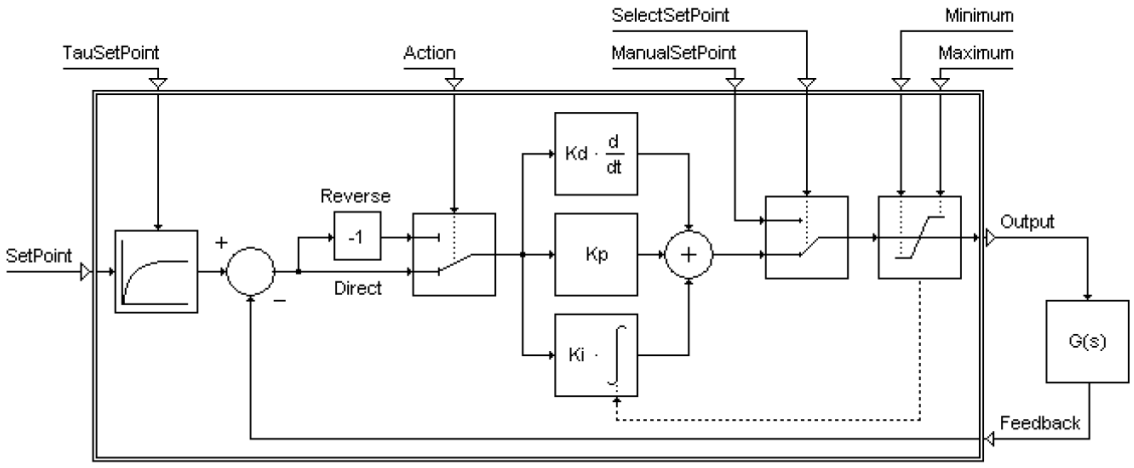
On the positive transition edge in EN, Output receives zero value, and the block executes its functionality as EN is at TRUE level.

When enabled, this block performs a routine PID control with the Kp, Ki and Kd parameters chosen. The PID topology used may be the Academic or Parallel, depending on what is chosen in Type#.

Academic Form:



Parallel Form:



The levels of the output signal of the controller are saturated at value MinimumOutput and MaximumOutput. The SelectSetPoint input level FALSE causes the SetPoint reference be adopted, allowing the controller maintains control over the process. When SelectSetPoint goes to TRUE level, the controller has no more domain, and ManualSetPoint becomes to be considered the output signal of the controller.

Action# will define the feedback operation. If Action# is DIRECT, the operation will be SetPoint – Feedback. If Action# is REVERSE, the operation will be Feedback – SetPoint.

Feedback receives the process variable considered as the plant output. Ts# receives the sampling time for the controller and # TauSetPoint receives the time constant for the input filter of the automatic reference.

When EN has FALSE value, Output remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

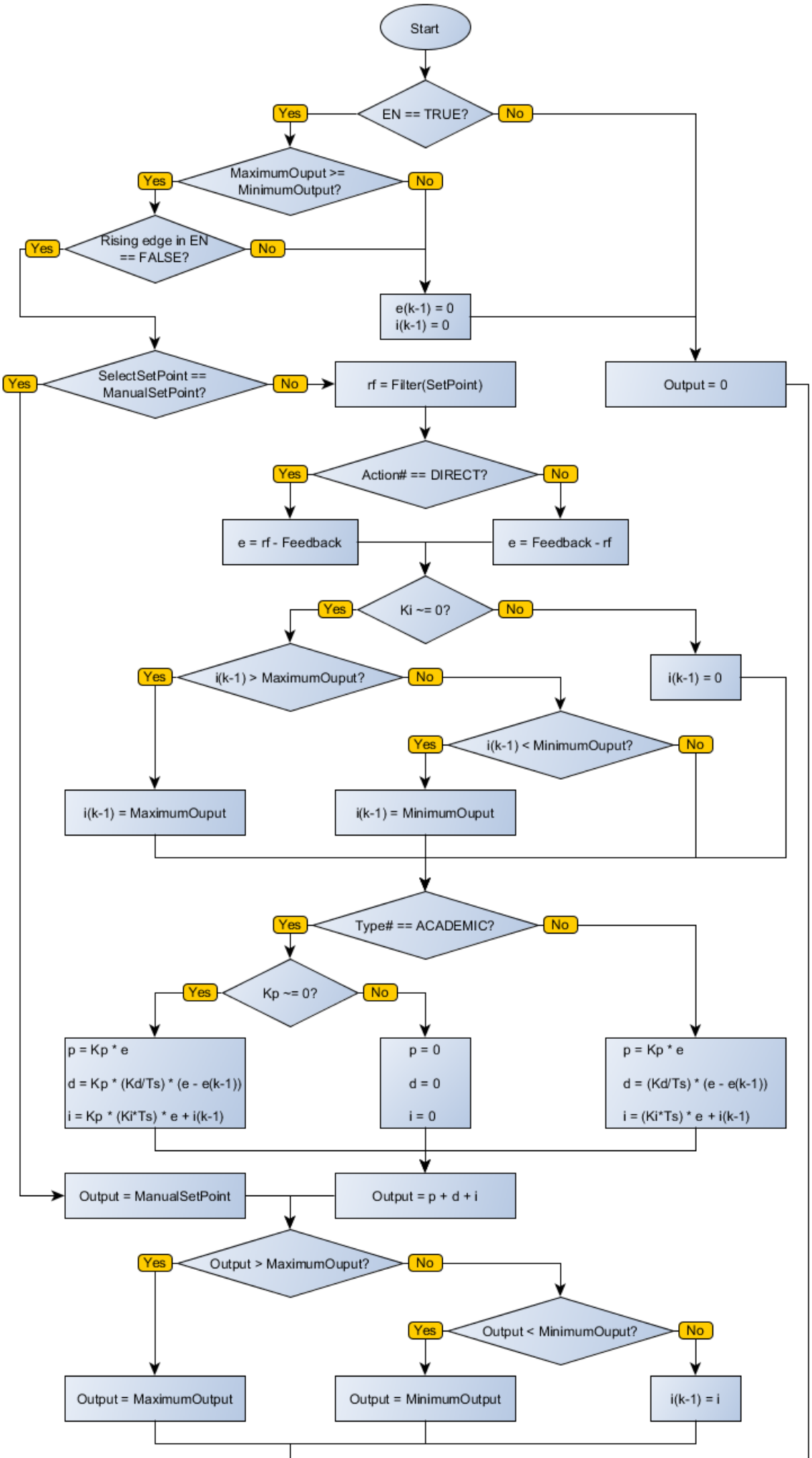
NOTE!
Effects of the alteration of gains on the process

- If Kp decreases, the process becomes slower; generally more stable or less oscillating; it has less overshoot.
- If Kp increases, the process responds faster; it may become more unstable or more oscillating; it has more overshoot.
- If Ki decreases, the process becomes slower, lagging to reach the "SetPoint"; it becomes more stable or less oscillating; it has less overshoot.
- If Ki increases, the process becomes faster, quickly reaching the "SetPoint"; it becomes more unstable or more oscillating; it has more overshoot.
- If Kd decreases, the process becomes slower; it has less overshoot.
- If Kd increases, it has more overshoot.

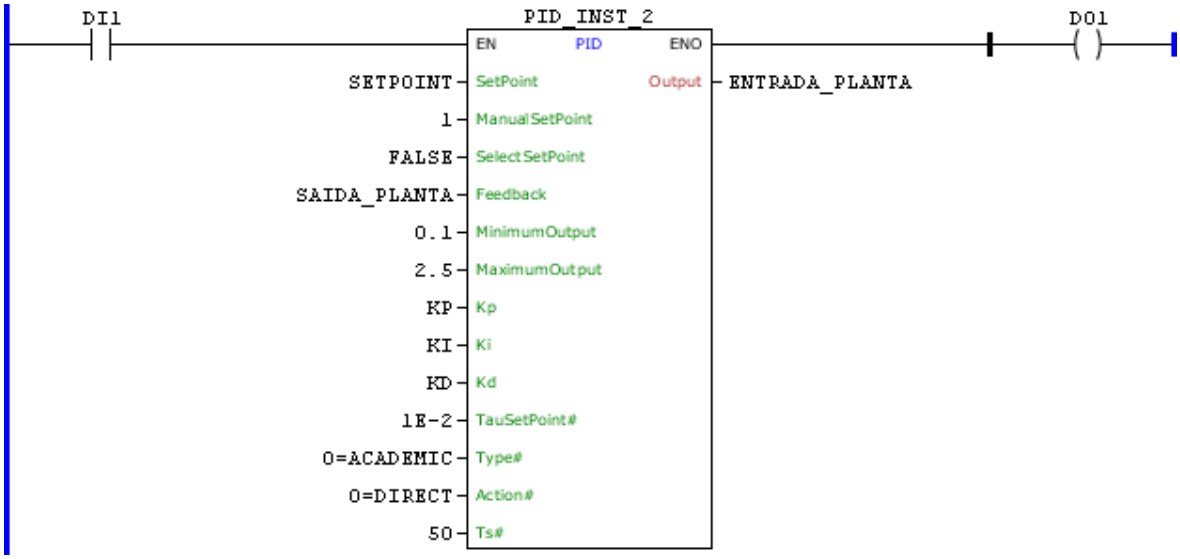
NOTE!
How to improve the performance of the process through the adjustment of gains (valid for the Academic PID)

- If the performance of the process is almost good, but the overshoot is a bit high, try to: (1) decrease Kp 20%, (2) decrease Ki 20% and/or (3) decrease Kd 50%.
- If the performance of the process is almost good, but it does not have overshoot and lags to reach the "SetPoint", try to: (1) increase Kp 20%, (2) increase Ki 20% and/or (3) increase Kd 50%.
- If the performance of the process is good, but the process output is varying too much, try to: (1) increase Kd 50%, (2) decrease Kp 20%.
- If the performance of the process is bad, i.e. after start up, the transitory lasts several periods of oscillation that reduce very slowly or never reduce at all, try to: (1) decrease Kp 50%.
- If the performance of the process is bad, i.e. after start up it slowly moves towards the "SetPoint" without overshoot, but is still very far and the process output is less than the rated value, try to: (1) increase Kp 50%, (2) increase Ki 50%, (3) increase Kd 70%.

Block Flowchart


Start
EN == TRUE?
Yes
No
MaximumOuput >= MinimumOutput?
Yes
No
Rising edge in EN == FALSE?
Yes
No
e(k-1) = 0
i(k-1) = 0
Output = 0
SelectSetPoint == ManualSetPoint?
Yes
No
rf = Filter(SetPoint)
Action# == DIRECT?
Yes
No
e = rf - Feedback
e = Feedback - rf
Ki ~= 0?
Yes
No
i(k-1) = 0
i(k-1) > MaximumOuput?
Yes
No
i(k-1) < MinimumOuput?
Yes
No
i(k-1) = MaximumOuput
i(k-1) = MinimumOuput
Type# == ACADEMIC?
Yes
No
Kp ~= 0?
Yes
No
p = Kp * e
d = Kp * (Kd/Ts) * (e - e(k-1))
i = Kp * (Ki*Ts) * e + i(k-1)
p = 0
d = 0
i = 0
p = Kp * e
d = (Kd/Ts) * (e - e(k-1))
i = (Ki*Ts) * e + i(k-1)
Output = ManualSetPoint
Output = p + d + i
Output > MaximumOuput?
Yes
No
Output < MinimumOuput?
Yes
No
Output = MaximumOutput
Output = MinimumOutput
i(k-1) = i

Example



The above example creates a loop of a digital PID form with sampling time 50 ms, using the constants KP, KI and KD for control. Automatic reference SETPOINT, filtered by a first order filter with time constant of 0:01 is used. The error signal is calculated as the difference between the filtered reference and variable SAIDA_PLANTA. The controller output is saturated between the values 0.1 and 2.5 and sent to the variable ENTRADA_PLANTA.

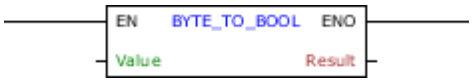
11.2.6.6 Conversion

11.2.6.6.1 BOOL

11.2.6.6.1.1 BYTE_TO_BOOL

Block that performs the conversion of a BYTE value into a BOOL value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT	Value in BYTE
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BOOL	Value in BOOL

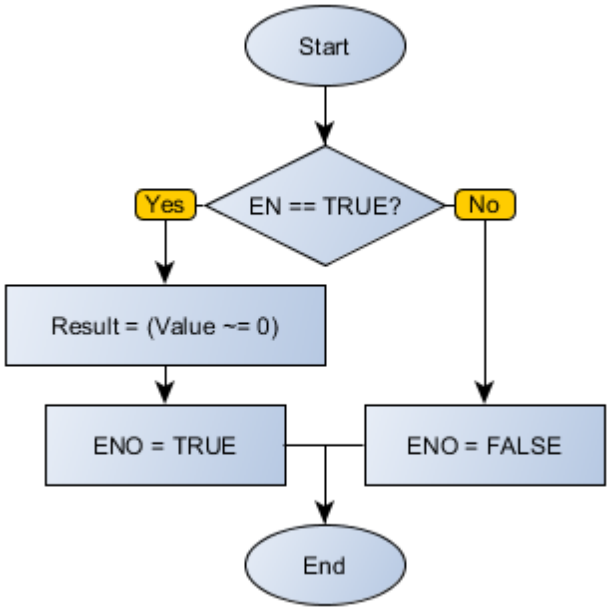
Operation

When this block has a TRUE value in EN, it interprets the Value value as BYTE and converts it into BOOL, storing in Result.

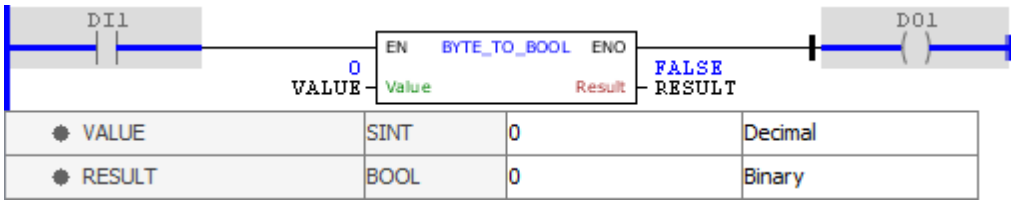
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

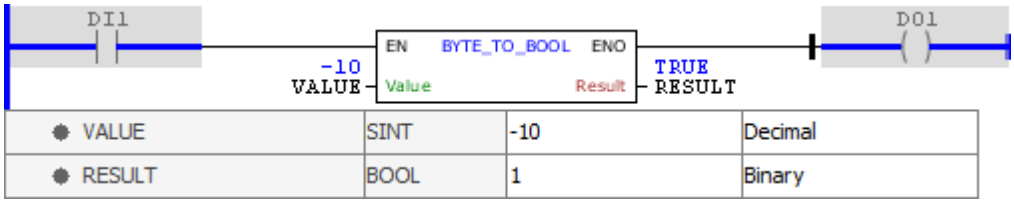
Block Flowchart



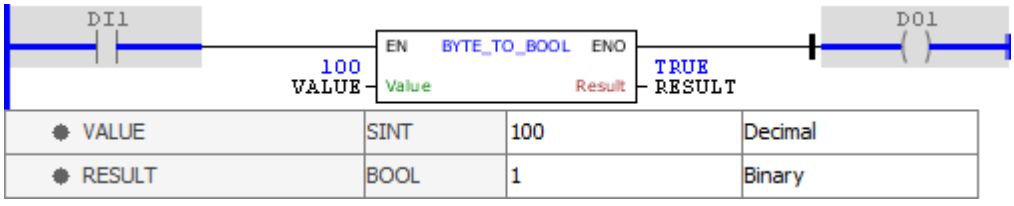
Example



VALUE	SINT	0	Decimal
RESULT	BOOL	0	Binary



VALUE	SINT	-10	Decimal
RESULT	BOOL	1	Binary



VALUE	SINT	100	Decimal
RESULT	BOOL	1	Binary

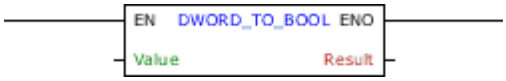
The examples above perform the conversion of VALUE variable, in BYTE, into a BOOL value storing

the final result in RESULT. The block ends with success and ENO output is activated.

11.2.6.6.1.2 DWORD_TO_BOOL

Block that performs the conversion of a DWORD value into a BOOL value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	DWORD UDINT DINT	Value in DWORD
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BOOL	Value in BOOL

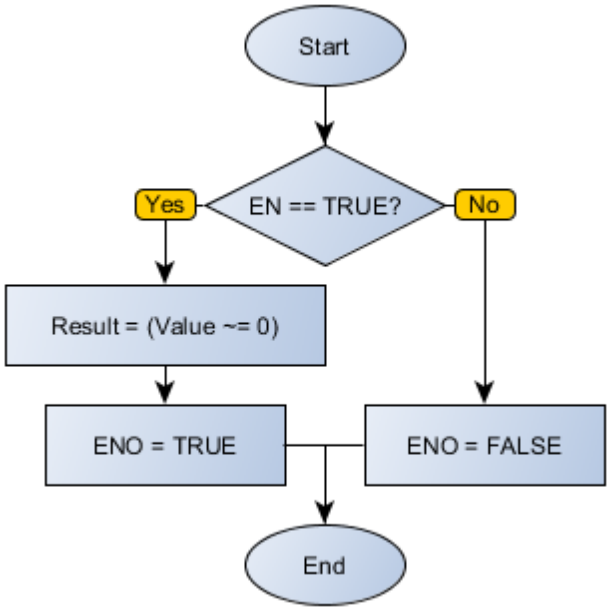
Operation

When this block has a TRUE value in EN, it interprets the Value value as DWORD and converts it into BOOL, storing in Result.

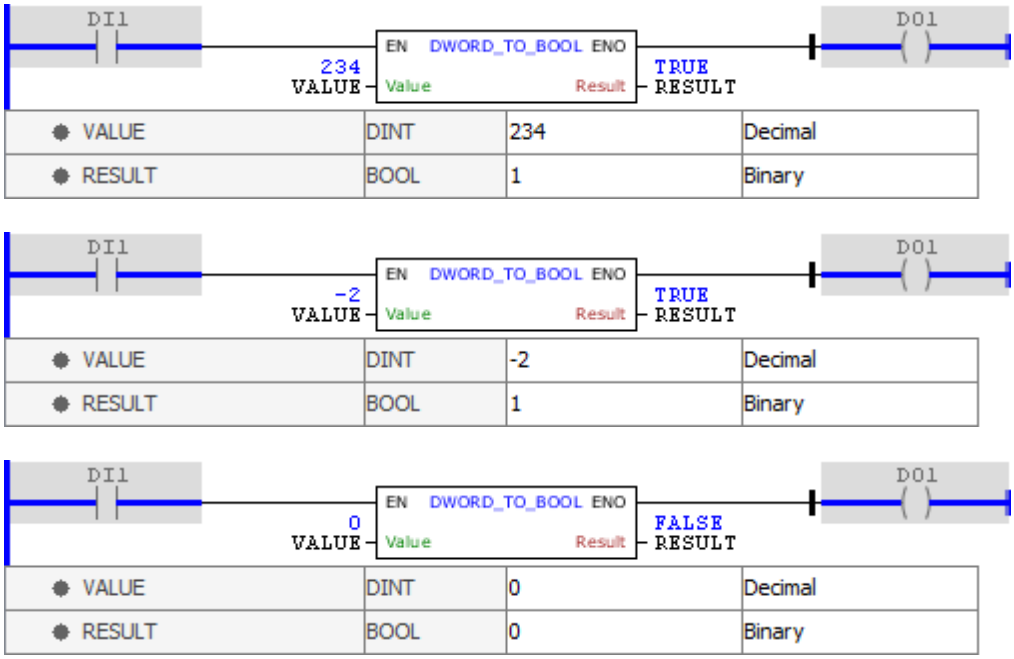
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example

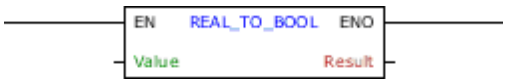


The examples above perform the conversion of VALUE variable, in DWORD, into a BOOL value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.2.6.6.1.3 REAL_TO_BOOL

Block that performs the conversion of a REAL value into a BOOL value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Value in REAL
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BOOL	Value in BOOL

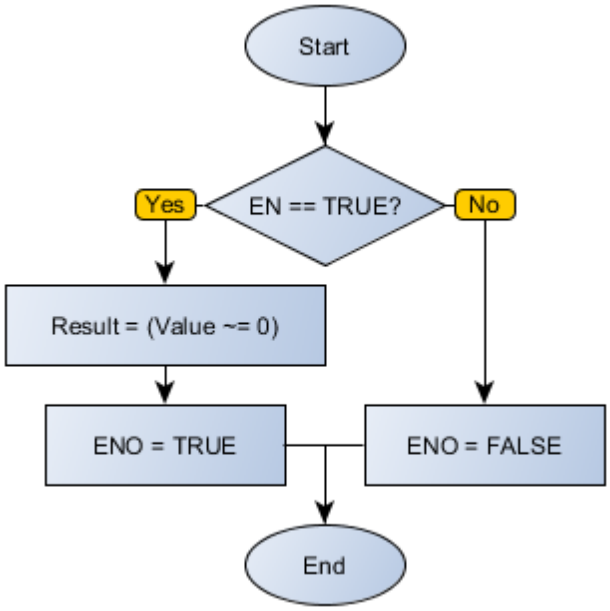
Operation

When this block has a TRUE value in EN, it interprets the Value value as REAL and converts it into BOOL, storing in Result.

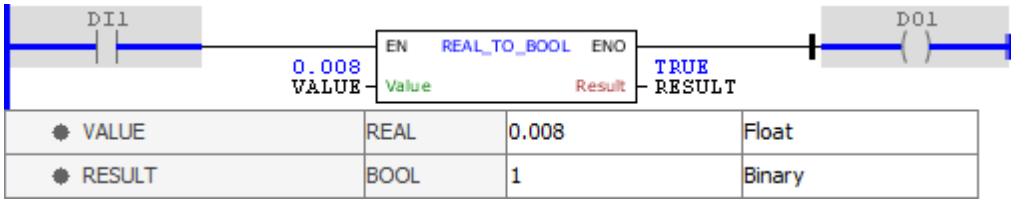
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

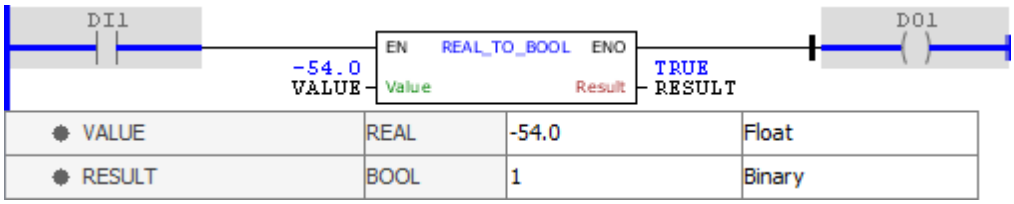
Block Flowchart



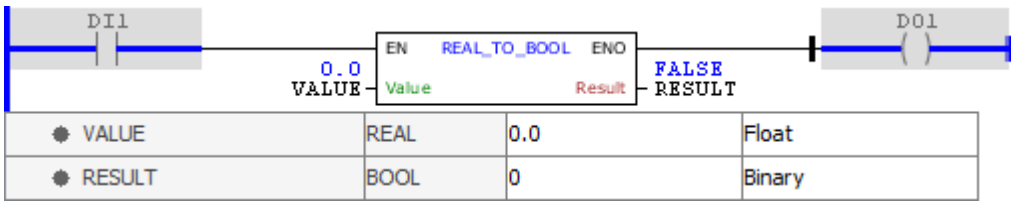
Example



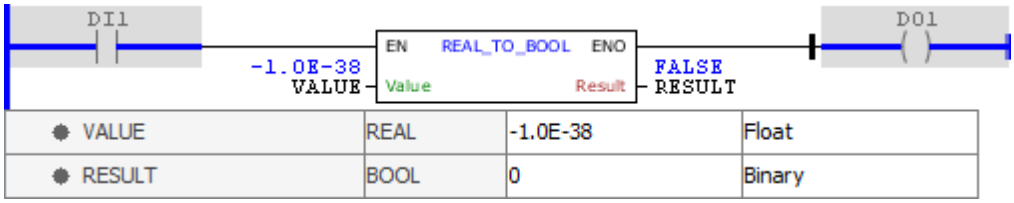
VALUE	REAL	0.008	Float
RESULT	BOOL	1	Binary



VALUE	REAL	-54.0	Float
RESULT	BOOL	1	Binary



VALUE	REAL	0.0	Float
RESULT	BOOL	0	Binary



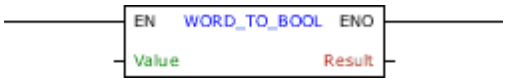
VALUE	REAL	-1.0E-38	Float
RESULT	BOOL	0	Binary

The examples above perform the conversion of VALUE variable, in REAL, into a BOOL value storing the final result in RESULT. The block ends with success and ENO output is activated. Notice in the last example that the values very close to the machine epsilon may result in an interpretation of the FALSE value.

11.2.6.6.1.4 WORD_TO_BOOL

Block that performs the conversion of a WORD value into a BOOL value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	WORD UINT INT	Value in WORD
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BOOL	Value in BOOL

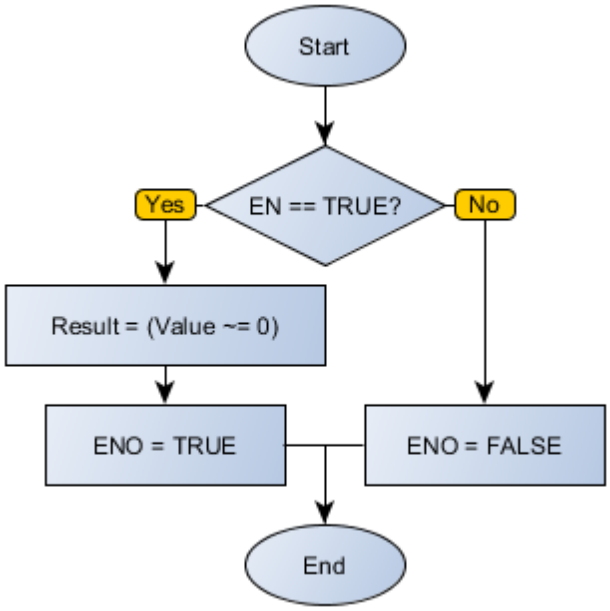
Operation

When this block has a TRUE value in EN, it interprets the Value value as WORD and converts it into BOOL, storing in Result.

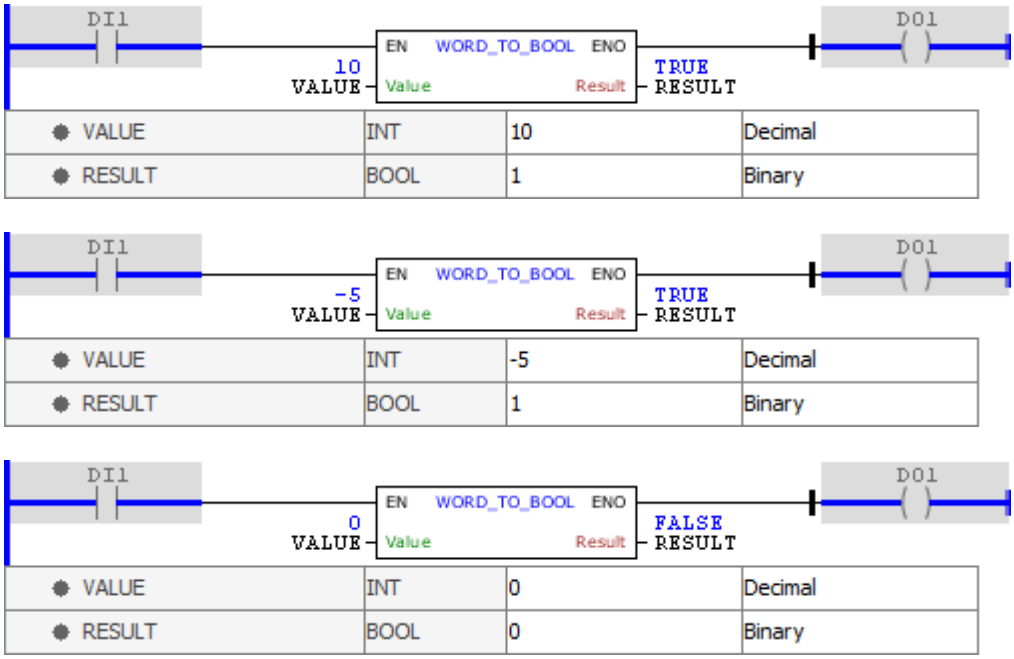
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



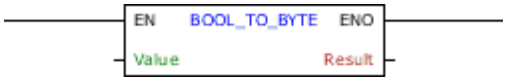
The examples above perform the conversion of VALUE variable, in WORD, into a BOOL value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.2.6.6.2 BYTE

11.2.6.6.2.1 BOOL_TO_BYTE

Block that performs the conversion of a BOOL value into a BYTE value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BOOL	Value in BOOL
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT	Value in BYTE

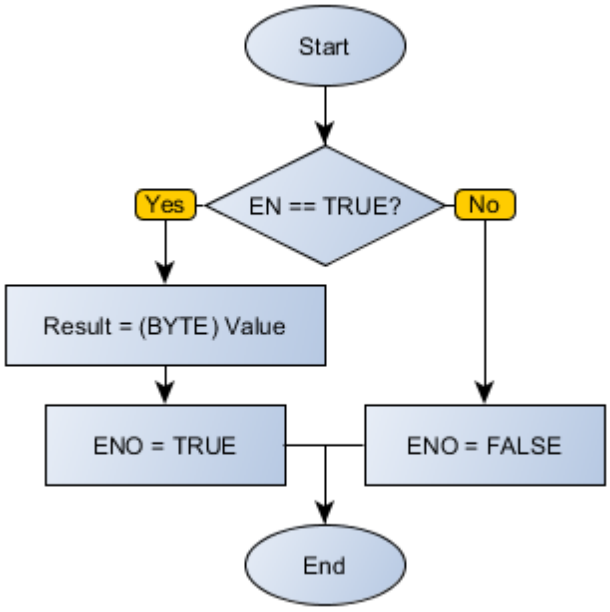
Operation

When this block has a TRUE value in EN, it interprets the Value value as BOOL and converts it into BYTE, storing in Result.

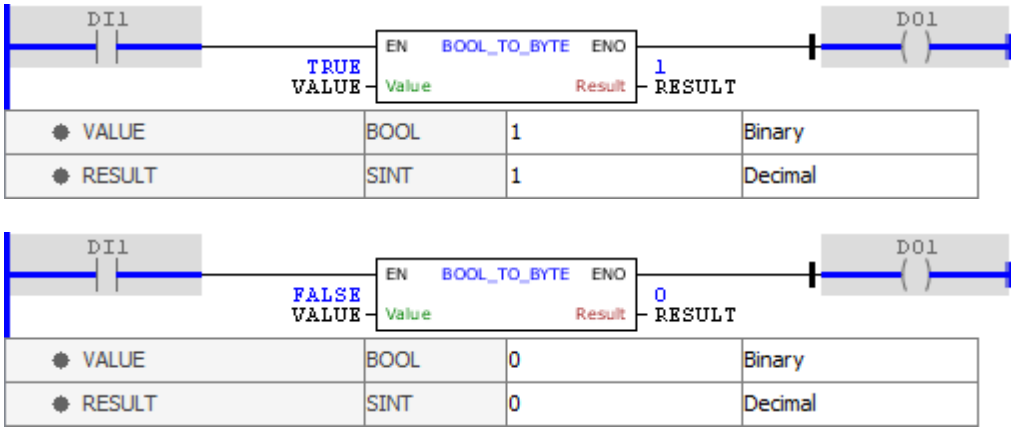
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example

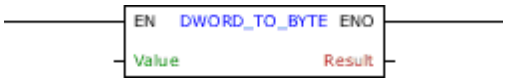


The examples above perform the conversion of variable VALUE, in BOOL, into a BYTE value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.2.6.6.2.2 DWORD_TO_BYTE

Block that performs the conversion of a DWORD value into a BYTE value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	DWORD UDINT DINT	Value in DWORD
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT	Value in BYTE

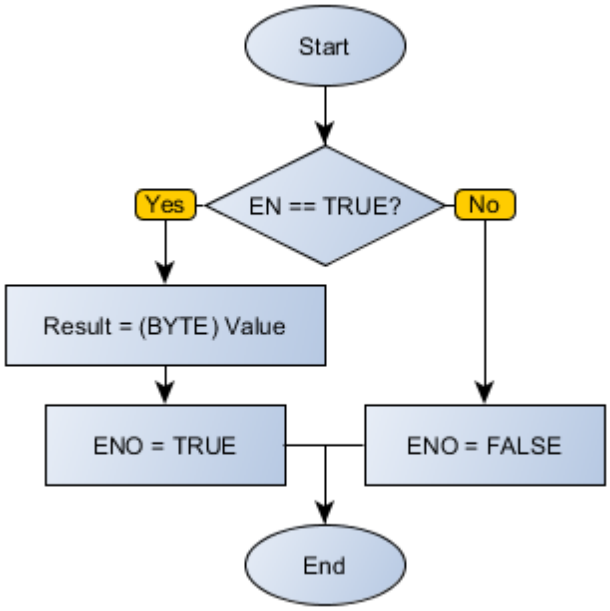
Operation

When this block has a TRUE value in EN, it interprets the Value value as DWORD and converts it into BYTE, storing in Result.

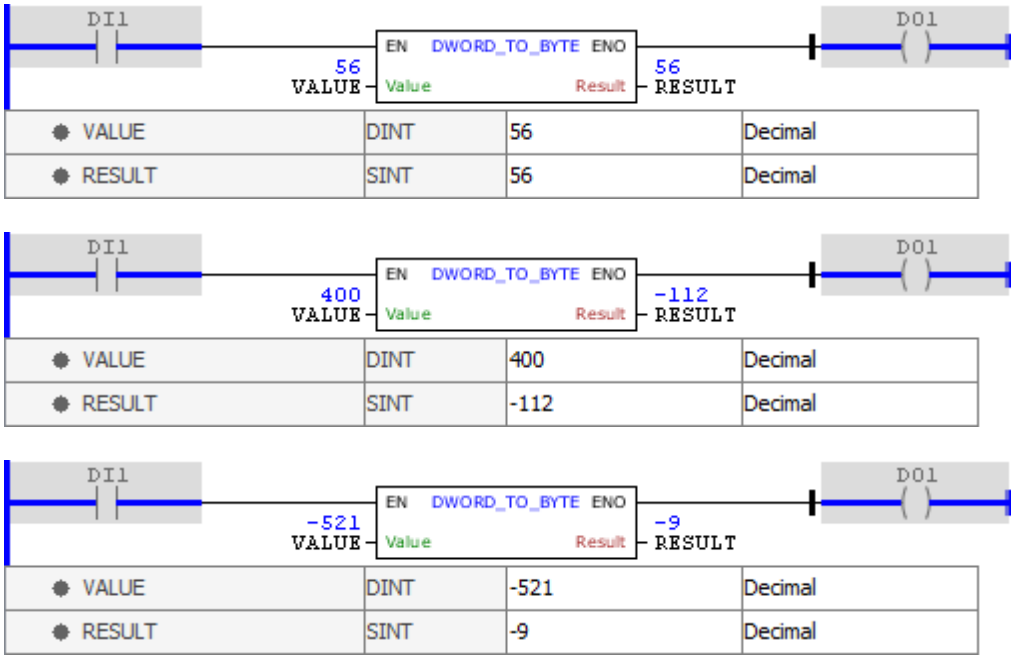
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example

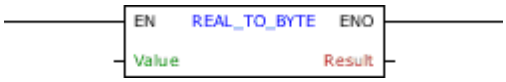


The examples above perform the conversion of variable VALUE, in DWORD, into a BYTE value storing the final result in RESULT. The block ends with success and ENO output is activated. Notice that only the eight least significant bits are taken into account.

11.2.6.6.2.3 REAL_TO_BYTE

Block that performs the conversion of a REAL value into a BYTE value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Value in REAL
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT	Value in BYTE

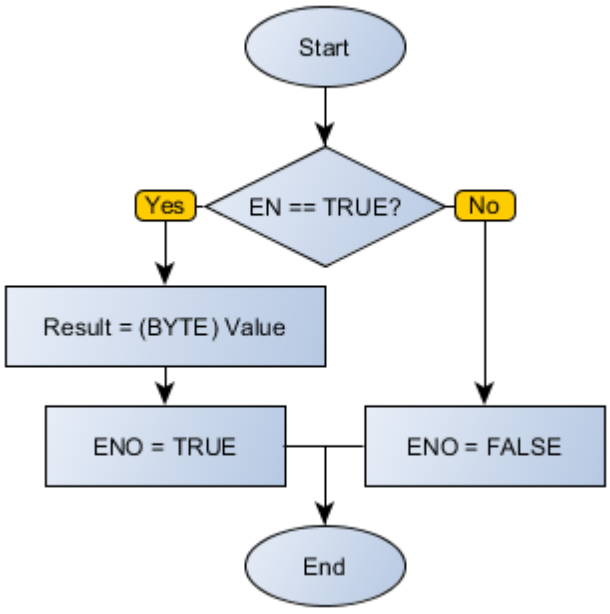
Operation

When this block has a TRUE value in EN, it interprets the Value value as REAL and converts it into BYTE, storing in Result.

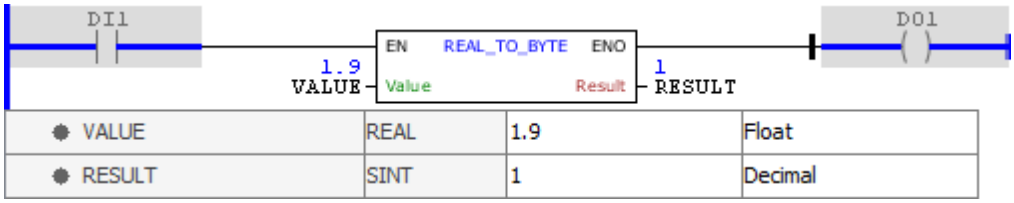
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

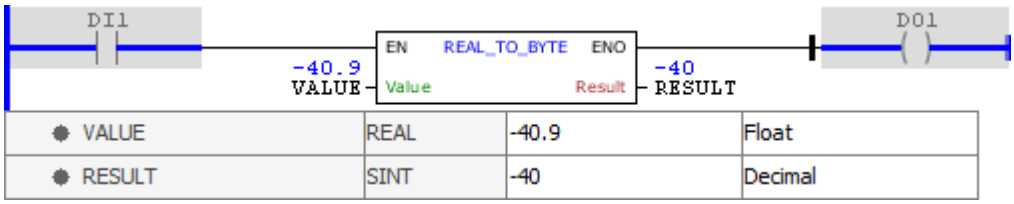
Block Flowchart



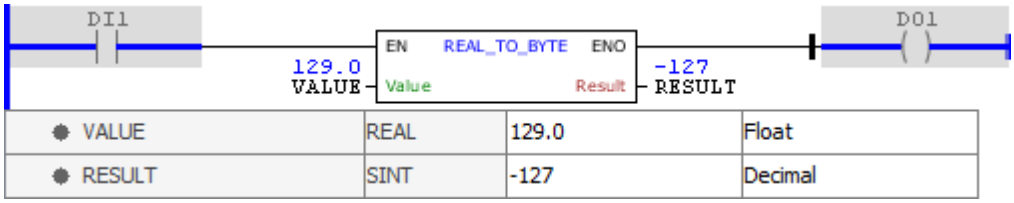
Example



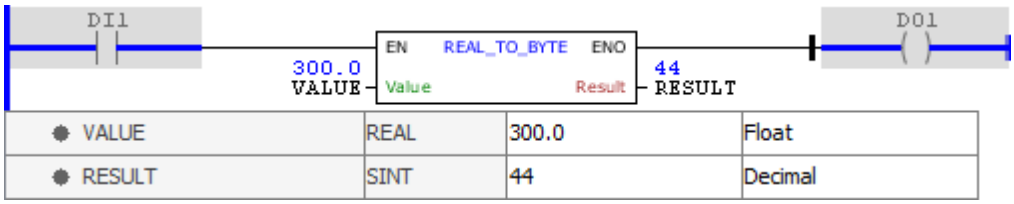
VALUE	REAL	1.9	Float
RESULT	SINT	1	Decimal



VALUE	REAL	-40.9	Float
RESULT	SINT	-40	Decimal



VALUE	REAL	129.0	Float
RESULT	SINT	-127	Decimal



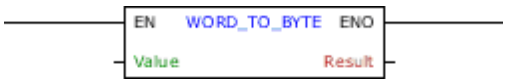
VALUE	REAL	300.0	Float
RESULT	SINT	44	Decimal

The examples above perform the conversion of variable VALUE, in REAL, into a BYTE value storing the final result in RESULT. The block ends with success and ENO output is activated. Notice that the results are truncated in decimal and only the eight least significant bits are taken into account.

11.2.6.6.2.4 WORD_TO_BYTE

Block that performs the conversion of a WORD value into a BYTE value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	WORD UINT INT	Value in WORD
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT	Value in BYTE

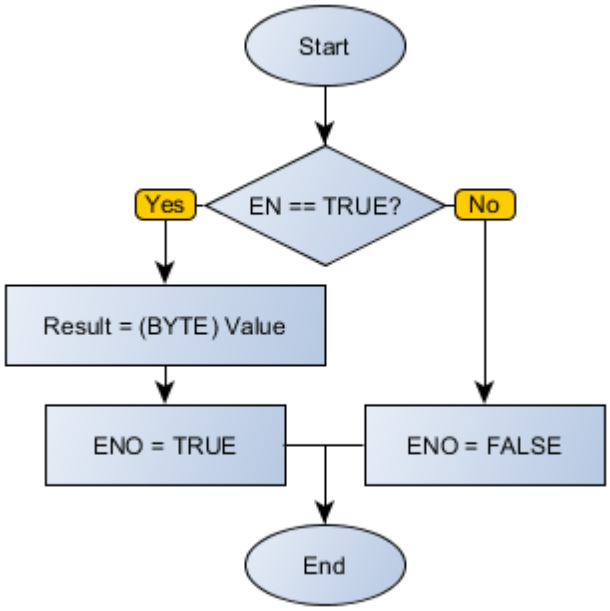
Operation

When this block has a TRUE value in EN, it interprets the Value value as WORD and converts it into BYTE, storing in Result.

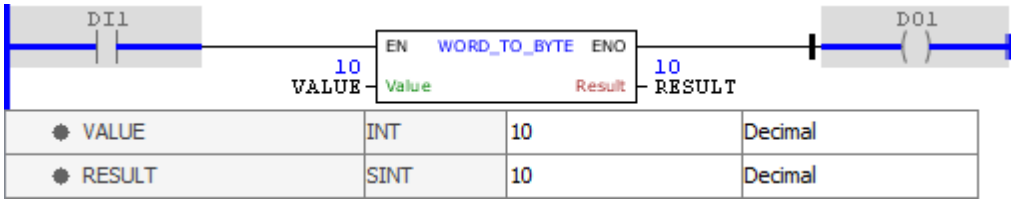
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

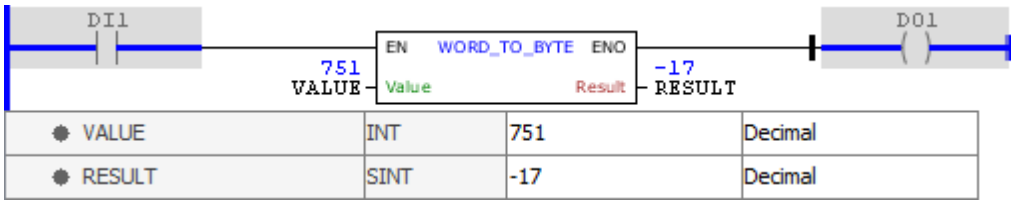
Block Flowchart



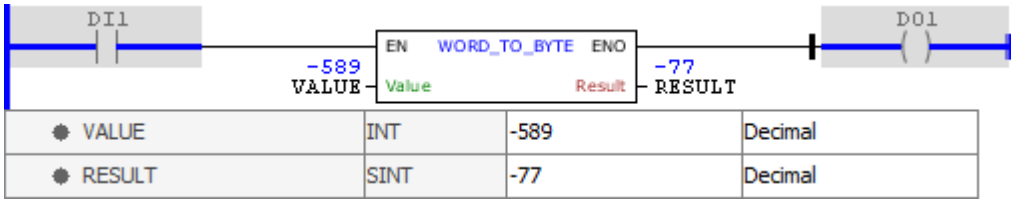
Example



VALUE	INT	10	Decimal
RESULT	SINT	10	Decimal



VALUE	INT	751	Decimal
RESULT	SINT	-17	Decimal



VALUE	INT	-589	Decimal
RESULT	SINT	-77	Decimal

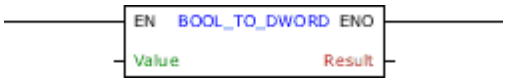
The examples above perform the conversion of variable VALUE, in WORD, into a BYTE value storing the final result in RESULT. The block ends with success and ENO output is activated. Notice that only the eight least significant bits are taken into account.

11.2.6.6.3 DWORD

11.2.6.6.3.1 BOOL_TO_DWORD

Block that performs the conversion of a BOOL value into a DWORD value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BOOL	Value in BOOL
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	DWORD UDINT DINT	Value in DWORD

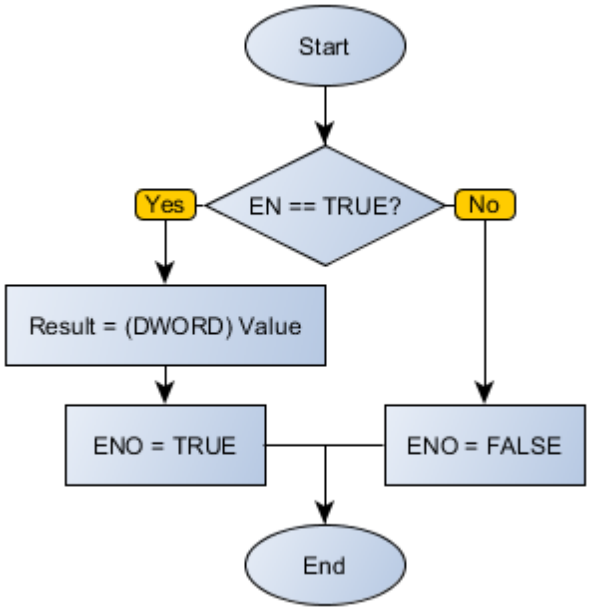
Operation

When this block has a TRUE value in EN, it interprets the Value value as BOOL and converts it into DWORD, storing in Result.

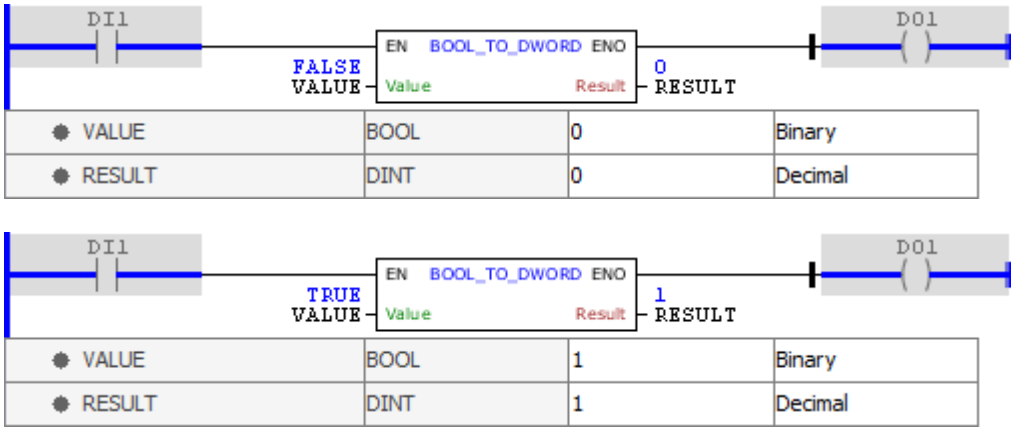
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example

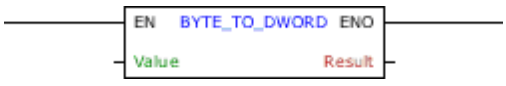


The examples above perform the conversion of VALUE variable, in BOOL, into a DWORD value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.2.6.6.3.2 BYTE_TO_DWORD

Block that performs the conversion of a BYTE value into a DWORD value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT	Value in BYTE
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	DWORD UDINT DINT	Value in DWORD

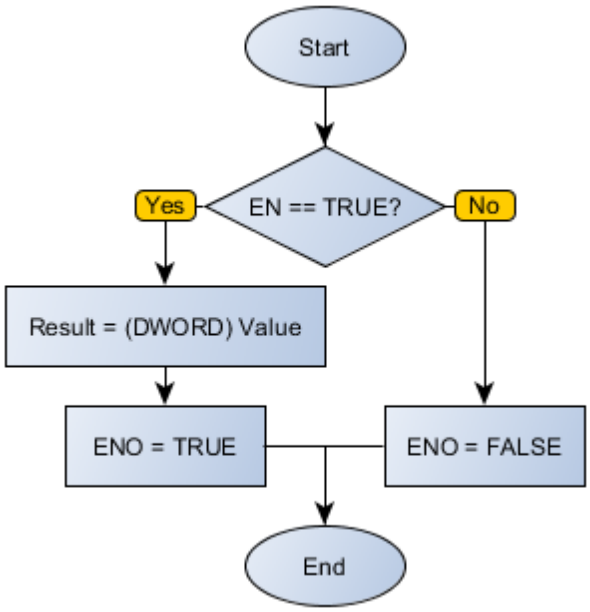
Operation

When this block has a TRUE value in EN, it interprets the Value value as BYTE and converts it into DWORD, storing in Result.

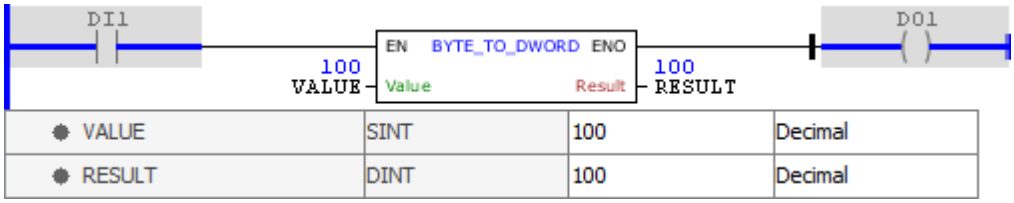
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

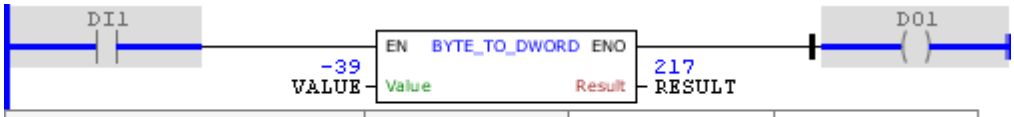
Block Flowchart



Example



VALUE	SINT	100	Decimal
RESULT	DINT	100	Decimal



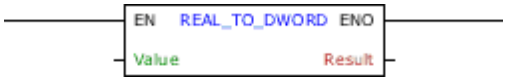
VALUE	SINT	-39	Decimal
RESULT	DINT	217	Decimal

The examples above perform the conversion of variable VALUE, in BYTE, into a DWORD value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.2.6.6.3.3 REAL_TO_DWORD

Block that performs the conversion of a REAL value into a DWORD value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Value in REAL
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	DWORD UDINT DINT	Value in DWORD

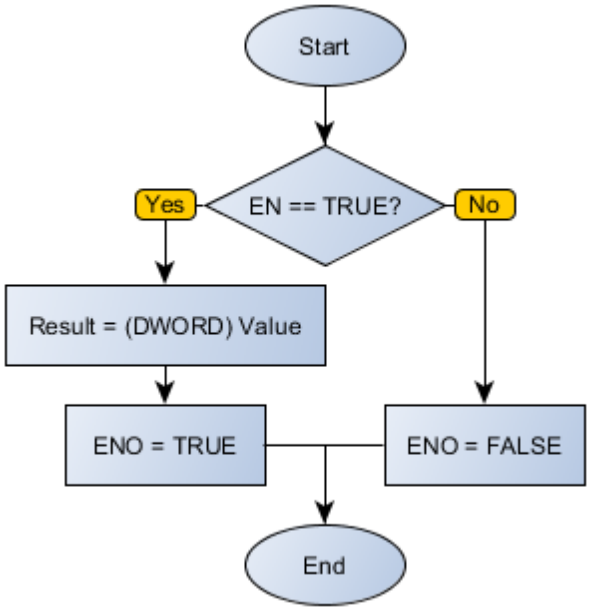
Operation

When this block has a TRUE value in EN, it interprets the Value value as REAL and converts it into DWORD, storing in Result.

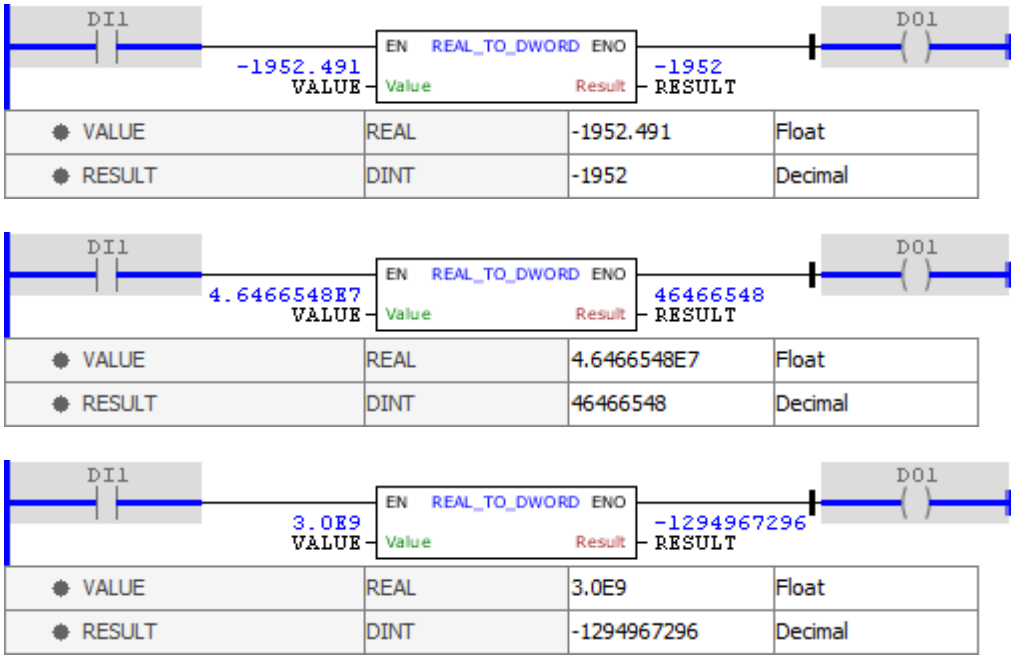
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example

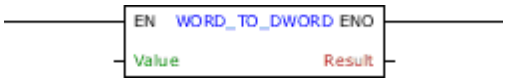


The examples above perform the conversion of variable VALUE, in REAL, into a DWORD value storing the final result in RESULT. The block ends with success and ENO output is activated. Note that the results are truncated in decimal and only the thirty-two least significant bits are taken into account.

11.2.6.6.3.4 WORD_TO_DWORD

Block that performs the conversion of a WORD value into a DWORD value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	WORD UINT INT	Value in WORD
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	DWORD UDINT DINT	Value in DWORD

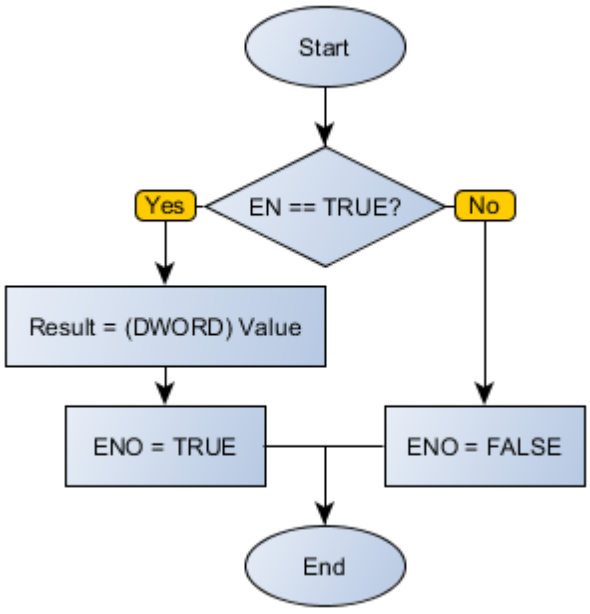
Operation

When this block has a TRUE value in EN, it interprets the Value value as WORD and converts it into DWORD, storing in Result.

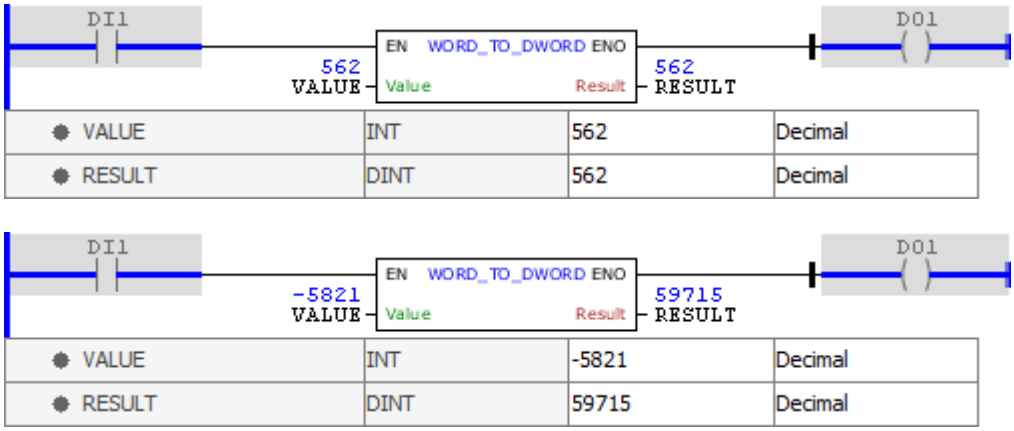
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



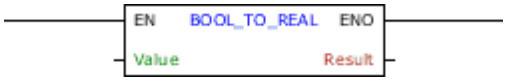
The examples above convert the VALUE variable, in WORD, into a DWORD value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.2.6.6.4 REAL

11.2.6.6.4.1 BOOL_TO_REAL

Block that performs the conversion of a BOOL value into a REAL value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BOOL	Value in BOOL
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	REAL	Value in REAL

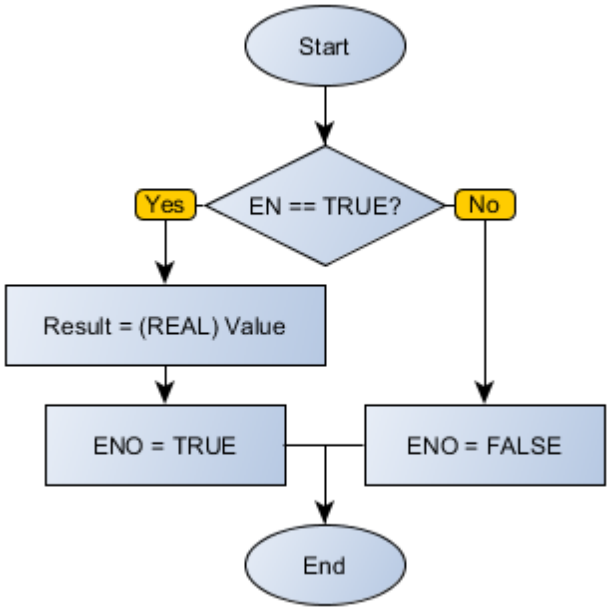
Operation

When this block has a TRUE value in EN, it interprets the Value value as BOOL and converts it into REAL, storing in Result.

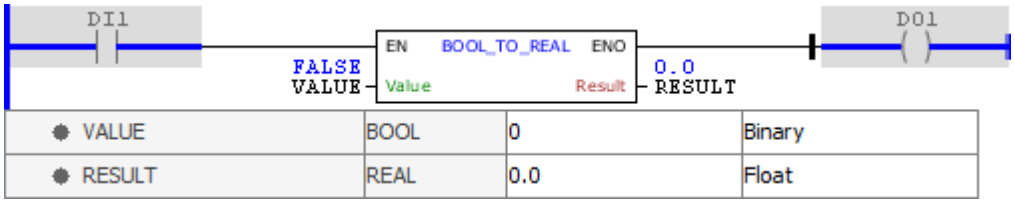
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

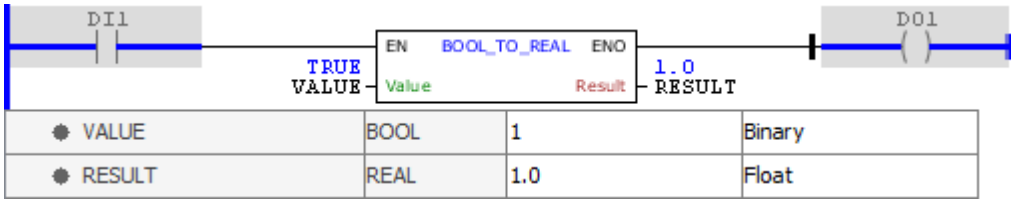
Block Flowchart



Example



VALUE	BOOL	0	Binary
RESULT	REAL	0.0	Float



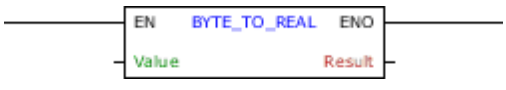
VALUE	BOOL	1	Binary
RESULT	REAL	1.0	Float

The examples above perform the conversion of variable VALUE, in BOOL, into a REAL value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.2.6.6.4.2 BYTE_TO_REAL

Block that performs the conversion of a BYTE value into a REAL value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT	Value in BYTE
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	REAL	Value in REAL

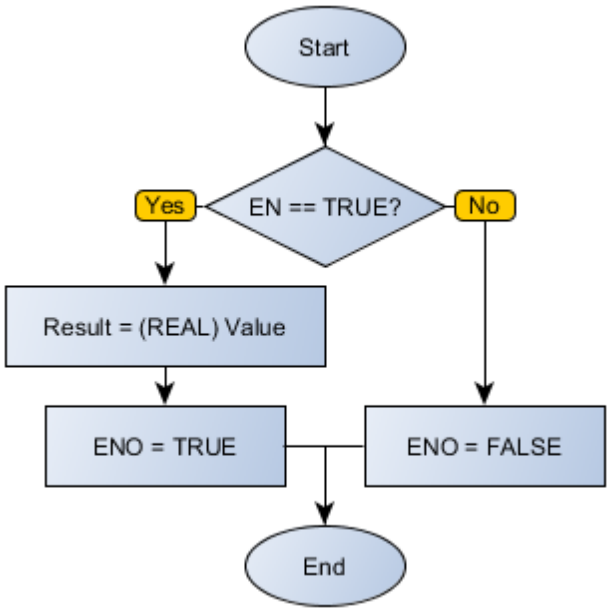
Operation

When this block has a TRUE value in EN, it interprets the Value value as BYTE and converts it into REAL, storing in Result.

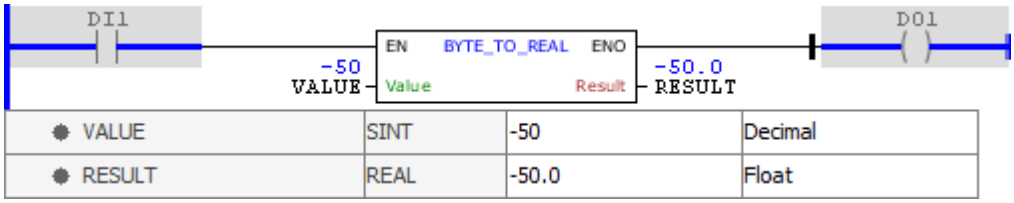
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

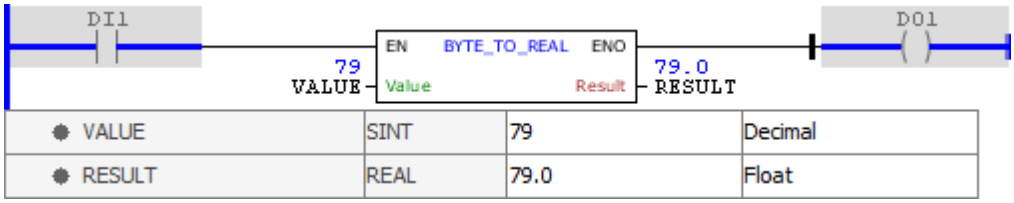
Block Flowchart



Example



VALUE	SINT	-50	Decimal
RESULT	REAL	-50.0	Float



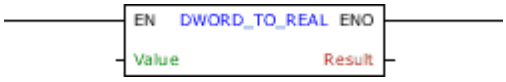
VALUE	SINT	79	Decimal
RESULT	REAL	79.0	Float

The examples above perform the conversion of variable VALUE, in BYTE, into a REAL value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.2.6.6.4.3 DWORD_TO_REAL

Block that performs the conversion of a DWORD value into a REAL value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	DWORD UDINT DINT	Value in DWORD
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	REAL	Value in REAL

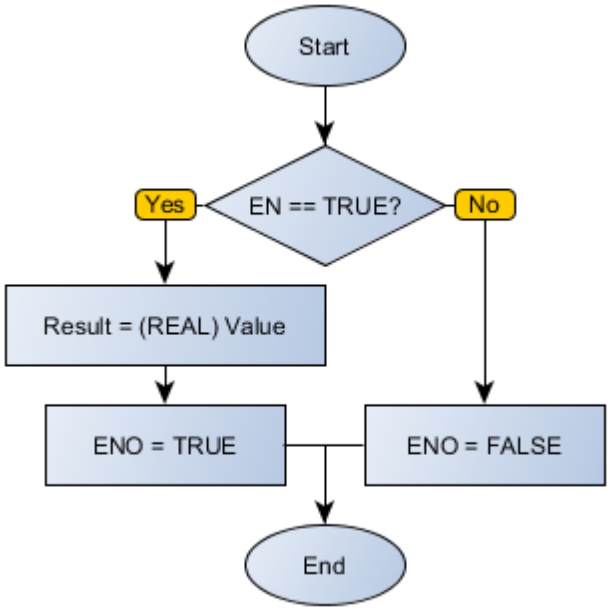
Operation

When this block has a TRUE value in EN, it interprets the Value value as DWORD and converts it into REAL, storing in Result.

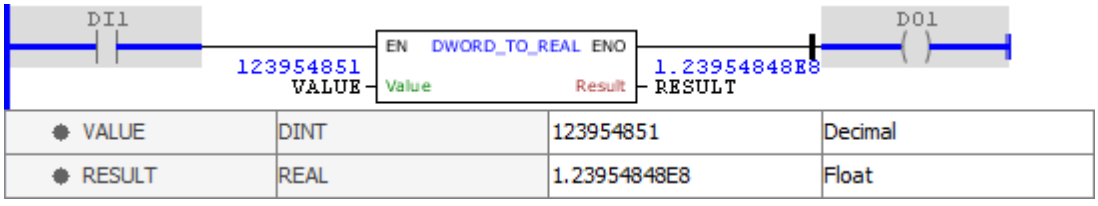
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

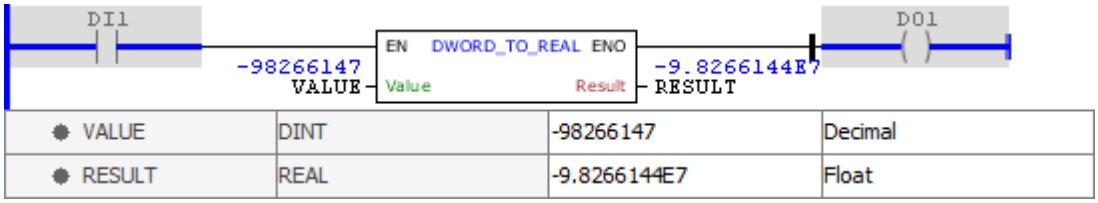
Block Flowchart



Example



VALUE	DINT	123954851	Decimal
RESULT	REAL	1.23954848E8	Float



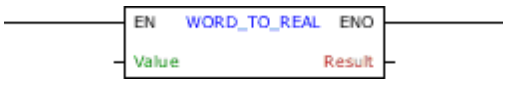
VALUE	DINT	-98266147	Decimal
RESULT	REAL	-9.8266144E7	Float

The examples above perform the conversion of variable VALUE, in DWORD, into a REAL value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.2.6.6.4.4 WORD_TO_REAL

Block that performs the conversion of a WORD value into a REAL value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	WORD UINT INT	Value in WORD
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	REAL	Value in REAL

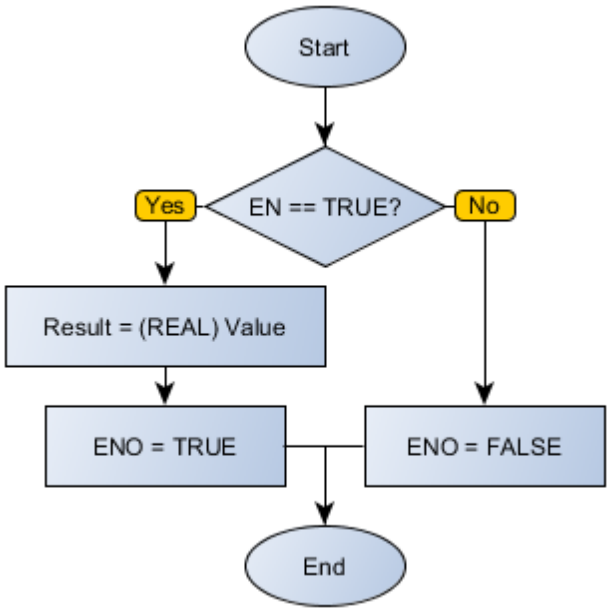
Operation

When this block has a TRUE value in EN, it interprets the Value value as WORD and converts it into REAL, storing in Result.

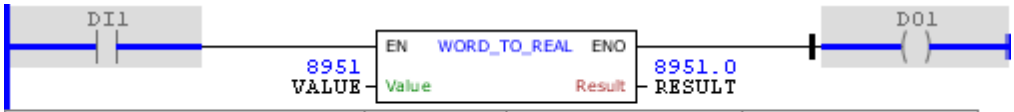
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

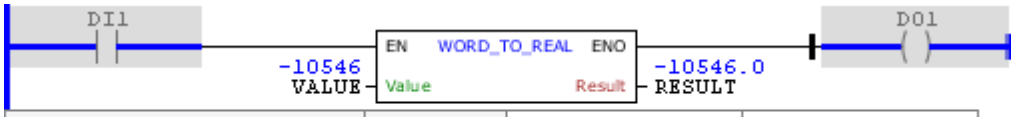
Block Flowchart



Example



VALUE	INT	8951	Decimal
RESULT	REAL	8951.0	Float



VALUE	INT	-10546	Decimal
RESULT	REAL	-10546.0	Float

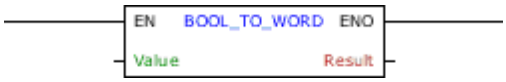
The examples above perform the conversion of variable VALUE, in WORD, into a REAL value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.2.6.6.5 WORD

11.2.6.6.5.1 BOOL_TO_WORD

Block that performs the conversion of a BOOL value into a WORD value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BOOL	Value in BOOL
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	WORD UINT INT	Value in WORD

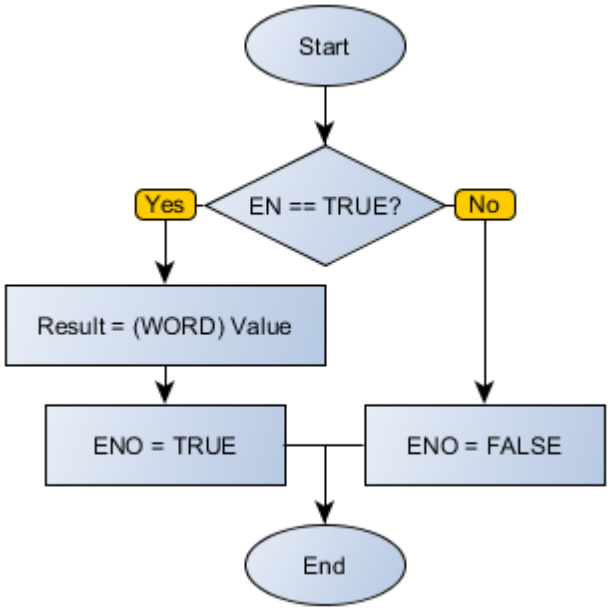
Operation

When this block has a TRUE value in EN, it interprets the Value value as BOOL and converts it into WORD, storing in Result.

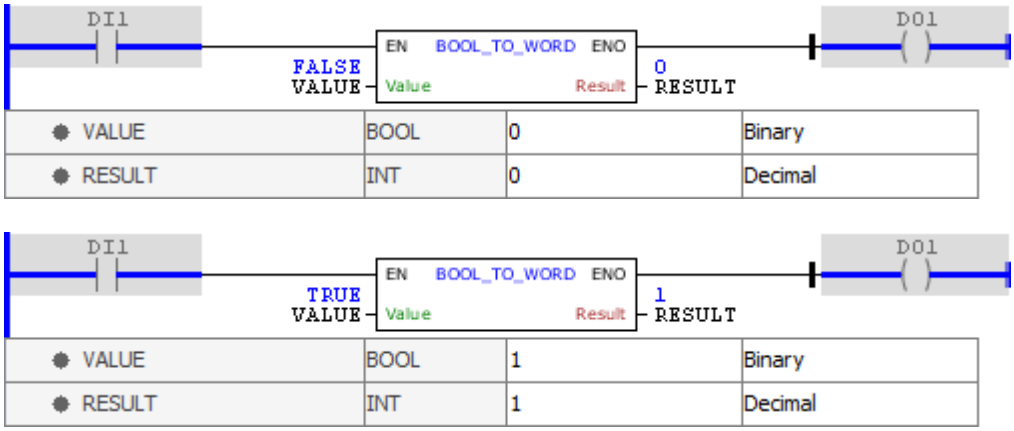
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example

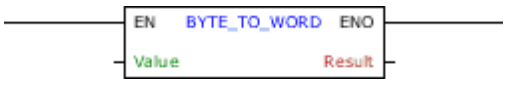


The examples above perform the conversion of VALUE variable, in BOOL, into a WORD value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.2.6.6.5.2 BYTE_TO_WORD

Block that performs the conversion of a BYTE value into a WORD value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT	Value in BYTE
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	WORD UINT INT	Value in WORD

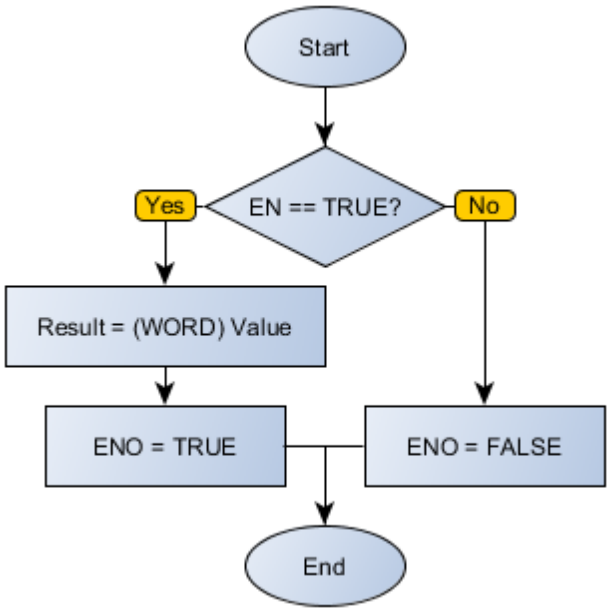
Operation

When this block has a TRUE value in EN, it interprets the Value value as BYTE and converts it into WORD, storing in Result.

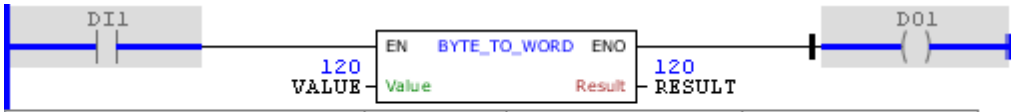
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

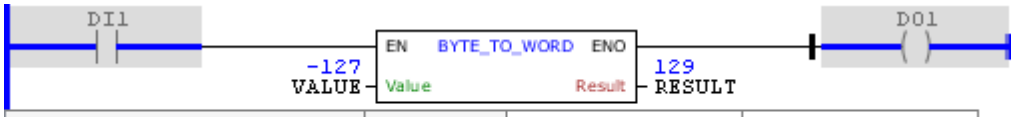
Block Flowchart



Example



VALUE	SINT	120	Decimal
RESULT	INT	120	Decimal



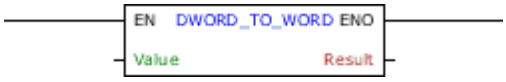
VALUE	SINT	-127	Decimal
RESULT	INT	129	Decimal

The examples above perform the conversion of variable VALUE, in BYTE, into a WORD value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.2.6.6.5.3 DWORD_TO_WORD

Block that performs the conversion of a DWORD value into a WORD value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	DWORD UDINT DINT	Value in DWORD
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	WORD UINT INT	Value in WORD

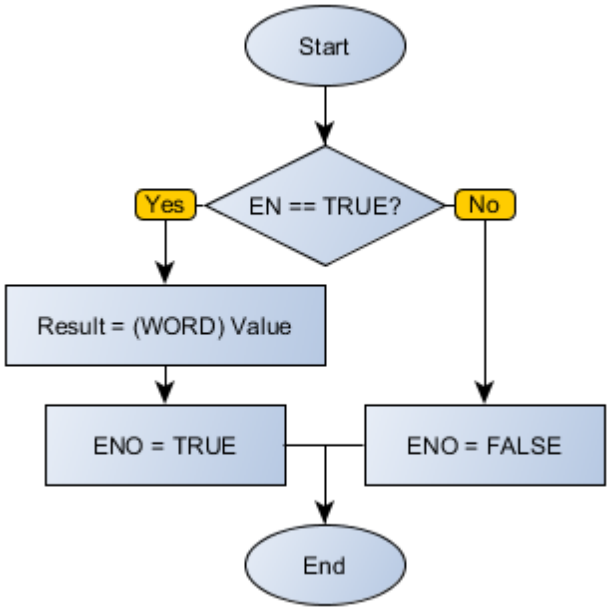
Operation

When this block has a TRUE value in EN, it interprets the Value value as DWORD and converts it into WORD, storing in Result.

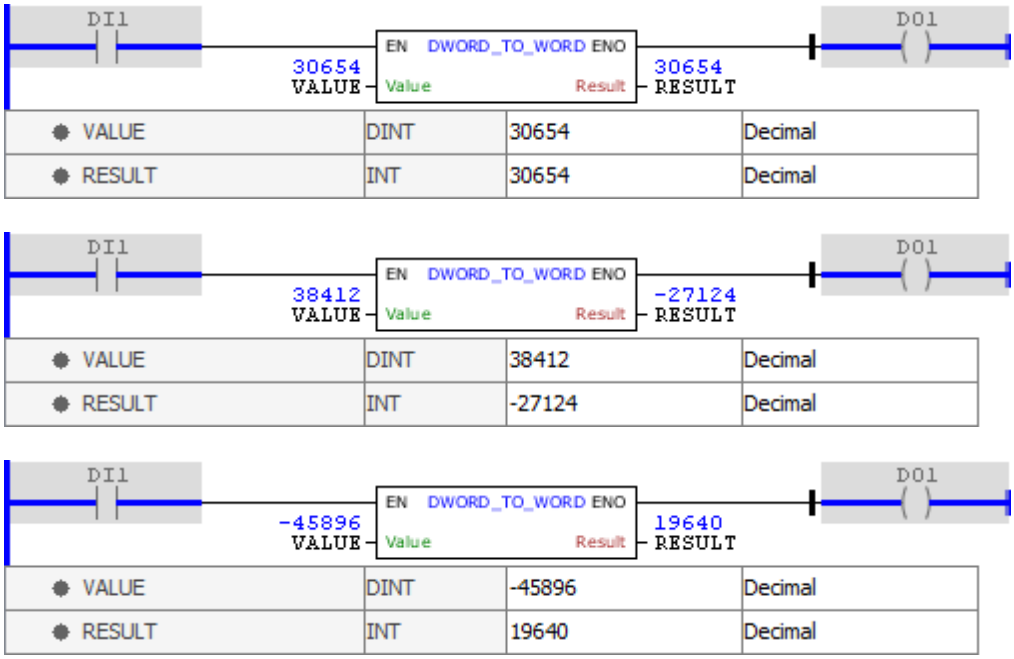
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example

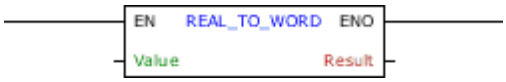


The examples above convert the VALUE variable, in DWORD, into a WORD value storing the final result in RESULT. The block ends with success and ENO output is activated. Notice that only the sixteen least significant bits are taken into account.

11.2.6.6.5.4 REAL_TO_WORD

Block that performs the conversion of a REAL value into a WORD value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Value in REAL
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	WORD UINT INT	Value in WORD

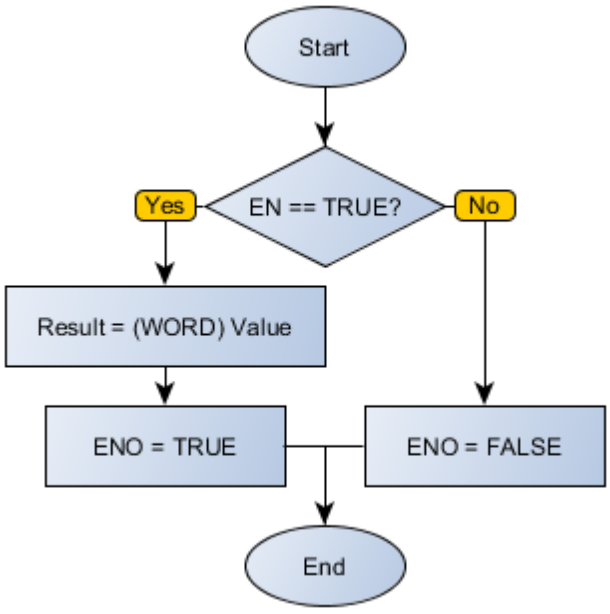
Operation

When this block has a TRUE value in EN, it interprets the Value value as REAL and converts it into WORD, storing in Result.

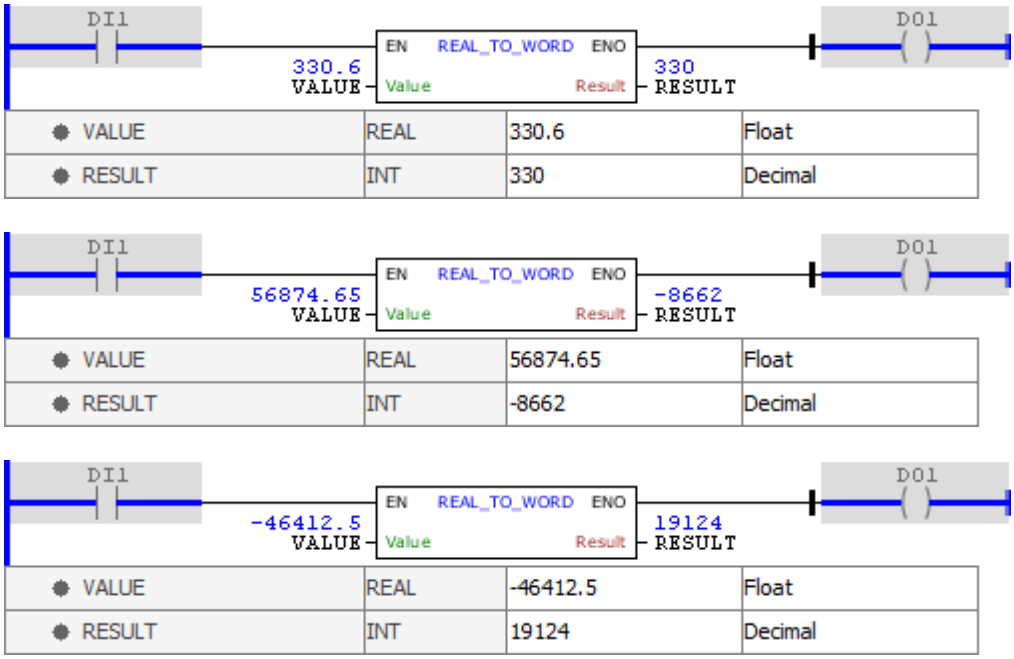
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



VALUE	REAL	330.6	Float
RESULT	INT	330	Decimal

VALUE	REAL	56874.65	Float
RESULT	INT	-8662	Decimal

VALUE	REAL	-46412.5	Float
RESULT	INT	19124	Decimal

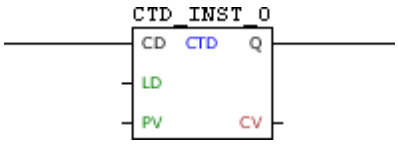
The examples above convert the VALUE variable, in DWORD, into a WORD value storing the final result in RESULT. The block ends with success and ENO output is activated. Note that the results are truncated in decimal and only the sixteen least significant bits are taken into account.

11.2.6.7 Counter

11.2.6.7.1 CTD

Countdown block of input pulses.

Ladder Representation



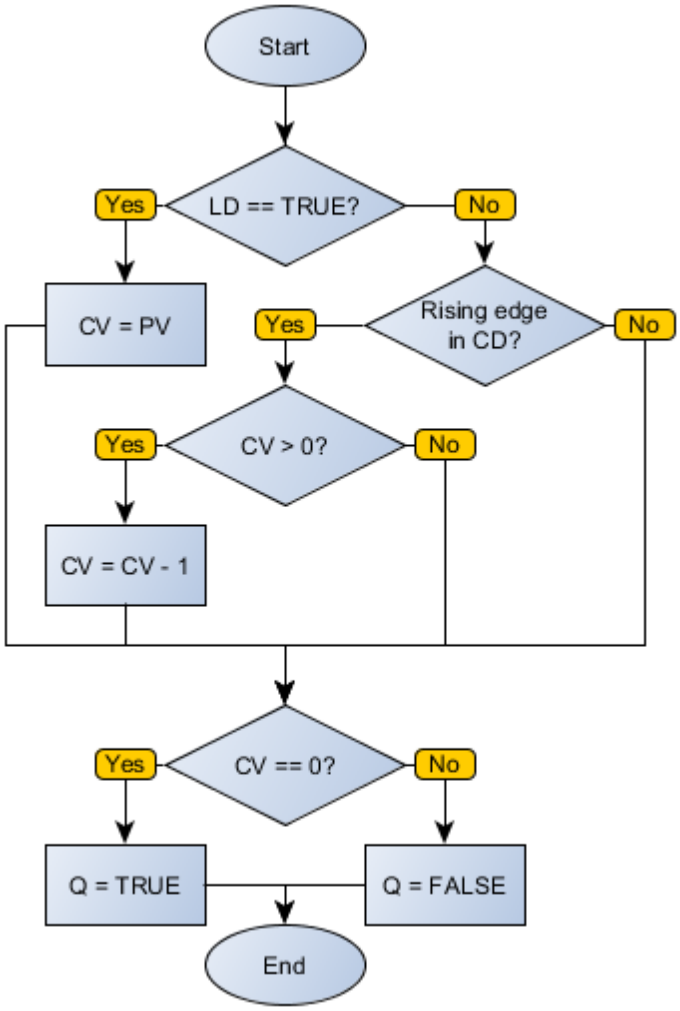
Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	CD	BOOL	Pulse identifier
	LD	BOOL	Loads the value of PV in CV
	PV	WORD UINT	Value of initial configuration
VAR_OUTPUT	Q	BOOL	Counter zeroed flag
	CV	WORD UINT	Current count value
VAR	CTD_INST_0	CTD	Instance of access to block structure

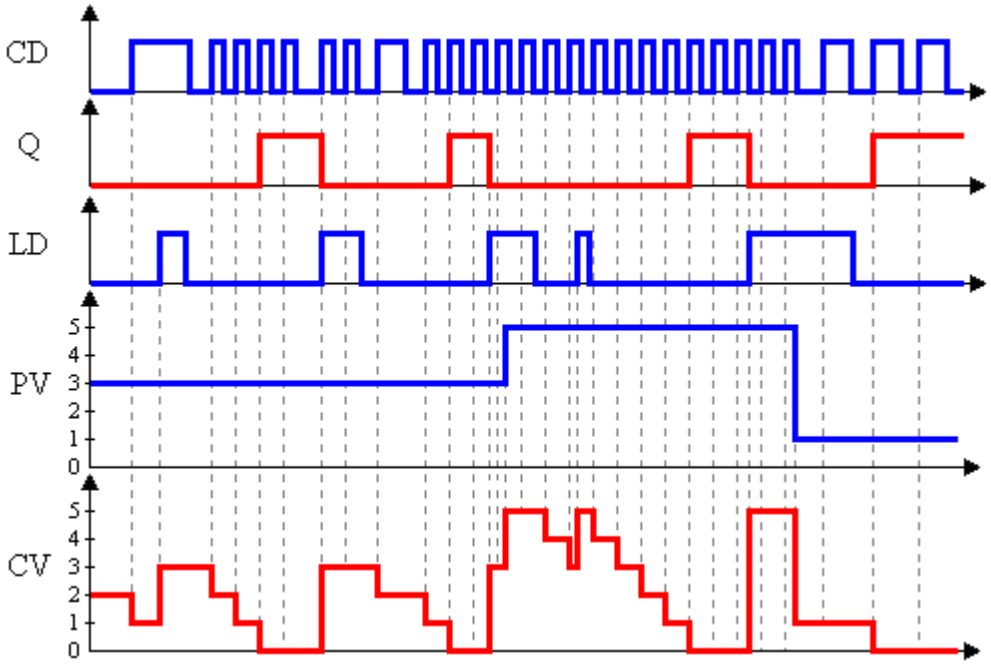
Operation

When this block identifies a leading edge in CD, it decrements the CV variable until it is zero. While CV equals zero, the output Q remains at TRUE level. By detecting high-level LD, the block loads the PV value in CV.

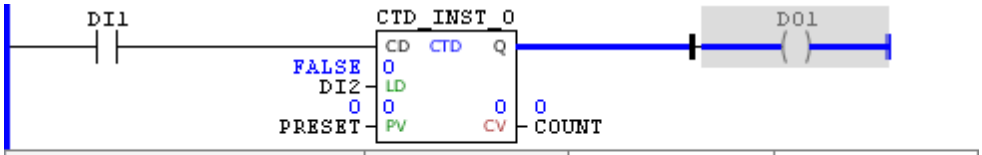
Block Flowchart



Operation Diagram

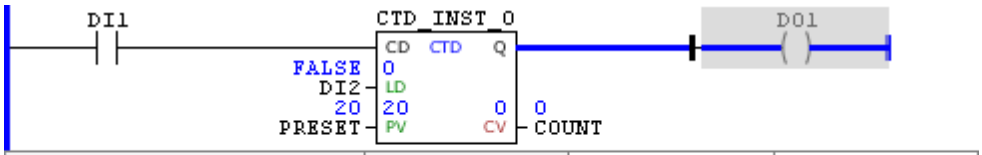


Example



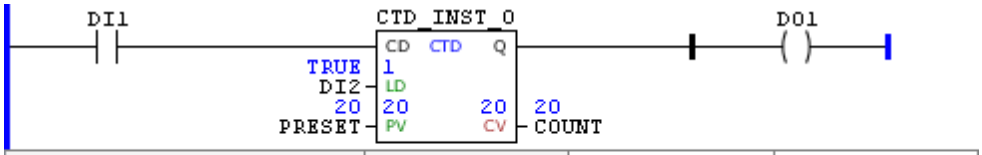
PRESET	UINT	0	Decimal
COUNT	UINT	0	Decimal

The above example shows the initial conditions of routine. As CV has a value of zero, the Q output is enabled.



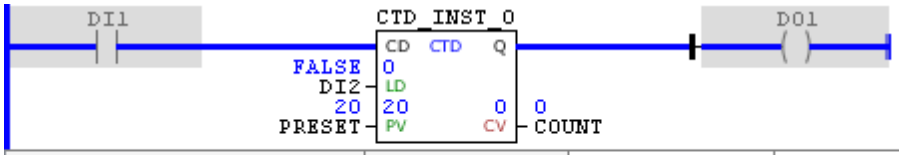
PRESET	UINT	20	Decimal
COUNT	UINT	0	Decimal

The value of the PV variable was changed to 20, but not yet loaded.



PRESET	UINT	20	Decimal
COUNT	UINT	20	Decimal

By identifying TRUE level in LD, the block loads the PV value to CV. Since this value is greater than zero, the Q output is disabled.



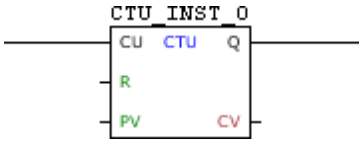
PRESET	UINT	20	Decimal
COUNT	UINT	0	Decimal

At each leading edge identified in CD, the value of COUNT is decremented until it reaches zero, when the Q output is enabled.

11.2.6.7.2 CTU

Block for gradual count of input pulses.

Ladder Representation



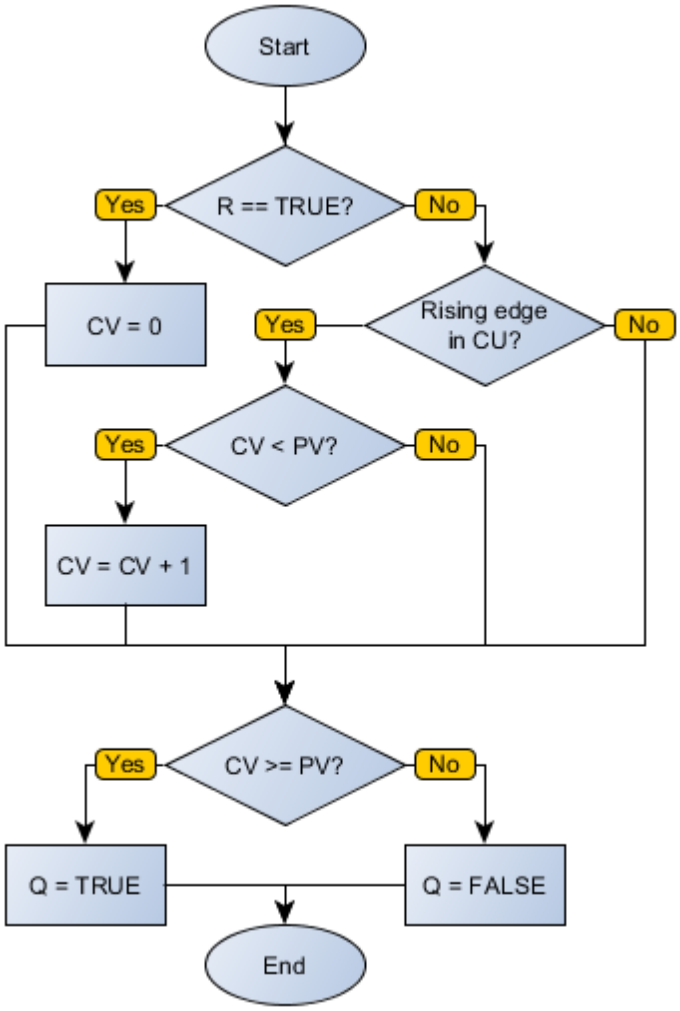
Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	CU	BOOL	Pulse identifier
	R	BOOL	Loads the zero value in CV
	PV	WORD UINT	Maximum count value
VAR_OUTPUT	Q	BOOL	Counter overrun flag
	CV	WORD UINT	Current count value
VAR	CTU_INST_0	CTU	Instance of access to block structure

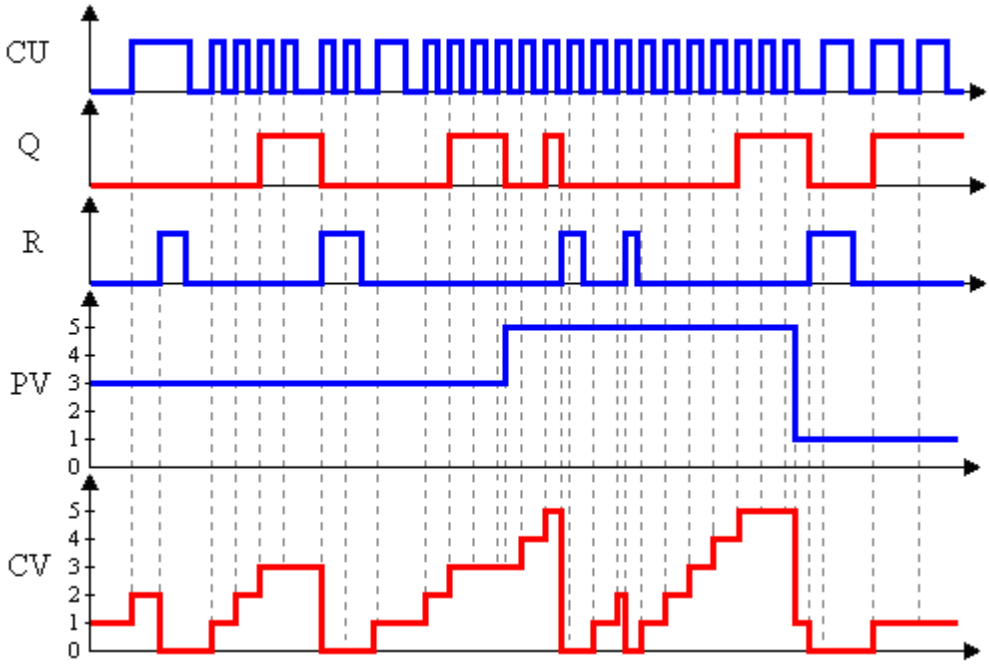
Operation

When this block identifies a leading edge in CD, it increments the CV variable until it is equal to PV. While CV equals PV, the output Q remains at TRUE level. By detecting high-level R, the block loads the zero value in CV.

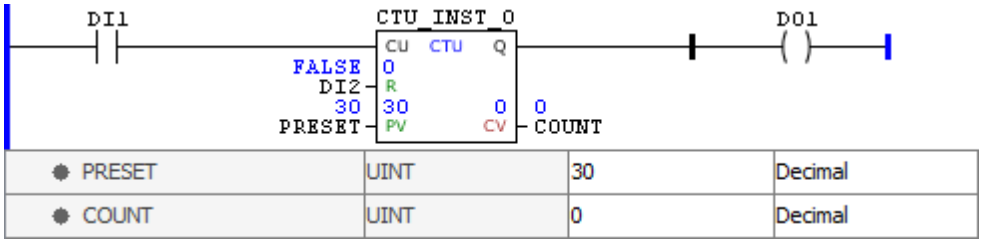
Block Flowchart



Operation Diagram

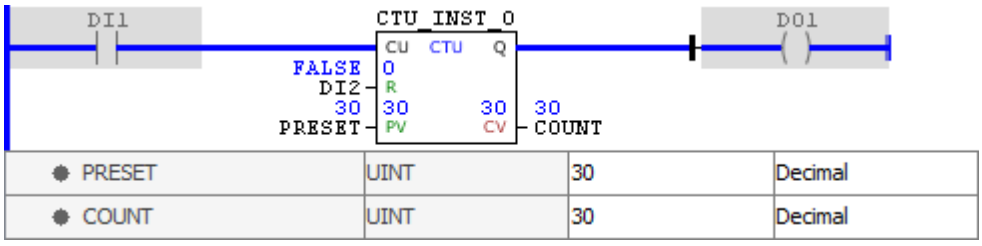


Example



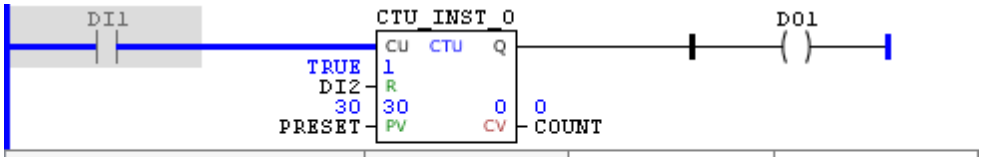
PRESET	UINT	30	Decimal
COUNT	UINT	0	Decimal

The above example shows the initial conditions of routine. Since CV has a lower value than of PV, the Q output is disabled.



PRESET	UINT	30	Decimal
COUNT	UINT	30	Decimal

At each leading edge identified in CU, the value of CV is incremented until it reaches the PV value, when the Q output is enabled.



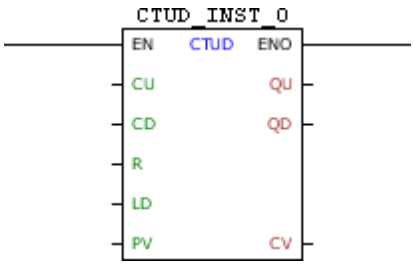
PRESET	UINT	30	Decimal
COUNT	UINT	0	Decimal

By identifying TRUE level in R, the block loads the zero value to CV. Since this value is lower than of PV, the Q output is disabled.

11.2.6.7.3 CTUD

Block for gradual count and countdown of input pulses.

Ladder Representation



Block Structure

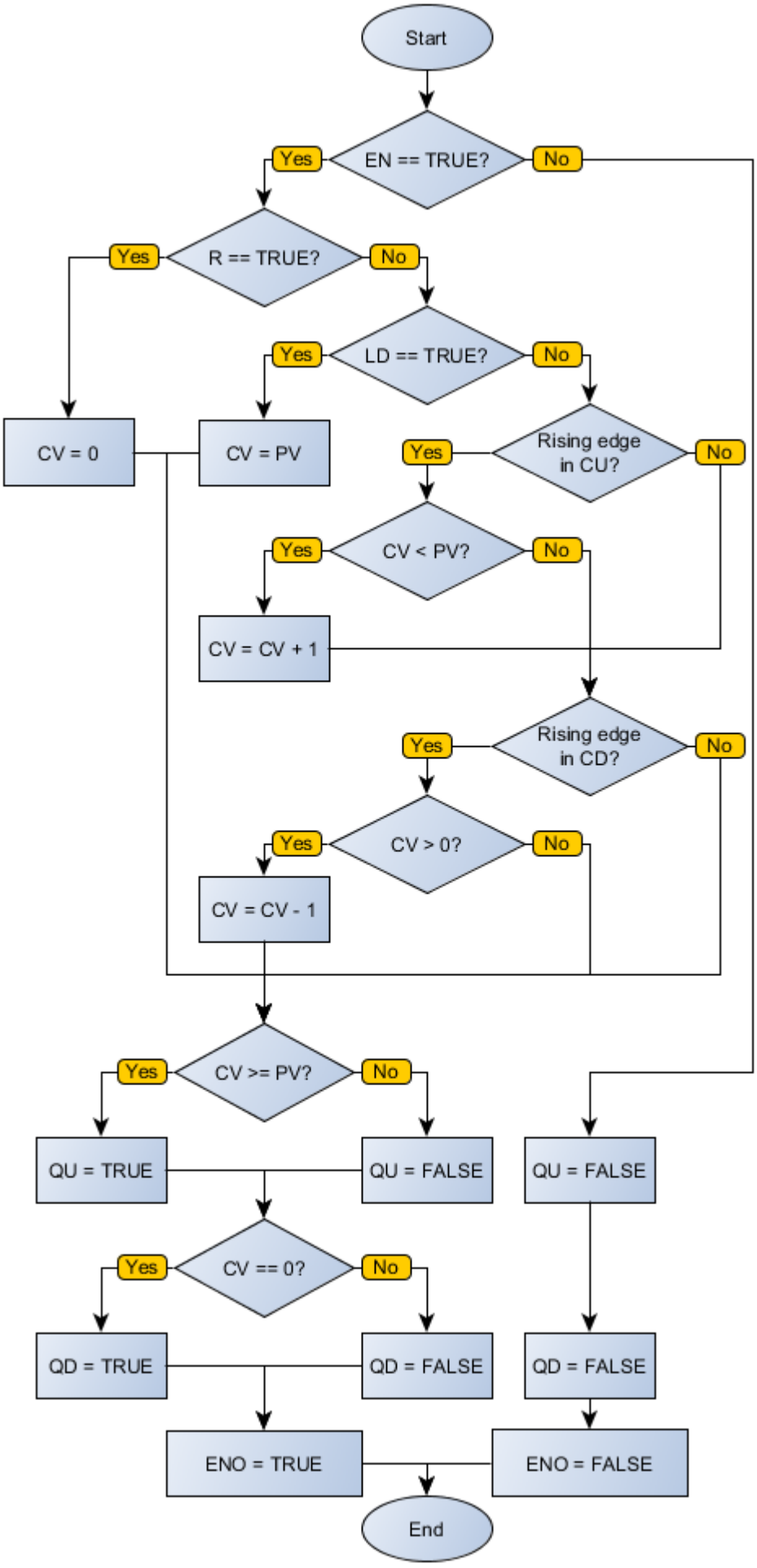
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	CU	BOOL	Pulse identifier for incremental
	CD	BOOL	Pulse identifier for decremental
	R	BOOL	Loads the zero value in CV
	LD	BOOL	Loads the value of PV in CV
	PV	WORD UINT	Reference value
VAR_OUTPUT	ENO	BOOL	Output enabling
	QU	BOOL	Counter overrun flag
	QD	BOOL	Counter zeroed flag
	CV	WORD UINT	Current count value
VAR	CTUD_INST_0	CTUD	Instance of access to block structure

Operation

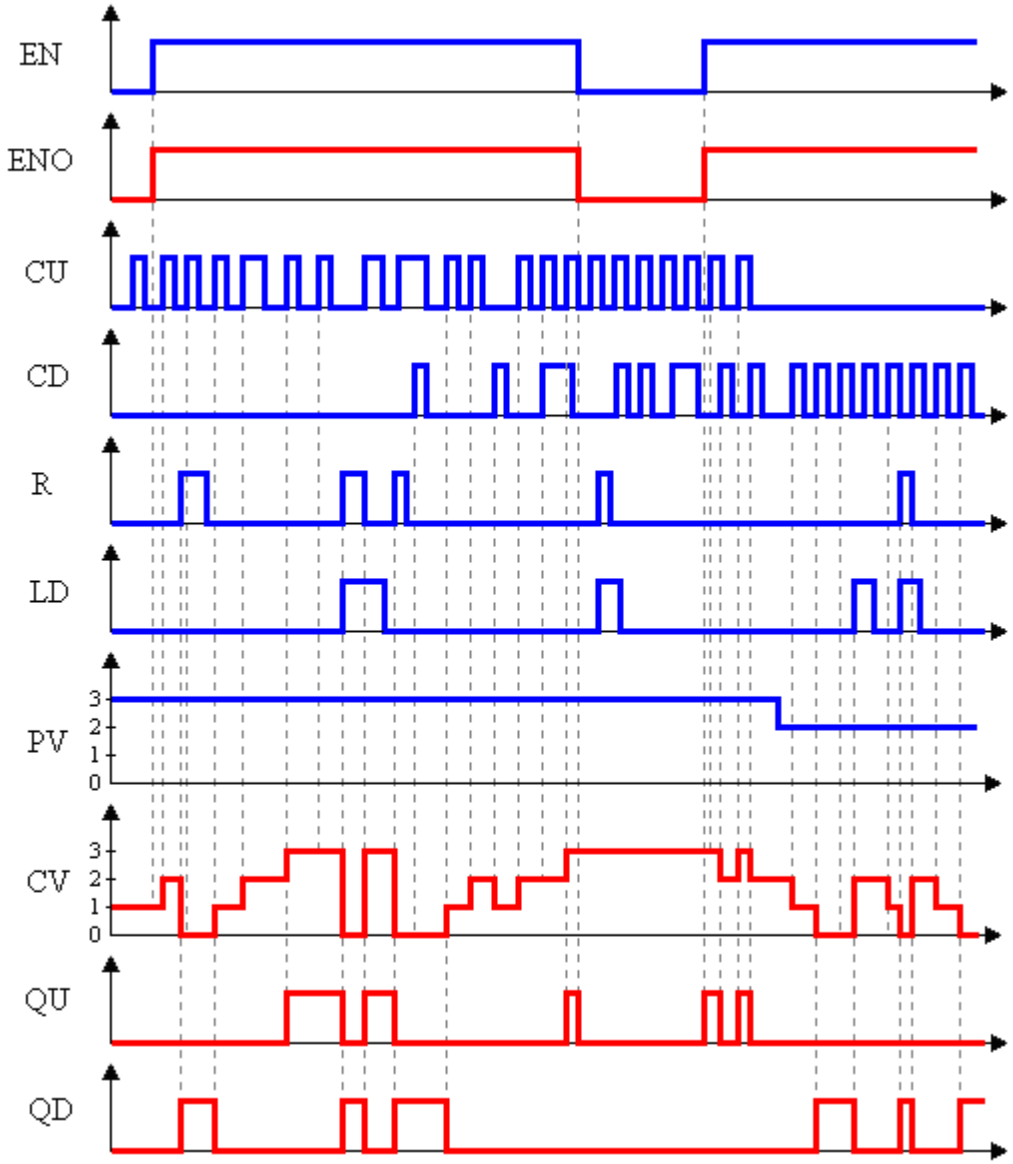
When this block has a TRUE value in EN, it acts as a CTD block and block CTU at the same time acting on the same CV counter. That is: increments CV until it reaches PV to the leading edges in CU and decrements CV until it reaches zero to the leading edges in CD. A positive transition in R carries zero in CV, while a leading edge in LD loads the PV value in CV. If CV has zero value, QD receives TRUE, and if CV has value equal to PV, QU receives TRUE.

The ENO value forwards to the next Ladder block the EN value.

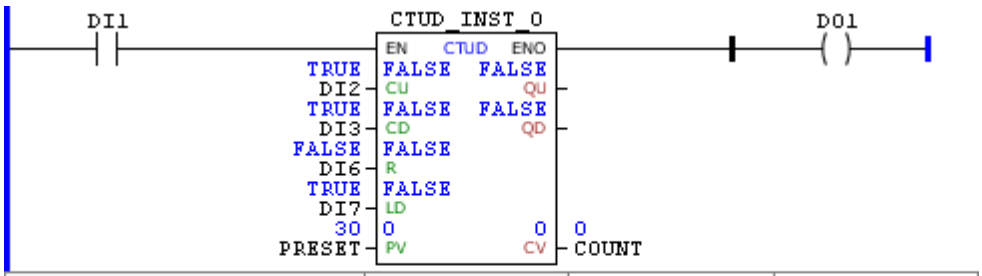
Block Flowchart


Start
EN == TRUE?
Yes
No
R == TRUE?
Yes
No
LD == TRUE?
Yes
No
CV = 0
CV = PV
Rising edge in CU?
Yes
No
CV < PV?
Yes
No
CV = CV + 1
Rising edge in CD?
Yes
No
CV > 0?
Yes
No
CV = CV - 1
CV >= PV?
Yes
No
QU = TRUE
QU = FALSE
QU = FALSE
CV == 0?
Yes
No
QD = TRUE
QD = FALSE
QD = FALSE
ENO = TRUE
ENO = FALSE
End

Operation Diagram

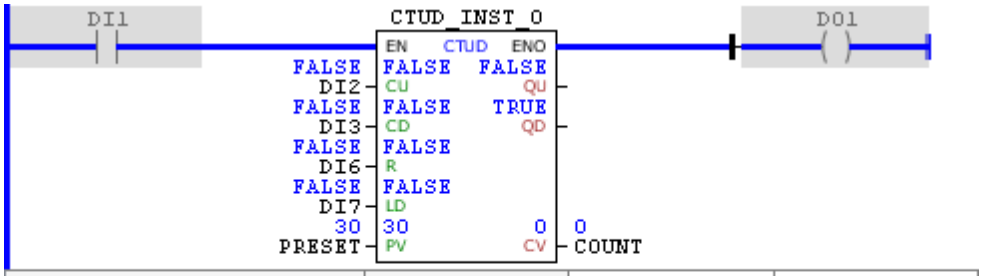


Example



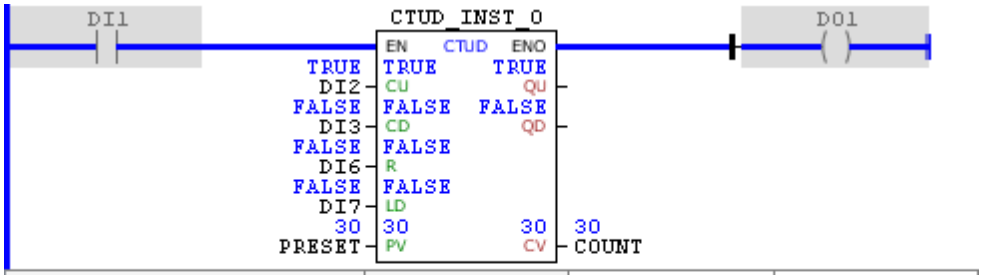
PRESET	UINT	30	Decimal
COUNT	UINT	0	Decimal

The example above shows the disabled block, with all its internal variables zeroed. Although the external controls are activated, these values are not forwarded to the instance of the block.



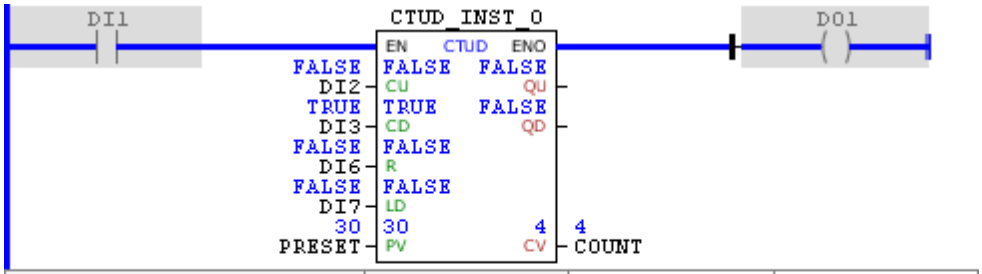
PRESET	UINT	30	Decimal
COUNT	UINT	0	Decimal

When activated, the block identifies the value of PRESET, loading it in PV, and identifies that the output is at zero, enabling the QD output. When execution is completed, the ENO output is activated.



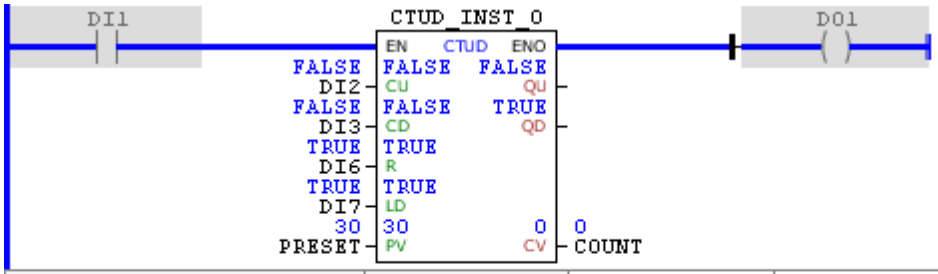
PRESET	UINT	30	Decimal
COUNT	UINT	30	Decimal

At each leading edge identified in CU, the value of CV is incremented until it reaches the PV value, when the QU output is enabled. When execution is completed, the ENO output is activated.



PRESET	UINT	30	Decimal
COUNT	UINT	4	Decimal

At each leading edge detected in CD, the CV value is decremented. When CV is a value between zero and PV, both QD and QU outputs are deactivated. When execution is completed, the ENO output is activated.



PRESET	UINT	30	Decimal
COUNT	UINT	0	Decimal

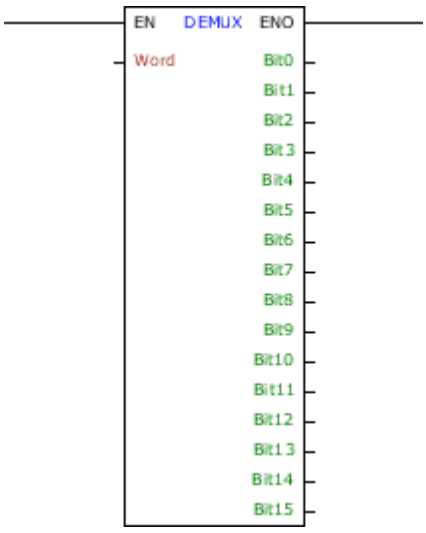
A TRUE value in R resets CV, while a TRUE value in LD loads the value of PV to CV. As we can see, R prevails over LD, leaving CV and enabling the QD output. When execution is completed, the ENO output is activated.

11.2.6.8 Data Transfer

11.2.6.8.1 DEMUX

Block that creates 16 new BOOL variables from the decomposition of a WORD variable.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Word	WORD UINT INT	Input variable of 15 bits
VAR_OUTPUT	ENO	BOOL	End of operation
	Bit0 – Bit15	BOOL	Bit of the corresponding position of Word

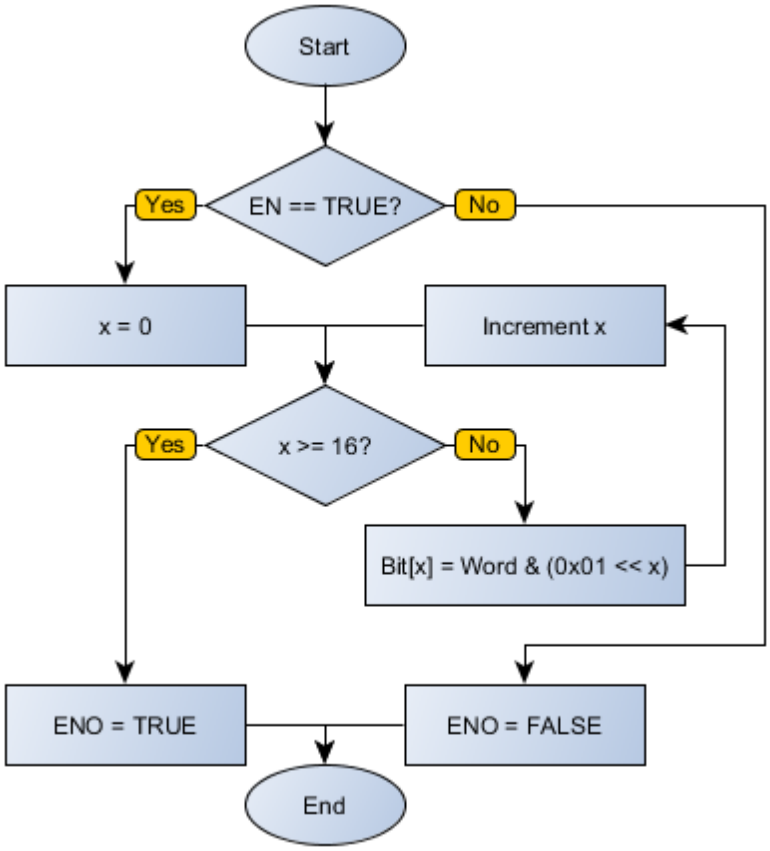
Operation

When this block has a TRUE value in EN, it decomposes the input variable in Word 15 Boolean values stored in Bit0 to Bit15 variables. Bit0 corresponds to the LSB (least significant bit) and Bit15 corresponds to the MSB (most significant bit).

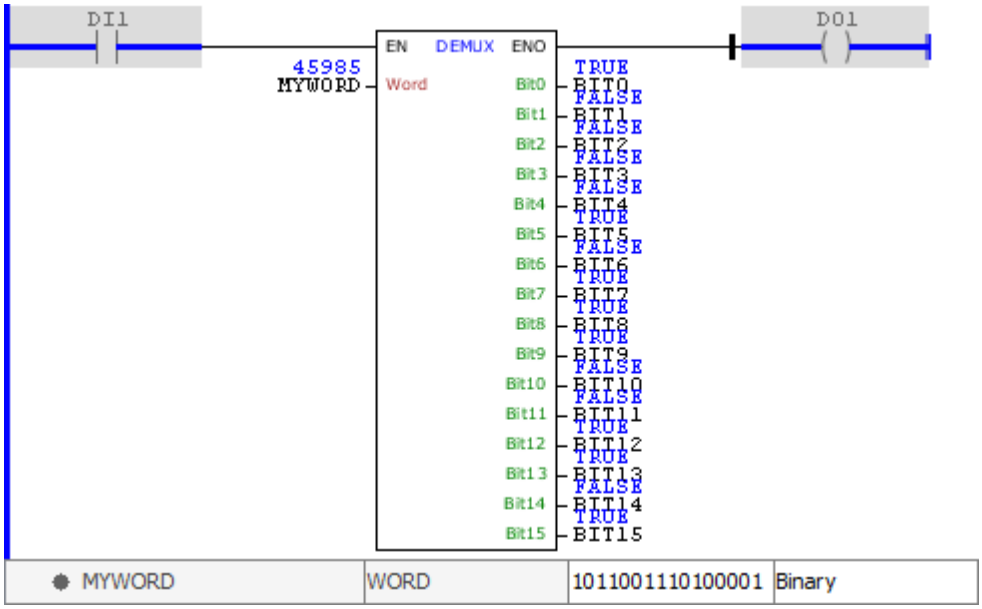
When EN has FALSE value, output variables remain unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example

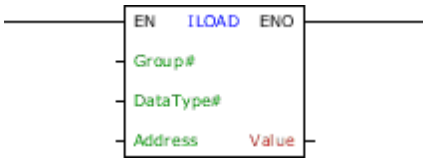


The example above decomposes the value of MYWORD in Boolean values, which are stored in the output variables BIT0 to Bit15. The block ends successfully and the ENO output is activated.

11.2.6.8.2 ILOAD

Block which indirectly loads the value of a variable and transfers it to Value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Group#	BYTE	Group w here the variable is stored
	DataType#	BYTE	Data type of the selected variable
	Address	DWORD UDINT DINT	Address of the global variable, as its group
VAR_OUTPUT	ENO	BOOL	End of operation
	Value	As selected in DataType#	Value of the selected variable

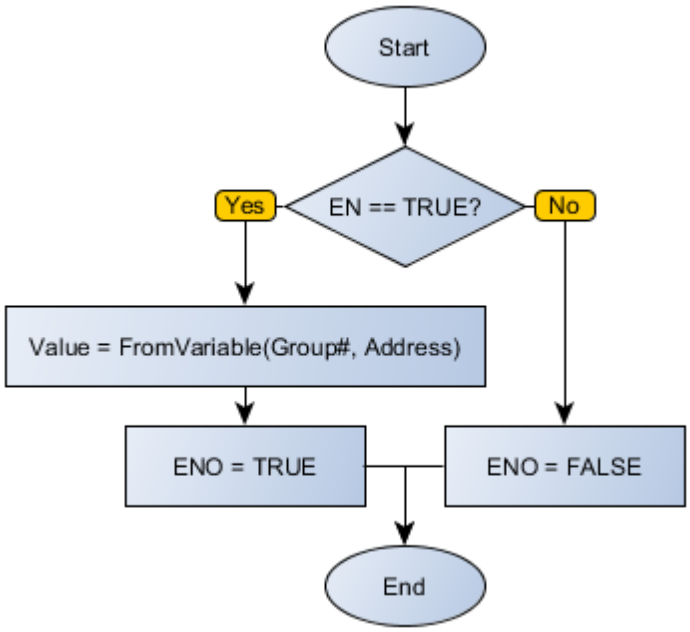
Operation

When this block has a TRUE value in EN, it loads, in Value, the of the Address variable belonging to the Group# group, as the selected DataType#.

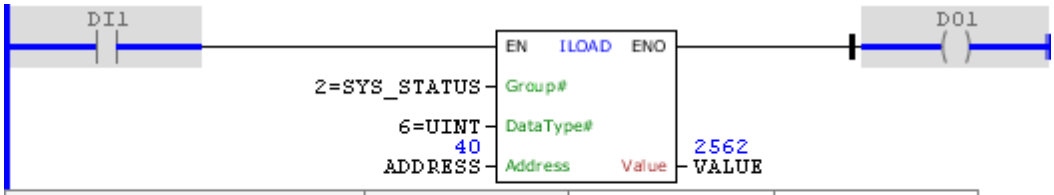
When EN has FALSE value, Value remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



ADDRESS	DINT	40	Decimal
VALUE	UINT	2562	Decimal

The above example loads the value of the address 40 of group 2 (GLOBAL_SYSTEM%S), which represents the status of ESC key in UINT format for the VALUE variable. The block ends with success and ENO output is activated.

11.2.6.8.3 ILOADBOOL

Block that indirectly loads the value of a bit in a global variable address.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Group#	BYTE	Group w here the variable is stored
	Address	DWORD UDINT DINT	Address of the global variable, as its group
	Bit	BYTE USINT SINT	Position of the bit to be checked
VAR_OUTPUT	ENO	BOOL	End of operation
	Value	BOOL	Value of the bit selected by the input arguments

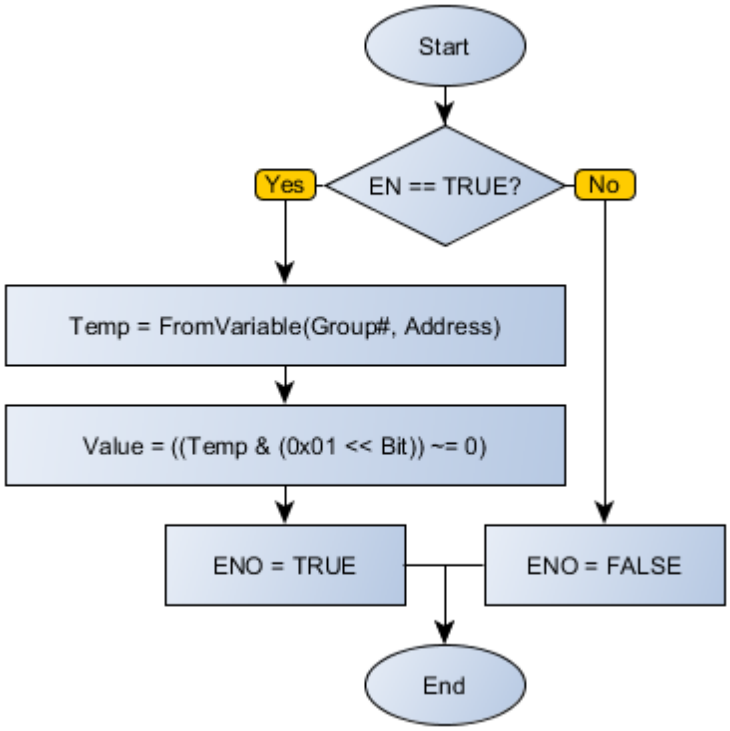
Operation

When this block has a TRUE value in EN, it loads, in Value, the Bit contents of the Address variable belonging to the Group# group.

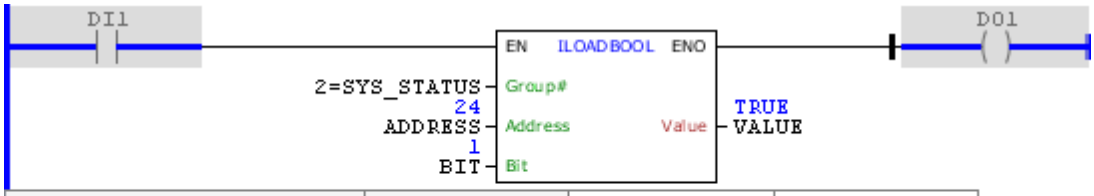
When EN has FALSE value, Value remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



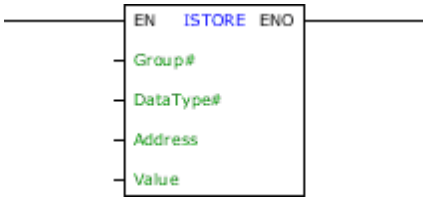
ADDRESS	DINT	24	Decimal
BIT	BYTE	1	Decimal
VALUE	BOOL	1	Binary

The above example loads the value of bit 1 of the address 24 of group 2 (S GLOBAL_SYSTEM%), which represents the status of ESC key for the VALUE variable. The block ends with success and ENO output is activated.

11.2.6.8.4 ISTORE

Block that indirectly loads the Value value in a variable.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Group#	BYTE	Group where the variable is stored
	DataType#	BYTE	Data type of the selected variable
	Address	DWORD UDINT DINT	Address of the global variable, as its group
	Value	Depending on the selection of the DataType#	Value to be written in the selected variable
VAR_OUTPUT	ENO	BOOL	End of operation

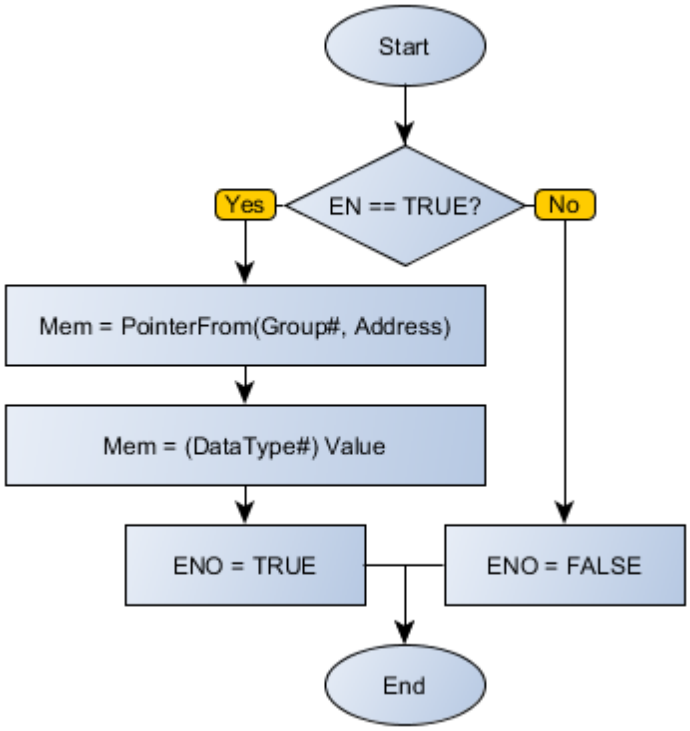
Operation

When this block has a TRUE value in EN, it loads the Value value in the contents of the Address variable belonging to the Group# group, as the selected DataType#.

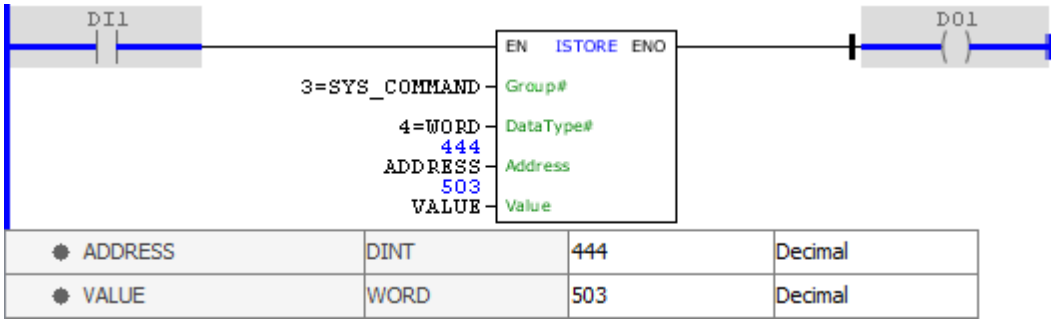
When EN has FALSE value, Value remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



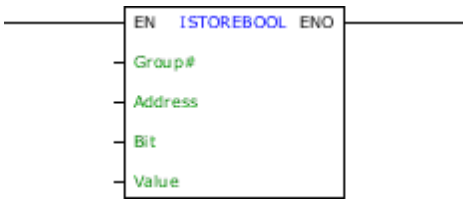
ADDRESS	DINT	444	Decimal
VALUE	WORD	503	Decimal

The example above stores the VALUE value in WORD format in address 444 of group 3 (GLOBAL_SYSTEM% C), which represents the index of the communication port Modbus TCP. The block ends with success and ENO output is activated.

11.2.6.8.5 ISTOREBOOL

Block that indirectly loads the Value value in a bit in a global variable address.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Group#	BYTE	Group where the variable is stored
	Address	DWORD UDINT DINT	Address of the global variable, as its group
	Bit	BYTE USINT SINT	Position of the bit to be modified
	Value	BOOL	New value of the selected bit
VAR_OUTPUT	ENO	BOOL	End of operation

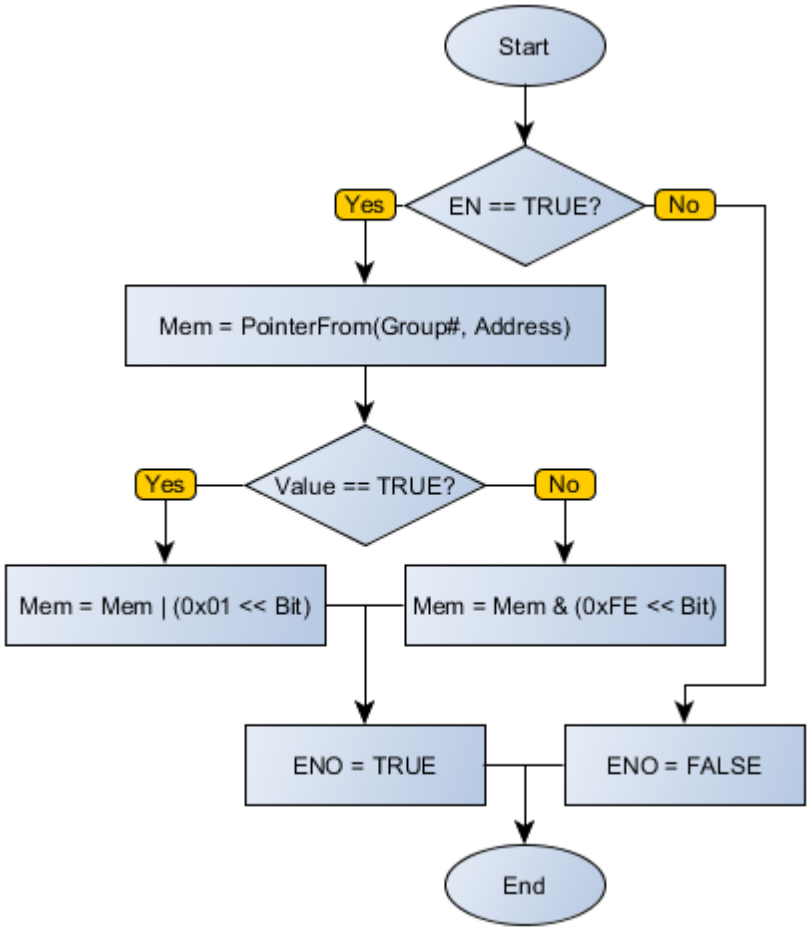
Operation

When this block has a TRUE value in EN, it loads the Value value in the Bit contents of the Address variable belonging to the Group# group.

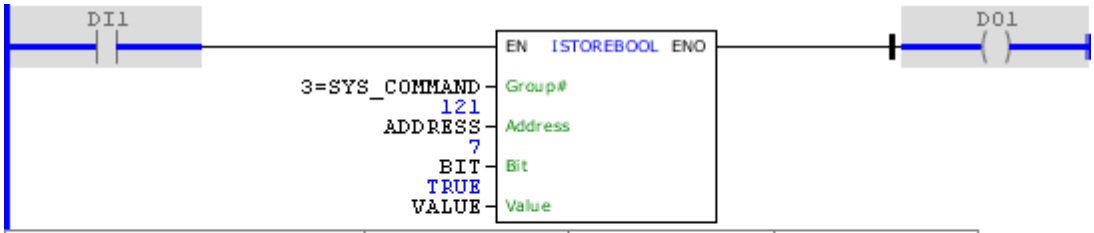
When EN has FALSE value, Value remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



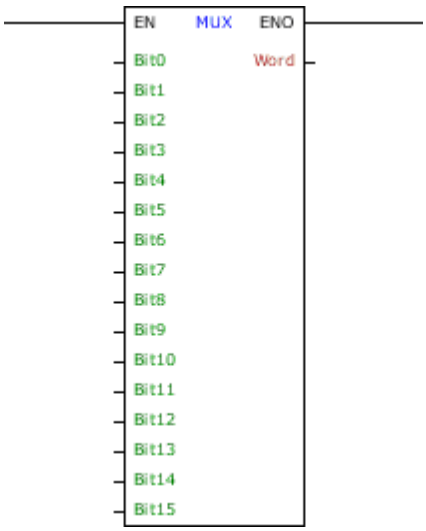
ADDRESS	DINT	121	Decimal
BIT	BYTE	7	Decimal
VALUE	BOOL	1	Binary

The example above stores the value of VALUE in bit 7 of the address 121 in group 3 (GLOBAL_SYSTEM% C), which represents the disable command of CANopen communication. The block ends with success and ENO output is activated.

11.2.6.8.6 MUX

Block that creates a new WORD variable from the concatenation of 16 BOOL variables.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Bit0 – Bit15	BOOL	Bit of the corresponding position in the new word
VAR_OUTPUT	ENO	BOOL	End of operation
	Word	WORD UINT INT	New word formed from the input bits

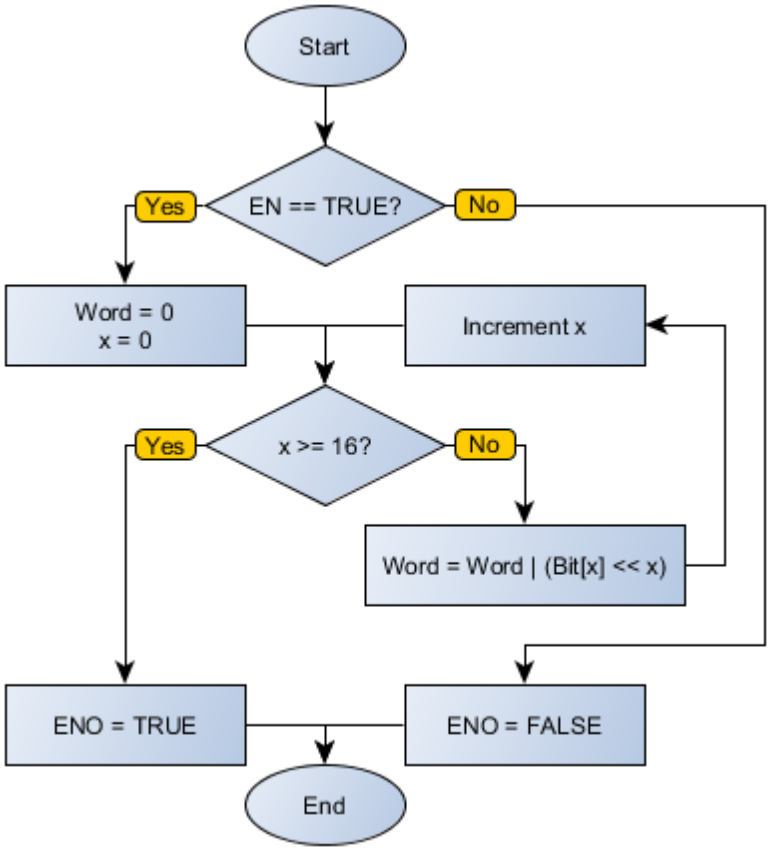
Operation

When this block has a TRUE value in EN, it concatenates Boolean values of the input variables and stores this value in the variable Word. Bit0 corresponds to LSB (least significant bit) and Bit15 corresponds to the MSB (most significant bit).

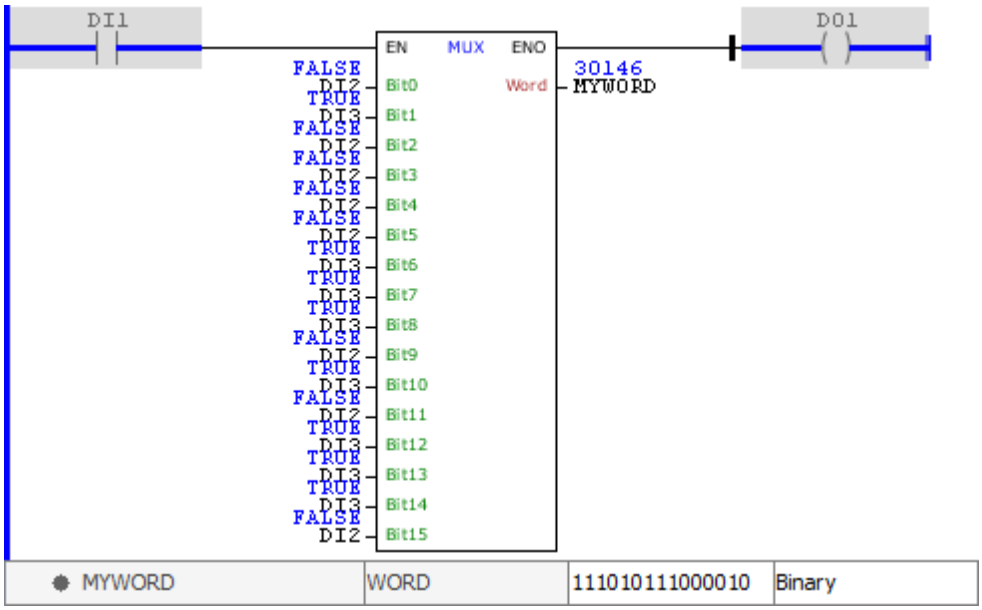
When EN has FALSE value, Word remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example

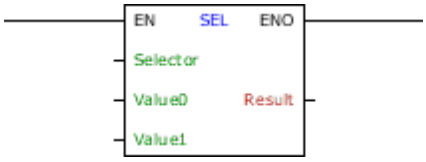


The above example concatenates the Boolean values of the input bits of the block to form the output word stored in MYWORD. The block ends with success and ENO output is activated.

11.2.6.8.7 SEL

Block that replicates to the output the value of an input variable (Value0 or Value1) according to the Selector selection.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Selector	BOOL	Variable that selects the input
	Value0	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Multiplexed input number 1
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Multiplexed input number 2
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Output value selected

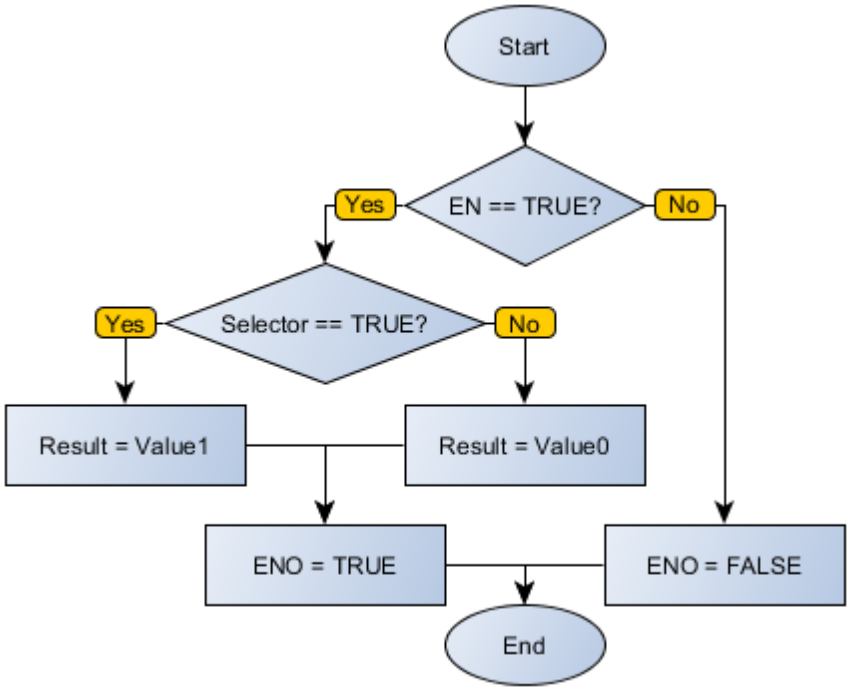
Operation

When this block has a TRUE value in EN, it replicates to the Result variable the Value0 value if selector is FALSE or the Value1 value if Selector is TRUE.

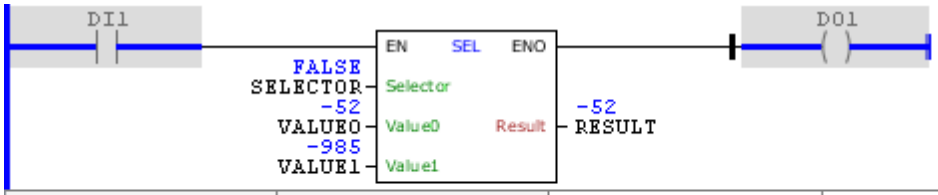
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart

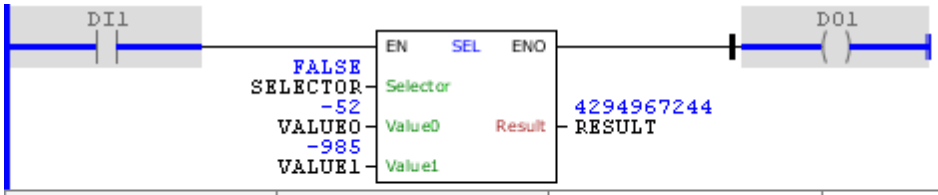


Example



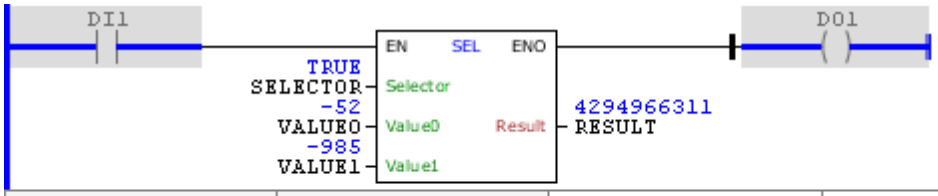
SELECTOR	BOOL	0	Binary
VALUE0	SINT	-52	Decimal
VALUE1	INT	-985	Decimal
RESULT	INT	-52	Decimal

The above example uses the SELECTOR variable as multiplexing channel selector. When it is at FALSE level, the RESULT output gets the value of VALUE0. The block ends successfully and the ENO output is activated.



SELECTOR	BOOL	0	Binary
VALUE0	SINT	-52	Decimal
VALUE1	INT	-985	Decimal
RESULT	DWORD	4294967244	Decimal

The above example uses the SELECTOR variable as multiplexing channel selector. When it is at FALSE level, the RESULT output gets the value of VALUE0. The block ends successfully and the ENO output is activated. Note that the binary pattern has been maintained even though the decimal representation is corrupted, since DWORD does not accept negative values.



SELECTOR	BOOL	1	Binary
VALUE0	SINT	-52	Decimal
VALUE1	INT	-985	Decimal
RESULT	DWORD	4294966311	Decimal

The above example uses the SELECTOR variable as multiplexing channel selector. When it is at TRUE level, the RESULT output gets the value of VALUE1. The block ends successfully and the ENO output is activated. Note that the binary pattern has been maintained even though the decimal representation is corrupted, since DWORD does not accept negative values.

11.2.6.8.8 STORE

Block that performs direct storage of data from a source to a destination.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	SRC	BYTE USINT SINT WORD UINT INT DWORD DINT DINT REAL	Data source
VAR_OUTPUT	ENO	BOOL	End of operation
	DST	BYTE USINT SINT WORD UINT INT DWORD DINT DINT REAL	Data destination

Operation

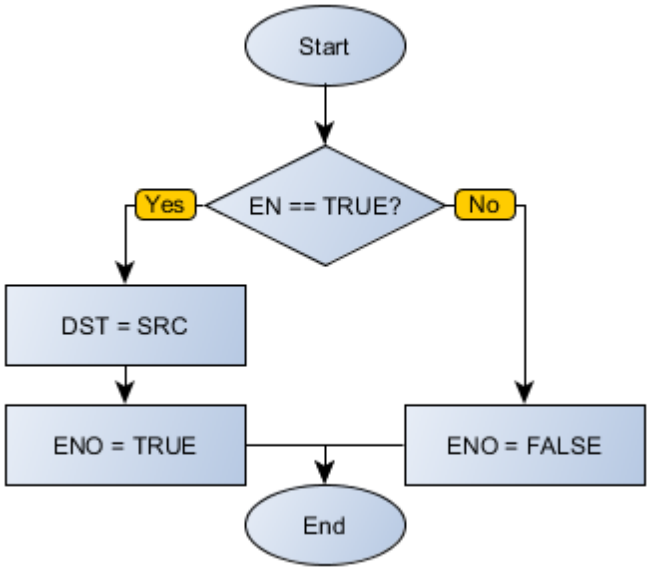
When this block has a TRUE value in EN, it stores the contents from SRC into DST.

NOTE!
SRC and DST must have data types of the same size.

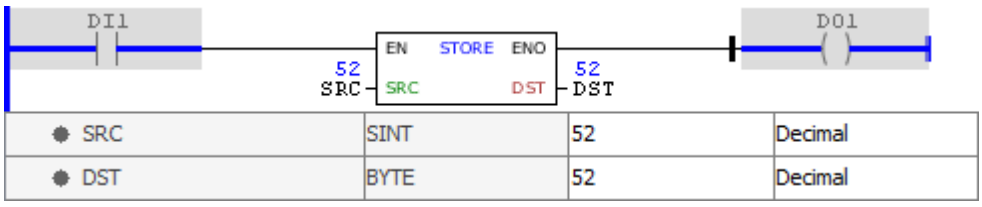
When EN has FALSE value, DST remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



The example above stores the value of the variable SRC in DST. The block ends with success and ENO output is activated.

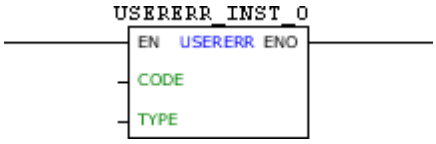
SRC	SINT	-15	Decimal
DST	BYTE	241	Decimal

The example above stores the value of the variable SRC in DST. The block ends with success and ENO output is activated. Note that the binary pattern is maintained between variables of different types.

11.2.6.8.9 USERERR

Block that generates an alarm or fault with the number programmed by the user.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	CODE	WORD UINT	Error code generated (950 - 999)
	TYPE	BYTE	Error type generated (0 - Alarm) (1 - Fault)
VAR_OUTPUT	ENO	BOOL	Success in the generation of error
VAR	USERERR_INST_0	USERERR	(*) Instance of access to block structure

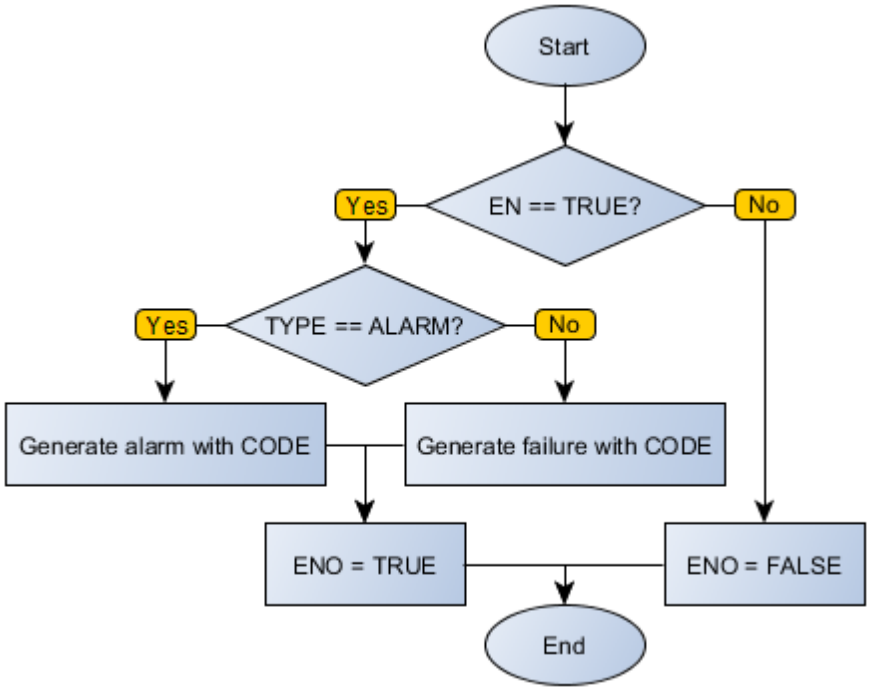
NOTE!
(*) USERERR_INST_0 instance must be configurated to SCA06 and LDW900.

Operation

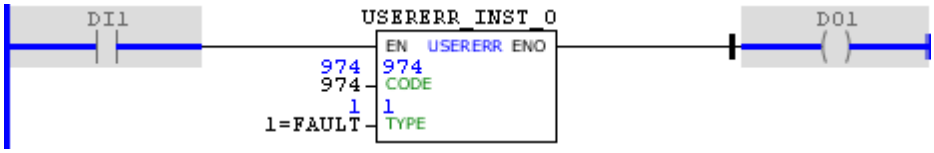
When this block has a TRUE value in EN, it generates an alarm or equipment failure, depending on the type defined in TYPE with CODE code.

The value of ENO informs if the generation of alarm or fault has been executed successfully.

Block Flowchart



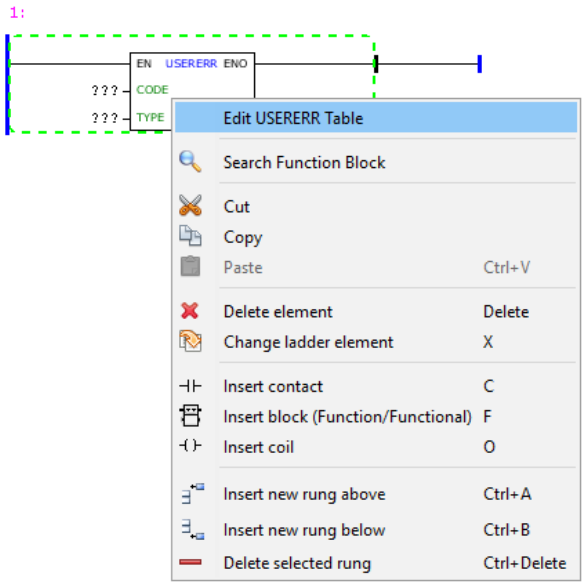
Example



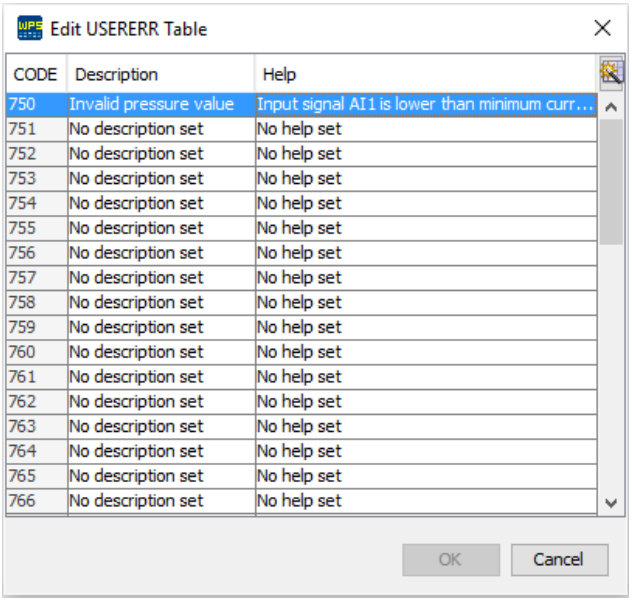
The above example, when identifying TRUE level in DI1, generates a fault with the code 974 and sets the DO1 output.

USERERR table configuration

On devices that have text-based HMI, messages can be configured through an editor. To access the editor, right click on the USERERR block and select the "Edit USERERR Table" option.



The texts configured in the table will be displayed on the HMI when the block USERERR is enabled.



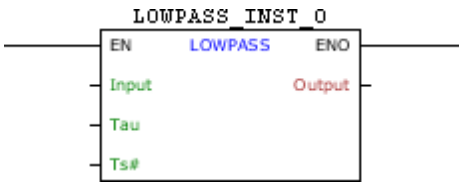
After editing the table, select the argument CODE of the block equal to the CODE column of the table.

11.2.6.9 Filter

11.2.6.9.1 LOWPASS

Block that filters the input using a low pass filter of first order and inserts the result in the output.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Input	REAL	Input signal
	Tau	REAL	Filter time constant
	Ts#	UINT	Sampling time [ms]
VAR_OUTPUT	ENO	BOOL	Output enabling
	Output	REAL	Filter output
VAR	LOWPASS_INST_0	LOWPASS	Instance of access to block structure

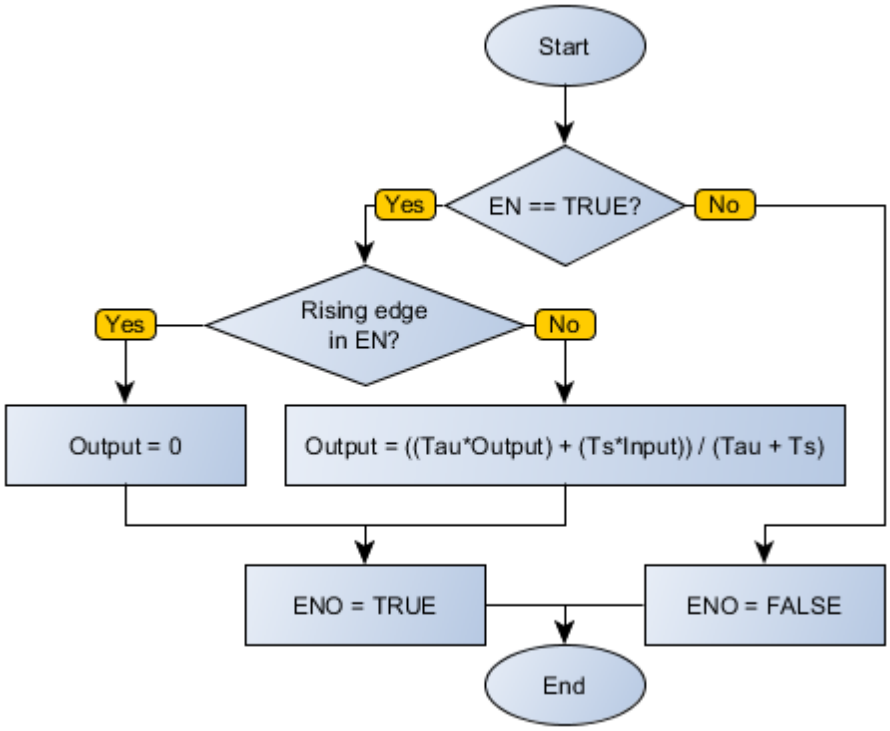
Operation

When this block has a TRUE value in EN, filters the input value of Input using a low pass first order filter described by Tau and Ts#, inserting the result in Output. On the leading edge of EN, Output receives zero.

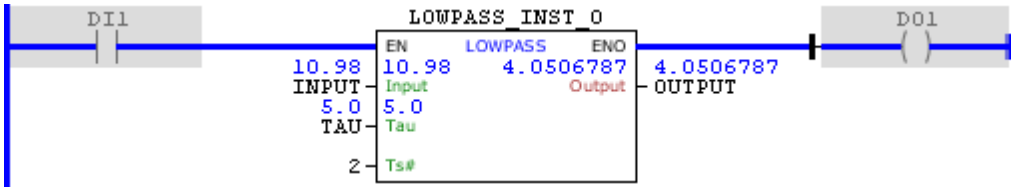
When EN has FALSE value, Output remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

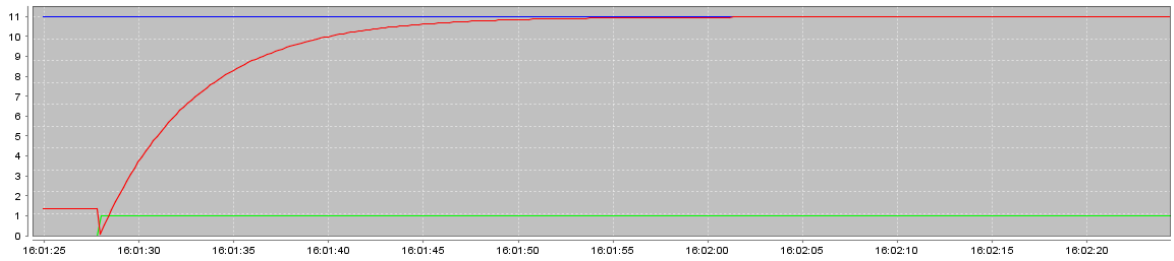
Block Flowchart



Example



The above example causes OUTPUT, by identifying a leading edge in EN, to display a behavior of first order with time constant equal to Tau and the sampling time of 2 ms, in order to achieve the reference set to INPUT. At each calculation completed successfully, the ENO output is activated.



11.2.6.10 Logic

11.2.6.10.1 Logic Bit

11.2.6.10.1.1 RESETBIT

Logical block used to perform reset of a specific bit in a field.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_IN_OUT	Data	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable w hose bit w ill be changed
VAR_INPUT	EN	BOOL	Block enabling
	Position	BYTE USINT	Position of the bit that w ill be changed
VAR_OUTPUT	DONE	BOOL	Operation successful

Operation

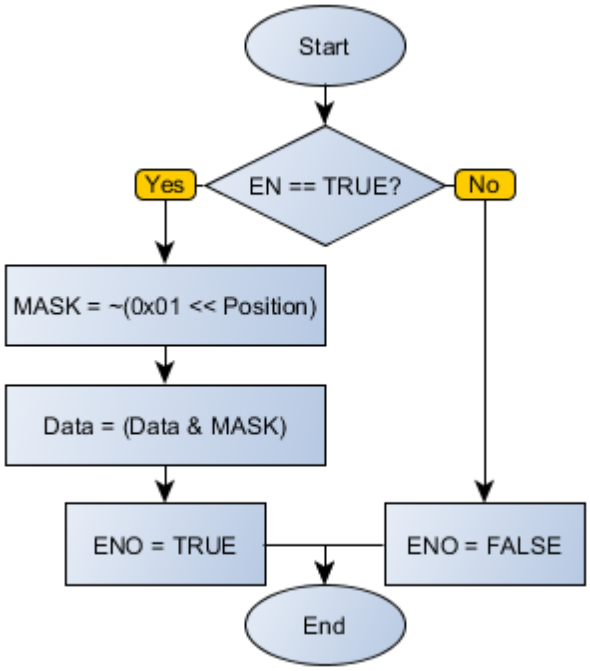
This block when it has a TRUE value in EN, resets the bit indicated in Position in the Data variable that is forwarded to the output already with its updated value.

When EN has FALSE value, Data remains unchanged.

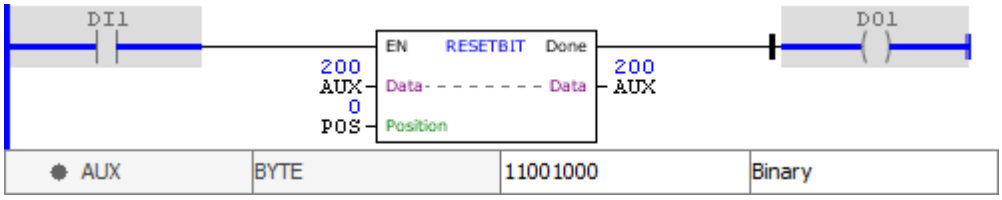
The DONE variable receives the same EN value, except when there is an error in the reset of the bit, then getting a FALSE value.

NOTE!
It is important to notice that Position is within the range of values of bits corresponding to variable type in Data. For example: if Data is a BYTE, it has 8 bits, and Position must contain a value between 0 and 7.

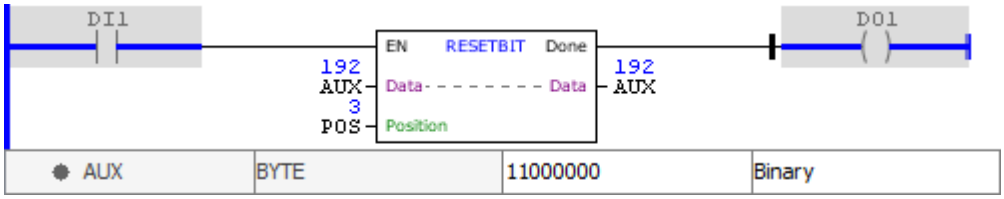
Block Flowchart



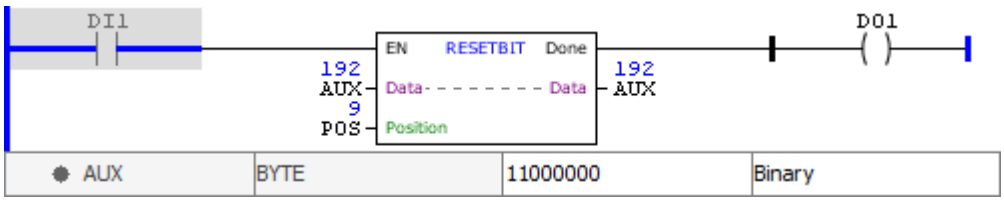
Example



The example above resets the bit of AUX zero position, whose initial value is 200 (1100 1000, in binary). Since this bit already had FALSE value, nothing has changed.



The example above resets the bit in position three of AUX by changing its binary value and, therefore, its decimal representation.



The example above resets the bit in position nine of AUX. Since AUX is a variable BYTE type, it has

only eight bits. Thus, the example above creates a runtime error in the block and therefore the output is not enabled.

11.2.6.10.1.2 SETBIT

Logical block used to perform the set of a specific bit in a field.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_IN_OUT	Data	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable w hose bit w ill be changed
VAR_INPUT	EN	BOOL	Block enabling
	Position	BYTE USINT	Position of the bit that w ill be changed
VAR_OUTPUT	DONE	BOOL	Operation successful

Operation

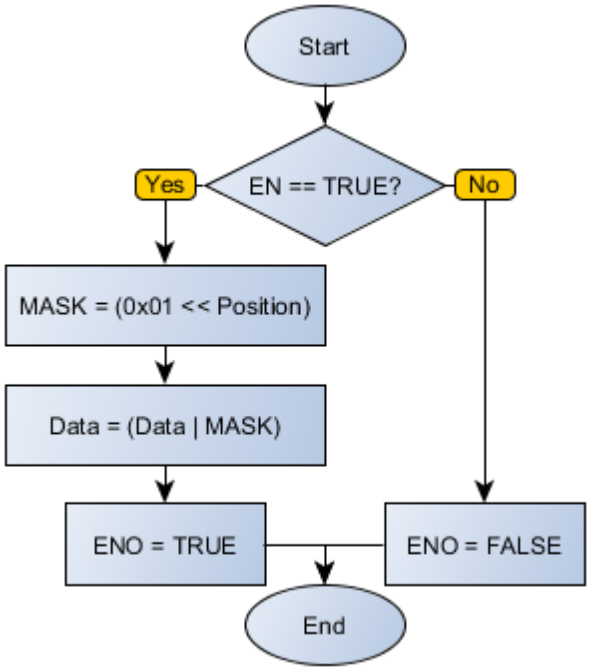
This block when it has a TRUE value in EN, sets the bit indicated in Position in the Data variable that is forwarded to the output already with its updated value.

When EN has FALSE value, Data remains unchanged.

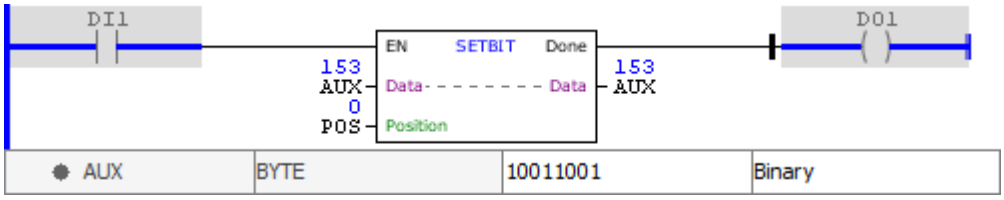
The DONE variable receives the same EN value, except when there is an error in the set of the bit, then getting a FALSE value.

NOTE!
It is important to notice that Position is within the range of values of bits corresponding to variable type in Data. For example: if Data is a BYTE, it has 8 bits, and Position must contain a value between 0 and 7.

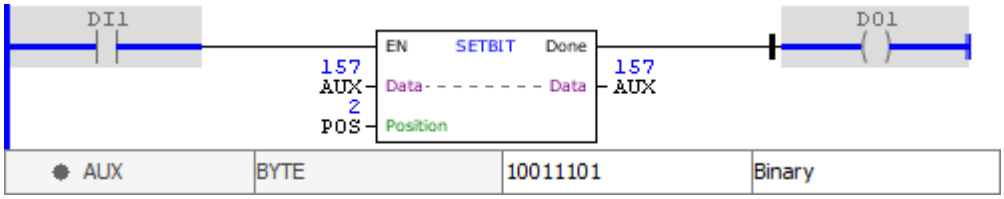
Block Flowchart



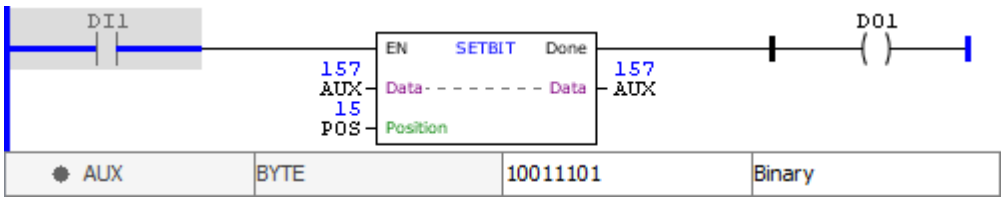
Example



The example above sets the bit of AUX zero position, whose initial value is 153 (1001 1001, in binary). Since this bit already had TRUE value, nothing has changed.



The example above sets the bit in position three of AUX by changing its binary value and, therefore, its decimal representation.



The example above sets the bit in position fifteen of AUX. Since AUX is a variable BYTE type, it has

only eight bits. Thus, the example above creates a runtime error in the block and therefore the output is not enabled.

11.2.6.10.1.3 TESTBIT

Logical block that revolutions the value of a specific bit in a field.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	Data	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable w hose bit w ill be tested
	EN	BOOL	Block enabling
	Position	BYTE USINT	Position of the bit that w ill be changed
VAR_OUTPUT	Q	BOOL	Value of the tested bit

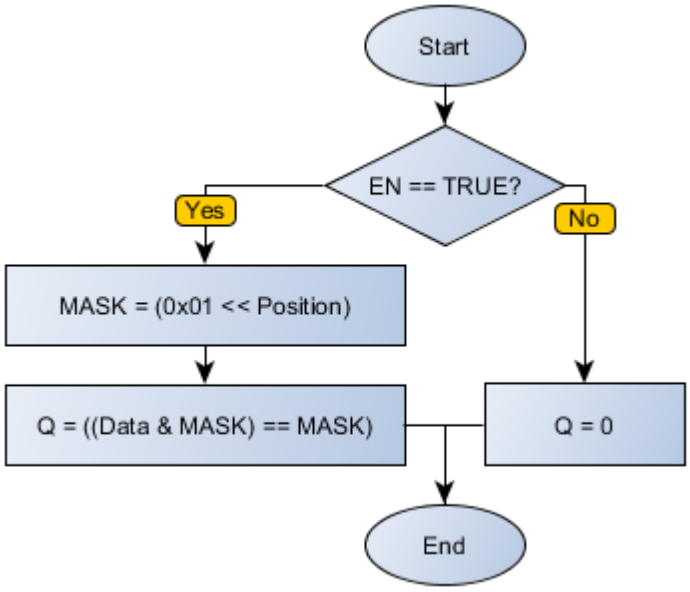
Operation

This block when it has a TRUE value in EN, sends to the output Q the bit value indicated in Position in the Data variable.

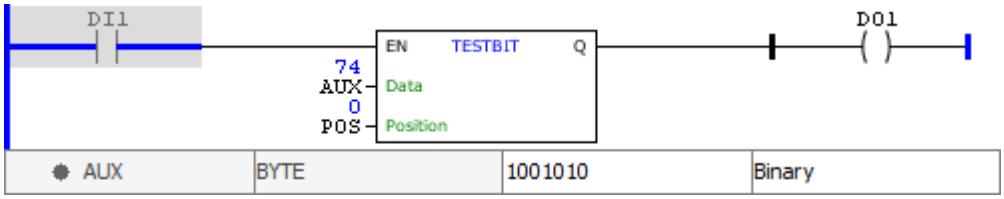
When EN has FALSE value, Q also receives FALSE.

NOTE!
It is important to notice that Position is within the range of values of bits corresponding to variable type in Data. For example: if Data is a BYTE, it has 8 bits, and Position must contain a value between 0 and 7.

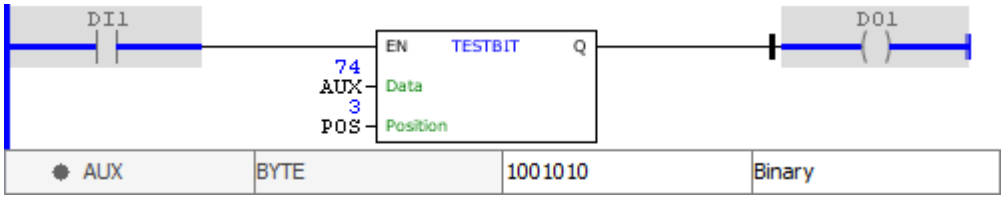
Block Flowchart



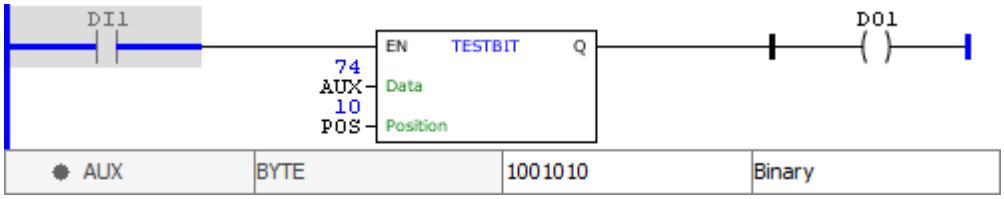
Example



The example above sets the bit value of zero position of AUX, whose initial value is 74 (0100 1010 in binary) to the output Q. Since this bit has value 0, the output is disabled.



The example above sets the value of the bit of position three of AUX to the output Q. Since this bit has value 1, the output is enabled.



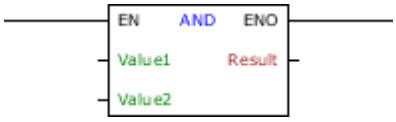
The example above sets the bit value of position ten of AUX to output Q. Since AUX is a variable of BYTE type, it has only eight bits. Thus, the example above creates a runtime error in the block and therefore the output is disabled.

11.2.6.10.2 Logic Boolean

11.2.6.10.2.1 AND

Logical block that performs an boolean "and" operation between two variables, storing the result in a third one.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable for the operation
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable for the operation
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable that stores the result of the operation

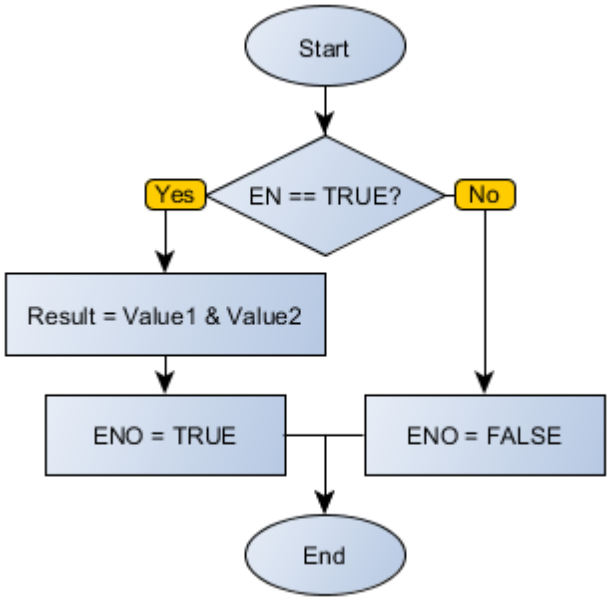
Operation

When this block has a TRUE value in EN, it sends to the Result output the “and” Boolean operation of input variables Value1 and Value2.

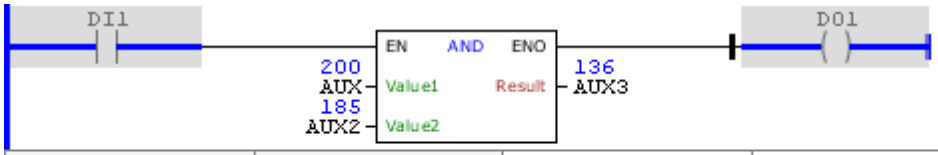
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



AUX	BYTE	11001000	Binary
AUX2	BYTE	10111001	Binary
AUX3	BYTE	10001000	Binary

The example above performs an "and" Boolean operation between AUX and AUX2, storing the result in AUX3.

11.2.6.10.2.2 NOT

Block that performs a logical operation of boolean "not" in a variable, storing the result in another.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Reference variable for the operation
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable that stores the result of the operation

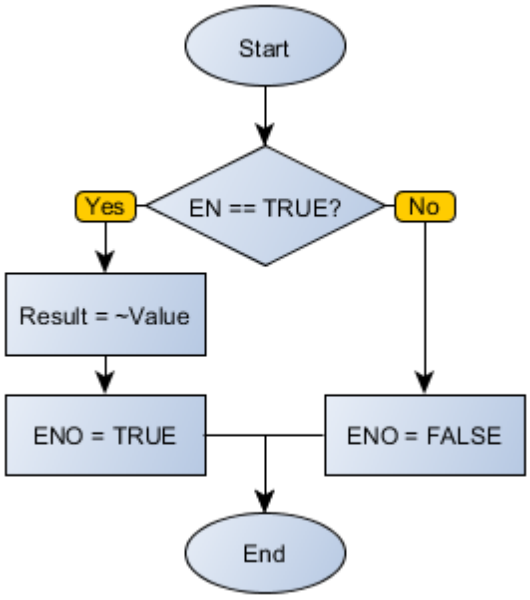
Operation

When this block has a TRUE value in EN, it sends to the Result output the denied Boolean value of the Value input variable.

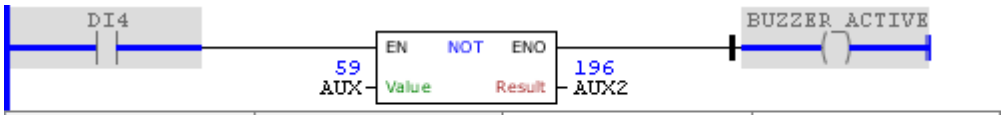
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



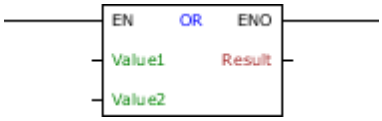
AUX	BYTE	111011	Binary
AUX2	BYTE	11000100	Binary

The example above performs a boolean "not" operation in AUX, storing the result in AUX2.

11.2.6.10.2.3 OR

Logical block that performs an Boolean "or" operation between two variables, storing the result in a third one.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable for the operation
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable for the operation
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable that stores the result of the operation

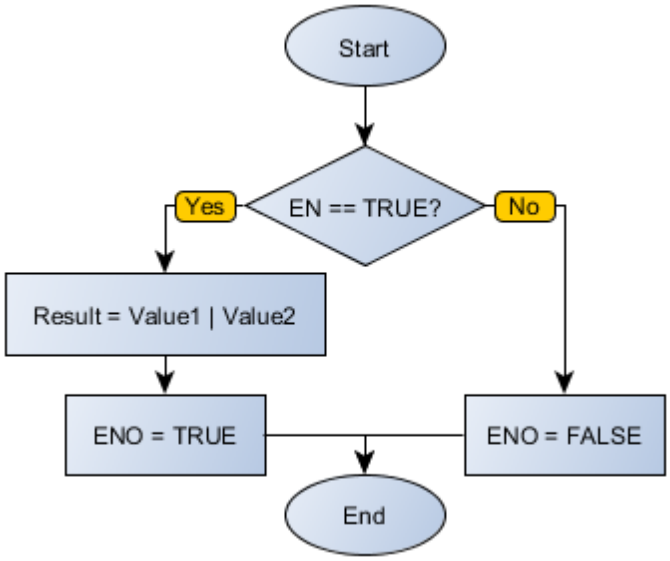
Operation

When this block has a TRUE value in EN, it sends to the Result output the “or” Boolean operation of input variables Value1 and Value2.

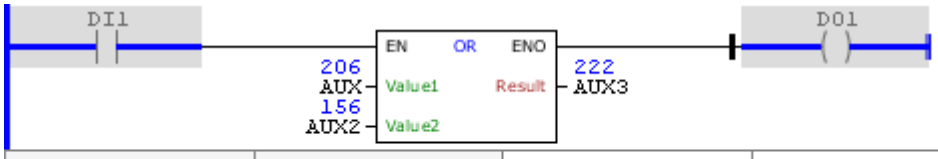
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



AUX	BYTE	11001110	Binary
AUX2	BYTE	10011100	Binary
AUX3	BYTE	11011110	Binary

The example above performs an "or" Boolean operation between AUX and AUX2, storing the result in AUX3.

11.2.6.10.2.4 XNOR

Logical block that performs an Boolean "not exclusive or" operation between two variables, storing the result in a third one.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable for the operation
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable for the operation
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable that stores the result of the operation

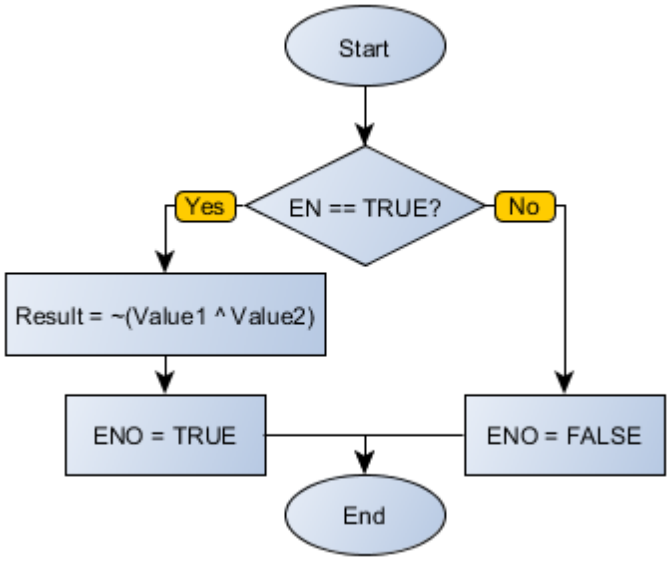
Operation

When this block has a TRUE value in EN, it sends to the Result output the “denied exclusive or” Boolean operation of input variables Value1 and Value2.

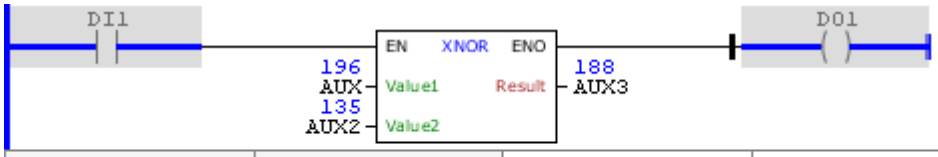
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



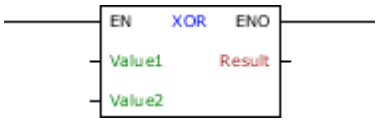
AUX	BYTE	11000100	Binary
AUX2	BYTE	10000111	Binary
AUX3	BYTE	10111100	Binary

The example above performs a "denied exclusive or" Boolean operation between AUX and AUX2, storing the result in AUX3.

11.2.6.10.2.5 XOR

Logical block that performs an Boolean "exclusive or" operation between two variables, storing the result in a third one.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable for the operation
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable for the operation
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable that stores the result of the operation

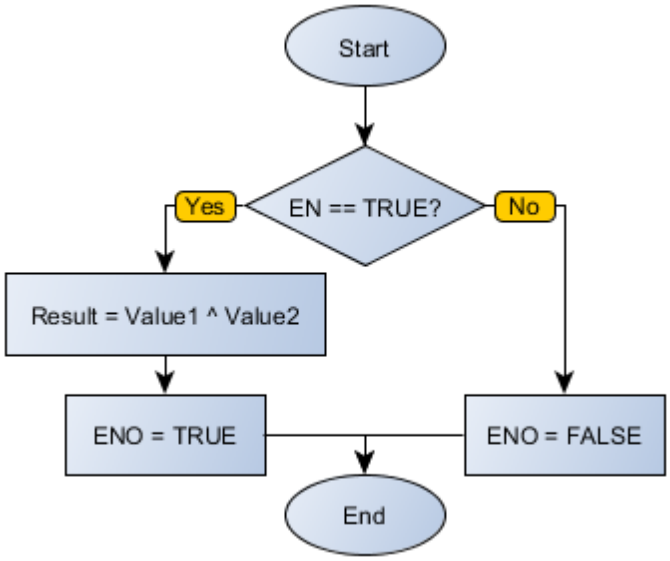
Operation

When this block has a TRUE value in EN, it sends to the Result output the “xor” Boolean operation of input variables Value1 and Value2.

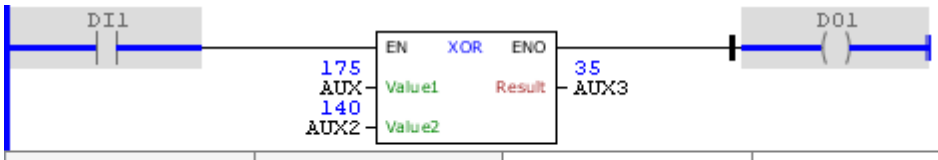
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



AUX	BYTE	10101111	Binary
AUX2	BYTE	10001100	Binary
AUX3	BYTE	100011	Binary

The example above performs a "xor" Boolean operation between AUX and AUX2, storing the result in AUX3.

11.2.6.10.3 Logic Rotate

11.2.6.10.3.1 ROL

Block that performs a logical left rotation operation in a value passed by Value, storing the result in Result.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable to undergo rotation
	Shift	BYTE USINT	Shift index
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable that stores the result of the operation

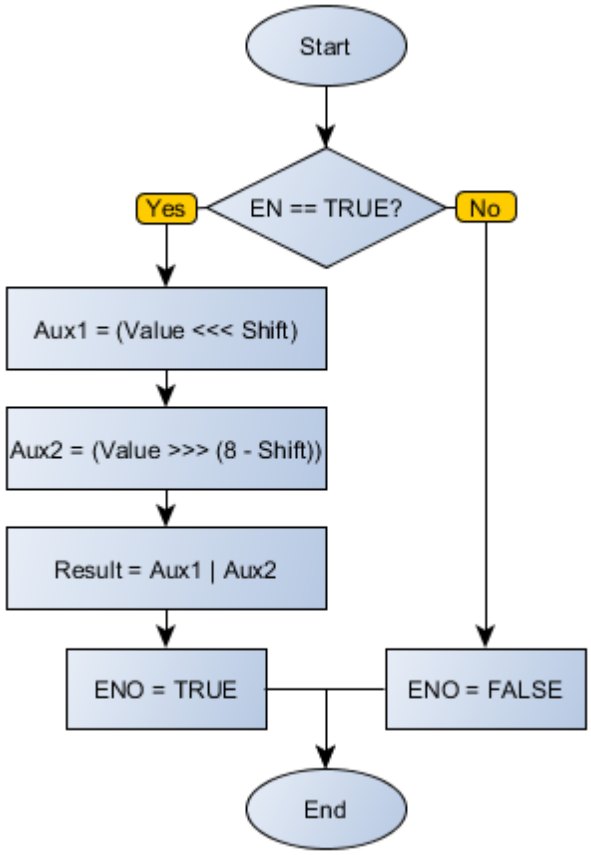
Operation

When this block has a TRUE value in EN, it sends to the Result output the value of the Value variable after performing a number of logical left shifts, according to the Shift value. The most significant bits that are being discarded are returned to the least significant bits, characterizing the rotation.

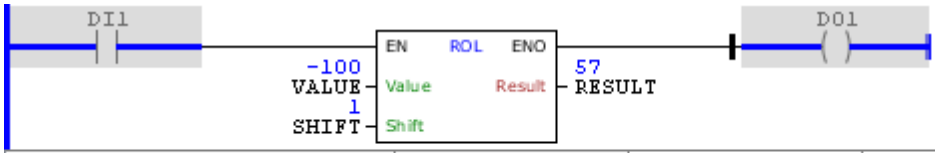
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart

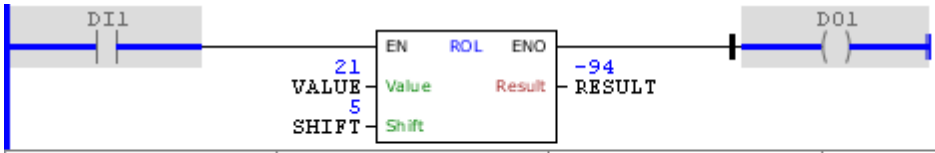


Example



VALUE	SINT	10011100	Binary
SHIFT	BYTE	1	Decimal
RESULT	SINT	111001	Binary

The above example performs a logical left shift by one position in the VALUE variable whose initial value is -100 (1001 1100 in binary). The discarded bits on the left are reinserted on the right. The final result (0011 1001 in binary) is stored in RESULT.



VALUE	SINT	10101	Binary
SHIFT	BYTE	5	Decimal
RESULT	SINT	10100010	Binary

The above example performs a logical left rotation by five positions in the VALUE variable whose initial value is 21 (0001 0101 in binary). The discarded bits on the left are reinserted on the right. The final result (1010 0010 in binary) is stored in RESULT.

11.2.6.10.3.2 ROR

Block that performs a logical right rotation operation in a value passed by Value, storing the result in Result.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable to undergo rotation
	Shift	BYTE USINT	Shift index
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable that stores the result of the operation

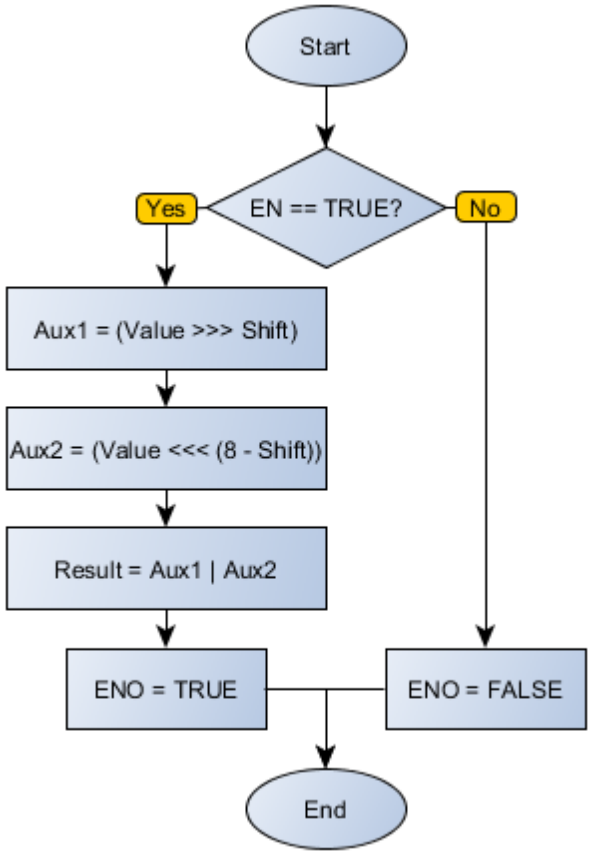
Operation

When this block has a TRUE value in EN, it sends to the Result output the value of the Value variable after performing a number of logical right shifts, according to the Shift value. The least significant bits that are being discarded are returned to the most significant bits, characterizing the rotation.

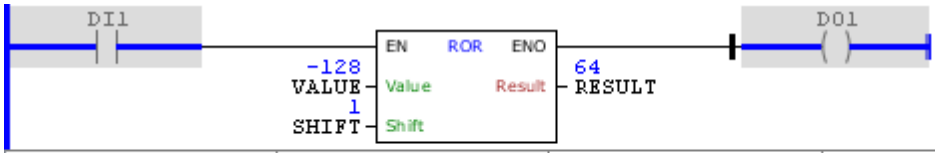
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart

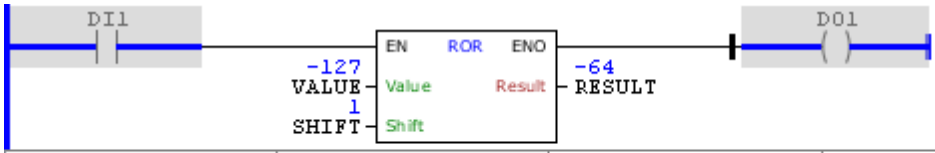


Example



VALUE	SINT	10000000	Binary
SHIFT	BYTE	1	Decimal
RESULT	SINT	1000000	Binary

The above example performs a logic right shift by one position in the VALUE variable whose initial value is -128 (1000 0000 in binary). The discarded bits on the right are reinserted on the left. The final result (0100 0000 in binary) is stored in RESULT. Notice that the sign is not preserved in this operation.



VALUE	SINT	10000001	Binary
SHIFT	BYTE	1	Decimal
RESULT	SINT	11000000	Binary

The above example performs a logical right rotation by one position in the VALUE variable whose initial value is -127 (1000 0001 in binary). The discarded bits on the right are reinserted on the left. The final result (1100 0000 in binary) is stored in RESULT.

11.2.6.10.4 Logic Shift

11.2.6.10.4.1 ASHL

Block that performs a binary left shift operation in a value passed by Value, storing the result in Result.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	SINT INT DINT	Variable to undergo shift
	Shift	BYTE USINT	Shift index
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	SINT INT DINT	Variable that stores the result of the operation

Operation

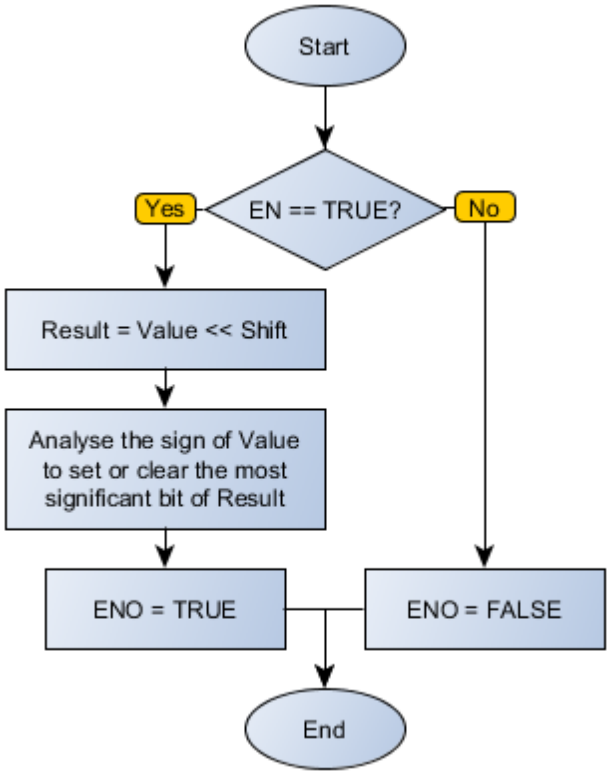
When this block has a TRUE value in EN, it sends to the Result output the value of the Value variable after performing a number of arithmetic left shifts, according to the Shift value.

NOTE!
All arithmetic shifts implemented maintain the sign of the variable.

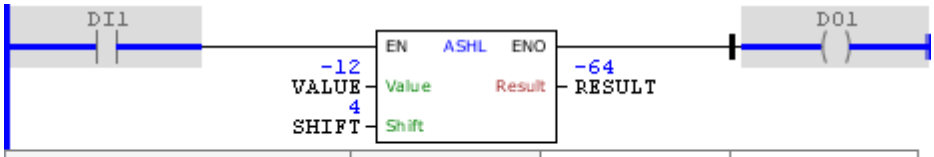
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart

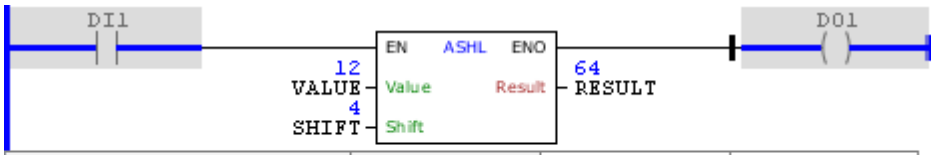


Example



VALUE	SINT	11110100	Binary
SHIFT	USINT	100	Binary
RESULT	SINT	11000000	Binary

Description of exemple.



VALUE	SINT	1100	Binary
SHIFT	USINT	100	Binary
RESULT	SINT	1000000	Binary

Description of exemple.

11.2.6.10.4.2 ASHR

Block that performs arithmetic left shift operation in a value passed by Value, storing the result in Result.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	SINT INT DINT	Variable to undergo shift
	Shift	BYTE USINT	Shift index
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	SINT INT DINT	Variable that stores the result of the operation

Operation

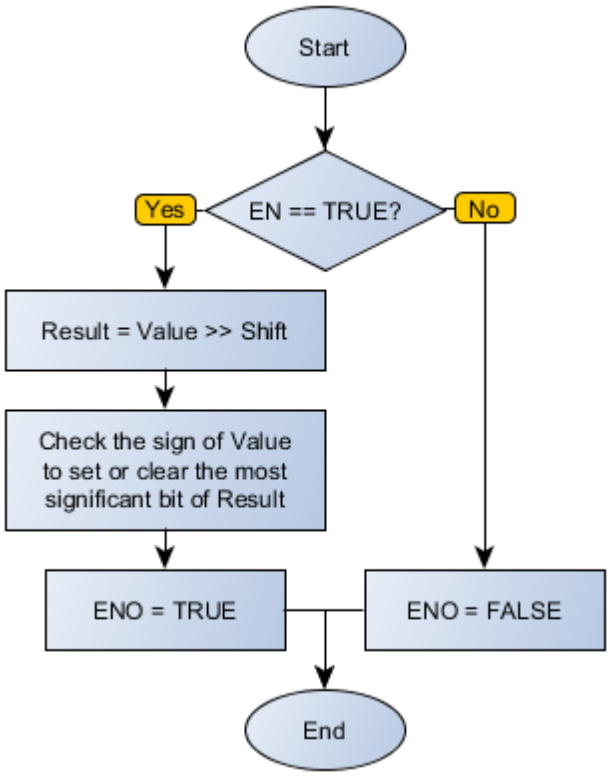
When this block has a TRUE value in EN, it sends to the Result output the value of the Value variable after performing a number of arithmetic right shifts, according to the Shift value.

NOTE!
All arithmetic shifts implemented maintain the sign of the variable.

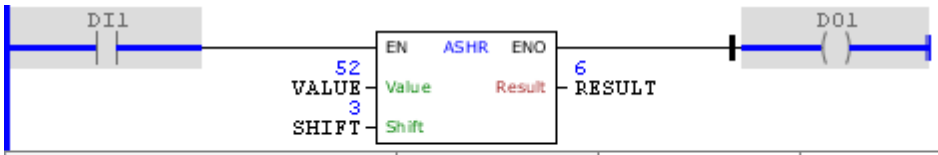
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart

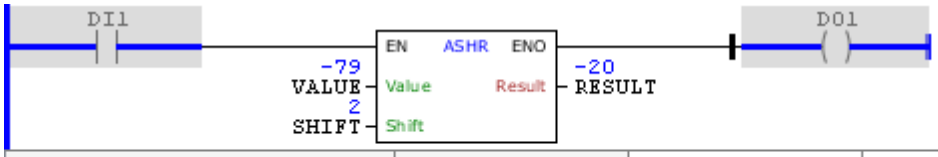


Example



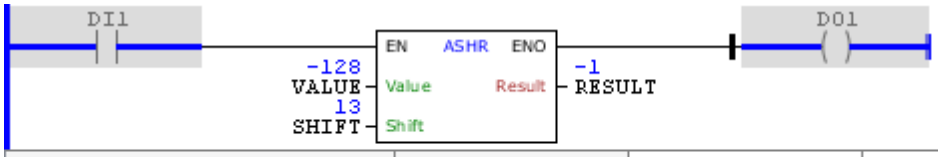
VALUE	SINT	110100	Binary
SHIFT	BYTE	3	Decimal
RESULT	INT	110	Binary

The above example performs an arithmetic right shift by three positions in the VALUE variable whose initial value is 52 (0011 0100 in binary). The bits on the right are being discarded, and on the left new zeros are inserted. The final result (0000 0110 in binary) is stored in RESULT.



VALUE	SINT	10110001	Binary
SHIFT	BYTE	2	Decimal
RESULT	INT	1111111111101100	Binary

The above example performs an arithmetic right shift by two positions in the VALUE variable whose initial value is -79 (1011 0001 in binary). The bits on the right will be discarded and new ones on the left are inserted, since the arithmetic right shifts preserve the sign of the variable. The final result (1111 0110 in binary) is stored in RESULT.



VALUE	SINT	10000000	Binary
SHIFT	BYTE	13	Decimal
RESULT	INT	1111111111111111	Binary

The above example performs an arithmetic right shift by thirteen positions in the VALUE variable whose initial value is -128 (1000 0000 in binary). The bits on the right are being discarded, and on the left new ones are inserted. The final result (1111 1111 in binary) is stored in RESULT.

11.2.6.10.4.3 SHL

Block that performs a binary logical left shift operation in a value passed by Value, storing the result in Result.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable to undergo shift
	Shift	BYTE USINT	Shift index
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable that stores the result of the operation

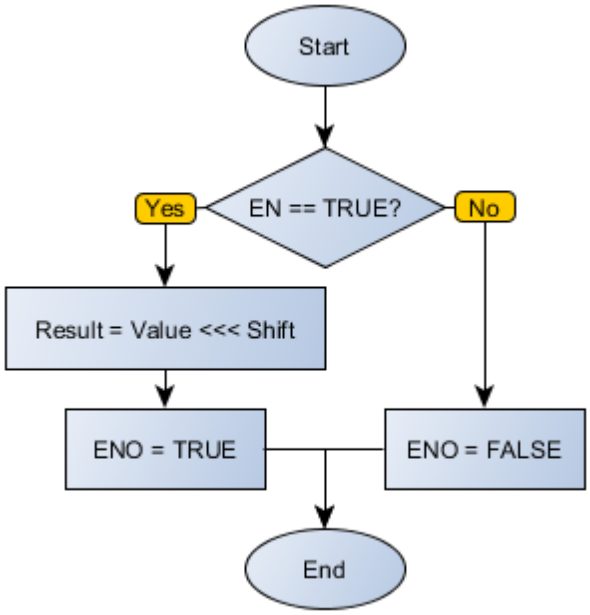
Operation

When this block has a TRUE value in EN, it sends to the Result output the value of the Value variable after performing a number of logical shifts left, according to the Shift value.

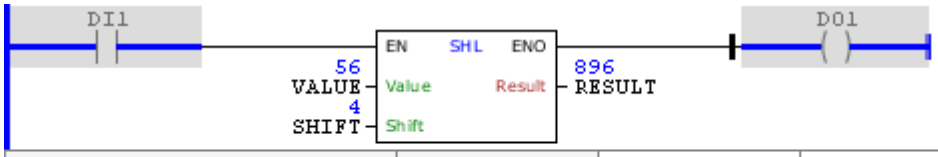
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart

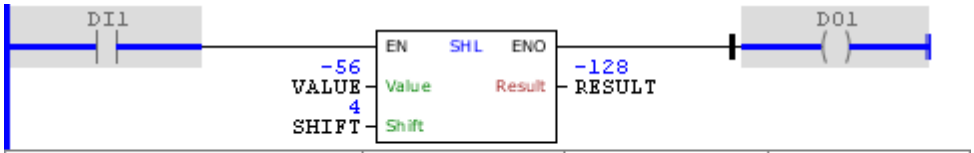


Example



VALUE	INT	111000	Binary
SHIFT	BYTE	4	Decimal
RESULT	INT	1110000000	Binary

The above example performs a logical right shift by four positions in the VALUE variable whose initial value is 56 (0011 1000 in binary). The bits on the left are being discarded, and on the left new zeros are inserted. The final result (0011 1000 0000 in binary) is stored in RESULT.



VALUE	SINT	11001000	Binary
SHIFT	USINT	4	Decimal
RESULT	SINT	10000000	Binary

The above example performs a logical right shift by four positions in the VALUE variable whose initial value is -56 (1100 1000 in binary). The bits on the left are being discarded, and on the left new zeros are inserted. The final result (1100 1000 0000 in binary) is stored in RESULT. Since RESULT is SINT type, it only accepts the first eight bits (1000 0000).

11.2.6.10.4.4 SHR

Block that performs a binary logical right shift operation in a value passed by Value, storing the result in Result.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable to undergo shift
	Shift	BYTE USINT	Shift index
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable that stores the result of the operation

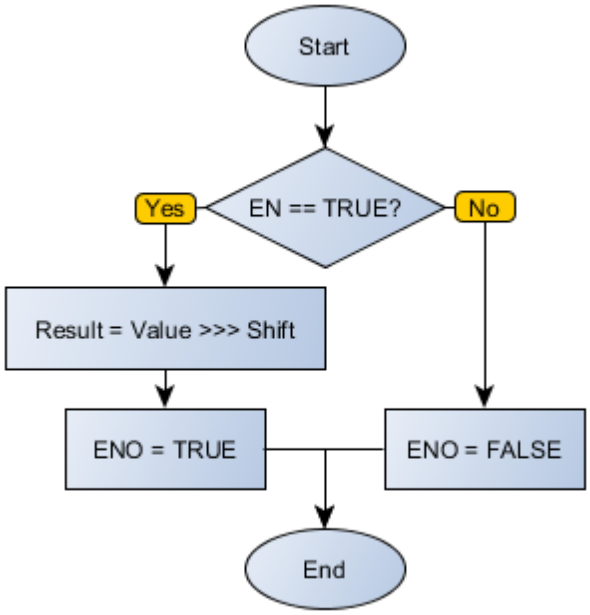
Operation

When this block has a TRUE value in EN, it sends to the Result output the value of the Value variable after performing a number of logical shifts right, according to the Shift value.

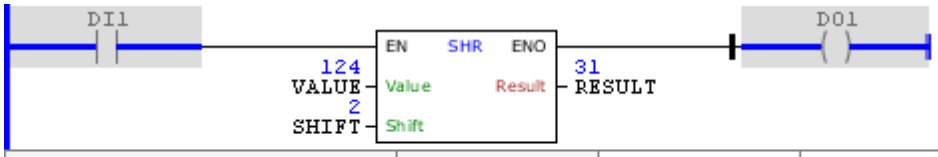
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart

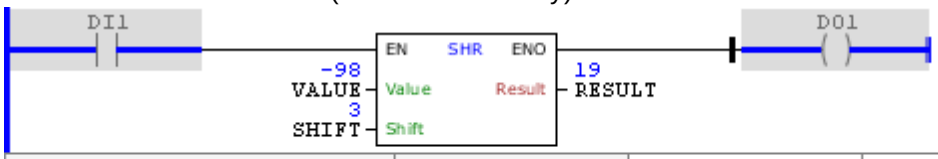


Example



VALUE	SINT	1111100	Binary
SHIFT	BYTE	2	Decimal
RESULT	SINT	11111	Binary

The above example performs a logical right shift by two positions in the VALUE variable whose initial value is 124 (0111 1100 in binary). The bits on the right are being discarded, and on the left new zeros are inserted. The final result (0001 1111 in binary) is stored in RESULT.



VALUE	SINT	10011110	Binary
SHIFT	BYTE	3	Decimal
RESULT	INT	10011	Binary

The above example performs a logical right shift by three positions in the VALUE variable whose initial value is -98 (1001 1110 in binary). The bits on the right are being discarded, and on the left new zeros are inserted. The final result (0001 0011 in binary) is stored in RESULT.

11.2.6.11 Math

11.2.6.11.1 Math Basic

11.2.6.11.1.1 ABS

Block that calculates the Value module, storing the result in Result.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Reference variable for the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Variable that stores the result of the operation

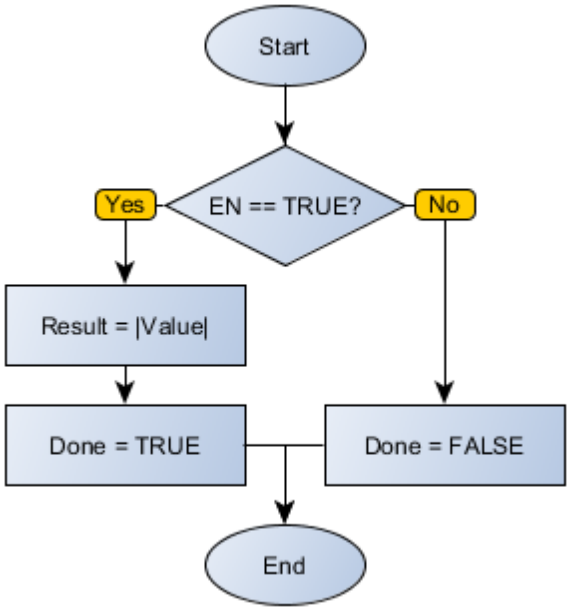
Operation

When this block has a TRUE value in EN, it sends to the Result output the absolute value of the

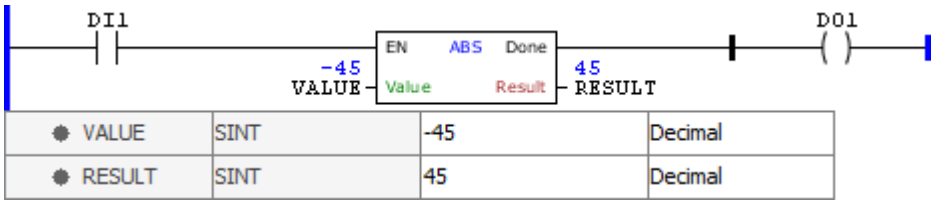
Value variable. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

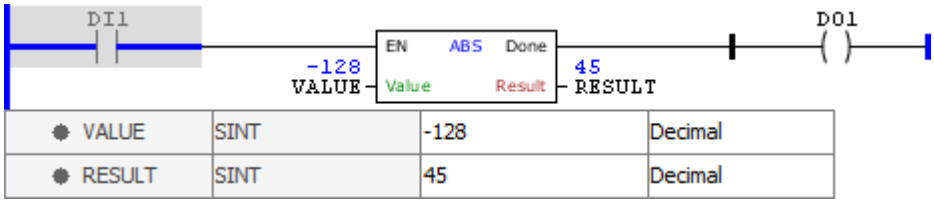


Example



VALUE	SINT	-45	Decimal
RESULT	SINT	45	Decimal

The above example calculates the absolute value of the VALUE variable whose initial value is -45, storing the final result, 45, in RESULT.



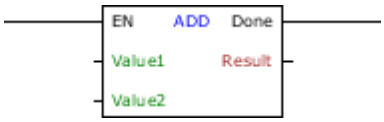
VALUE	SINT	-128	Decimal
RESULT	SINT	45	Decimal

The above example calculates the absolute value of the VALUE variable whose initial value is -45. The final result, 128, cannot be stored in RESULT, because it is outside the limits of accepted values by SINT type. Therefore, RESULT remains unchanged and the output is disabled.

11.2.6.11.1.2 ADD

Block that calculates the sum of the values of Value1 and Value2, storing the result in Result.

Ladder Representation



Block Structure

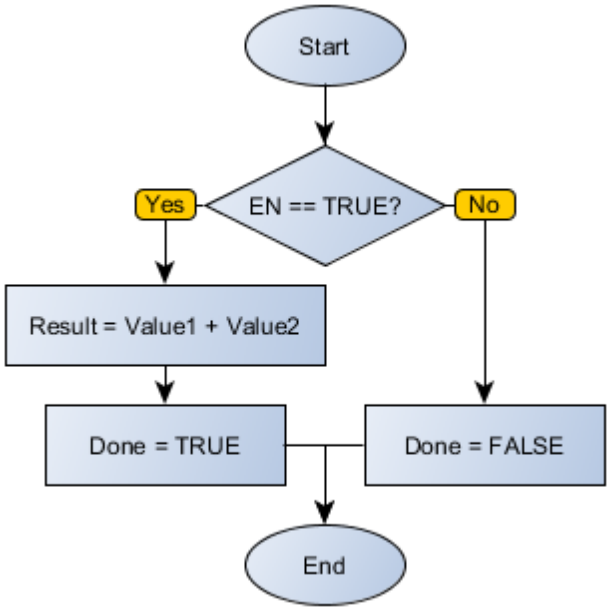
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	First addend of the operation
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Second addend of the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Variable that stores the result of the operation

Operation

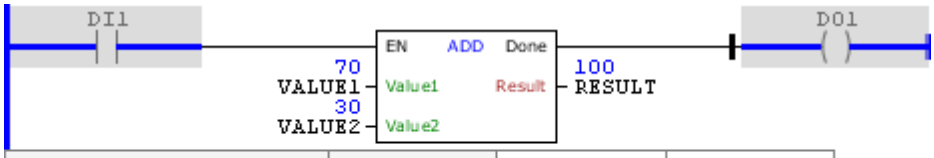
When this block has a TRUE value in EN, it sends to the Result output the sum of Value1 and Value2 variables. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

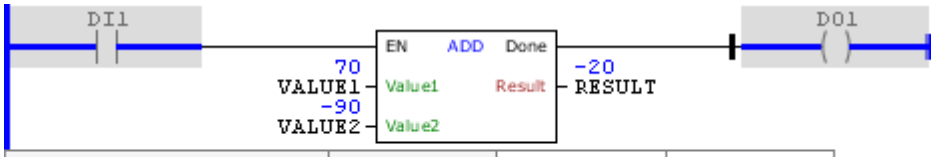


Example



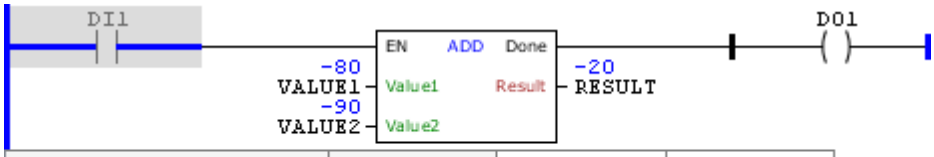
VALUE1	SINT	70	Decimal
VALUE2	SINT	30	Decimal
RESULT	SINT	100	Decimal

The above example calculates the sum of VALUE 1 and VALUE2 variables, storing the final result in RESULT.



VALUE1	SINT	70	Decimal
VALUE2	SINT	-90	Decimal
RESULT	SINT	-20	Decimal

The above example calculates the sum of VALUE 1 and VALUE2 variables, storing the final result in RESULT. Notice that the block accepts arguments of both signs.



VALUE1	SINT	-80	Decimal
VALUE2	SINT	-90	Decimal
RESULT	SINT	-20	Decimal

The above example calculates the sum of VALUE1 and VALUE2 variables. The final result -170 cannot be stored in RESULT, because it is outside the limits of accepted values by SINT type. Therefore, RESULT remains unchanged and the output is disabled.

11.2.6.11.1.3 DIV

Block that calculates the division of the values of Value1 and Value2, storing the result in Result.

Ladder Representation

Block Structure

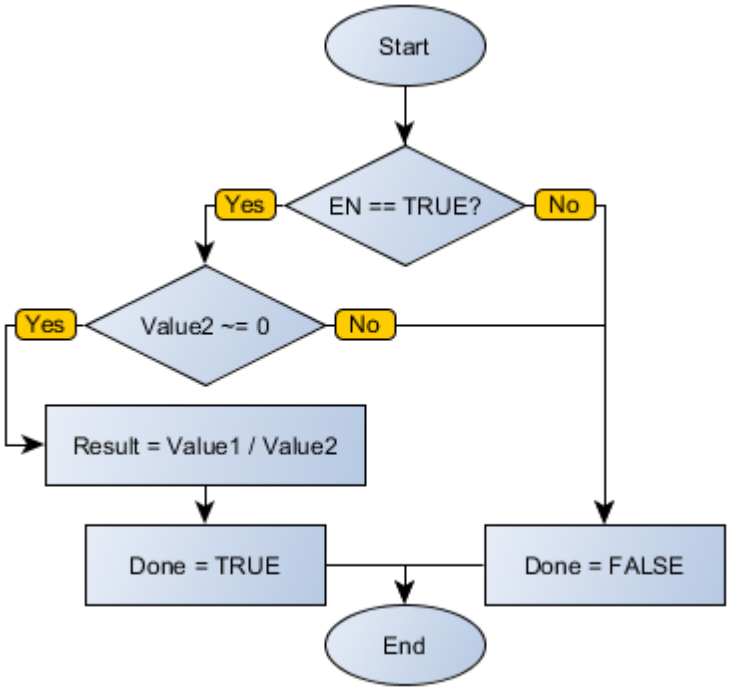
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Dividend of the operation
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Divisor of the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Variable that stores the result of the operation

Operation

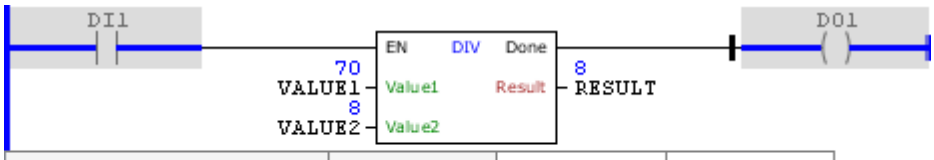
When this block has a TRUE value in EN, it sends to the Result output the division of Value1 and Value2 variables. The value stored will be the exact division if Result is REAL, or, in other cases, only the quotient. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

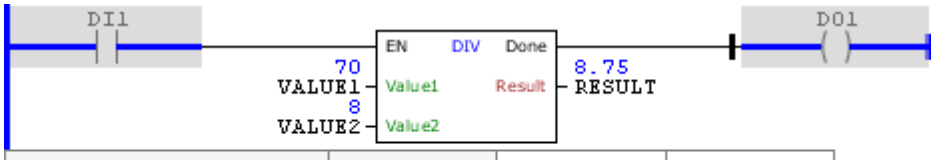


Example



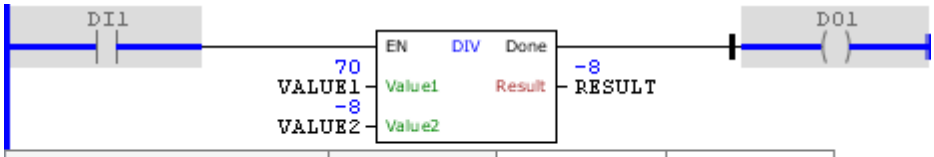
VALUE1	SINT	70	Decimal
VALUE2	SINT	8	Decimal
RESULT	SINT	8	Decimal

The above example calculates the division of VALUE 1 and VALUE2 variables, storing the final result in RESULT. Since RESULT is SINT type, only the quotient is stored in it.



VALUE1	SINT	70	Decimal
VALUE2	SINT	8	Decimal
RESULT	REAL	8.75	Float

The above example calculates the division of VALUE 1 and VALUE2 variables, storing the final result in RESULT. Since RESULT is of REAL type, the exact value of the division is stored in it.



VALUE1	SINT	70	Decimal
VALUE2	SINT	-8	Decimal
RESULT	SINT	-8	Decimal

The above example calculates the division of VALUE 1 and VALUE2 variables, storing the final result in RESULT. Since RESULT is SINT type, only the quotient is stored in it. Notice that the block accepts arguments of both signs.

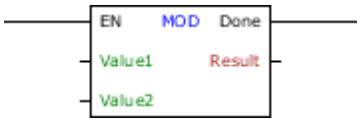
VALUE1	SINT	70	Decimal
VALUE2	SINT	0	Decimal
RESULT	SINT	-8	Decimal

The above example calculates the division of VALUE 1 and VALUE2 variables, storing the final result in RESULT. Since VALUE2 is zero, the block generates a runtime error, RESULT remains unchanged and the output is disabled.

11.2.6.11.1.4 MOD

Block that calculates the remainder of the values of Value1 and Value2, storing the result in Result.

Ladder Representation



Block Structure

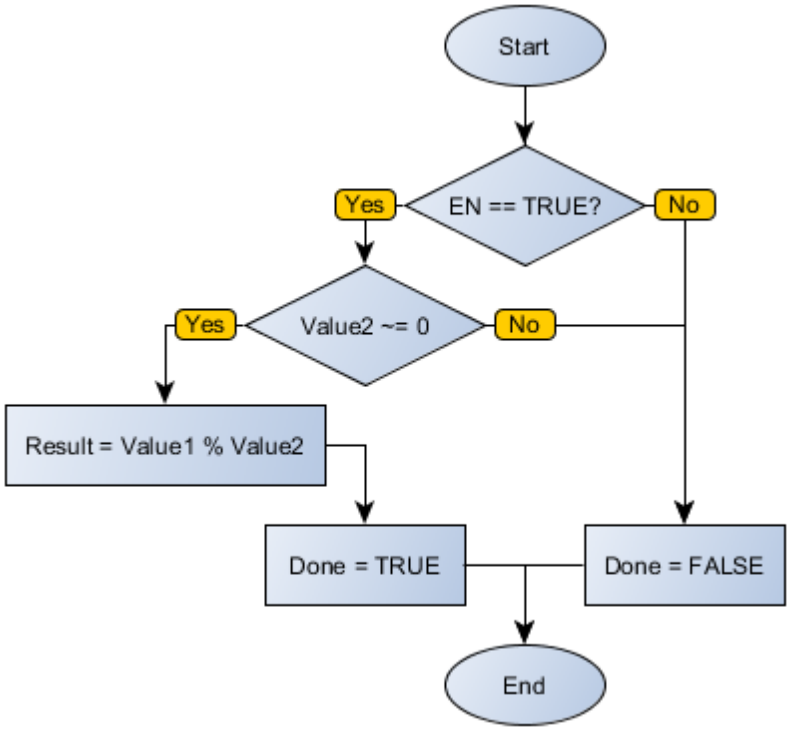
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Dividend of the operation
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Divisor of the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable that stores the result of the operation

Operation

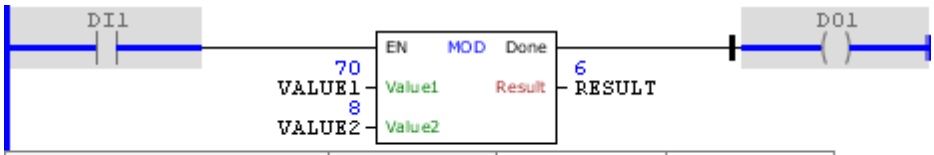
When this block has a TRUE value in EN, it sends to the Result output the remainder of Value1 and Value2 variables. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

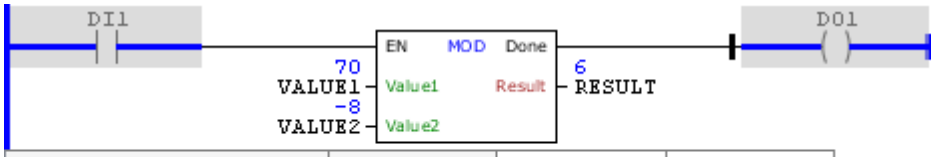


Example



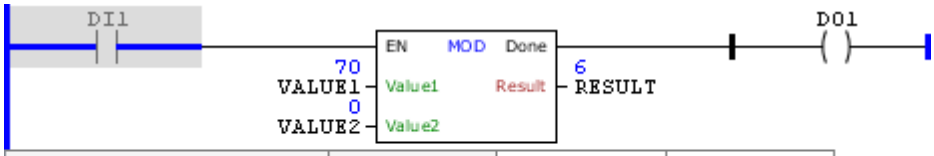
VALUE1	SINT	70	Decimal
VALUE2	SINT	8	Decimal
RESULT	SINT	6	Decimal

The above example calculates the remainder of VALUE 1 and VALUE2 variables, storing the final result in RESULT.



VALUE1	SINT	70	Decimal
VALUE2	SINT	-8	Decimal
RESULT	SINT	6	Decimal

The above example calculates the remainder of VALUE 1 and VALUE2 variables, storing the final result in RESULT. Notice that the block accepts arguments of both signs.



VALUE1	SINT	70	Decimal
VALUE2	SINT	0	Decimal
RESULT	SINT	6	Decimal

The above example calculates the remainder of VALUE 1 and VALUE2 variables, storing the final result in RESULT. Since VALUE2 is zero, the block generates a runtime error, RESULT remains unchanged and the output is disabled.

11.2.6.11.1.5 MUL

Block that calculates the multiplication of the values of Value1 and Value2, storing the result in Result.

Ladder Representation

Block Structure

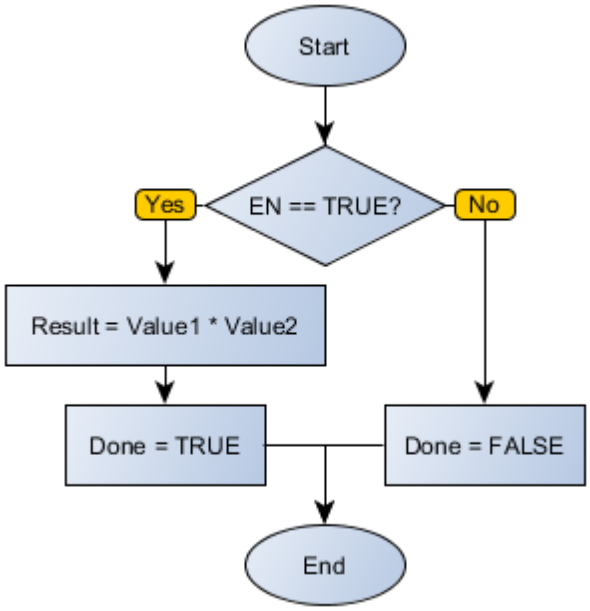
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	First factor of the operation
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Second factor of the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Variable that stores the result of the operation

Operation

When this block has a TRUE value in EN, it sends to the Result output the multiplication of Value1 and Value2 variables. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

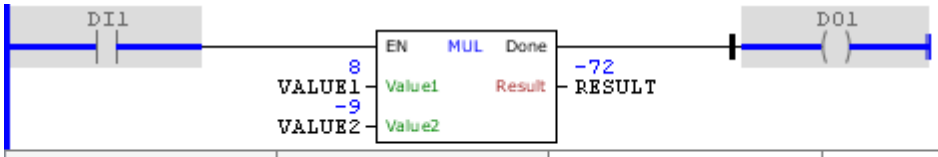
Block Flowchart



Example

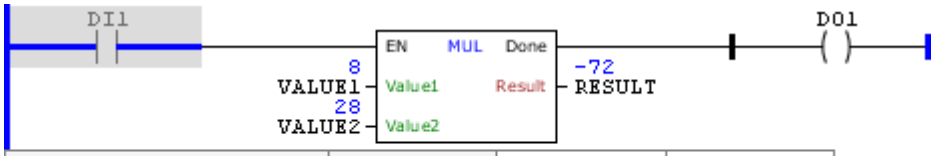
VALUE1	SINT	8	Decimal
VALUE2	SINT	12	Decimal
RESULT	SINT	96	Decimal

The above example calculates the product of VALUE 1 and VALUE2 variables, storing the final result in RESULT.



VALUE1	SINT	8	Decimal
VALUE2	SINT	-9	Decimal
RESULT	SINT	-72	Decimal

The above example calculates the product of VALUE 1 and VALUE2 variables, storing the final result in RESULT. Notice that the block accepts arguments of both signs.



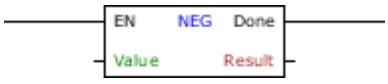
VALUE1	SINT	8	Decimal
VALUE2	SINT	28	Decimal
RESULT	SINT	-72	Decimal

The above example calculates the product of VALUE1 and VALUE2 variables. The final result 224 cannot be stored in RESULT, because it is outside the limits of accepted values by SINT type. Therefore, RESULT remains unchanged and the output is disabled.

11.2.6.11.1.6 NEG

Block that calculates the opposite (i.e., the product with -1) of a value passed by Value, storing the result in Result.

Ladder Representation



Block Structure

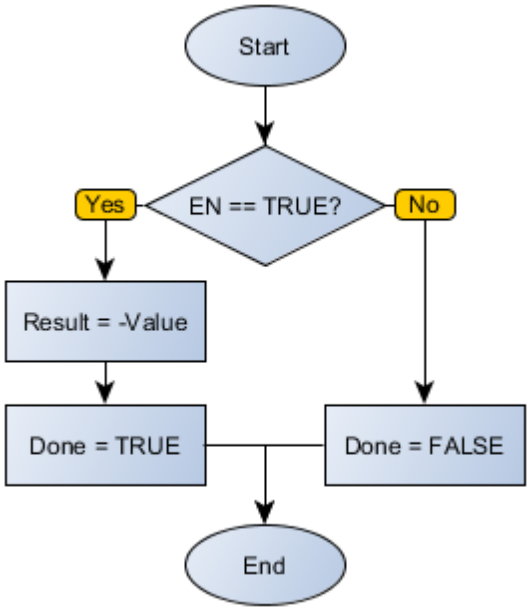
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Reference variable for the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Variable that stores the result of the operation

Operation

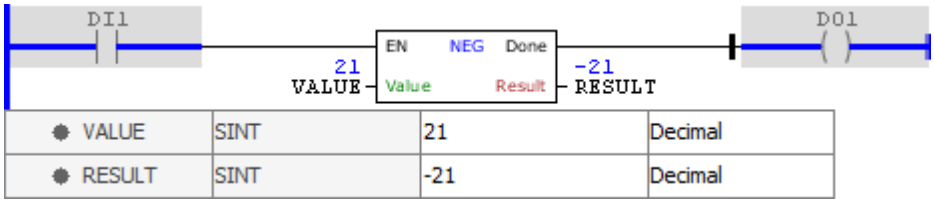
When this block has a TRUE value in EN, it sends to the Result output the opposite of the Value variable. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

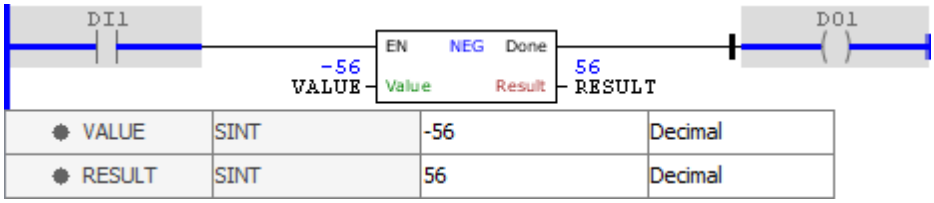


Example



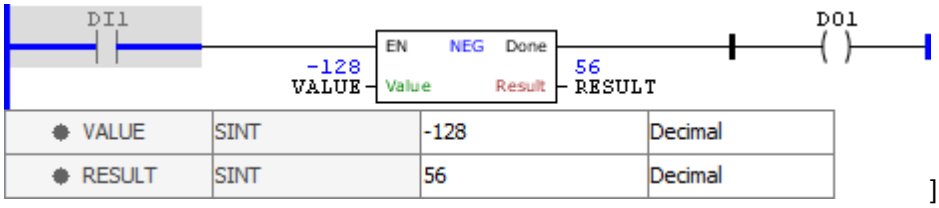
VALUE	SINT	21	Decimal
RESULT	SINT	-21	Decimal

The above example calculates the opposite of the VALUE variable whose initial value is 21, storing the final result, -21, in RESULT.



VALUE	SINT	-56	Decimal
RESULT	SINT	56	Decimal

The above example calculates the opposite of the VALUE variable whose initial value is -56, storing the final result, 56, in RESULT.



VALUE	SINT	-128	Decimal
RESULT	SINT	56	Decimal

]

The above example calculates the opposite of the VALUE variable whose initial value is -128. The final result, 128, cannot be stored in RESULT, because it is outside the limits of accepted values by SINT type. Therefore, RESULT remains unchanged and the output is disabled.

11.2.6.11.1.7 SUB

Block that calculates the subtraction between the Value1 and Value2 values, storing the result in Result.

Ladder Representation

Block Structure

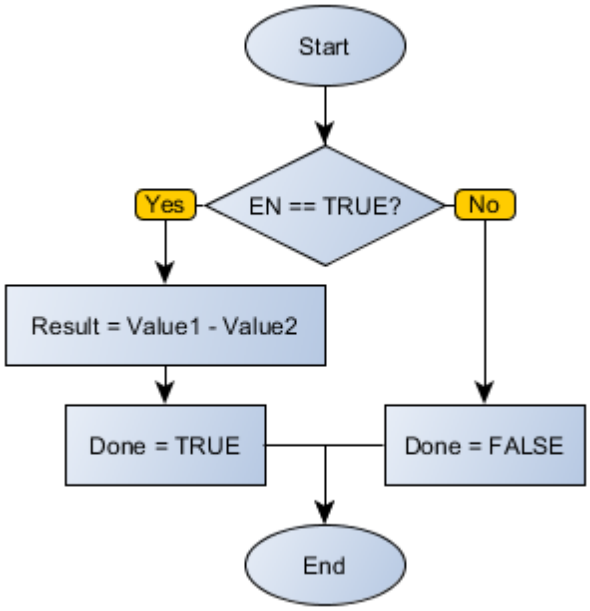
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Minuend of operation
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Subtrahend of operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Variable that stores the result of the operation

Operation

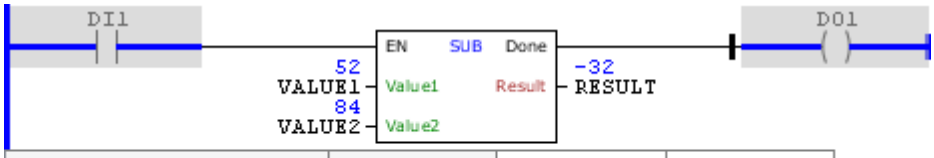
When this block has a TRUE value in EN, it sends to the Result output the subtraction of Value1 and Value2 variables. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

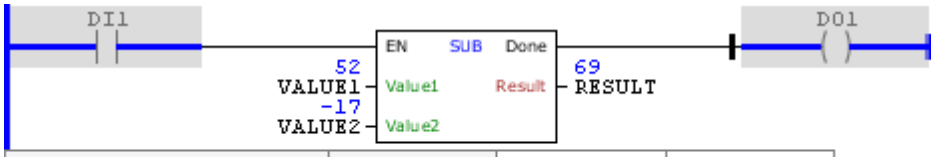


Example



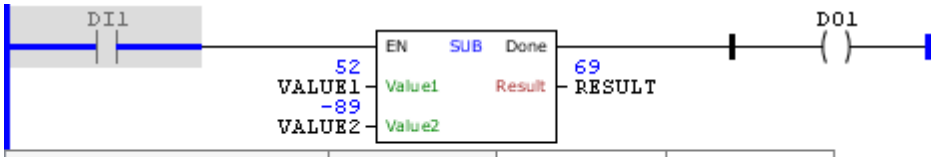
VALUE1	SINT	52	Decimal
VALUE2	SINT	84	Decimal
RESULT	SINT	-32	Decimal

The above example calculates the subtraction of VALUE 1 and VALUE2 variables, storing the final result in RESULT.



VALUE1	SINT	52	Decimal
VALUE2	SINT	-17	Decimal
RESULT	SINT	69	Decimal

The above example calculates the subtraction of VALUE 1 and VALUE2 variables, storing the final result in RESULT. Notice that the block accepts arguments of both signs.



VALUE1	SINT	52	Decimal
VALUE2	SINT	-89	Decimal
RESULT	SINT	69	Decimal

The above example calculates the subtraction of VALUE1 and VALUE2 variables. The final result 141 cannot be stored in RESULT, because it is outside the limits of accepted values by SINT type. Therefore, RESULT remains unchanged and the output is disabled.

11.2.6.11.2 Math Extended

11.2.6.11.2.1 ALOG10

Block that calculates the antilogarithm (exponent with base 10) of the Value value, storing the result in Result.

Ladder Representation

Block Structure

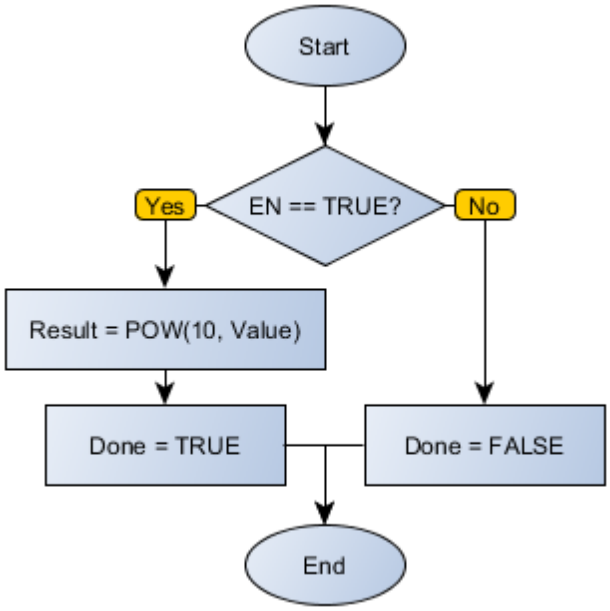
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Reference variable for the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	REAL	Variable that stores the result of the operation

Operation

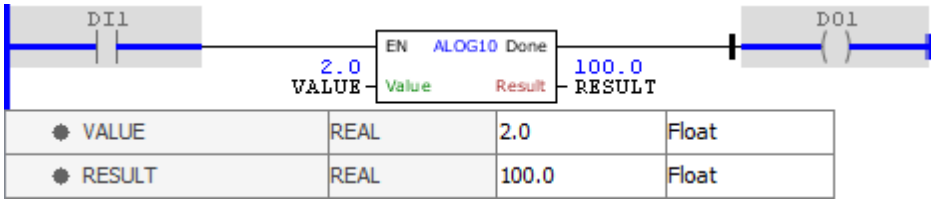
When this block has a TRUE value in EN, it sends to the Result output the antilogarithm of the Value variable. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

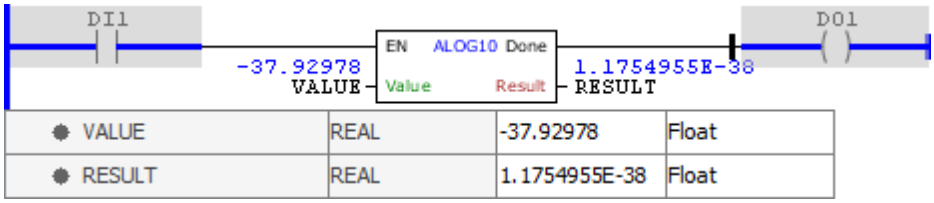


Example



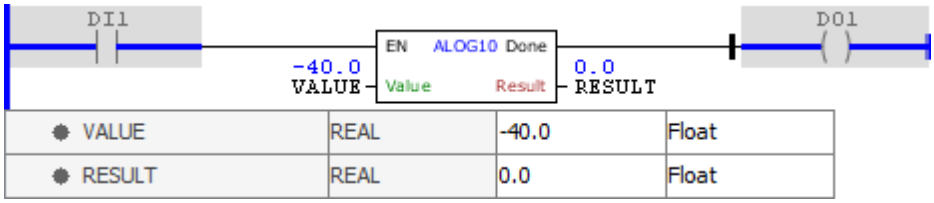
VALUE	REAL	2.0	Float
RESULT	REAL	100.0	Float

The above example calculates the antilogarithm of the VALUE variable, storing the final result in RESULT. The block ends with success and Done output is activated.



VALUE	REAL	-37.92978	Float
RESULT	REAL	1.1754955E-38	Float

The above example calculates the antilogarithm of the VALUE variable, storing the final result in RESULT. The indicated value is the minimum input value for which the block revolutions a nonzero result. The block ends with success and Done output is activated.



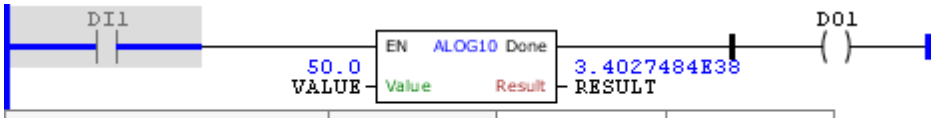
VALUE	REAL	-40.0	Float
RESULT	REAL	0.0	Float

The above example calculates the antilogarithm of the VALUE variable, storing the final result in

RESULT. Below the minimum values cause the block to return a null value. The block ends with success and Done output is activated.

VALUE	REAL	38.53183	Float
RESULT	REAL	3.4027484E38	Float

The above example calculates the antilogarithm of the VALUE variable, storing the final result in RESULT. The indicated value is the maximum input value for which the block revolutions a valid result. The block ends with success and Done output is activated.



VALUE	REAL	50.0	Float
RESULT	REAL	3.4027484E38	Float

The above example calculates the antilogarithm of the VALUE variable, storing the final result in RESULT. Values higher than the maximum cause the block to generate an error, the RESULT output remains unchanged and Done output is disabled.

11.2.6.11.2.2 EXP

Block that calculates the exponential of the Euler number "and" raised to the value of Value, storing the result in Result.

Ladder Representation

Block Structure

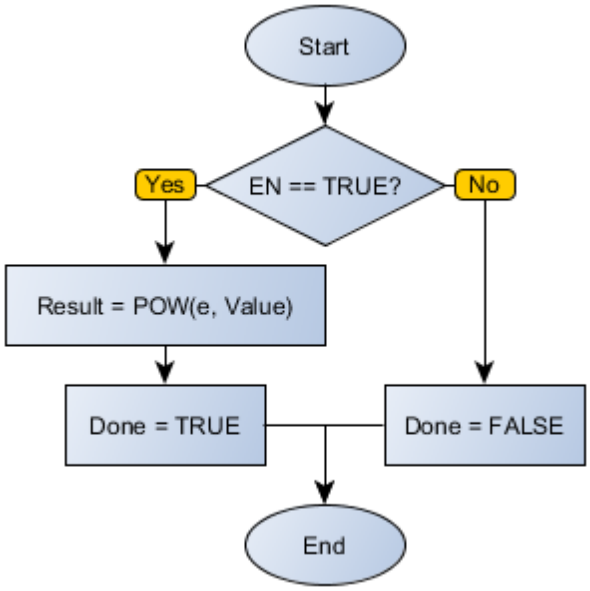
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Reference variable for the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	REAL	Variable that stores the result of the operation

Operation

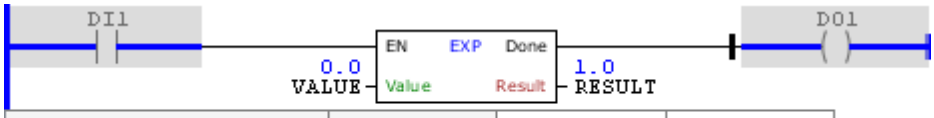
When this block has a TRUE value in EN, it sends to the Result output the exponent of the Euler number "and" raised to the Value variable. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart



Example



VALUE	REAL	0.0	Float
RESULT	REAL	1.0	Float

The above example calculates the exponent of the VALUE variable, storing the final result in RESULT. The block ends with success and Done output is activated.

VALUE	REAL	-87.33654	Float
RESULT	REAL	1.1754997E-38	Float

The above example calculates the exponent of the VALUE variable, storing the final result in RESULT. The indicated value is the minimum input value for which the block revolutions a nonzero result. The block ends with success and Done output is activated.

VALUE	REAL	-50000.0	Float
RESULT	REAL	0.0	Float

The above example calculates the exponent of the VALUE variable, storing the final result in RESULT. Values below the minimum cause the block to return to a null value. The block ends with success and Done output is activated.

VALUE	REAL	88.72283	Float
RESULT	REAL	3.4027985E38	Float

The above example calculates the exponent of the VALUE variable, storing the final result in RESULT. The indicated value is the maximum input value for which the block revolutions a valid result. The block ends with success and Done output is activated.

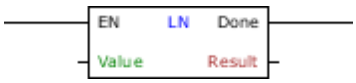
VALUE	REAL	90.0	Float
RESULT	REAL	3.4027985E38	Float

The above example calculates the exponent of the VALUE variable, storing the final result in RESULT. Values higher than the maximum cause the block to generate an error, the RESULT output remains unchanged and Done output is disabled.

11.2.6.11.2.3 LN

Block that calculates the natural logarithm of the Value value, storing the result in Result.

Ladder Representation



Block Structure

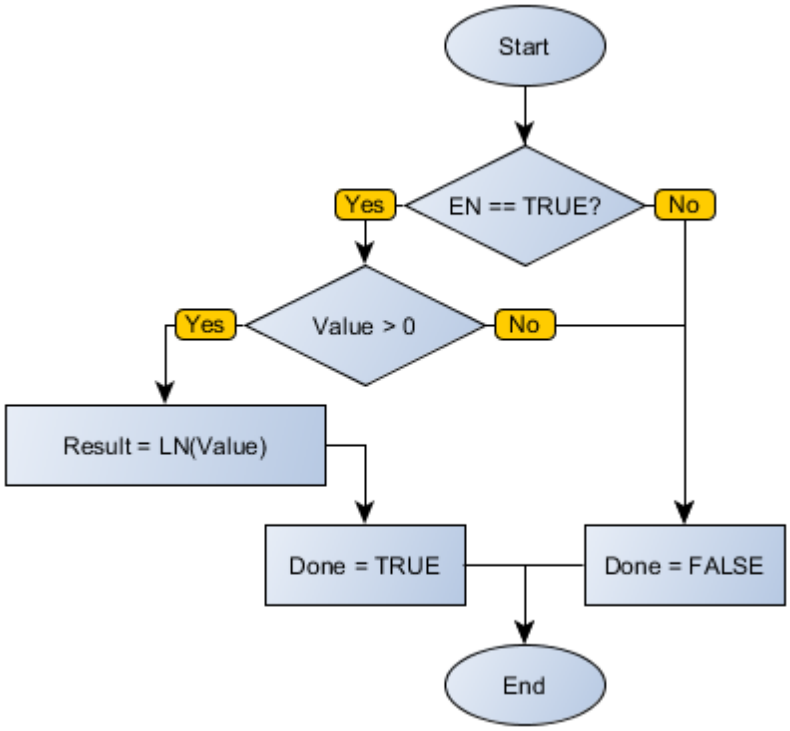
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Reference variable for the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	REAL	Variable that stores the result of the operation

Operation

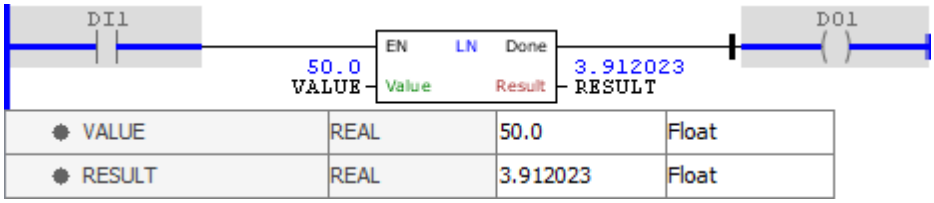
When this block has a TRUE value in EN, it sends to the Result output the natural logarithm of the Value variable. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart



Example



VALUE	REAL	50.0	Float
RESULT	REAL	3.912023	Float

The above example calculates the natural logarithm of the VALUE variable, storing the final result in RESULT. The block ends with success and Done output is activated.

VALUE	REAL	0.0	Float
RESULT	REAL	3.912023	Float

The above example calculates the natural logarithm of the VALUE variable, storing the final result in RESULT. The block generates a runtime error, since VALUE has value zero, and Done output is disabled.

11.2.6.11.2.4 LOG10

Block that calculates the common logarithm (base 10) of the Value value, storing the result in Result.

Ladder Representation

Block Structure

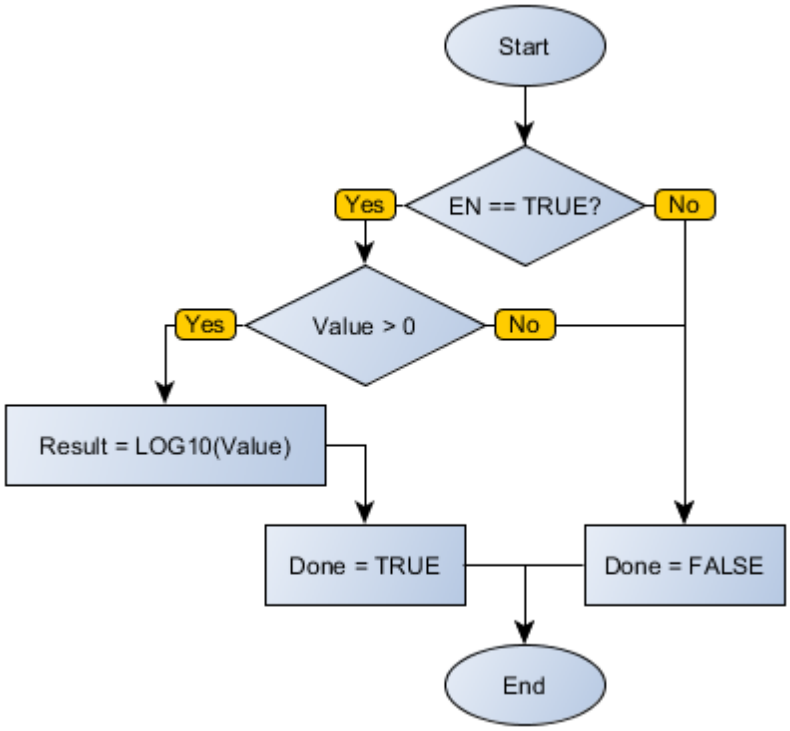
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Reference variable for the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	REAL	Variable that stores the result of the operation

Operation

When this block has a TRUE value in EN, it sends to the Result output the common logarithm of the Value variable. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart



Example

VALUE	REAL	1000.0	Float
RESULT	REAL	3.0	Float

The above example calculates the common logarithm of the VALUE variable, storing the final result in RESULT. The block ends with success and Done output is activated.

VALUE	REAL	-5.0	Float
RESULT	REAL	3.0	Float

The above example calculates the common logarithm of the VALUE variable, storing the final result in RESULT. The block generates a runtime error, since VALUE has negative value, and Done output is disabled.

11.2.6.11.2.5 POW

Block that calculates the value of Value raised to the exponent Power, storing the result in Result.

Ladder Representation

Block Structure

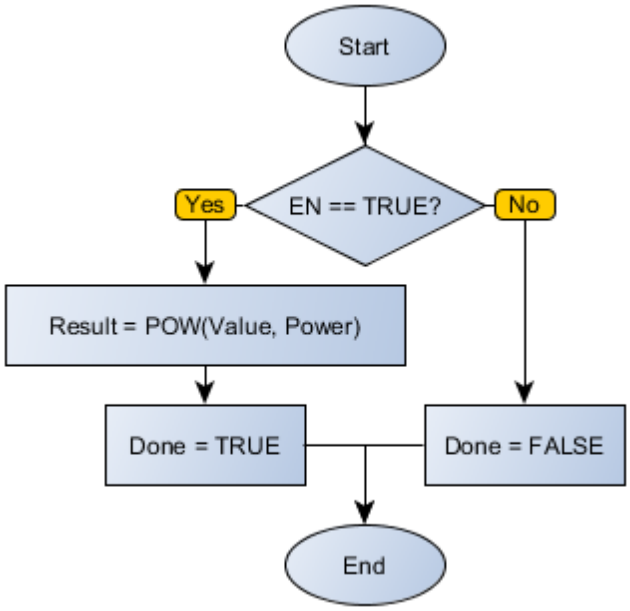
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Base of the operation
	Power	REAL	Exponent of the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	REAL	Variable that stores the result of the operation

Operation

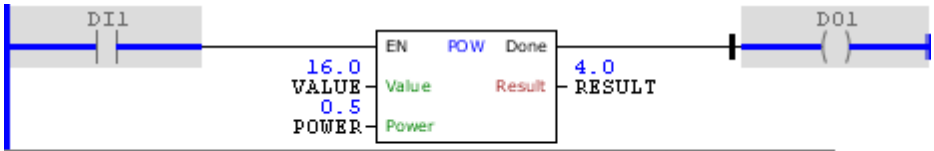
When this block has a TRUE value in EN, it sends to the Result output the value of Value raised to the exponent Power. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

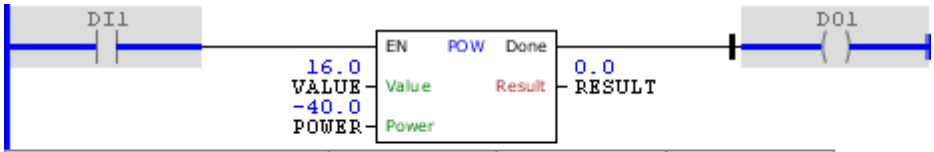


Example



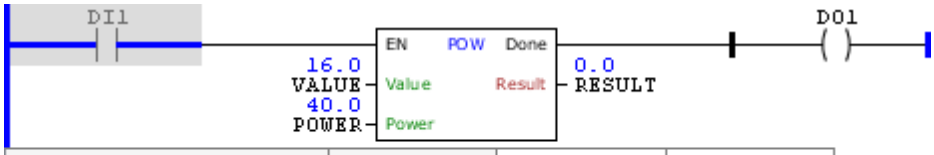
VALUE	REAL	16.0	Float
POWER	REAL	0.5	Float
RESULT	REAL	4.0	Float

The above example calculates the value of VALUE raised to the POWER variable, storing the final result in RESULT. The block ends with success and Done output is activated.



VALUE	REAL	16.0	Float
POWER	REAL	-40.0	Float
RESULT	REAL	0.0	Float

The above example calculates the value of VALUE raised to the POWER variable, storing the final result in RESULT. The block ends with success and Done output is activated.



VALUE	REAL	16.0	Float
POWER	REAL	40.0	Float
RESULT	REAL	0.0	Float

The above example calculates the value of VALUE raised to the POWER variable, storing the final result in RESULT. Since the result is higher than the maximum supported by REAL type, the block generates an error and Done output is disabled.

11.2.6.11.2.6 ROUND

Block that rounds the value of Value, storing the result in Result.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Reference variable for the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	REAL	Variable that stores the result of the operation

Operation

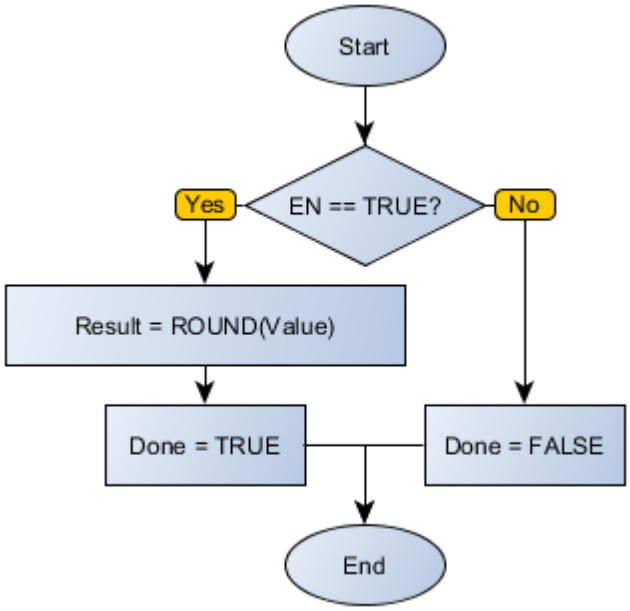
When this block has a TRUE value in EN, it sends to the Result output the rounded value of Value. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

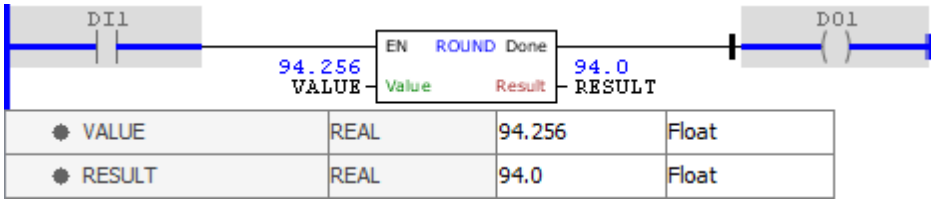
Compatibility

Device	Version
PLC300	2.10 or higher
SCA06	2.00 or higher

Block Flowchart

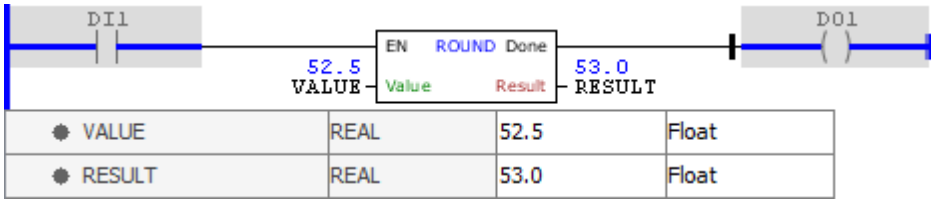


Example



VALUE	REAL	94.256	Float
RESULT	REAL	94.0	Float

The above example rounds the value of the VALUE variable, storing the final result in RESULT. Decimals less than 0.5 are discarded. The block ends with success and Done output is activated.



VALUE	REAL	52.5	Float
RESULT	REAL	53.0	Float

The above example rounds the value of the VALUE variable, storing the final result in RESULT. Decimals greater than or equal to 0.5 promote unity value immediately above. The block ends with success and Done output is activated.

11.2.6.11.2.7 SQRT

Block that calculates the square root value of Value, storing the result in Result.

Ladder Representation

Block Structure

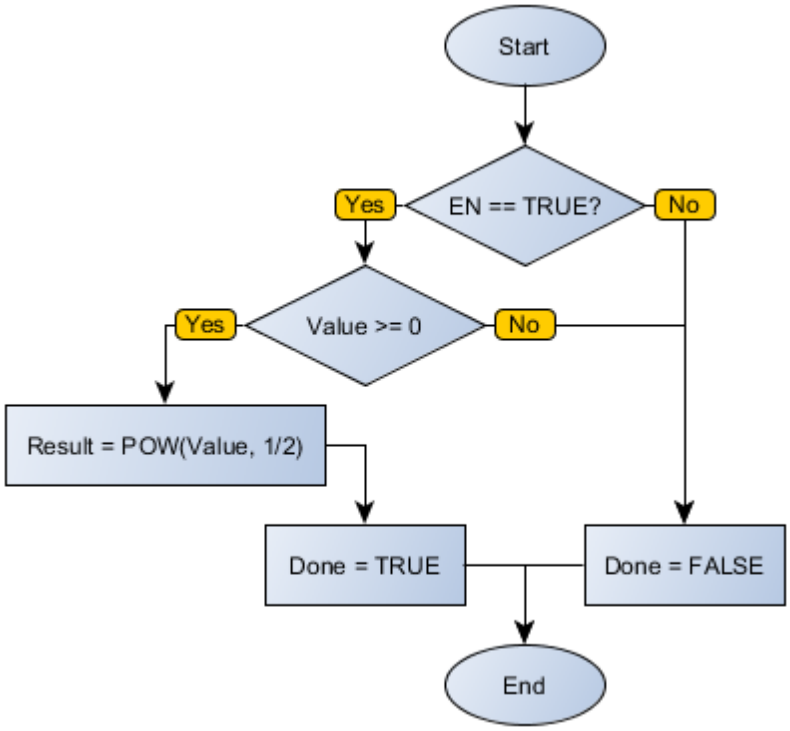
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Reference variable for the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	REAL	Variable that stores the result of the operation

Operation

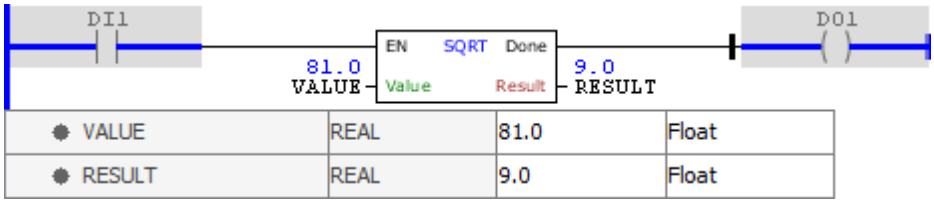
When this block has a TRUE value in EN, it sends to the Result output the square root value of Value. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

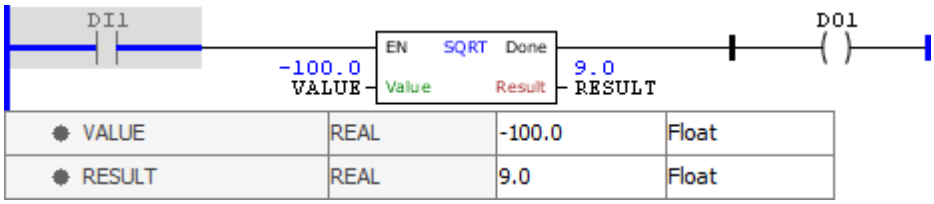


Example



VALUE	REAL	81.0	Float
RESULT	REAL	9.0	Float

The above example calculates the square root value of the VALUE variable, storing the final result in RESULT. The block ends with success and Done output is activated.



VALUE	REAL	-100.0	Float
RESULT	REAL	9.0	Float

The above example calculates the square root value of the VALUE variable, storing the final result in RESULT. The block generates a runtime error, since VALUE has negative value, and Done output is disabled.

11.2.6.11.2.8 TRUNC

Block that truncates the value of Value, storing the result in Result.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Reference variable for the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	REAL	Variable that stores the result of the operation

Operation

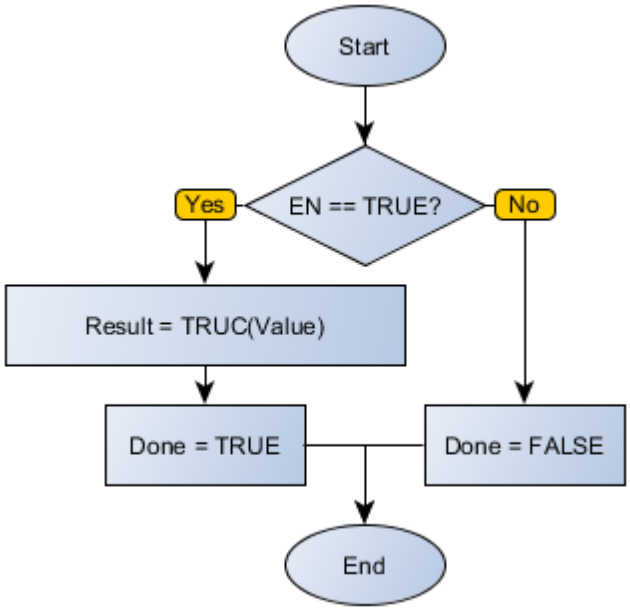
When this block has a TRUE value in EN, it sends to the Result output the truncated value of Value. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

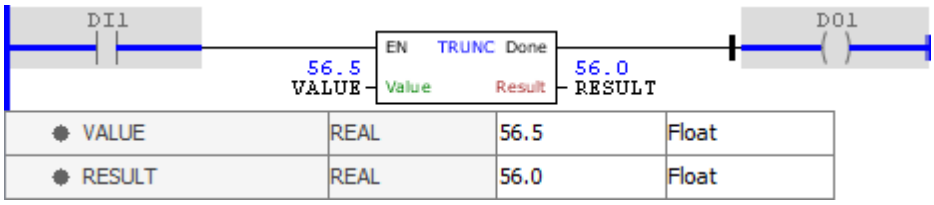
Compatibility

Device	Version
PLC300	2.10 or higher
SCA06	2.00 or higher

Block Flowchart



Example



VALUE	REAL	56.5	Float
RESULT	REAL	56.0	Float

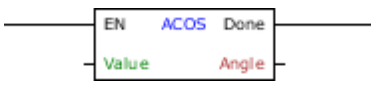
The above example truncates the value of the VALUE variable, storing the final result in RESULT. Decimals are discarded. The block ends with success and Done output is activated.

11.2.6.11.3 Math Trigonometry

11.2.6.11.3.1 ACOS

Block that calculates the arccosine of Value, storing the result in Angle.

Ladder Representation



Block Structure

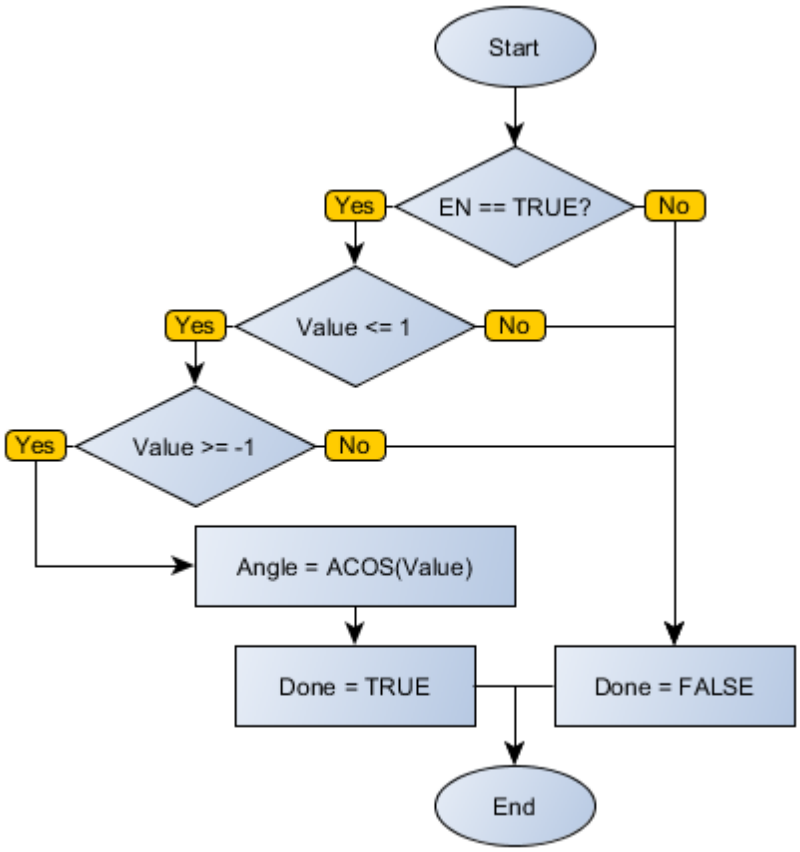
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Value of cosine
VAR_OUTPUT	Done	BOOL	End of operation
	Angle	REAL	Value of the angle w hose cosine is equal to Value (in radians)

Operation

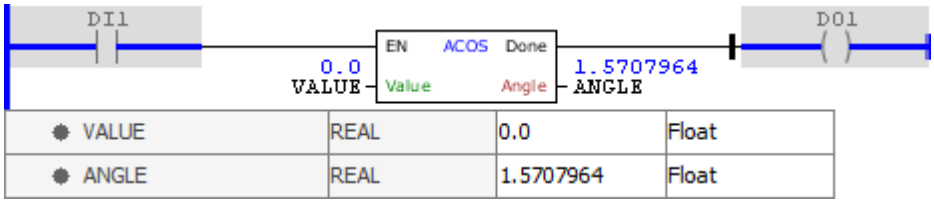
When this block has a TRUE value in EN, it sends to the Angle output the arccosine of Value. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Angle remains with its value unchanged.

When EN has FALSE value, Angle remains unchanged and Done remains in FALSE.

Block Flowchart

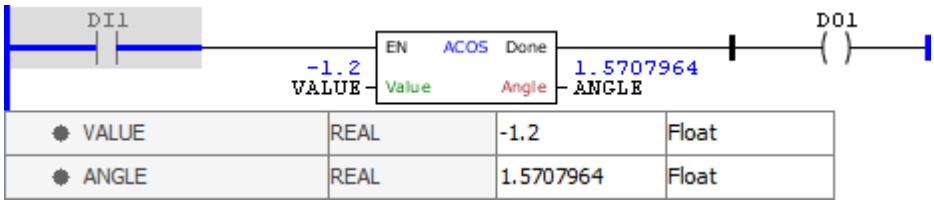


Example



VALUE	REAL	0.0	Float
ANGLE	REAL	1.5707964	Float

The above example calculates the arc, in radians, whose cosine is the VALUE variable, storing the final result in RESULT. The block ends with success and Done output is activated.



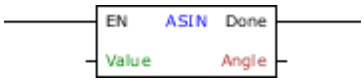
VALUE	REAL	-1.2	Float
ANGLE	REAL	1.5707964	Float

The above example calculates the arc, in radians, whose cosine is the VALUE variable, storing the final result in RESULT. The block generates a runtime error, since VALUE has value inferior to 1, and Done output is disabled.

11.2.6.11.3.2 ASIN

Block that calculates the arcsine of Value, storing the result in Angle.

Ladder Representation



Block Structure

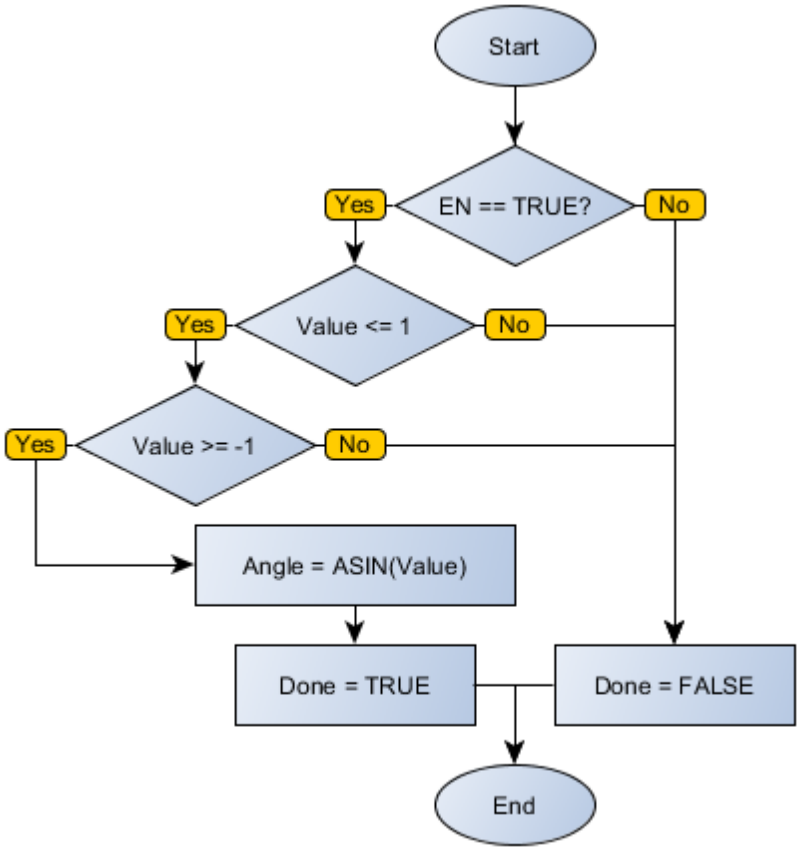
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Value of sine
VAR_OUTPUT	Done	BOOL	End of operation
	Angle	REAL	Value of the angle whose sine is equal to Value (in radians)

Operation

When this block has a TRUE value in EN, it sends to the Angle output the arcsine of Value. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Angle remains with its value unchanged.

When EN has FALSE value, Angle remains unchanged and Done remains in FALSE.

Block Flowchart



Example

VALUE	REAL	-1.0	Float
ANGLE	REAL	-1.5707964	Float

The above example calculates the arc, in radians, whose sine is the VALUE variable, storing the final result in RESULT. The block ends with success and Done output is activated.

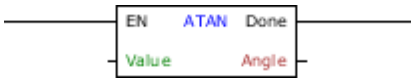
VALUE	REAL	2.0	Float
ANGLE	REAL	-1.5707964	Float

The above example calculates the arc, in radians, whose sine is the VALUE variable, storing the final result in RESULT. The block generates a runtime error, since VALUE has value superior to 1, and Done output is disabled.

11.2.6.11.3.3 ATAN

Block that calculates the arctangent of Value, storing the result in Angle.

Ladder Representation



Block Structure

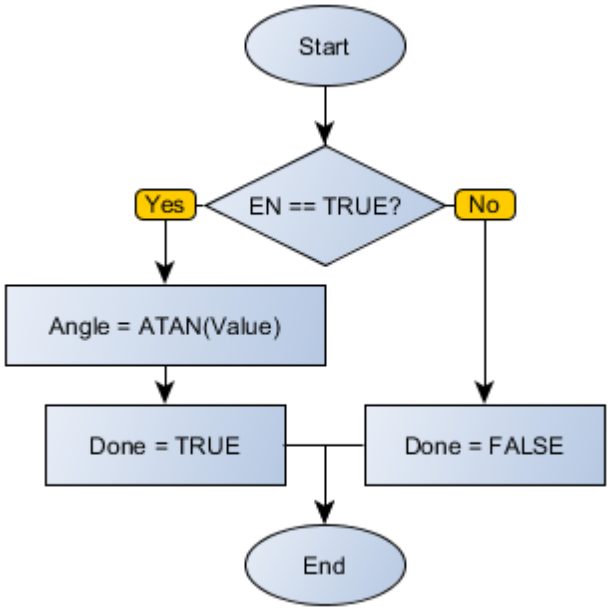
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Value of tangent
VAR_OUTPUT	Done	BOOL	End of operation
	Angle	REAL	Value of the angle whose tangent is equal to Value (in radians)

Operation

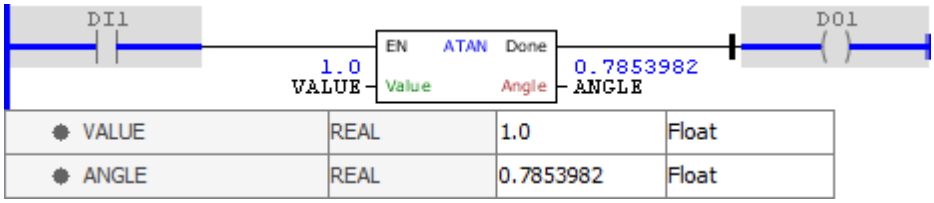
When this block has a TRUE value in EN, it sends to the Angle output the arctangent of Value. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Angle remains with its value unchanged.

When EN has FALSE value, Angle remains unchanged and Done remains in FALSE.

Block Flowchart

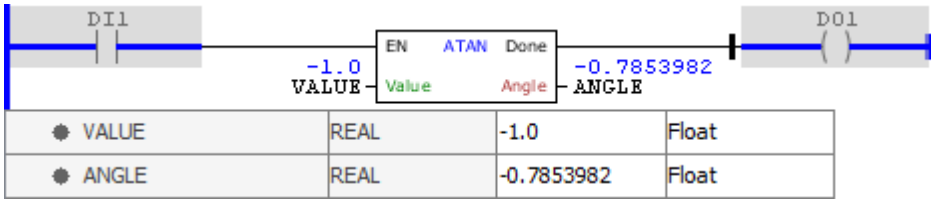


Example



VALUE	REAL	1.0	Float
ANGLE	REAL	0.7853982	Float

The above example calculates the arc, in radians, whose tangent is the VALUE variable, storing the final result in RESULT. The arc, for positive values, is always in the first quadrant. The block ends with success and Done output is activated.



VALUE	REAL	-1.0	Float
ANGLE	REAL	-0.7853982	Float

The above example calculates the arc, in radians, whose tangent is the VALUE variable, storing the final result in RESULT. The arc, for negative values, is always in the fourth quadrant. The block ends with success and Done output is activated.

11.2.6.11.3.4 ATAN2

Block that calculates the arctangent of Y/X, storing the result in Angle.

Ladder Representation

Block Structure

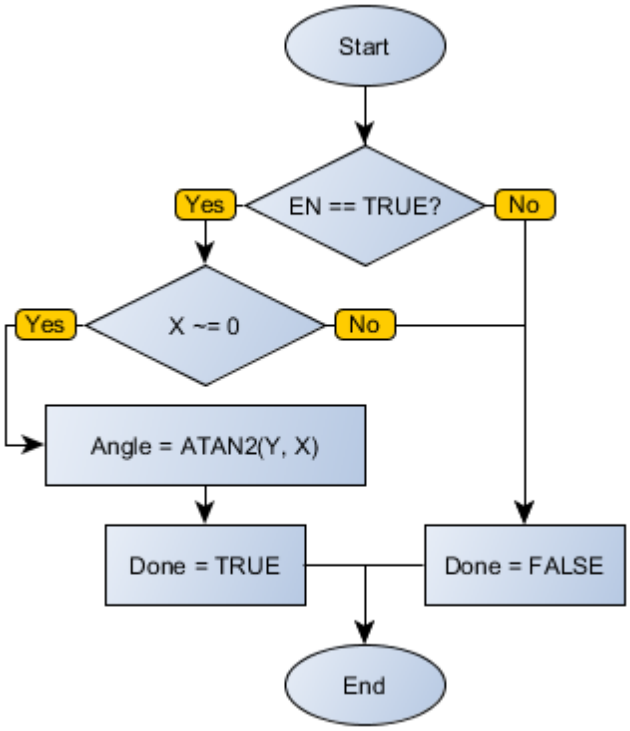
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	X	REAL	Parameter X of the function
	Y	REAL	Parameter Y of the function
VAR_OUTPUT	Done	BOOL	End of operation
	Angle	REAL	Value of the angle whose tangent is equal to (Y/X) (in radians)

Operation

When this block has a TRUE value in EN, it sends to the Angle output the arctangent of Y/X. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Angle remains with its value unchanged.

When EN has FALSE value, Angle remains unchanged and Done remains in FALSE.

Block Flowchart



Example

X	REAL	1.0	Float
Y	REAL	1.0	Float
ANGLE	REAL	0.7853982	Float

The above example calculates the arc, in radians, whose tangent is the Y/X variable, storing the final result in RESULT. The arc, for positive values of X and Y, is always in the first quadrant. The block ends with success and Done output is activated.

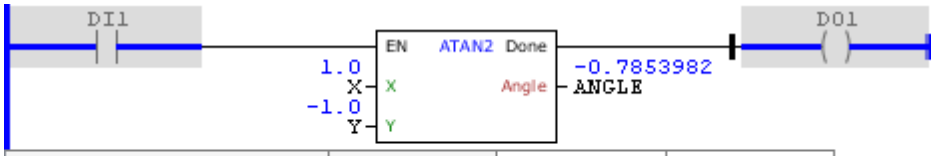
X	REAL	-1.0	Float
Y	REAL	1.0	Float
ANGLE	REAL	2.3561945	Float

The above example calculates the arc, in radians, whose tangent is the Y/X variable, storing the final

result in RESULT. The arc, for negative values of X and positive values of Y, is always in the second quadrant. The block ends with success and Done output is activated.

X	REAL	-1.0	Float
Y	REAL	-1.0	Float
ANGLE	REAL	-2.3561945	Float

The above example calculates the arc, in radians, whose tangent is the Y/X variable, storing the final result in RESULT. The arc, for negative values of X and Y, is always in the third quadrant. The block ends with success and Done output is activated.



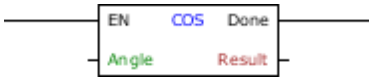
X	REAL	1.0	Float
Y	REAL	-1.0	Float
ANGLE	REAL	-0.7853982	Float

The above example calculates the arc, in radians, whose tangent is the Y/X variable, storing the final result in RESULT. The arc, for positive values of X and negative values of Y, is always in the fourth quadrant. The block ends with success and Done output is activated.

11.2.6.11.3.5 COS

Block that calculates the cosine of Angle, storing the result in Result.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Angle	REAL	Angle (in radians)
VAR_OUTPUT	Done	BOOL	End of operation
	Result	REAL	Variable that stores the result of the operation

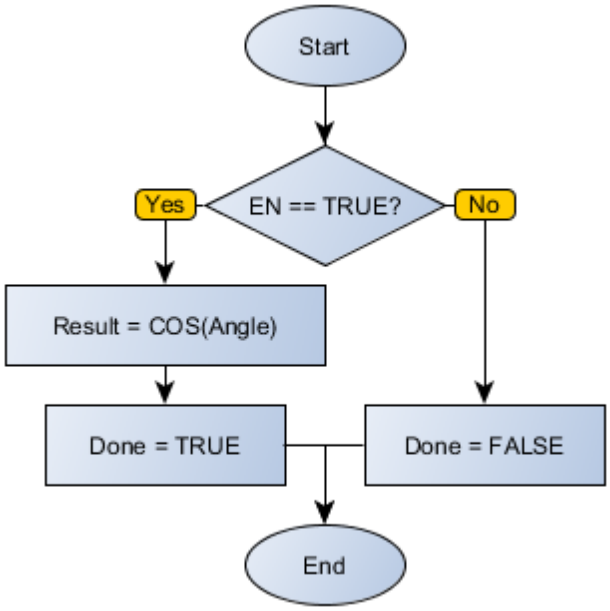
Operation

When this block has a TRUE value in EN, it sends to the Result output the cosine of Angle. If no

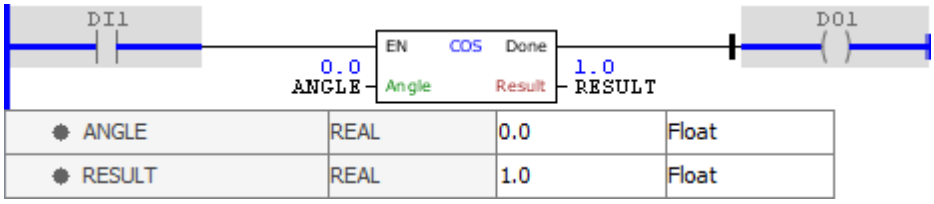
errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

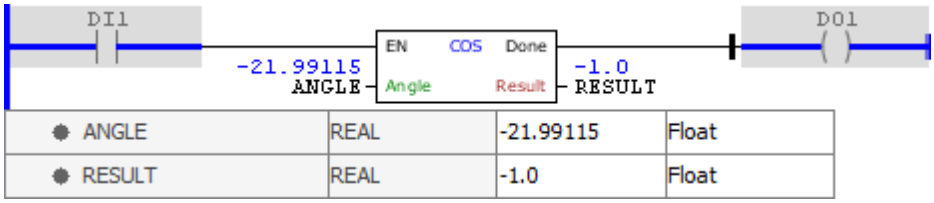


Example



ANGLE	REAL	0.0	Float
RESULT	REAL	1.0	Float

The above example calculates the cosine of the VALUE variable, interpreted in radians, storing the final result in RESULT. The block ends with success and Done output is activated.



ANGLE	REAL	-21.99115	Float
RESULT	REAL	-1.0	Float

The above example calculates the cosine of the VALUE variable, interpreted in radians, storing the final result in RESULT. The block ends with success and Done output is activated. Notice that the block accepts negative input values and greater than one turn.

11.2.6.11.3.6 SIN

Block that calculates the sine of Angle, storing the result in Result.

Ladder Representation

Block Structure

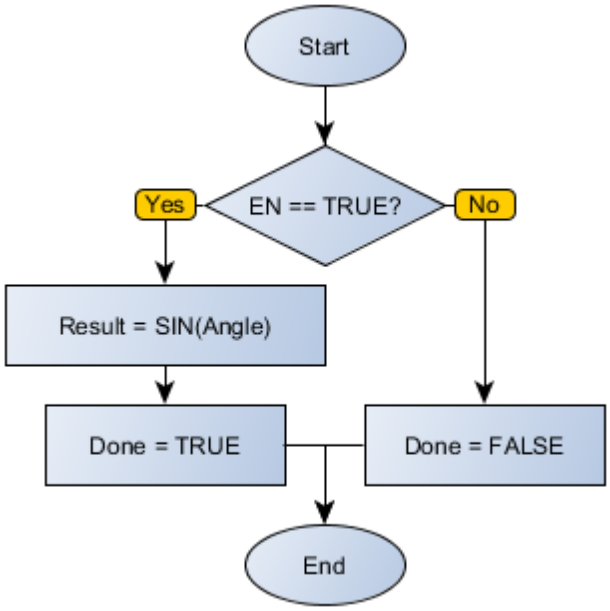
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Angle	REAL	Angle (in radians)
VAR_OUTPUT	Done	BOOL	End of operation
	Result	REAL	Variable that stores the result of the operation

Operation

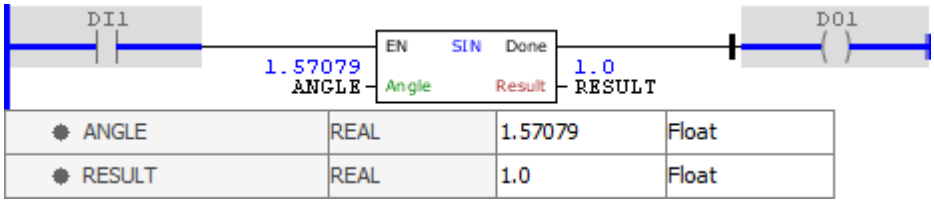
When this block has a TRUE value in EN, it sends to the Result output the sine of Angle. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

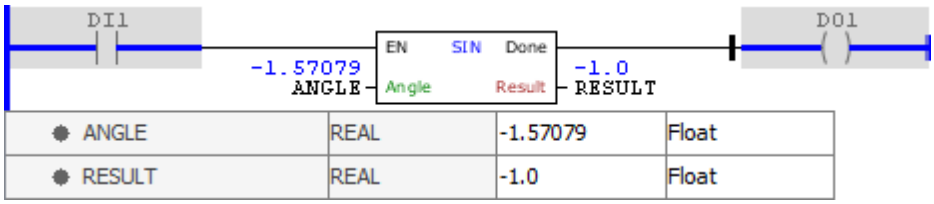


Example



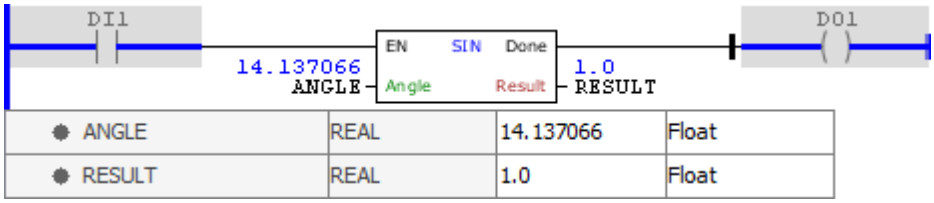
ANGLE	REAL	1.57079	Float
RESULT	REAL	1.0	Float

The above example calculates the sine of the VALUE variable, interpreted in radians, storing the final result in RESULT. The block ends with success and Done output is activated.



ANGLE	REAL	-1.57079	Float
RESULT	REAL	-1.0	Float

The above example calculates the sine of the VALUE variable, interpreted in radians, storing the final result in RESULT. The block ends with success and Done output is activated. Notice that the block accepts negative input values.



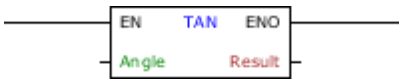
ANGLE	REAL	14.137066	Float
RESULT	REAL	1.0	Float

The above example calculates the sine of the VALUE variable, interpreted in radians, storing the final result in RESULT. The block ends with success and Done output is activated. Notice that the block accepts values greater than one full turn.

11.2.6.11.3.7 TAN

Block that calculates the tangent of Angle, storing the result in Result.

Ladder Representation



Block Structure

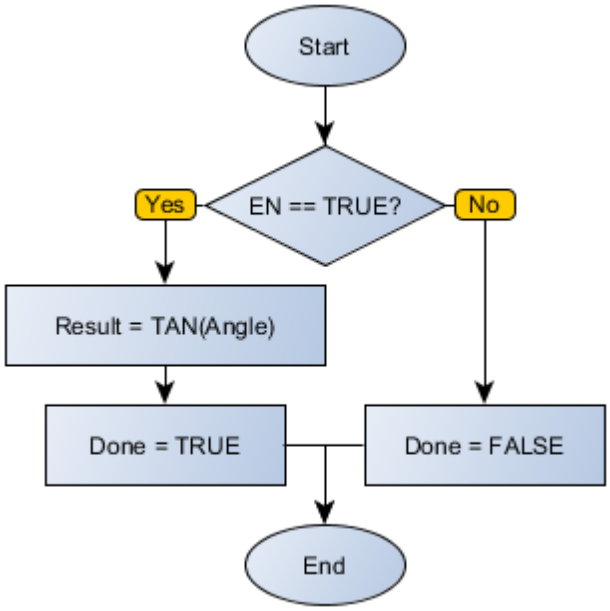
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Angle	REAL	Angle (in radians)
VAR_OUTPUT	Done	BOOL	End of operation
	Result	REAL	Variable that stores the result of the operation

Operation

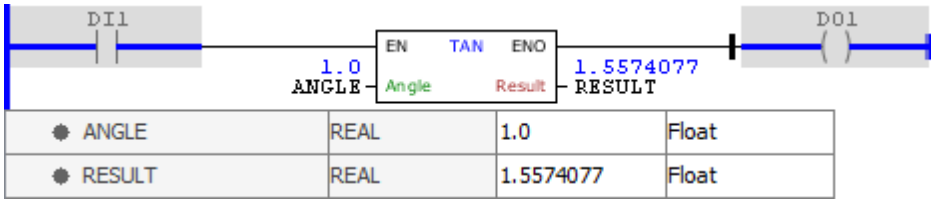
When this block has a TRUE value in EN, it sends to the Result output the tangent of Angle. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

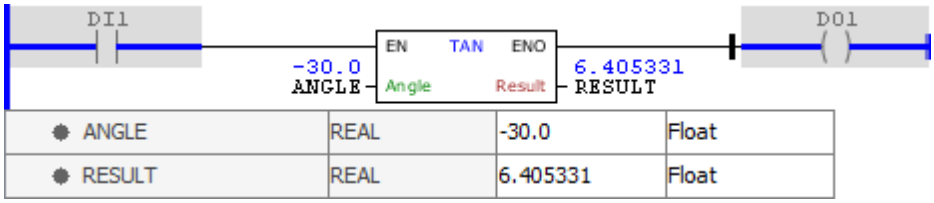


Example



ANGLE	REAL	1.0	Float
RESULT	REAL	1.5574077	Float

The above example calculates the tangent of the VALUE variable, interpreted in radians, storing the final result in RESULT. The block ends with success and Done output is activated.



ANGLE	REAL	-30.0	Float
RESULT	REAL	6.405331	Float

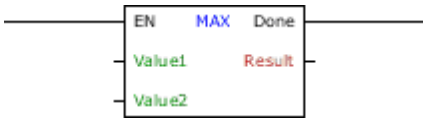
The above example calculates the tangent of the VALUE variable, interpreted in radians, storing the final result in RESULT. The block ends with success and Done output is activated. Notice that the block accepts negative input values and greater than one turn.

11.2.6.11.4 Math Util

11.2.6.11.4.1 MAX

Block that compares the values of Value1 and Value2 and stores the highest of them in Result.

Ladder Representation



Block Structure

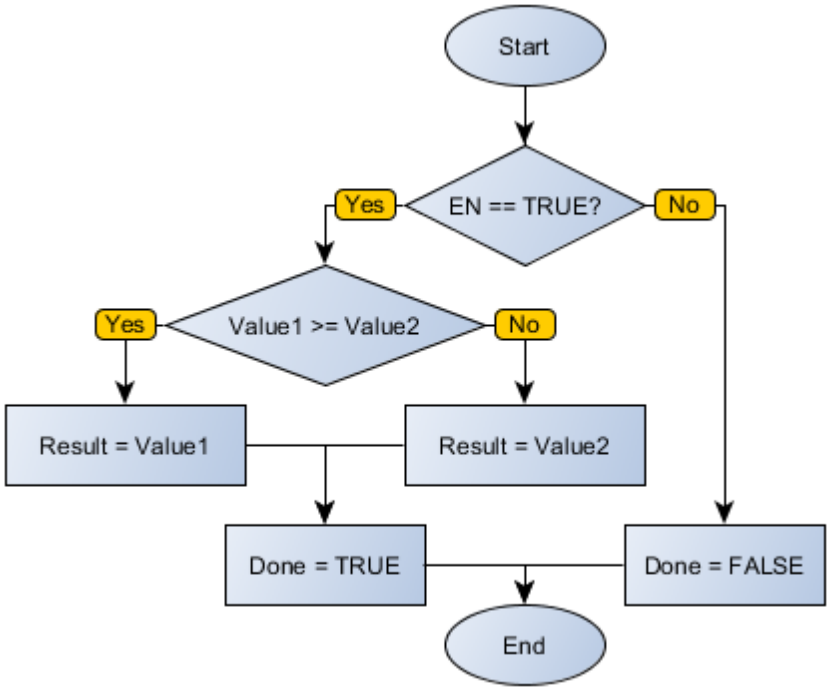
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	First value of comparison
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Second value of comparison
VAR_OUTPUT	Done	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Highest of the values compared

Operation

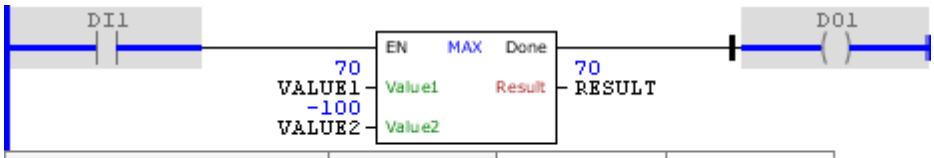
When this block has a TRUE value in EN, it sends to the Result output the highest value in the comparison between Value1 and Value2. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

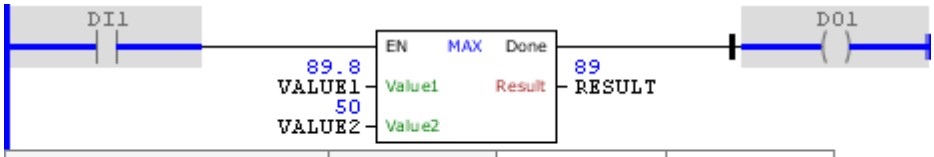


Example



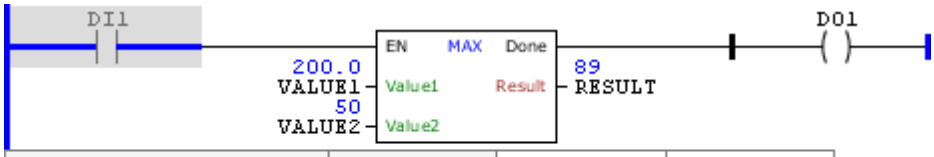
VALUE1	SINT	70	Decimal
VALUE2	SINT	-100	Decimal
RESULT	SINT	70	Decimal

The above example calculates the maximum value between VALUE 1 and VALUE2 variables, storing the final result in RESULT. The block ends with success and Done output is activated.



VALUE1	REAL	89.8	Float
VALUE2	SINT	50	Decimal
RESULT	SINT	89	Decimal

The above example calculates the maximum value between VALUE 1 and VALUE2 variables, storing the final result in RESULT. The block ends with success and Done output is activated. Notice that the types of the input variables can be different without causing execution problems.



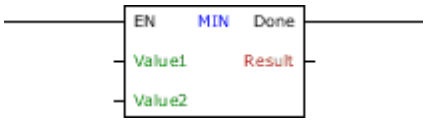
VALUE1	REAL	200.0	Float
VALUE2	SINT	50	Decimal
RESULT	SINT	89	Decimal

The above example calculates the maximum value between VALUE 1 and VALUE2 variables, storing the final result in RESULT. Since the result is higher than the maximum supported by SINT type, the block generates an error and Done output is disabled.

11.2.6.11.4.2 MIN

Block that compares the values of Value1 and Value2 and stores the lowest of them in Result.

Ladder Representation



Block Structure

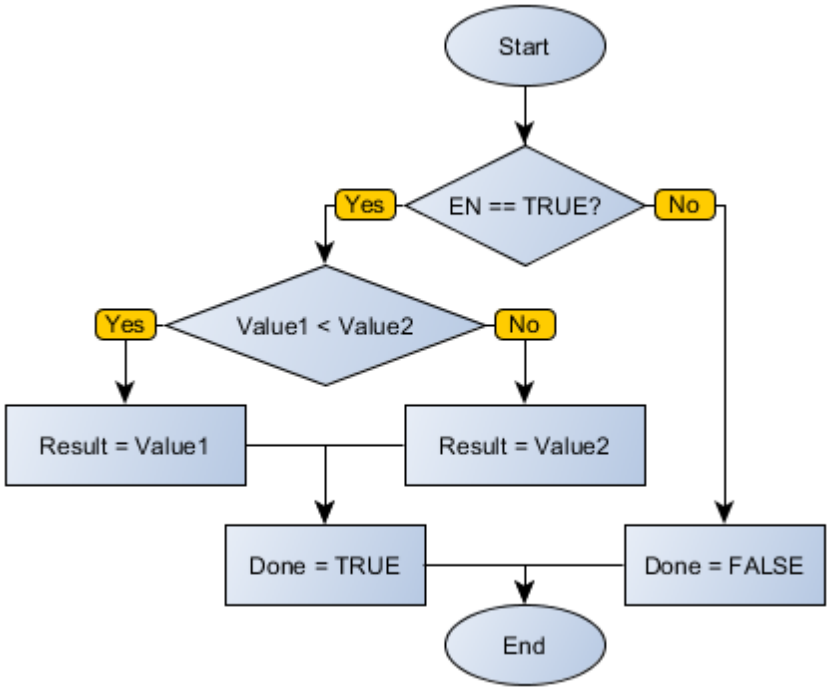
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	First value of comparison
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Second value of comparison
VAR_OUTPUT	Done	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Low est of the values compared

Operation

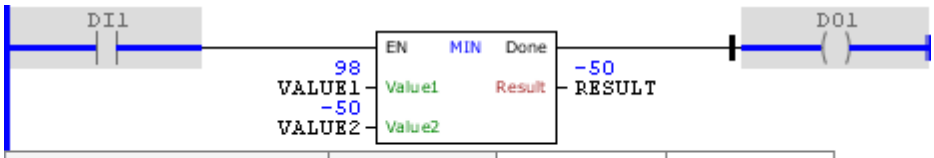
When this block has a TRUE value in EN, it sends to the Result output the lowest value in the comparison between Value1 and Value2. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

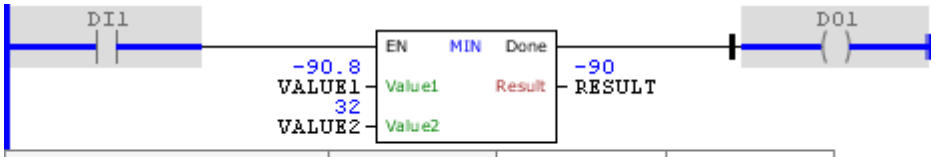


Example



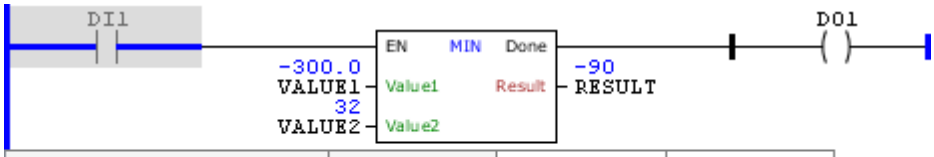
VALUE1	SINT	98	Decimal
VALUE2	SINT	-50	Decimal
RESULT	SINT	-50	Decimal

The above example calculates the minimum value between VALUE 1 and VALUE2 variables, storing the final result in RESULT. The block ends with success and Done output is activated.



VALUE1	REAL	-90.8	Float
VALUE2	SINT	32	Decimal
RESULT	SINT	-90	Decimal

The above example calculates the minimum value between VALUE 1 and VALUE2 variables, storing the final result in RESULT. The block ends with success and Done output is activated. Notice that the types of the input variables can be different without causing execution problems.



VALUE1	REAL	-300.0	Float
VALUE2	SINT	32	Decimal
RESULT	SINT	-90	Decimal

The above example calculates the minimum value between VALUE 1 and VALUE2 variables, storing the final result in RESULT. Since the result is lower than the minimum supported by SINT type, the block generates an error and Done output is disabled.

11.2.6.11.4.3 SAT

Block that performs a routine for saturation of the value found in Value in accordance with the limits for Minimum and Maximum, storing the result in Result.

Ladder Representation

Block Structure

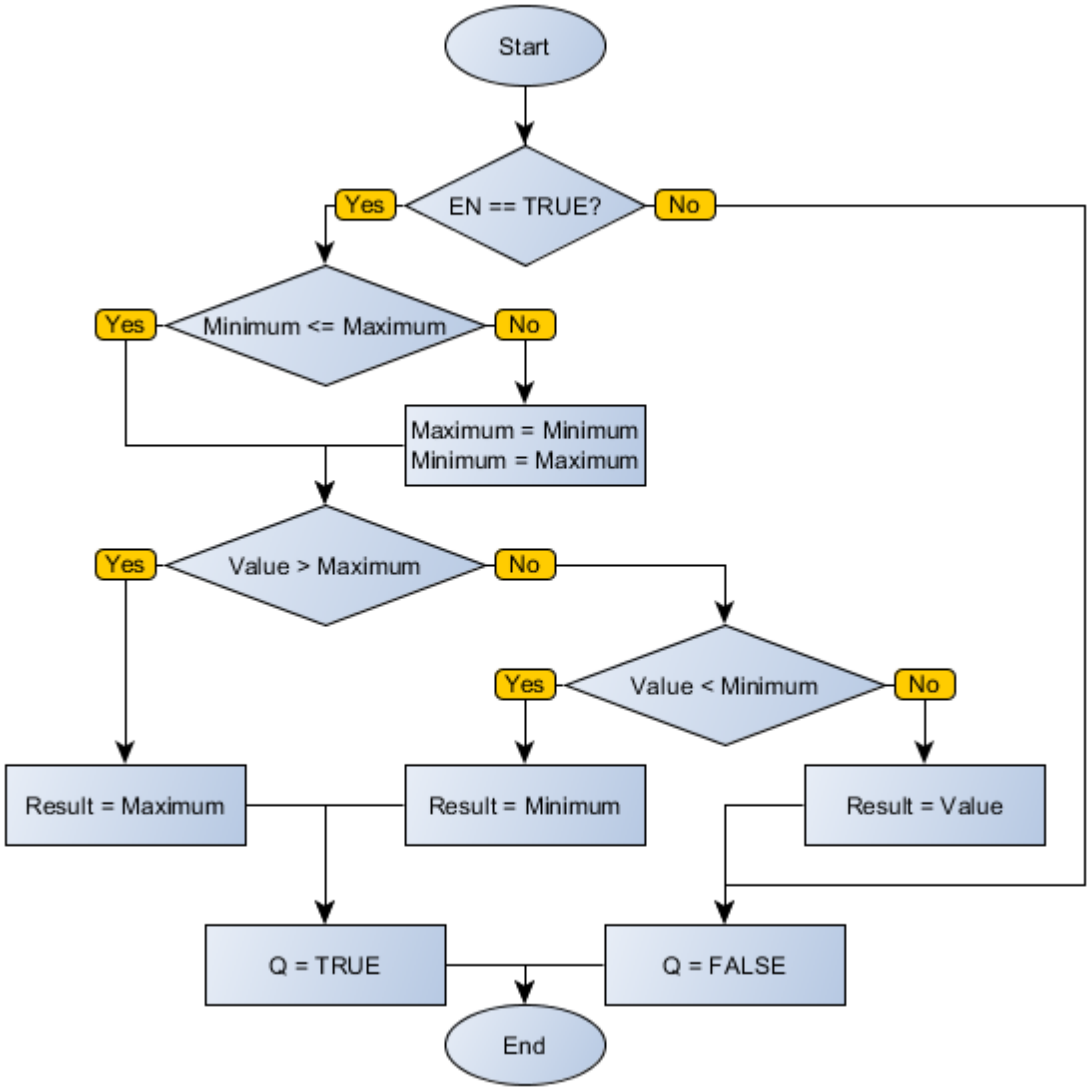
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Reference value
	Minimum	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Inferior saturation value
	Maximum	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Superior saturation value
VAR_OUTPUT	Q	BOOL	Indicator that there w as saturation in the process
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Result of operation

Operation

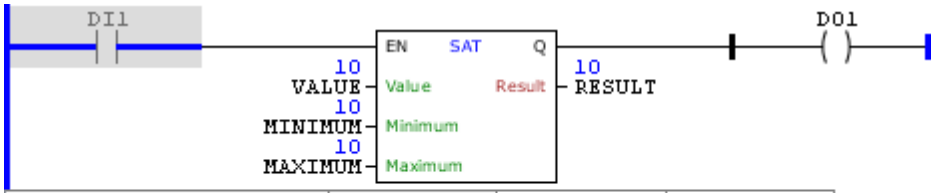
When this block has a TRUE value in EN, it performs a comparison between Value and Minimum and Maximum. If Value is in the range between Minimum and Maximum, Result receives the value of Value and Q remains FALSE. If Value is higher than Maximum, Result receives Maximum and Q receives TRUE. If Value is lower than Minimum, Result receives Minimum and Q receives TRUE. If there is any error in the operation, Q is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Q remains in FALSE.

Block Flowchart

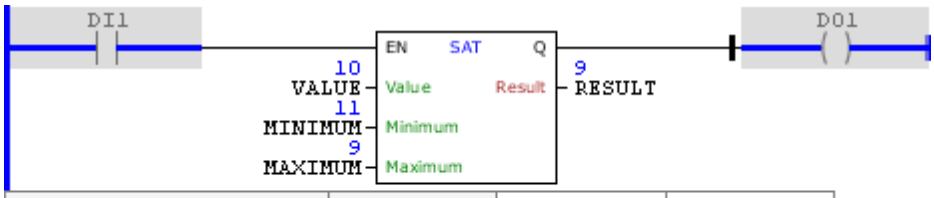


Example



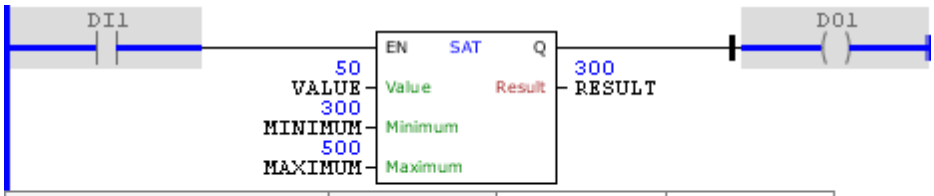
VALUE	INT	10	Decimal
MINIMUM	INT	10	Decimal
MAXIMUM	INT	10	Decimal
RESULT	INT	10	Decimal

The above example passes the VALUE value to RESULT, since it is not lower than MINIMUM or higher than MAXIMUM. The block ends successfully and the Q output is disabled, since there was no saturation.



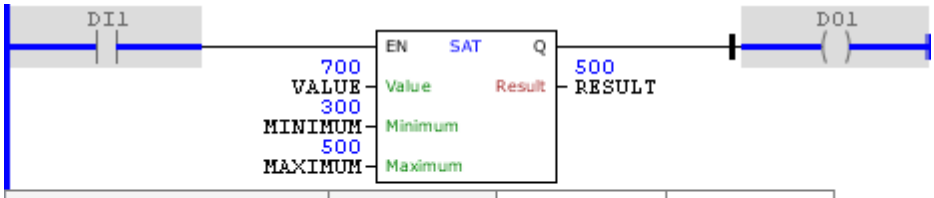
VALUE	INT	10	Decimal
MINIMUM	INT	11	Decimal
MAXIMUM	INT	9	Decimal
RESULT	INT	9	Decimal

The above example passes the MAXIMUM to RESULT, since VALUE is higher than MAXIMUM. The block ends successfully and the Q output is activated, since there was saturation.



VALUE	INT	50	Decimal
MINIMUM	INT	300	Decimal
MAXIMUM	INT	500	Decimal
RESULT	INT	300	Decimal

The above example passes the MINIMUM to RESULT, since VALUE is lower than MINIMUM. The block ends successfully and the Q output is activated, since there was saturation.



VALUE	INT	700	Decimal
MINIMUM	INT	300	Decimal
MAXIMUM	INT	500	Decimal
RESULT	INT	500	Decimal

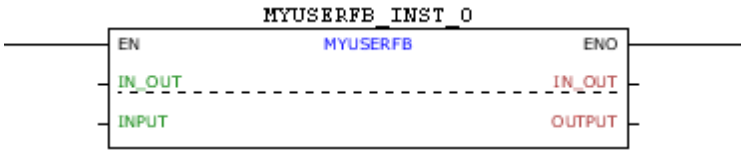
The above example passes the MAXIMUM value to RESULT, since VALUE is higher than MAXIMUM. The block ends successfully and the Q output is activated, since there was saturation.

11.2.6.12 Module

11.2.6.12.1 USERFB

Block that performs a subroutine programmed by the user.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	INPUT	According to user programming	Block inputs
VAR_OUTPUT	ENO	BOOL	End of operation
	OUTPUT	According to user programming	Block outputs
VAR_IN_OUT	IN_OUT	According to user programming	Block inputs/outputs
VAR	MYUSERFB_INST_0	MYUSERFB	Instance of access to block structure

Operation

When this block has a TRUE value in EN, it updates the values of internal fields with the input variables, performs the Ladder routine programmed by the user and updates the values of the outputs after completing routine.

When EN has FALSE value, outputs remain unchanged.

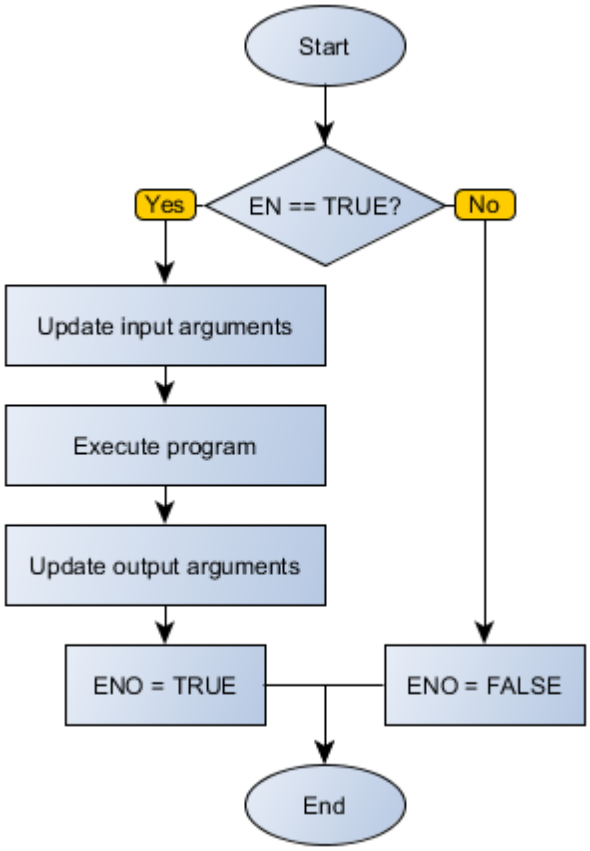
The ENO value forwards to the next Ladder block the EN value after the operation is completed.

NOTE!
Refer to section Working with USERFBs for further information.

Compatibility

Device	Version
PLC300	1.50 or higher
SCA06	2.00 or higher

Block Flowchart



11.2.6.13 Motion Control

11.2.6.13.1 MW_RefVelocity

This block sends speed reference to drive.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Velocity	DINT INT REAL	Sets speed reference to drive if block is enabled
	VelocityUnit	0 = 13Bits 1 = RPM 2 = HZ(x10)	Sets the speed unit: 13 Bits – Sends the speed value in 13 bits; RPM – Sends the speed value in RPM; HZ – Sends the speed value in Hz (x10).
	RunAutomatic	0 = FALSE 1 = TRUE	Define if block will run the Run/Stop (CFW_CMD_RUN_STOP) when it is enabled: FALSE – Do not send Run/Stop command with block enabling (it is necessary to use the marker CFW_CMD_RUN_STOP in ladder's logic to send the Run/Stop command); TRUE – Send Run command with block enabling and Stop command with block disabling.
VAR_OUTPUT	ENO	BOOL	End of operation. Conditions for ENO = 1 • Does not exist another active block MW_RefVelocity; • Drive is enabled and stop mode set "Stopping by inertia".

Operation

When this block has a "0" value in EN, it does not execute and ENO output is zero.

RunAutomatic = TRUE

When this block has a "1" value in EN input, the drive is general enabled, no other motion block is active, the Run/Stop command goes to "1", the speed reference value is send to drive and the ENO output is set to "1".
If EN input has a "0" value, and this block is active, the Run/Stop command is set to "0" and ENO output goes to "0".

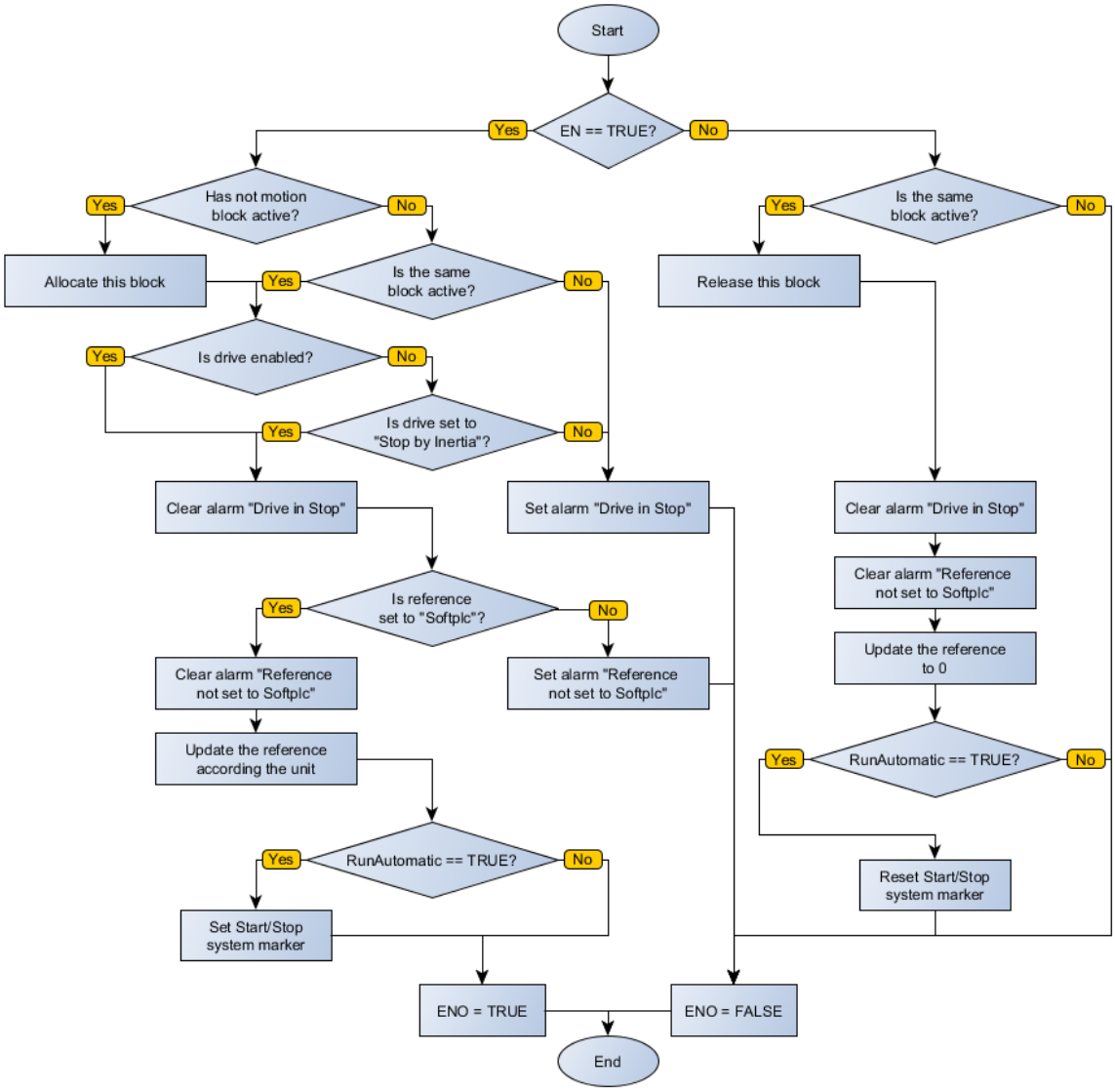
RunAutomatic = FALSE

When this block has a "1" value in EN input, the drive is general enabled, the Run/Stop command is set to "1", no other motion block is active, the speed reference value is send to drive and the ENO output is set to "1".
If EN input has a "0" value and this block is active, ENO output is set to "0".

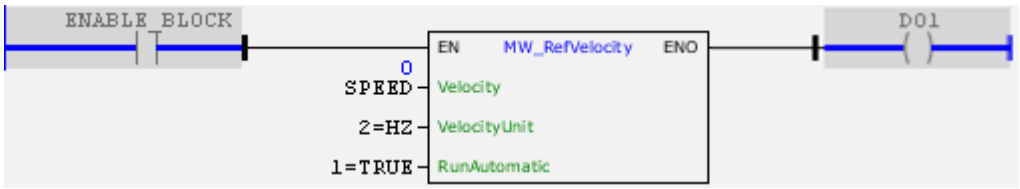
NOTE!

Check the source of speed reference and command Run/Stop for correct operation of this block

Block Flowchart

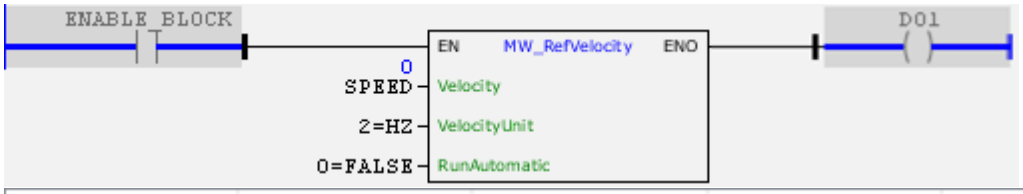


Example



Variável	Tipo	User	Monitoring	Visualização
Main Ladder				
SPEED	INT	0	500	Decimal
Global Variables				
PAR_001	UINT	0	500	Decimal

The above example shows the MW_RefVelocity block, set to Hz and the RunAutomatic command in TRUE, if drive is general enabled and the block is enabled, the speed reference is changed.



Variável	Tipo	User	Monitoring	Visualização
Main Ladder				
SPEED	INT	0	500	Decimal
Global Variables				
PAR_001	UINT	0	500	Decimal

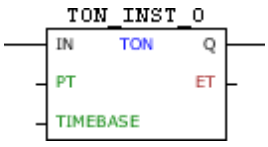
The above example shows the MW_RefVelocity block, set to Hz and RunAutomatic command in FALSE, if drive is general enabled, it is necessary the Run command. So, when the block be enabled, the speed reference would be changed.

11.2.6.14 Timer

11.2.6.14.1 TON

Timer block that, when energized, enables the output after a delay set by PT.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	IN	BOOL	Block enabling
	PT	WORD UINT DWORD UDINT	Delay of output drive
	TIMEBASE	WORD	Time base for PT and ET
VAR_OUTPUT	Q	BOOL	Block output
	ET	WORD UINT DWORD UDINT	Counter elapsed time
VAR	TON_INST_0	TON	Instance of access to block structure

NOTE!
In CFW300, the PT e ET fields can only be WORD ou UINT type.

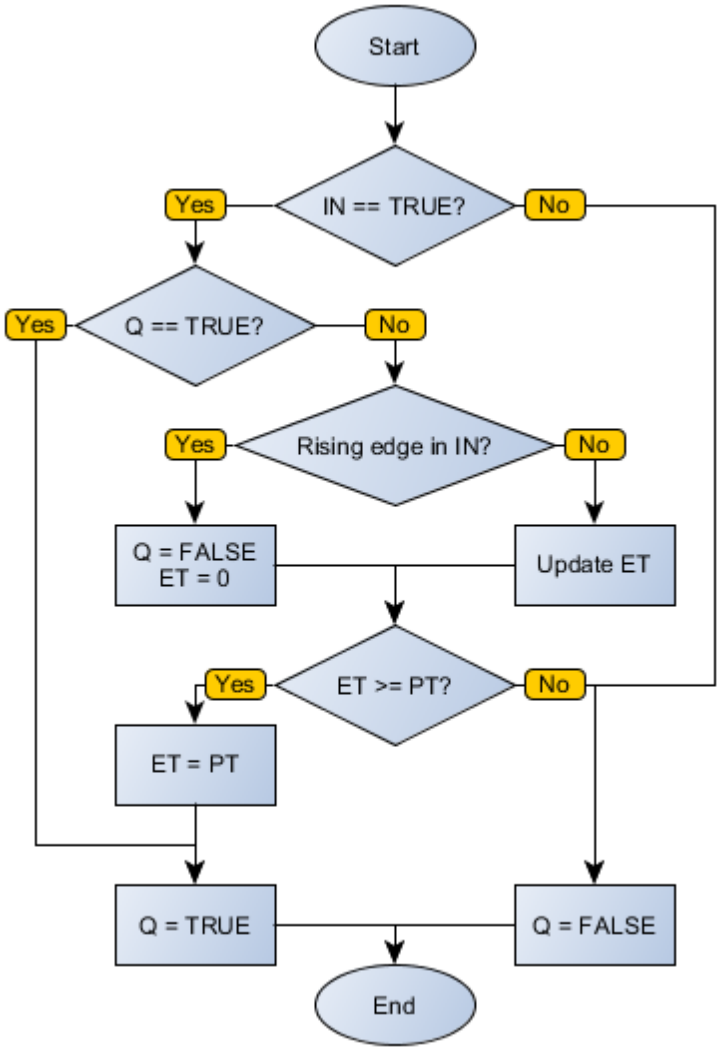
Operation

While the IN input is FALSE, the Q output is FALSE and ET also receives the value zero.
On the edge positive transition in IN, counting is triggered and ET is incremented according to TIMEBASE. When ET equals PT, the Q output goes to state TRUE until IN revolutions to FALSE.

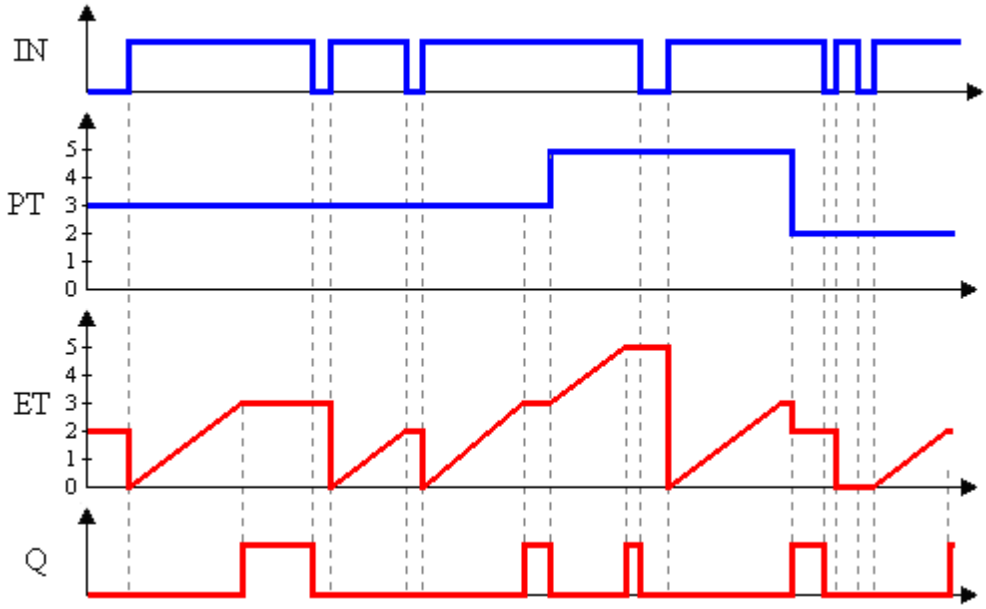
Compatibility

Device	Version
PLC300	1.50 or higher
SCA06	2.00 or higher

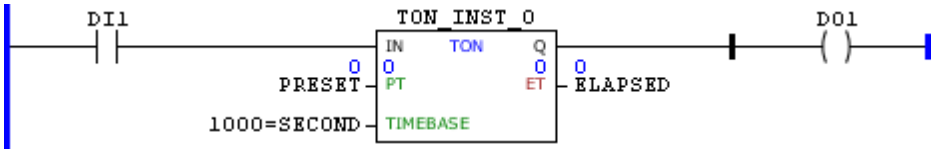
Block Flowchart



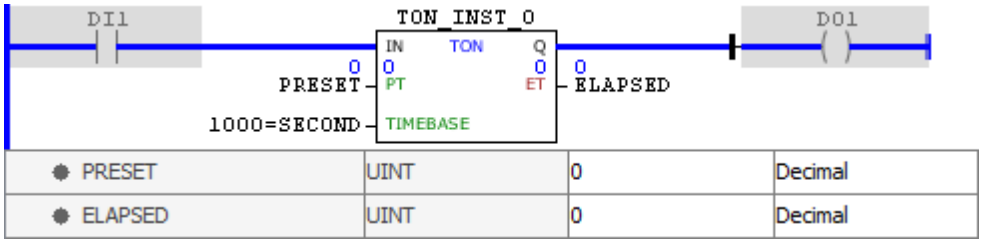
Operation Diagram



Example

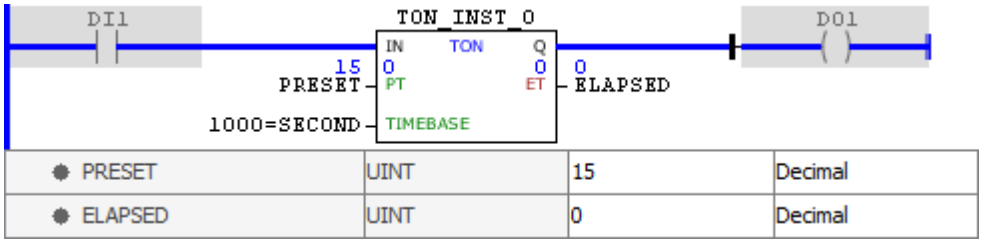


The above example shows the initial conditions of the block and of the routine variables.



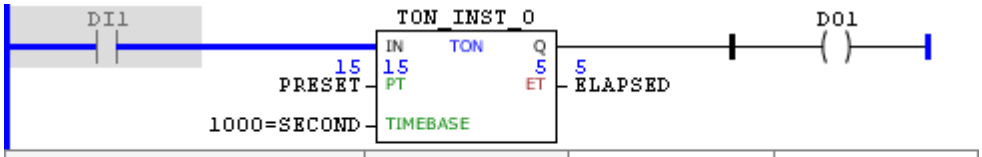
PRESET	UINT	0	Decimal
ELAPSED	UINT	0	Decimal

When activated the IN input, counting is triggered. Since ET equals PT, the Q output is enabled.



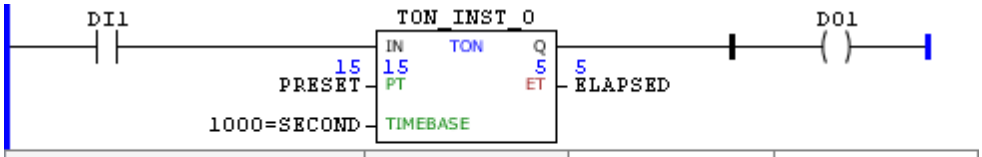
PRESET	UINT	15	Decimal
ELAPSED	UINT	0	Decimal

Note that a change in PRESET variable is not forwarded to the PT field while the IN entry remains enabled.



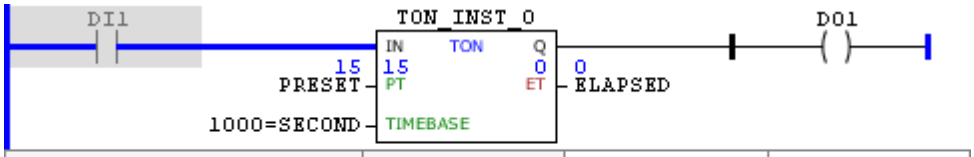
PRESET	UINT	15	Decimal
ELAPSED	UINT	5	Decimal

Disabling the IN input, the value of PT is updated and the Q output is disabled. When activating it again, counting is triggered.



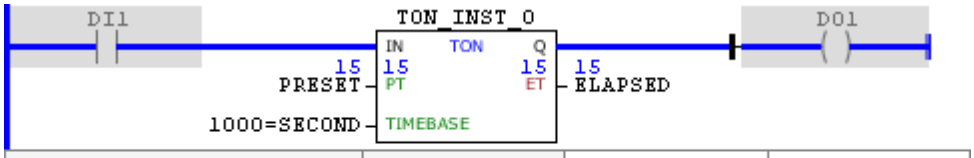
PRESET	UINT	15	Decimal
ELAPSED	UINT	5	Decimal

Disabling the IN input, the value of ET remains saved.



PRESET	UINT	15	Decimal
ELAPSED	UINT	0	Decimal

Enabling the IN input, the value of ET is reset and counting is triggered.



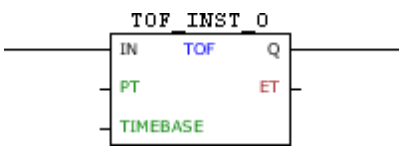
PRESET	UINT	15	Decimal
ELAPSED	UINT	15	Decimal

When ET reaches the value PT, the Q is output enabled and remains so while IN is at TRUE level.

11.2.6.14.2 TOF

Timer block that, when energized, disables the output after a delay set by PT.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	IN	BOOL	Block enabling
	PT	WORD UINT DWORD UDINT	Delay of output deactivating
	TIMEBASE	WORD	Time base for PT and ET
VAR_OUTPUT	Q	BOOL	Block output
	ET	WORD UINT DWORD UDINT	Counter elapsed time
VAR	TOF_INST_0	TOF	Instance of access to block structure

NOTE!
In CFW300, the PT e ET fields can only be WORD ou UINT type.

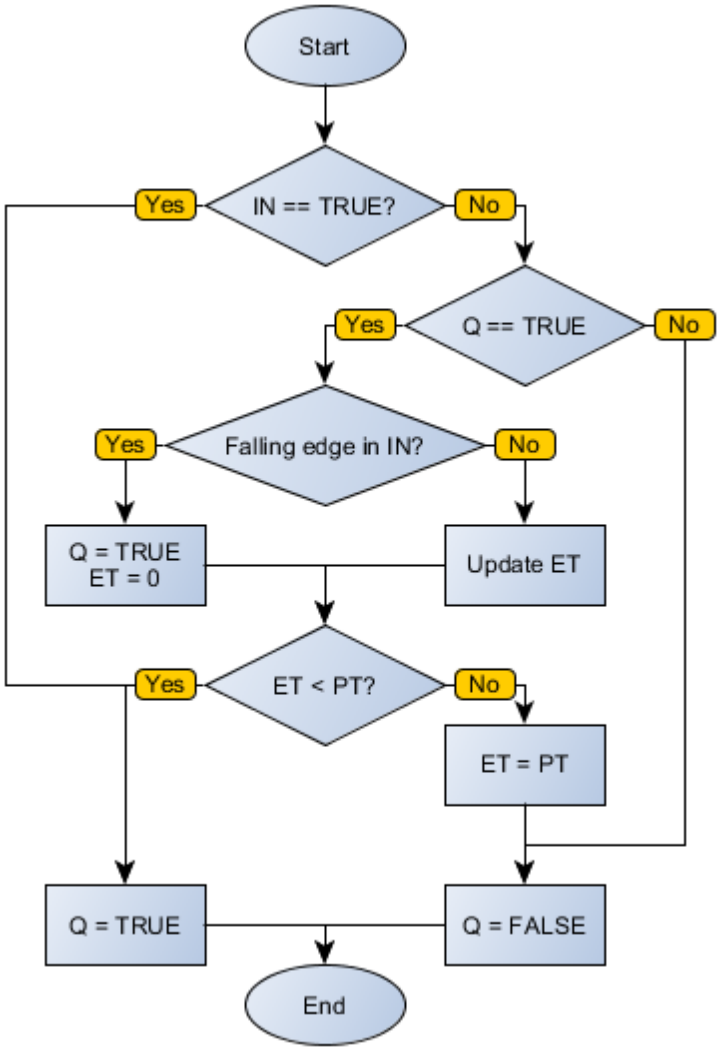
Operation

While the IN input is TRUE, the Q output is also TRUE and ET also receives the value zero.
On the negative transition edge in IN, counting is triggered and ET is incremented according to TIMEBASE. When ET equals PT, the Q output goes to state FALSE until IN revolutions to FALSE.

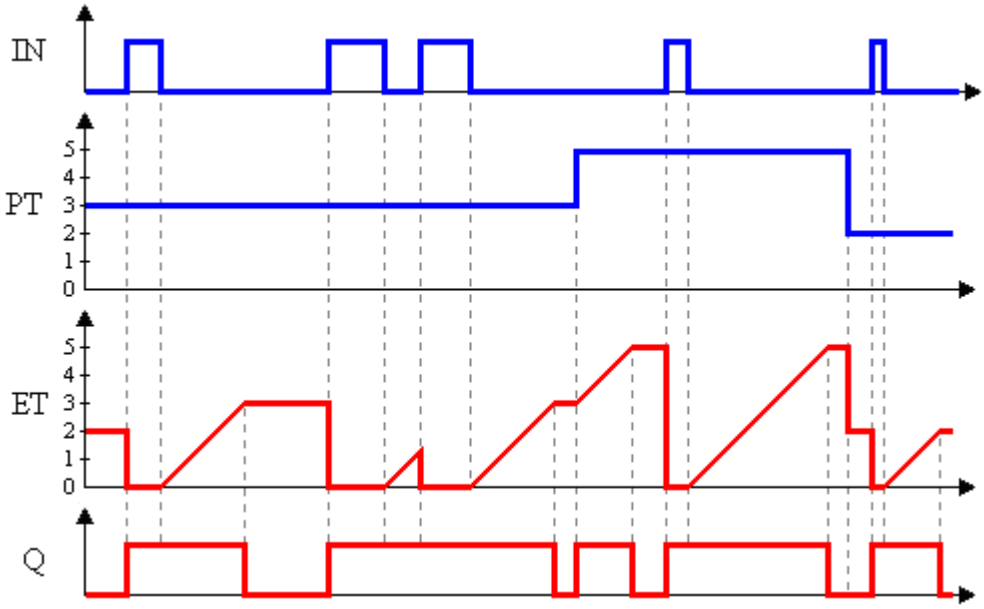
Compatibility

Device	Version
PLC300	1.50 or higher
SCA06	2.00 or higher

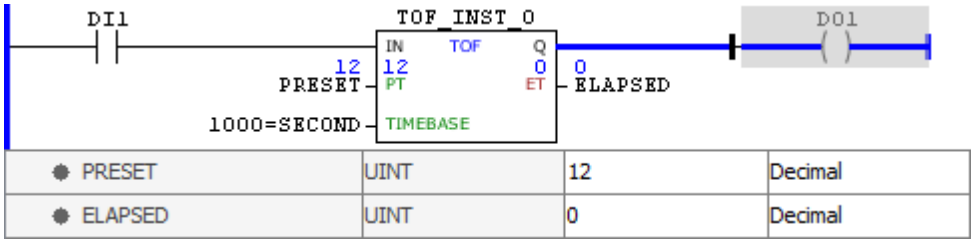
Block Flowchart



Operation Diagram



Example



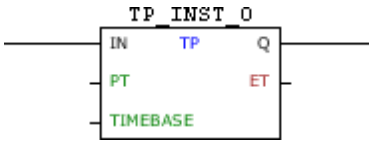
PRESET	UINT	12	Decimal
ELAPSED	UINT	0	Decimal

The above example disables the DO1 output to identify a low level in DI1 for 12 seconds, remaining disabled until DI1 again be TRUE.

11.2.6.14.3 TP

Timer block that, when identifies it is energized, enables the output after a delay set by PT.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	IN	BOOL	Block enabling
	PT	WORD UINT DWORD UDINT	Time w hile the output is enabled
	TIMEBASE	WORD	Time base for PT and ET
VAR_OUTPUT	Q	BOOL	Block output
	ET	WORD UINT DWORD UDINT	Counter elapsed time
VAR	TP_INST_0	TP	Instance of access to block structure

NOTE!
In CFW300, the PT e ET fields can only be WORD ou UINT type.

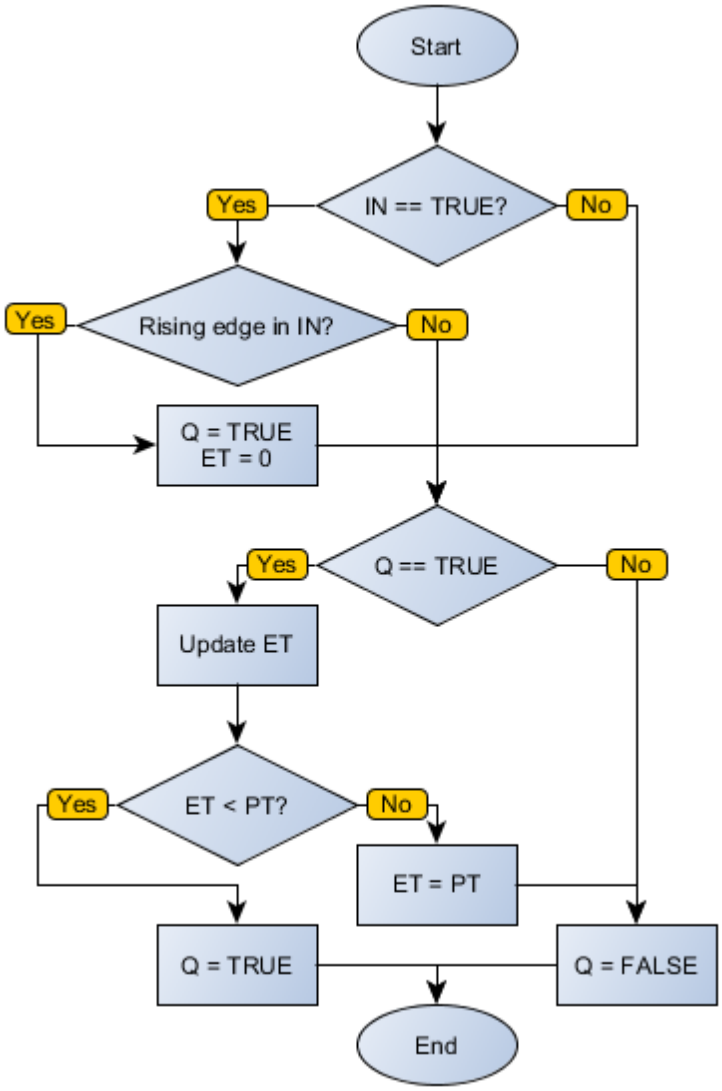
Operation

On the edge positive transition in IN, Q receives TRUE value, counting is triggered and ET is incremented according to TIMEBASE. When ET equals PT, the Q output goes to state FALSE until IN revolutions to FALSE. At that moment, if IN is at TRUE level, nothing happens. On the edge positive transition in IN, ET is automatically reset.

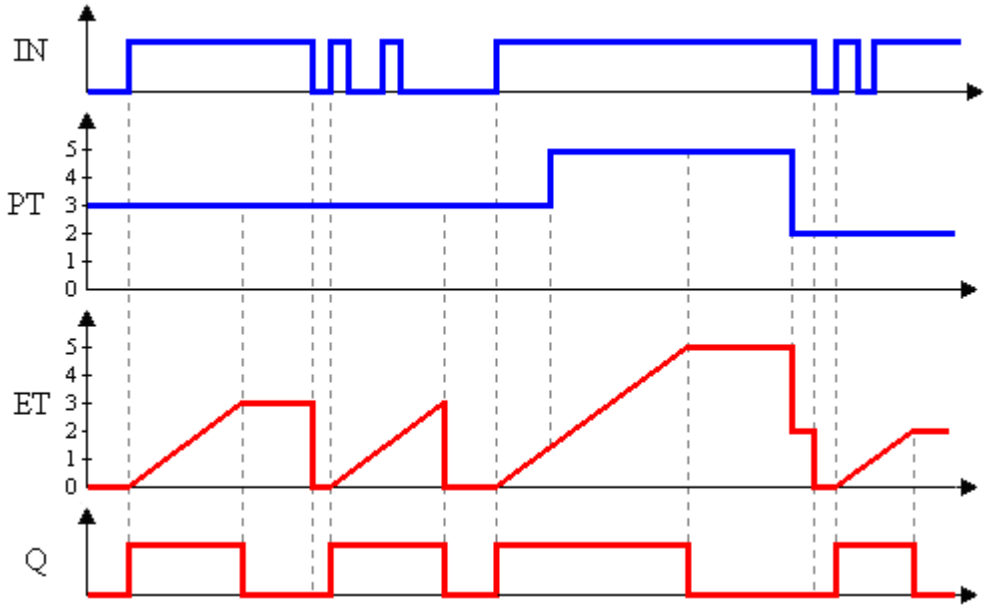
Compatibility

Device	Version
PLC300	1.50 or higher
SCA06	2.00 or higher

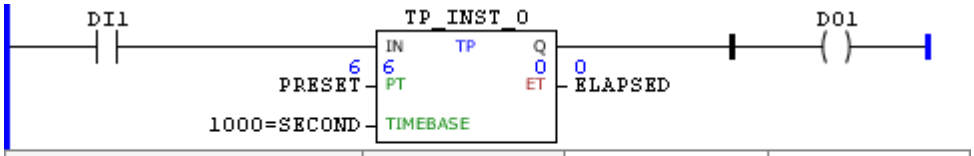
Block Flowchart



Operation Diagram



Example



PRESET	UINT	6	Decimal
ELAPSED	UINT	0	Decimal

The above example enables the DO1 output for six seconds at each DI1 positive transition.

11.2.6.15 Structures

Structure is a data grouping used to define a recipe or an object.

In the Ladder program, it is possible to create variables of the structure type and use them in the blocks. To access the internal members of the structure, the '.' is used followed by its respective member.

Creating a structure

1. With the right button of the mouse on the folder **Structure**, click on **New file**.

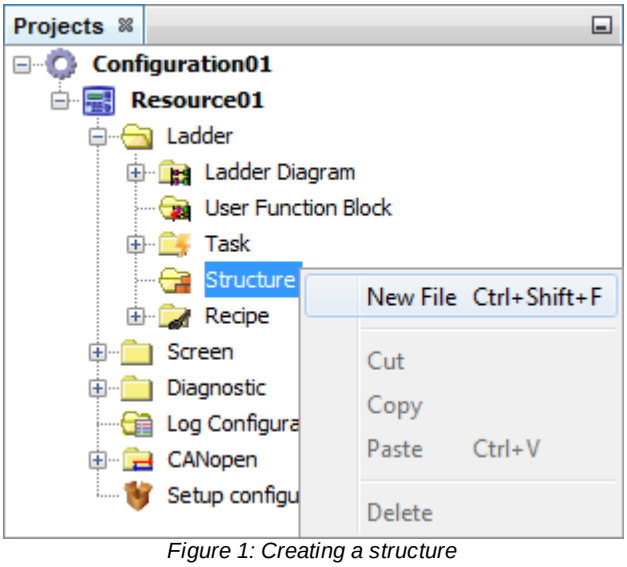


Figure 1: Creating a structure

2. Define the file name and press the **Next** button.

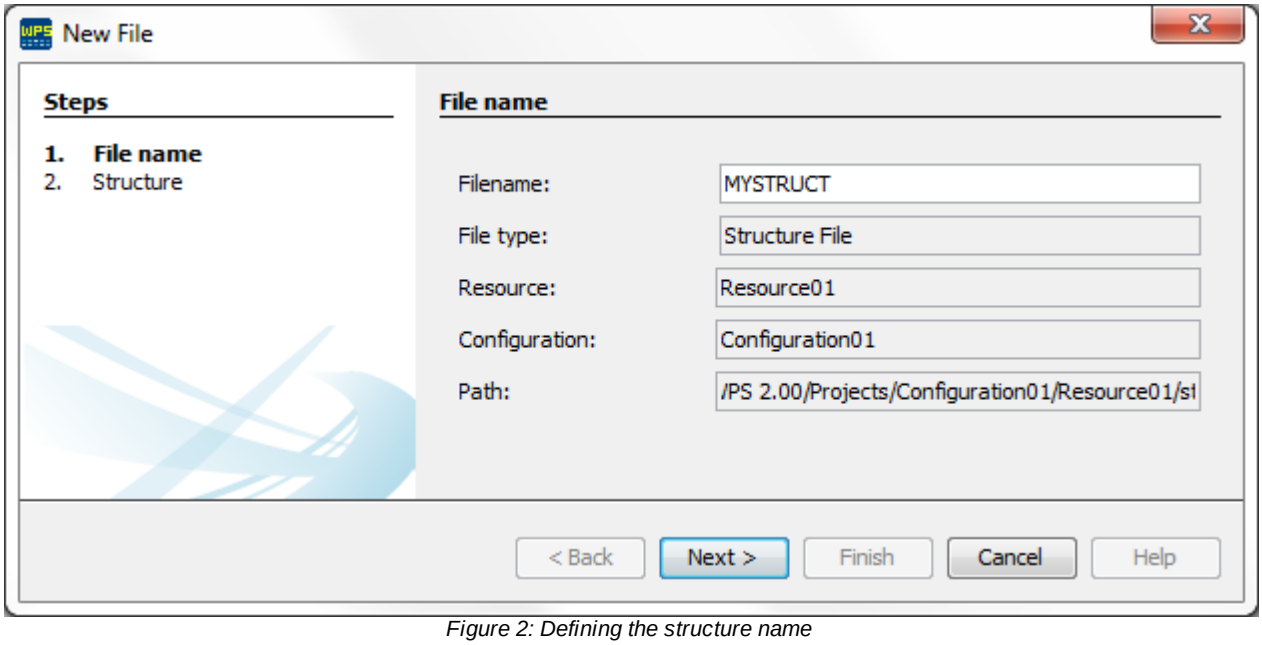


Figure 2: Defining the structure name

3. Configure the structure using the buttons presented in the figure below.

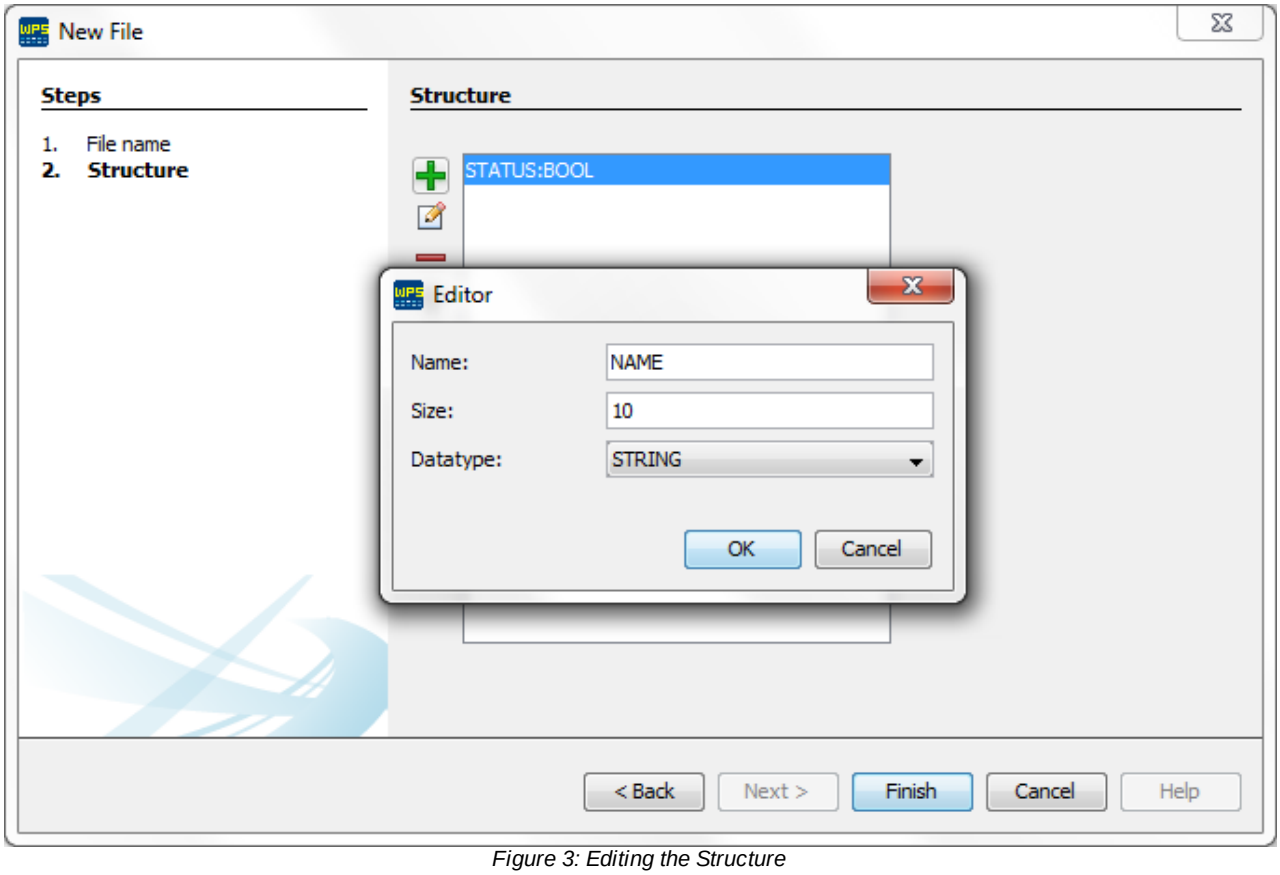


Figure 3: Editing the Structure

4. After finishing the edition of the structure, click on the button **Finish**.

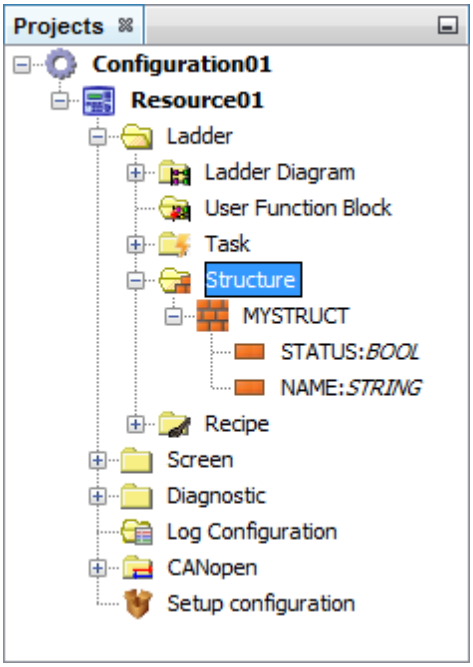


Figure 4: Structure created in the project

Editing a structure

Just double click on the desired structure, as shown in figure 4, and a window will open as shown in figure 3, allowing to insert new data, erase or move the position of the data.

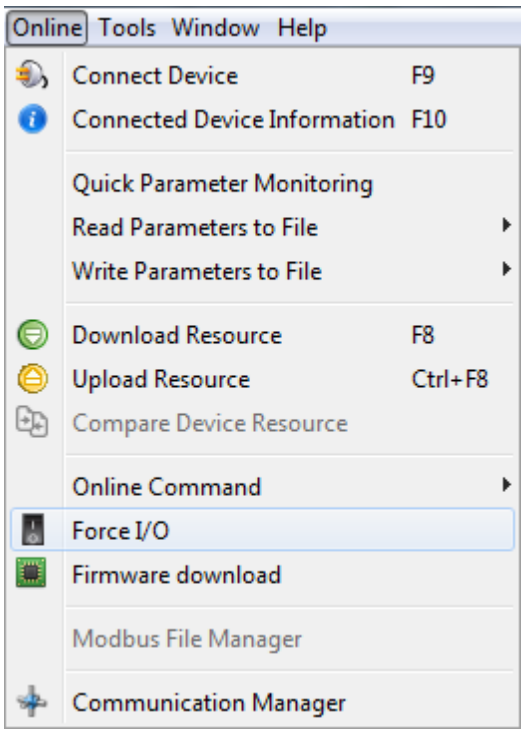
11.2.7 Communication

11.2.7.1 Force I/O

Overview

The force inputs and outputs window is used for the values of the digital and analog inputs to be read by the program, by values manipulated by the user, regardless their physical state. It also allows the manipulation of the physical states of the digital and analog outputs by the user independently of the values calculated by the program.
In order to force the device inputs and outputs, it is necessary that the online monitoring be active and the option **Run cyclically** be enabled. The data are sent to the device every 2 seconds.
The values can be edited with the device disconnected. The configurations are stored in the resources and recorded whenever the main resource selection is changed.
The data displayed on the force I/O window contain the values belonging to the resource (and configuration) selected as main.

The force I/O window is open trough the menu **Online** > **Force I/O**:



Toolbar

The toolbar of the force window has the options to run cyclically, upload the device force configuration, enable

all and disable all:

Run cyclically: Sends the user's configurations to the device and updates the state of the inputs and outputs in a cyclic way.

Upload configuration: Allows the current configuration of the device to be read. For this option to be enabled, it is necessary that the online monitoring be active and the option run cyclically be disabled.

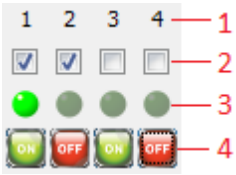
Enable all: Enables the force I/O of all of the inputs and outputs of the device.

Disable all: Disables the force I/O of all of the inputs and outputs of the device.

Input and Output commands

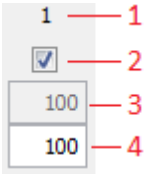
For each digital and analog input and output there is a selection box linked to enable the force, a status field and an edition field.

Digital:



1. Number of the digital inputs/output
2. Enable/disable Force I/O
3. Current status of the I/O: It has three statuses: 1. light green LED: activated; 2. dark green LED: deactivated; 3. gray LED: the value is not being read.
4. Enable/disable the input/output

Analog:



1. Number of the analog input/output
2. Enable/disable Force I/O
3. Current value of the input/output
4. Value of the input/output configured by the user

> **NOTE!**
> The analog signal scale has 15 bits plus 1 bit for signal, except for SSW900 which it has only 10 unsigned bits.

11.3 CFW500

Enter topic text here.

11.3.1 Description

With a modern design and power range of 0.25 to 30 hp, the CFW500 is a high-performance variable speed drive that assists in the speed and torque control of three-phase induction motors. The device also features sensorless vector, vector encoder or scalar, SoftPLC, which adds functions of PLC (Programmable Logic Controller), Pump Genius, which features dedicated functions for pumping and selectable plug-in modules, which provide a flexible and optimized for any application.

Refer to the user's manual of the CFW500 for further details about the product.

NOTE!

This product does not have the Ladder tool available in WPS.
You can use the WLP application if this feature is required.

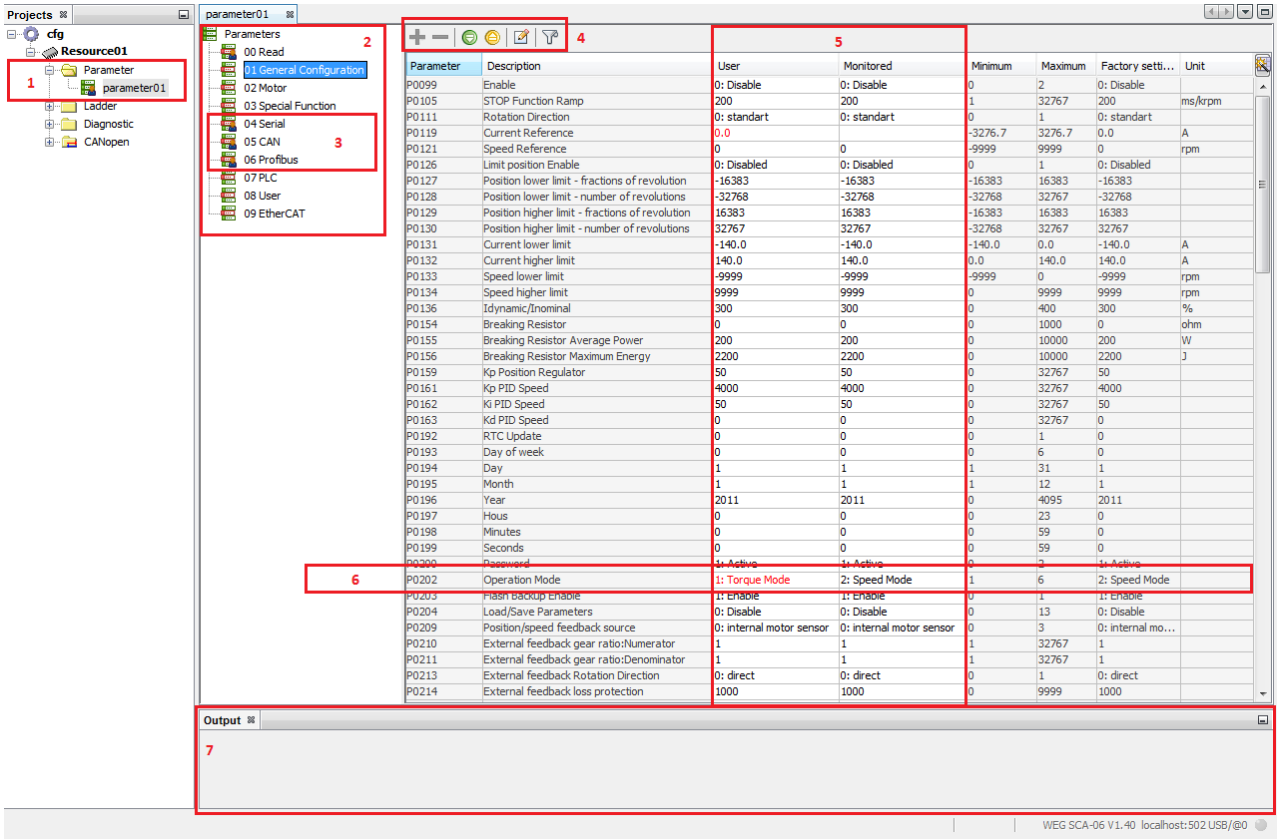
11.3.2 Parameters

11.3.2.1 Overview

The parameter configuration screen is used to configure and monitor all the parameters of the equipment, including the user parameters.

NOTE!
The reading and writing of such parameters is done on this screen; only the user parameter configuration must be sent the first time or whenever modified by means of the resource download routine.

Below is an overview of the parameter configuration screen.



1. **Parameter files**. In this part are all the parameter configuration files created by the user. Notice that when the file features a person figure on the table, it means this parameter table contains hidden parameters/ group of parameters.
2. **Group of parameters**. This tree shows all the group of parameters. Notice that the same parameter can be in more than one group, and when its value is modified, it will be modified in all the groups to which it belongs.
3. **Modified group of parameters**. Group of parameters which contain the figure of a person on the table means they have hidden parameters.
4. **Commands**. The commands are described below in the order they appear:
   4.1. **Unhide parameter**: In case some parameter has been hidden, this button allows making it visible again.
   4.2. **Hide parameter**: Just select one or more parameters on the table and trigger this command to hide them.
   4.3. **Save table**: It saves the values of the parameters shown on the equipment screen; the sent values are the ones in the User column. The flow is User -> Monitored (equipment)
   4.4. **Read table**: It reads the parameters of the equipment shown in the Monitored column and saves them in the parameter file in the User column. The flow is Monitored (equipment) -> User
   4.5. **User parameters**: It opens a screen to edit the user parameters.
   4.6. **Filter**: It opens a parameter filter option, and it can filter by parameter number or description.
   4.7. **User Parameters and Monitored Parameters**. These two columns show the off-line and on-line parameters, so to speak. The **User** column shows the values contained in the file located on the computer and the **Monitored** column shows the values that are effectively saved on the equipment. Whenever you use the **Save Parameter** option, the sent values will be from the **User** column to the **Monitored** column, that is, File -> Equipment. In case of reading, the flow is the opposite, from the **Monitored** column to the **User** column, that is, Equipment -> File. In case you wish to change the values directly on the equipment without changing it in the file, just click on the monitored column,

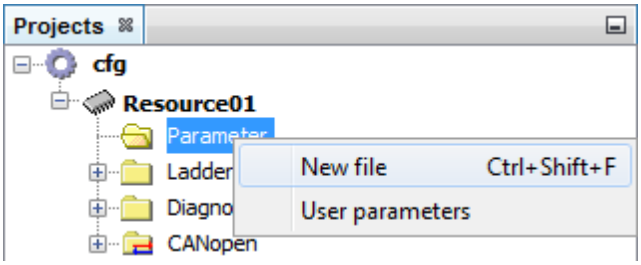
change the values and the modification will occur on-line.

5. **Modified parameters**: Whenever a parameter value in the **User** column is different from the **Monitored** column, it will be shown in red.
6. **Output**. This screen shows error information in case they occur during the writing or reading of the parameters.

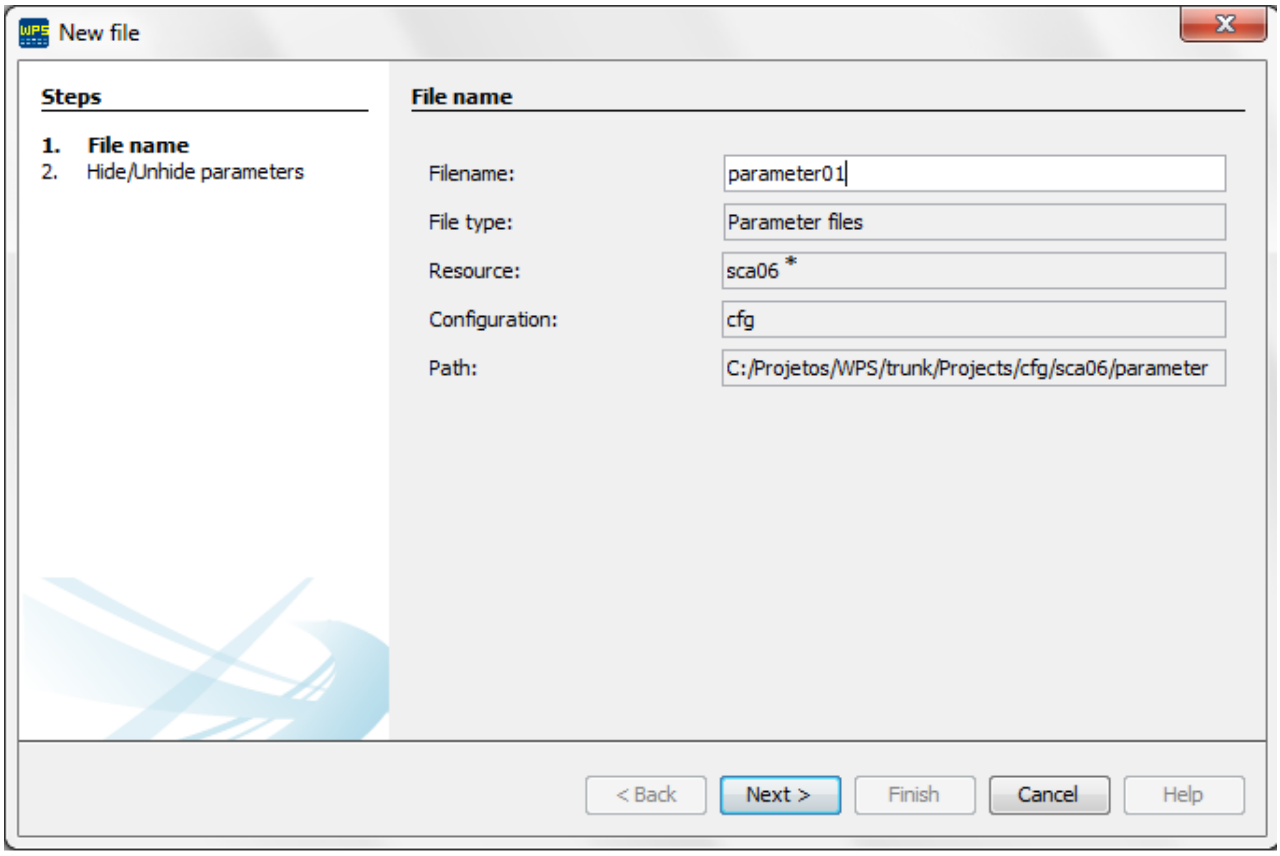
11.3.2.2 Configuration

Below is the list of the required steps to create a parameter file.

1. Create a new parameter file.

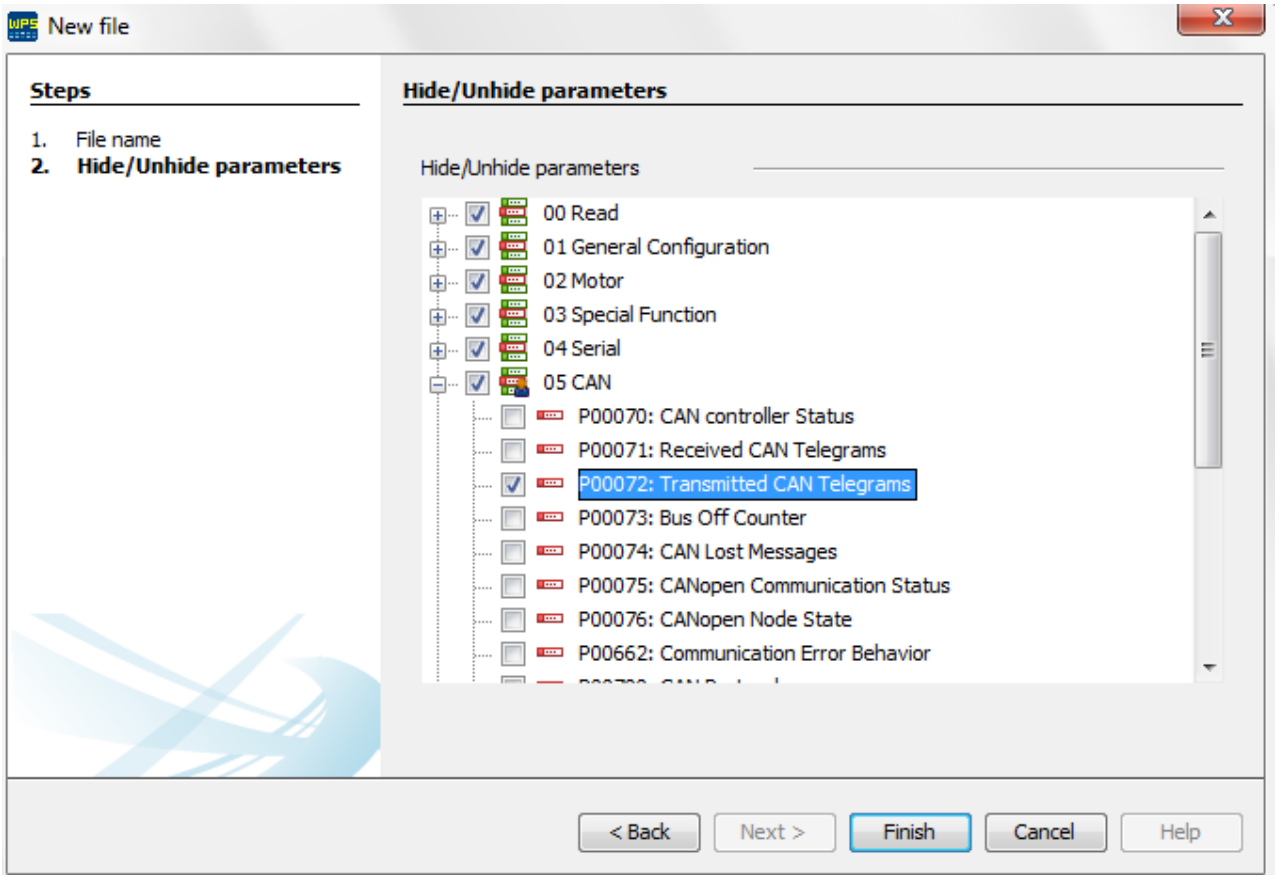


2. Define a name for the parameter file

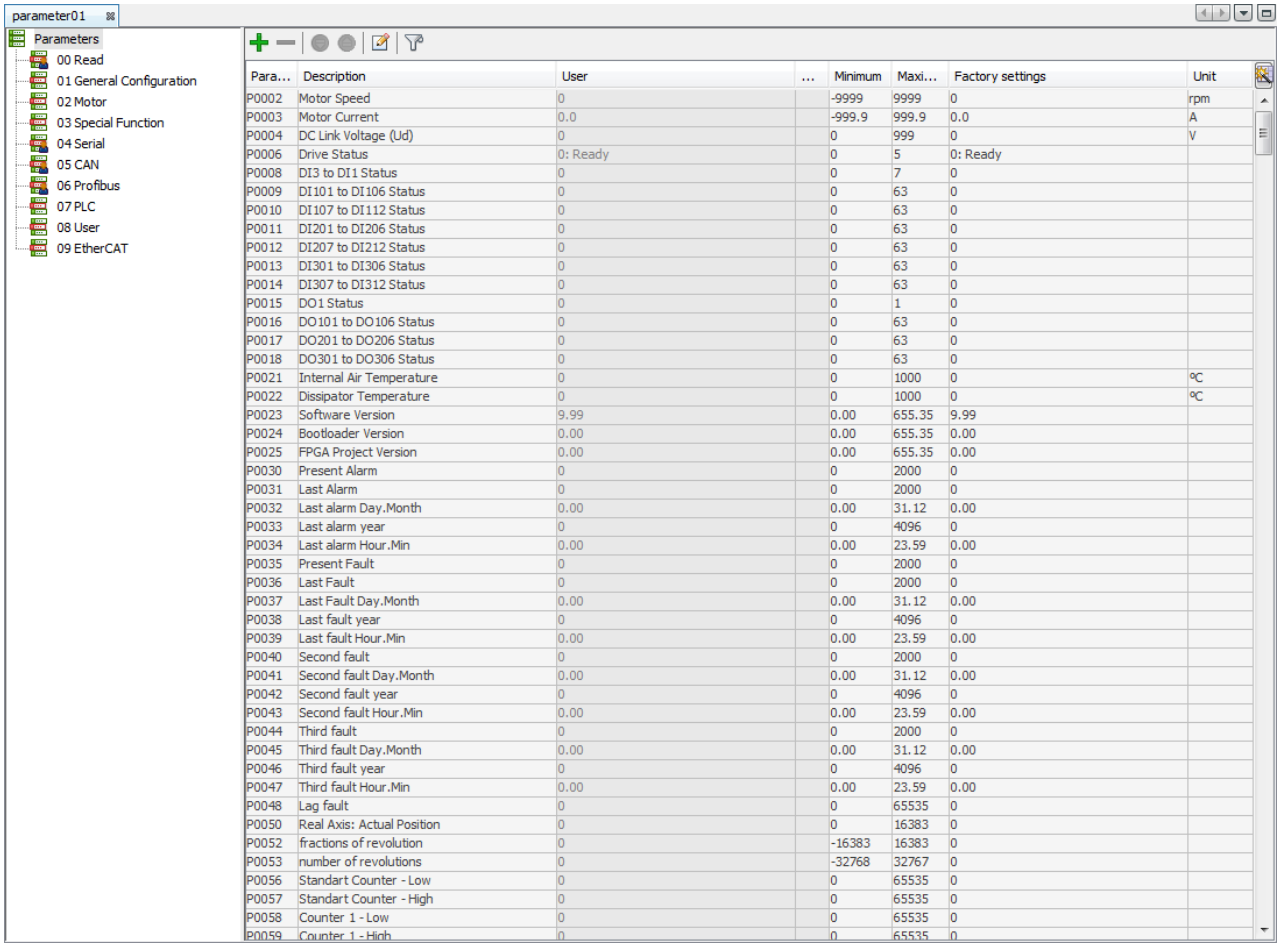


*** Resource: Resource01, SCA06, CFW300, etc.**

3. Configure which parameters you wish to view in your parameter table



4. After performing the steps above, the parameter file will be created and the equipment can be parameterized.

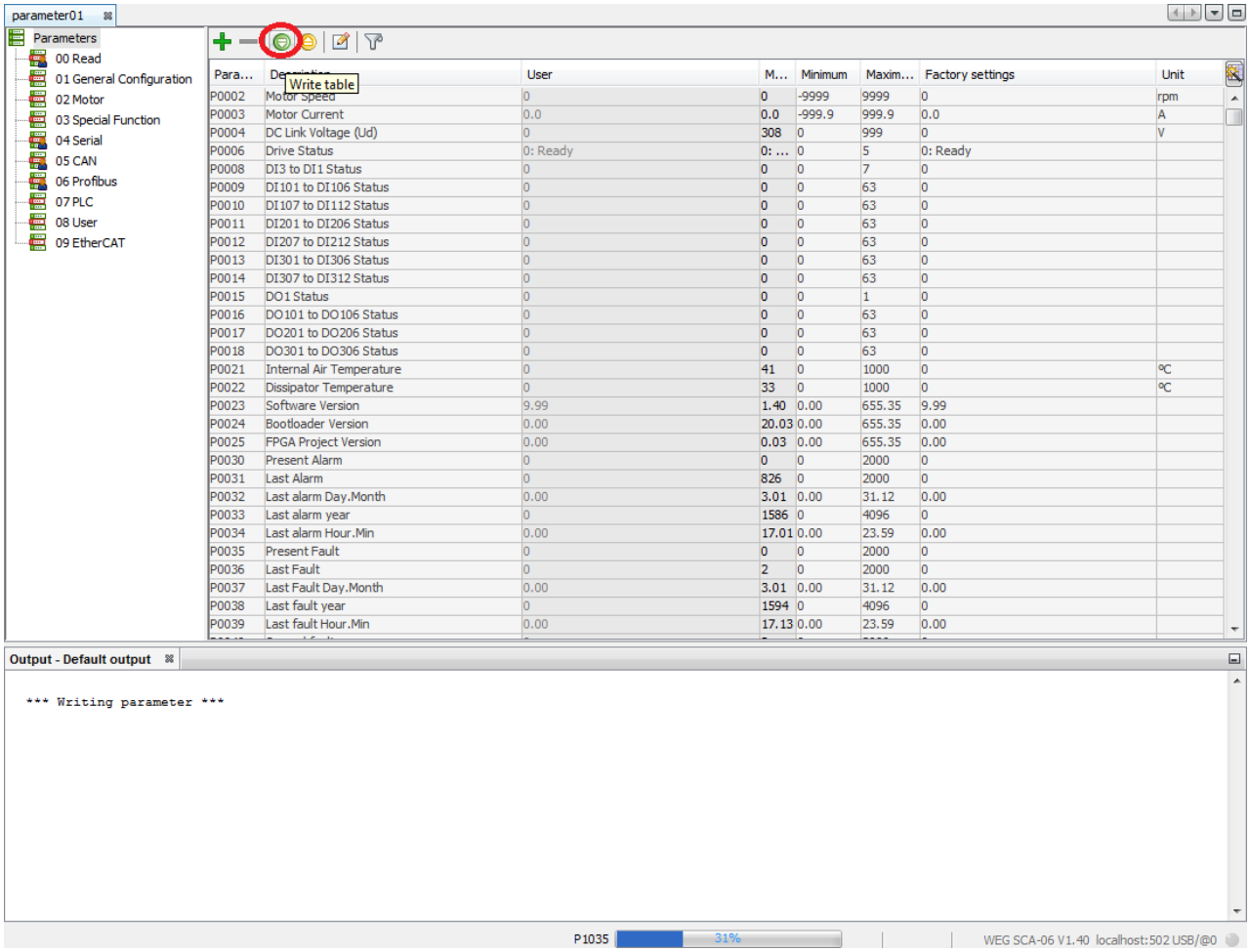


Para...	Description	User	...	Minimum	Maxi...	Factory settings	Unit
P0002	Motor Speed	0		-9999	9999	0	rpm
P0003	Motor Current	0.0		-999.9	999.9	0.0	A
P0004	DC Link Voltage (Ud)	0		0	999	0	V
P0006	Drive Status	0: Ready		0	5	0: Ready	
P0008	DI3 to DI1 Status	0		0	7	0	
P0009	DI101 to DI106 Status	0		0	63	0	
P0010	DI107 to DI112 Status	0		0	63	0	
P0011	DI201 to DI206 Status	0		0	63	0	
P0012	DI207 to DI212 Status	0		0	63	0	
P0013	DI301 to DI306 Status	0		0	63	0	
P0014	DI307 to DI312 Status	0		0	63	0	
P0015	DO1 Status	0		0	1	0	
P0016	DO101 to DO106 Status	0		0	63	0	
P0017	DO201 to DO206 Status	0		0	63	0	
P0018	DO301 to DO306 Status	0		0	63	0	
P0021	Internal Air Temperature	0		0	1000	0	ºC
P0022	Dissipator Temperature	0		0	1000	0	ºC
P0023	Software Version	9.99		0.00	655.35	9.99	
P0024	Bootloader Version	0.00		0.00	655.35	0.00	
P0025	FPGA Project Version	0.00		0.00	655.35	0.00	
P0030	Present Alarm	0		0	2000	0	
P0031	Last Alarm	0		0	2000	0	
P0032	Last alarm Day.Month	0.00		0.00	31.12	0.00	
P0033	Last alarm year	0		0	4096	0	
P0034	Last alarm Hour.Min	0.00		0.00	23.59	0.00	
P0035	Present Fault	0		0	2000	0	
P0036	Last Fault	0		0	2000	0	
P0037	Last Fault Day.Month	0.00		0.00	31.12	0.00	
P0038	Last fault year	0		0	4096	0	
P0039	Last fault Hour.Min	0.00		0.00	23.59	0.00	
P0040	Second fault	0		0	2000	0	
P0041	Second fault Day.Month	0.00		0.00	31.12	0.00	
P0042	Second fault year	0		0	4096	0	
P0043	Second fault Hour.Min	0.00		0.00	23.59	0.00	
P0044	Third fault	0		0	2000	0	
P0045	Third fault Day.Month	0.00		0.00	31.12	0.00	
P0046	Third fault year	0		0	4096	0	
P0047	Third fault Hour.Min	0.00		0.00	23.59	0.00	
P0048	Lag fault	0		0	65535	0	
P0050	Real Axis: Actual Position	0		0	16383	0	
P0052	fractions of revolution	0		-16383	16383	0	
P0053	number of revolutions	0		-32768	32767	0	
P0056	Standart Counter - Low	0		0	65535	0	
P0057	Standart Counter - High	0		0	65535	0	
P0058	Counter 1 - Low	0		0	65535	0	
P0059	Counter 1 - High	0		0	65535	0	

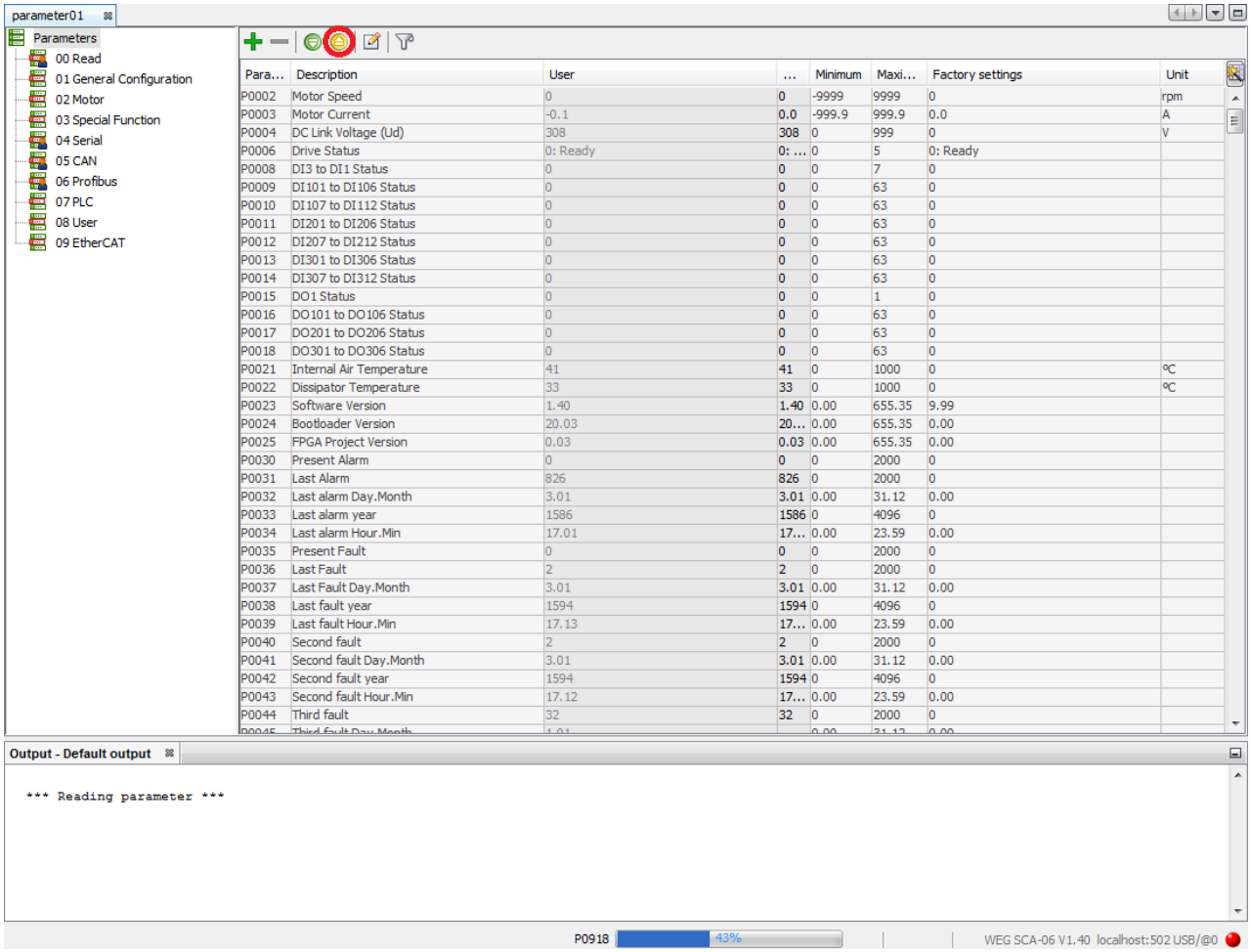
11.3.2.3 Read and Write of Parameters

There are 3 (three) ways to do the reading and writing of the parameters: by means of table, selection and group.

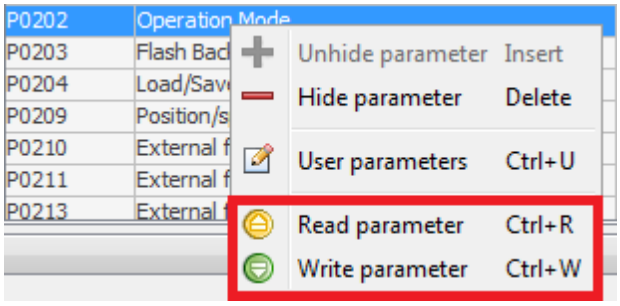
1. **Table writing**. The table writing command will send all visible parameters on the equipment screen. If and error occurs during the sending of some specific parameter, a message will be shown on the output window informing the error. It is important to notice that only visible parameters will be sent; therefore, it is necessary attention to which node of the group of parameters tree you are viewing. Example: If you wish to write all of them without filtering per group, just select the tree root.



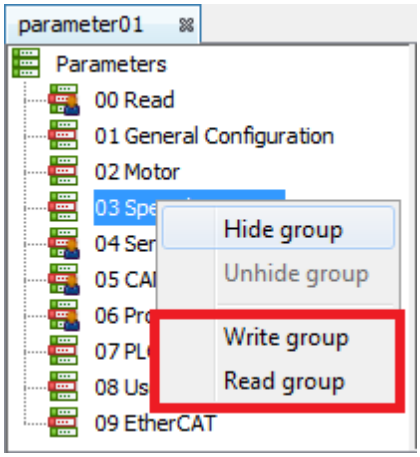
2. **Table reading**. The table reading command will read all the parameters of the equipment. If a error occurs during the reading of some specific parameter, a message will be shown on the output window informing the error. It is important to notice that only visible parameters will be read; therefore, it is necessary attention to which node of the group of parameters tree you are viewing. Example: If you wish to read all of them without filtering per group, just select the tree root.



3. **Reading/writing of specific parameters**. In order to read/write one or more specific parameters, just select them on the table, right click and choose the desired option: read or write parameter.



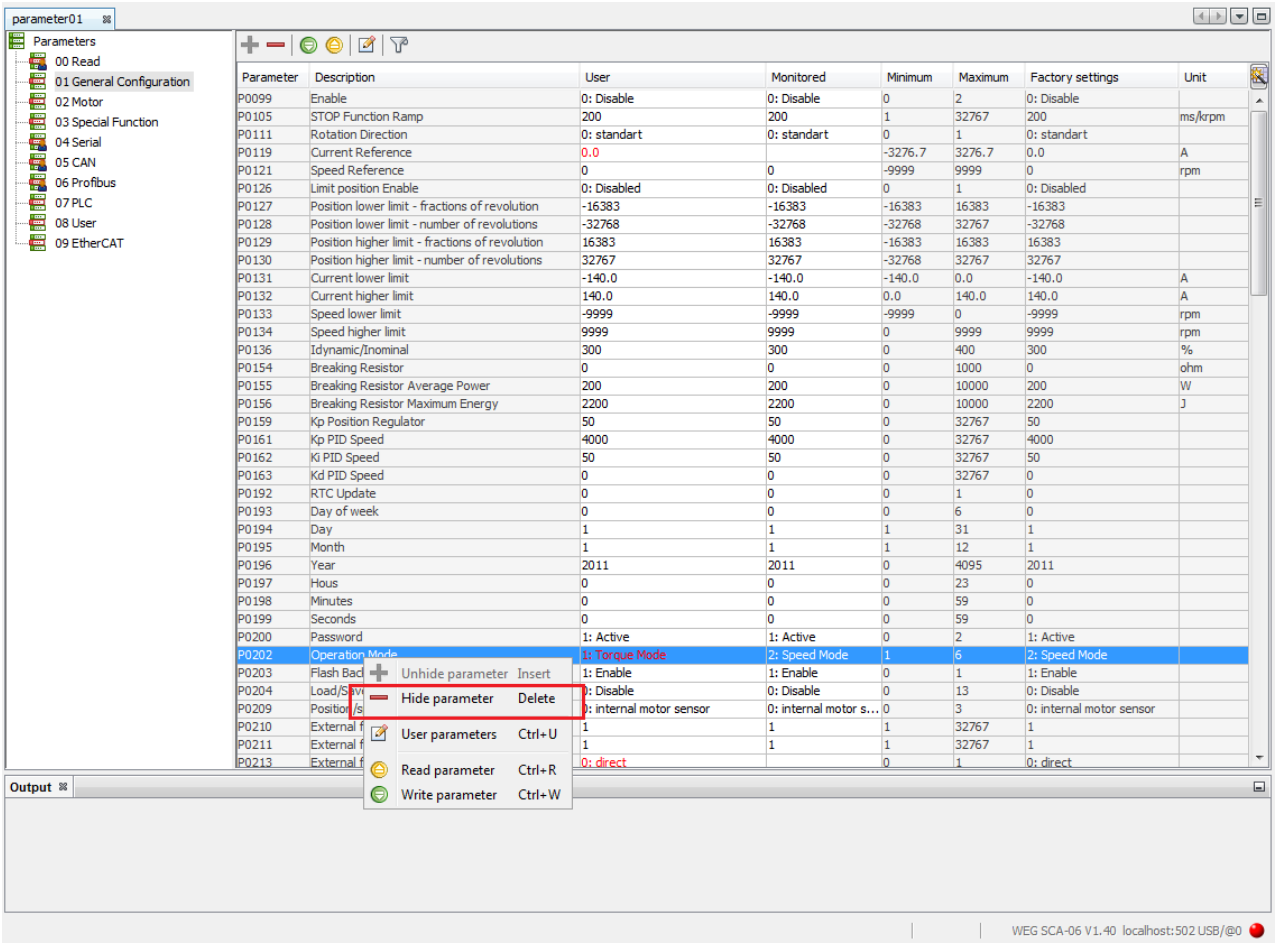
4. **Reading/writing of group of parameters**. In order to read/write only one group of parameters, just select it on the group tree, right click and choose the desired option: read or write group.



11.3.2.4 Hide/Unhide Parameters and Group of Parameters

The parameter can be hidden/unhidden in two ways: individually or in group.

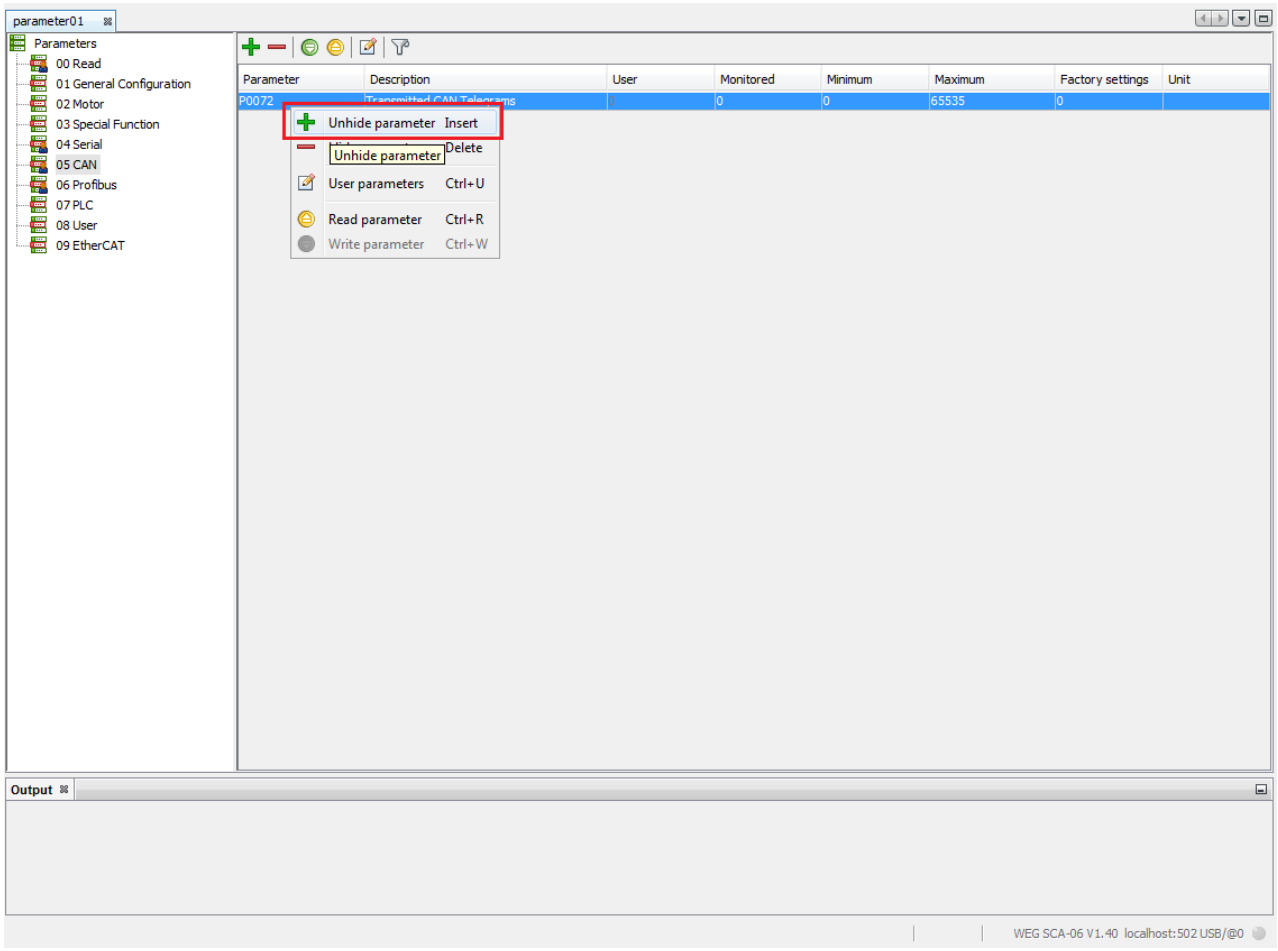
1. **Hide parameters**. In order to hide a parameter individually, just right click on the desired parameters and select the **Hide Parameter** option. You can also press the **Delete** key.

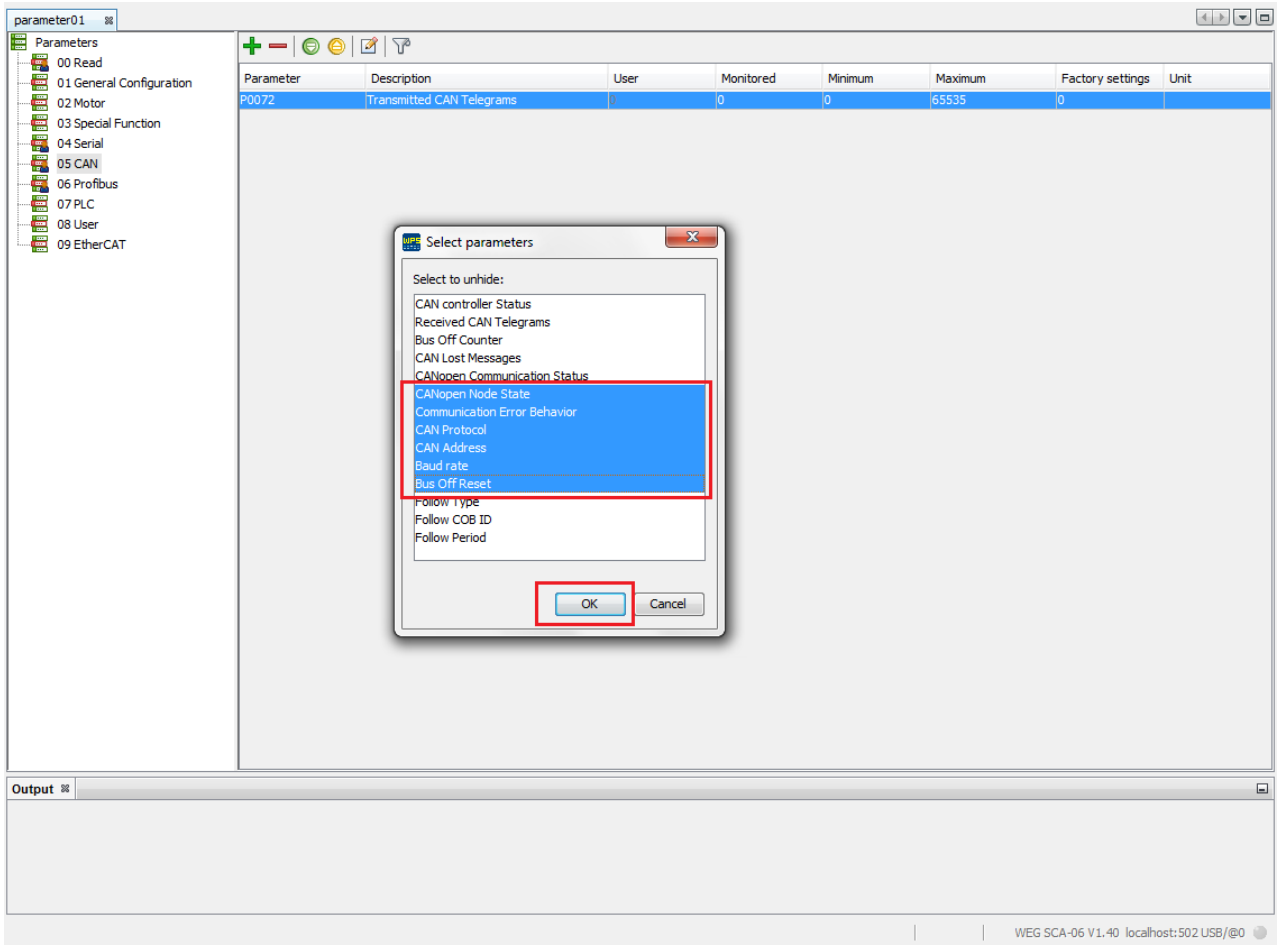


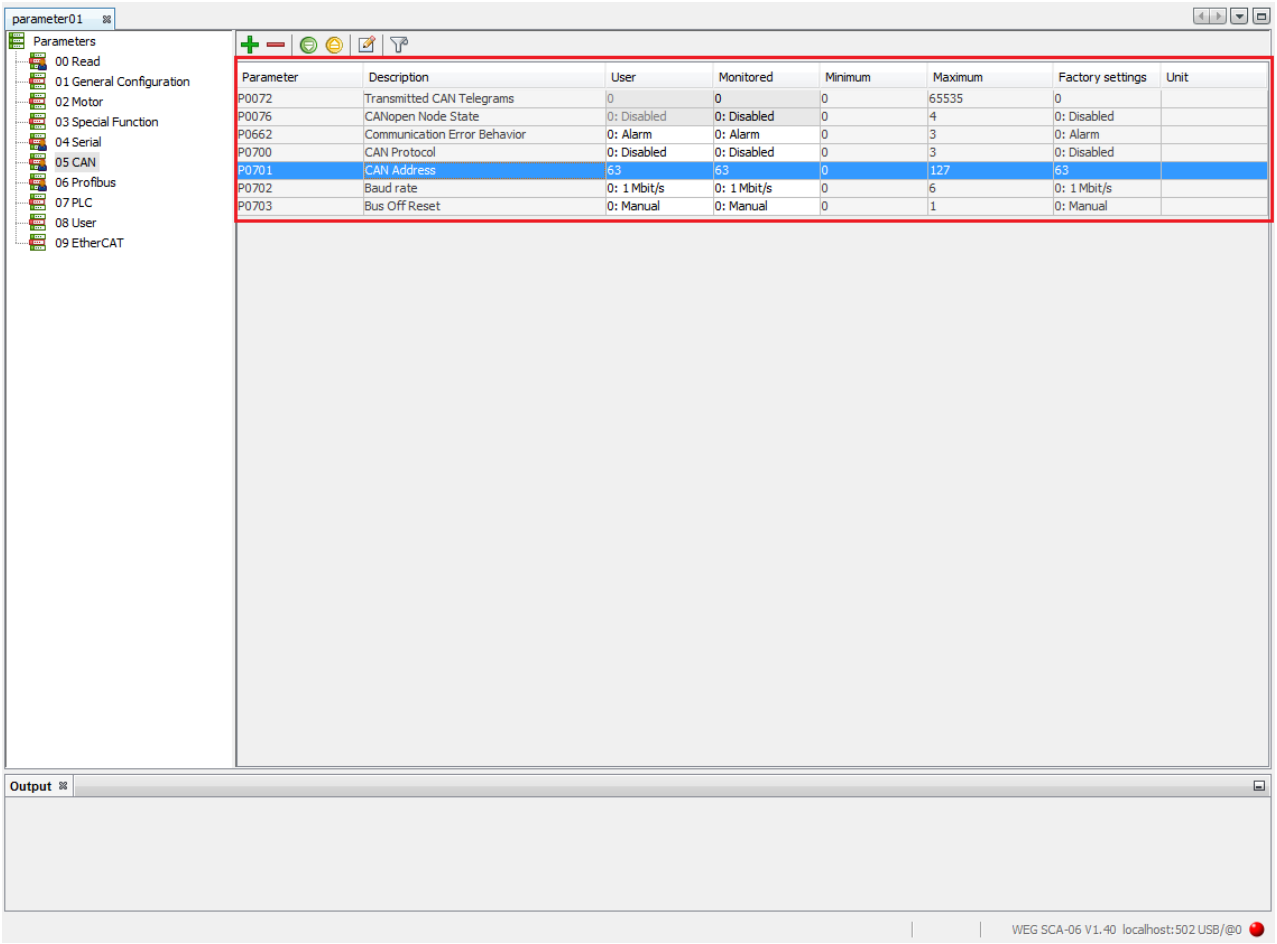
Parameter	Description	User	Monitored	Minimum	Maximum	Factory settings	Unit
P0099	Enable	0: Disable	0: Disable	0	2	0: Disable	
P0105	STOP Function Ramp	200	200	1	32767	200	ms/krpm
P0111	Rotation Direction	0: standart	0: standart	0	1	0: standart	
P0119	Current Reference	0.0		-3276.7	3276.7	0.0	A
P0121	Speed Reference	0	0	-9999	9999	0	rpm
P0126	Limit position Enable	0: Disabled	0: Disabled	0	1	0: Disabled	
P0127	Position lower limit - fractions of revolution	-16383	-16383	-16383	16383	-16383	
P0128	Position lower limit - number of revolutions	-32768	-32768	-32768	32767	-32768	
P0129	Position higher limit - fractions of revolution	16383	16383	-16383	16383	16383	
P0130	Position higher limit - number of revolutions	32767	32767	-32768	32767	32767	
P0131	Current lower limit	-140.0	-140.0	-140.0	0.0	-140.0	A
P0132	Current higher limit	140.0	140.0	0.0	140.0	140.0	A
P0133	Speed lower limit	-9999	-9999	-9999	0	-9999	rpm
P0134	Speed higher limit	9999	9999	0	9999	9999	rpm
P0136	Idynamic/Inominal	300	300	0	400	300	%
P0154	Breaking Resistor	0	0	0	1000	0	ohm
P0155	Breaking Resistor Average Power	200	200	0	10000	200	W
P0156	Breaking Resistor Maximum Energy	2200	2200	0	10000	2200	J
P0159	Kp Position Regulator	50	50	0	32767	50	
P0161	Kp PID Speed	4000	4000	0	32767	4000	
P0162	Ki PID Speed	50	50	0	32767	50	
P0163	Kd PID Speed	0	0	0	32767	0	
P0192	RTC Update	0	0	0	1	0	
P0193	Day of week	0	0	0	6	0	
P0194	Day	1	1	1	31	1	
P0195	Month	1	1	1	12	1	
P0196	Year	2011	2011	0	4095	2011	
P0197	Hous	0	0	0	23	0	
P0198	Minutes	0	0	0	59	0	
P0199	Seconds	0	0	0	59	0	
P0200	Password	1: Active	1: Active	0	2	1: Active	
P0202	Operation Mode	1: Torque Mode	2: Speed Mode	1	6	2: Speed Mode	
P0203	Flash Bac	1: Enable	1: Enable	0	1	1: Enable	
P0204	Load/S	0: Disable	0: Disable	0	13	0: Disable	
P0209	Position/s	0: internal motor sensor	0: internal motor s...	0	3	0: internal motor sensor	
P0210	External f	1	1	1	32767	1	
P0211	External f	1	1	1	32767	1	
P0213	External f	0: direct		0	1	0: direct	

2. **Unhide Parameters**. In order to show hidden parameters, right click and choose the **Unhide Parameters**

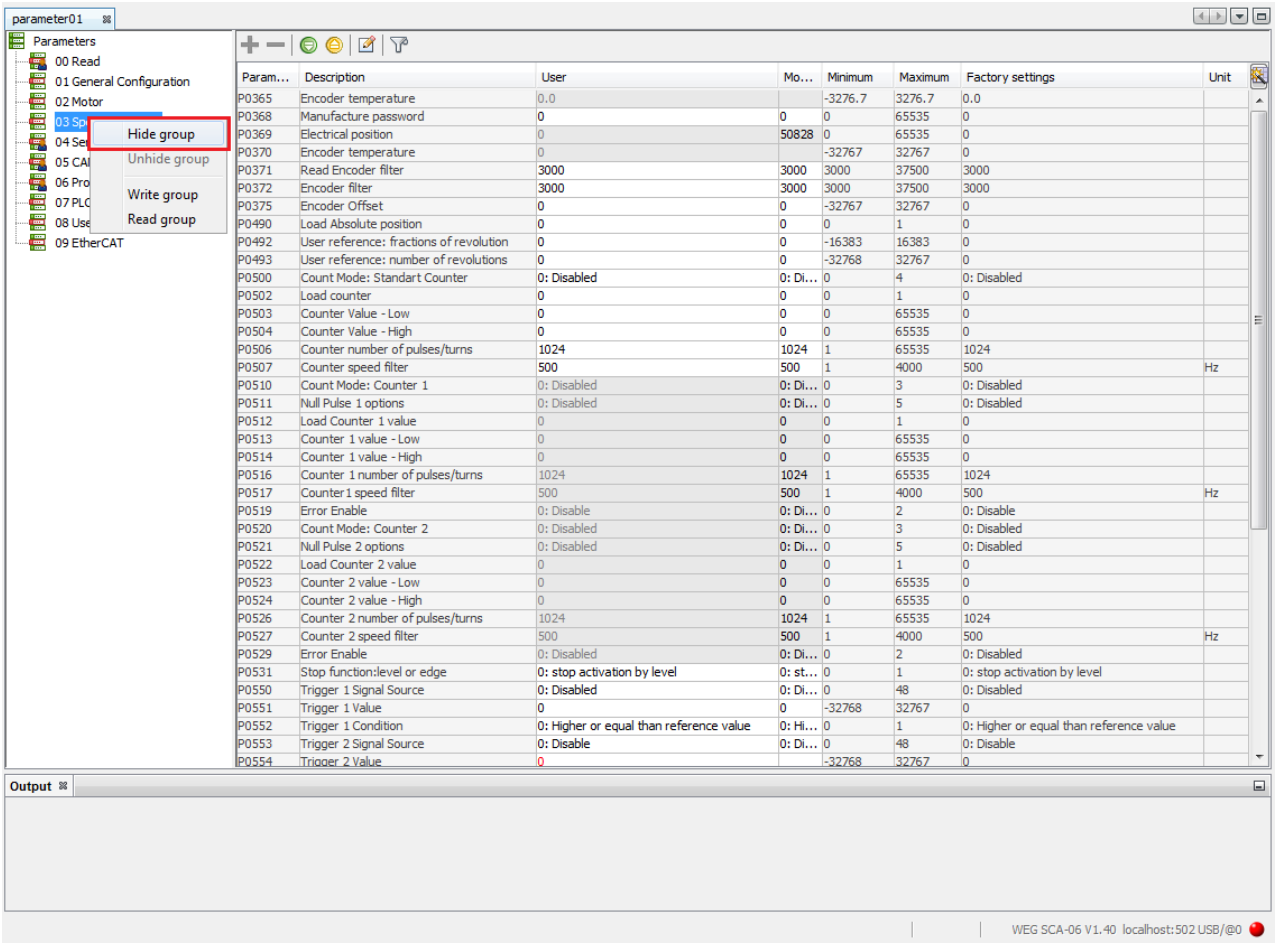
or press the **Insert** key. Then, a window will open and show the hidden parameters. Now, you just have to select the desired parameters and confirm.
Note: The parameters shown on this new window are only those which belong to the current filter according to the selection on the parameter group tree. In the figures below, the CAN group is selected; that means that only the hidden parameters of this group will be shown.

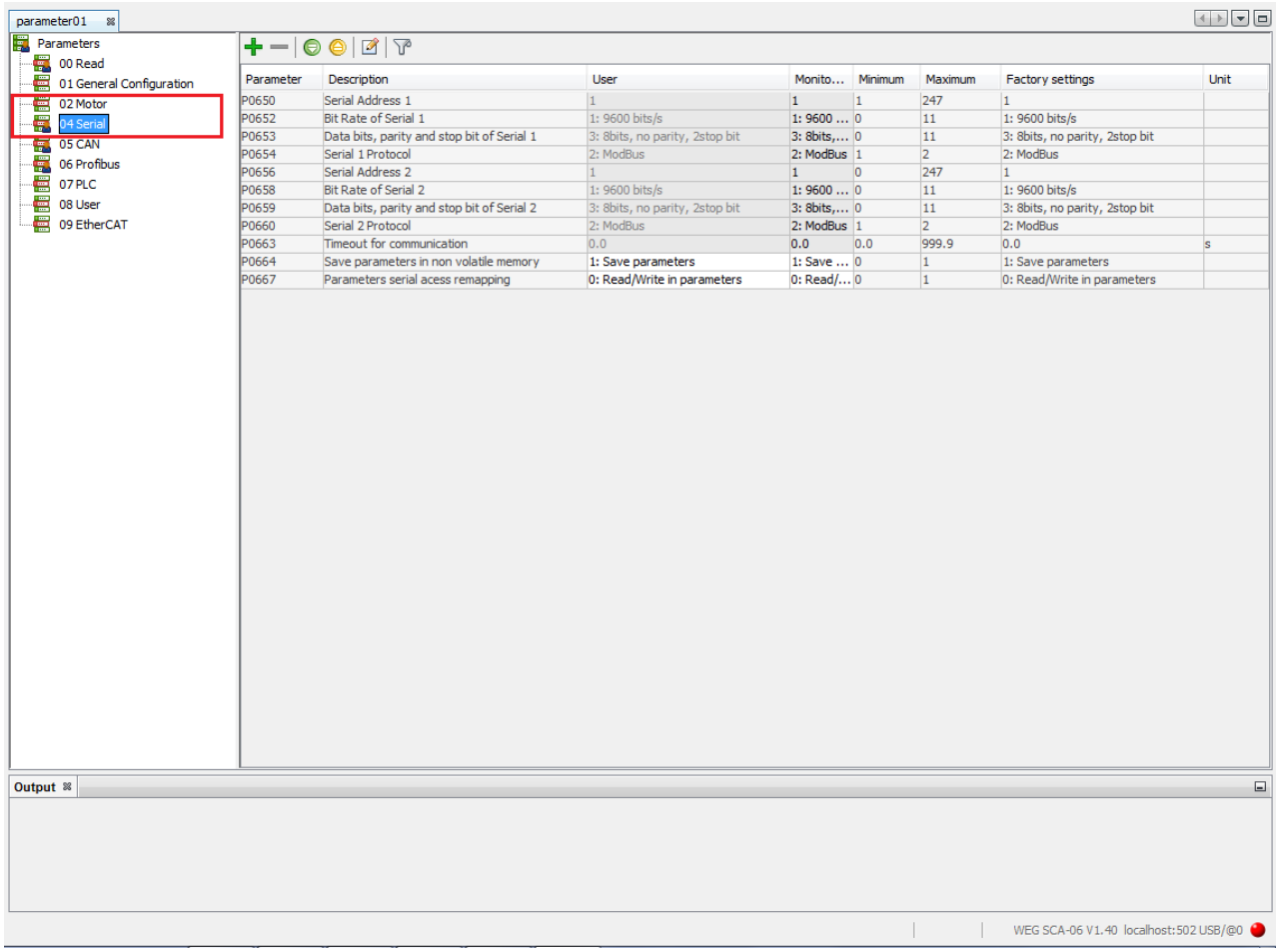



parameter01
Parameters
00 Read
01 General Configuration
02 Motor
03 Special Function
04 Serial
05 CAN
06 Profibus
07 PLC
08 User
09 EtherCAT
Parameter
Description
User
Monitored
Minimum
Maximum
Factory settings
Unit
P0072
Transmitted CAN Telegrams
0
0
0
65535
0
Select parameters
Select to unhide:
CAN controller Status
Received CAN Telegrams
Bus Off Counter
CAN Lost Messages
CANopen Communication Status
CANopen Node State
Communication Error Behavior
CAN Protocol
CAN Address
Baud rate
Bus Off Reset
Follow Type
Follow COB ID
Follow Period
OK
Cancel
Output
WEG SCA-06 V1.40 localhost:502 USB/@0

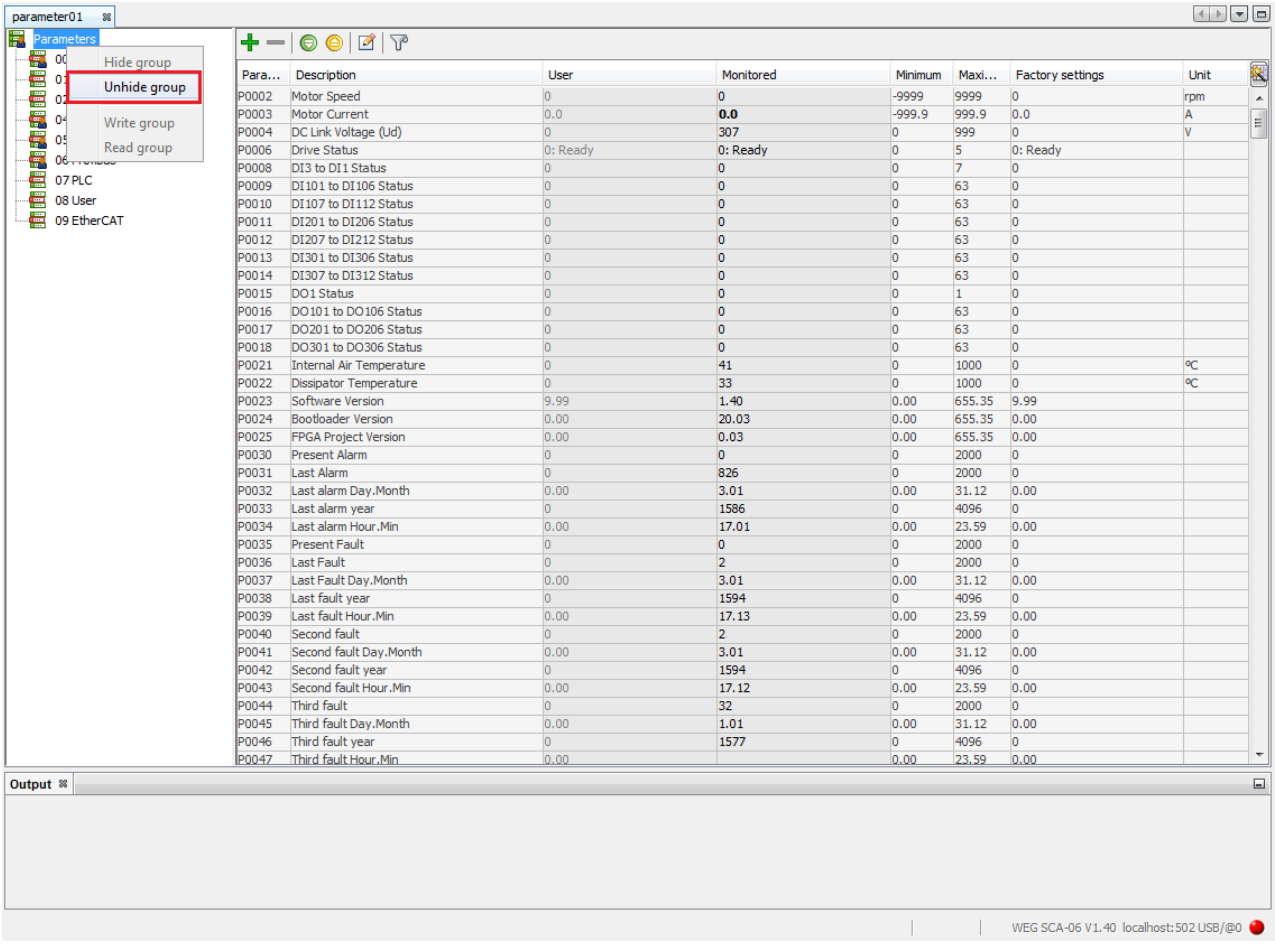


3. **Hide Group of Parameters**. In order to hide a group of parameters, just select the group on the tree and use the **Hide Group** option.

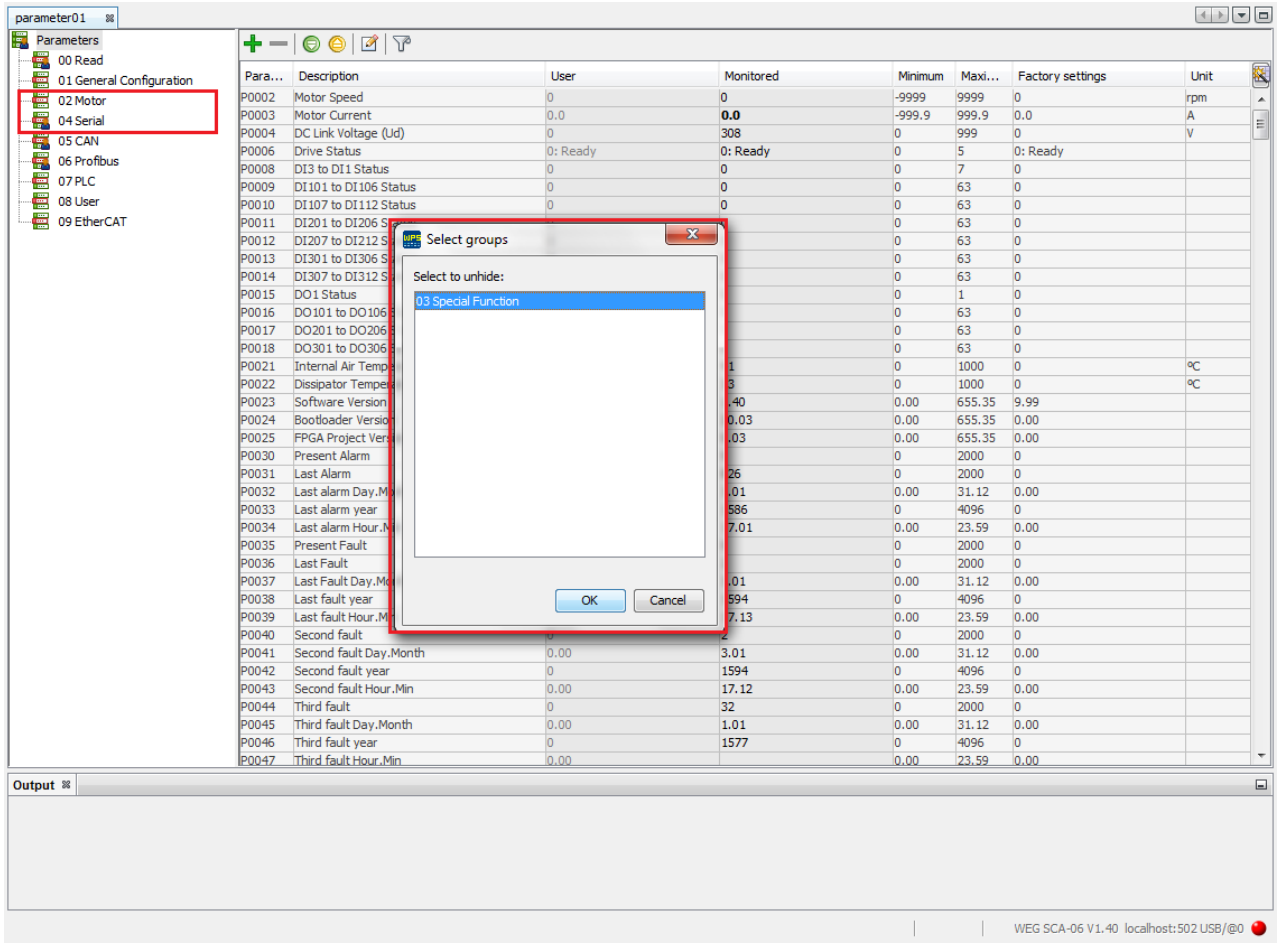

parameter01
Parameters
00 Read
01 General Configuration
02 Motor
03 Sp
04 Se
05 CA
06 Pro
07 PLC
08 Use
09 EtherCAT
Hide group
Unhide group
Write group
Read group
Param...
Description
User
Mo...
Minimum
Maximum
Factory settings
Unit
P0365 Encoder temperature 0.0 -3276.7 3276.7 0.0
P0368 Manufacture password 0 0 0 65535 0
P0369 Electrical position 0 50828 0 65535 0
P0370 Encoder temperature 0 -32767 32767 0
P0371 Read Encoder filter 3000 3000 3000 37500 3000
P0372 Encoder filter 3000 3000 3000 37500 3000
P0375 Encoder Offset 0 0 -32767 32767 0
P0490 Load Absolute position 0 0 0 1 0
P0492 User reference: fractions of revolution 0 0 -16383 16383 0
P0493 User reference: number of revolutions 0 0 -32768 32767 0
P0500 Count Mode: Standart Counter 0: Disabled 0: Di... 0 4 0: Disabled
P0502 Load counter 0 0 0 1 0
P0503 Counter Value - Low 0 0 0 65535 0
P0504 Counter Value - High 0 0 0 65535 0
P0506 Counter number of pulses/turns 1024 1024 1 65535 1024
P0507 Counter speed filter 500 500 1 4000 500 Hz
P0510 Count Mode: Counter 1 0: Disabled 0: Di... 0 3 0: Disabled
P0511 Null Pulse 1 options 0: Disabled 0: Di... 0 5 0: Disabled
P0512 Load Counter 1 value 0 0 0 1 0
P0513 Counter 1 value - Low 0 0 0 65535 0
P0514 Counter 1 value - High 0 0 0 65535 0
P0516 Counter 1 number of pulses/turns 1024 1024 1 65535 1024
P0517 Counter1 speed filter 500 500 1 4000 500 Hz
P0519 Error Enable 0: Disable 0: Di... 0 2 0: Disable
P0520 Count Mode: Counter 2 0: Disabled 0: Di... 0 3 0: Disabled
P0521 Null Pulse 2 options 0: Disabled 0: Di... 0 5 0: Disabled
P0522 Load Counter 2 value 0 0 0 1 0
P0523 Counter 2 value - Low 0 0 0 65535 0
P0524 Counter 2 value - High 0 0 0 65535 0
P0526 Counter 2 number of pulses/turns 1024 1024 1 65535 1024
P0527 Counter 2 speed filter 500 500 1 4000 500 Hz
P0529 Error Enable 0: Disabled 0: Di... 0 2 0: Disabled
P0531 Stop function:level or edge 0: stop activation by level 0: st... 0 1 0: stop activation by level
P0550 Trigger 1 Signal Source 0: Disabled 0: Di... 0 48 0: Disabled
P0551 Trigger 1 Value 0 0 -32768 32767 0
P0552 Trigger 1 Condition 0: Higher or equal than reference value 0: Hi... 0 1 0: Higher or equal than reference value
P0553 Trigger 2 Signal Source 0: Disable 0: Di... 0 48 0: Disable
P0554 Trigger 2 Value 0 -32768 32767 0
Output
WEG SCA-06 V1.40 localhost:502 USB/@0

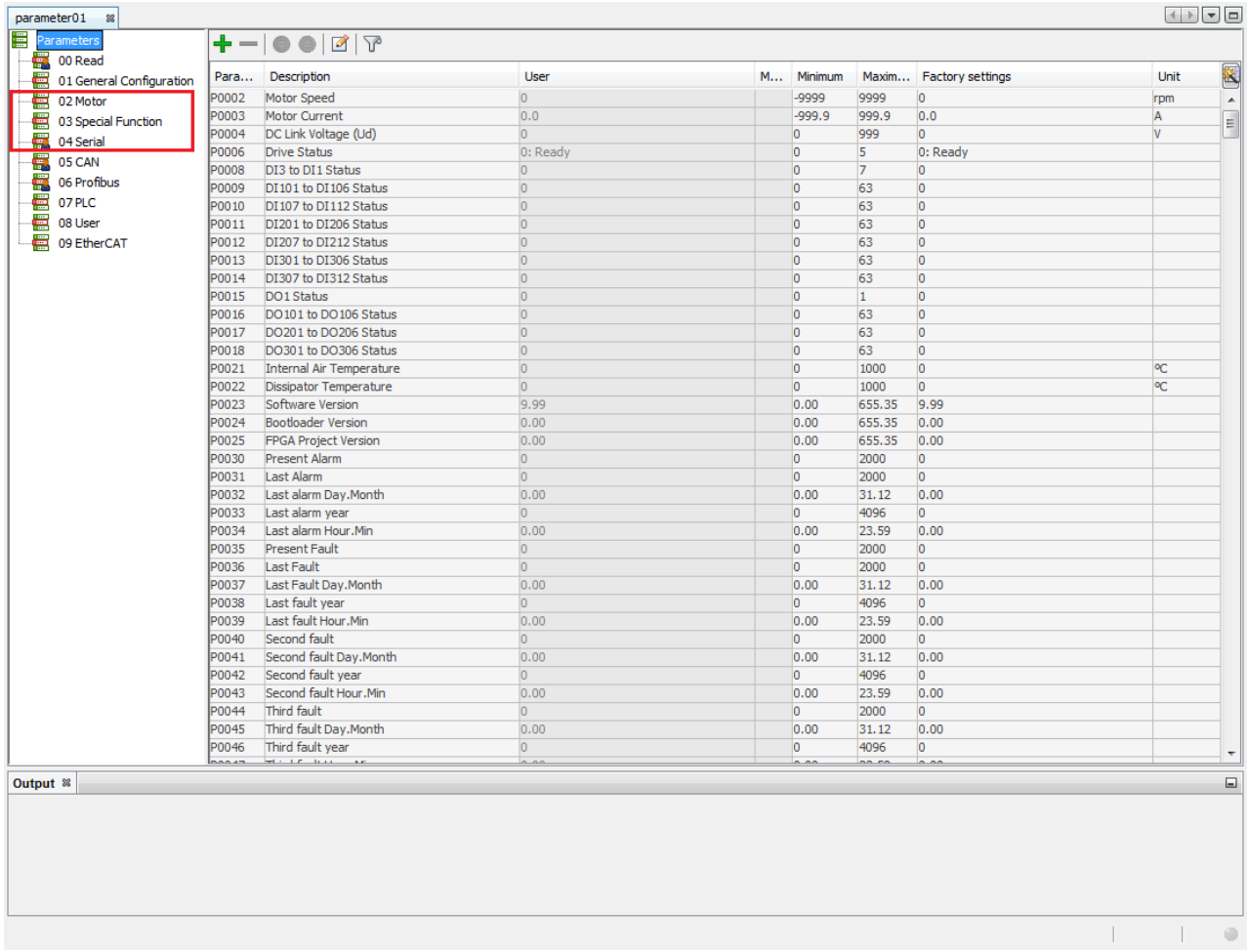


4. **Unhide Group of Parameters**. In order to show a hidden group of parameters, just select the root of the group tree and select the **Unhide Group** option. A window will open showing the groups that are hidden; then just select the group you wish to unhide.



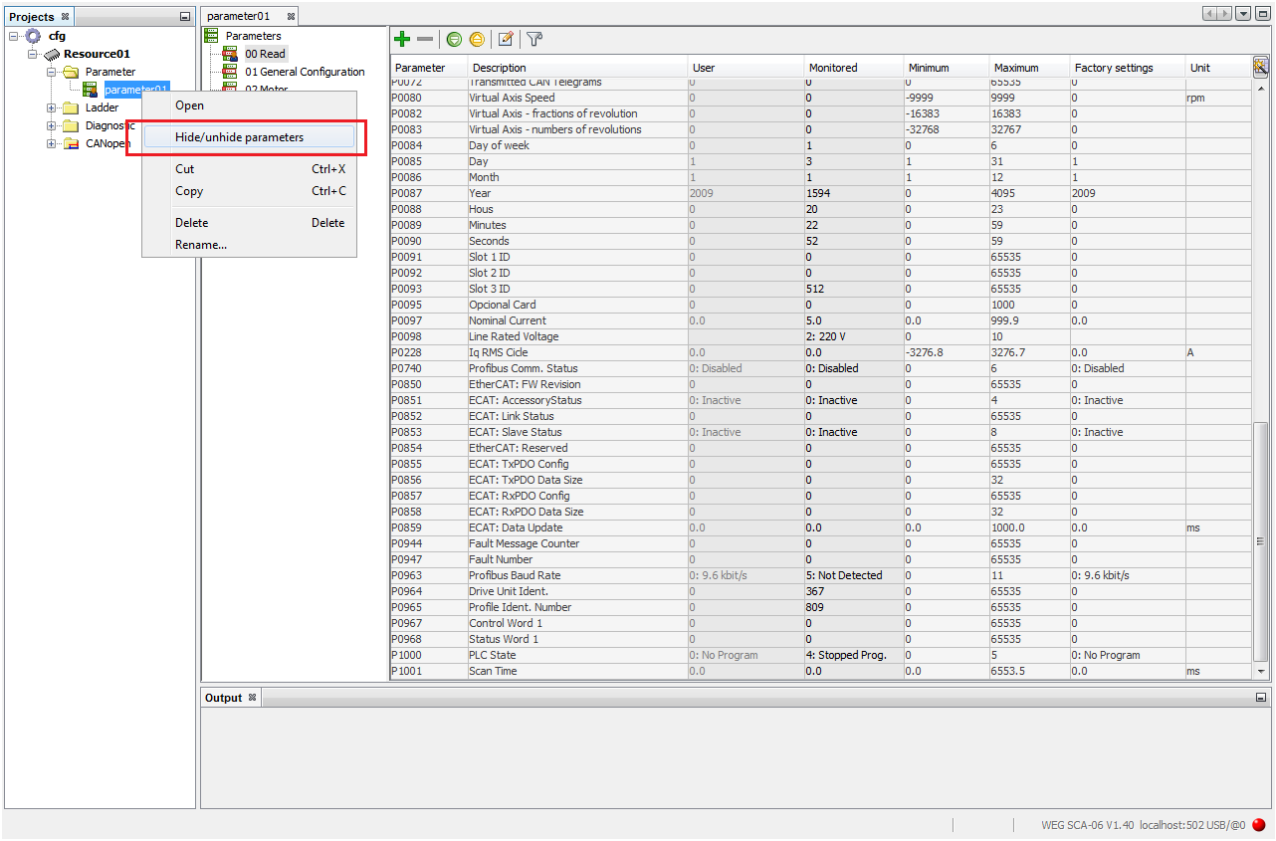
Para...	Description	User	Monitored	Minimum	Maxi...	Factory settings	Unit
P0002	Motor Speed	0	0	-9999	9999	0	rpm
P0003	Motor Current	0.0	**0.0**	-999.9	999.9	0.0	A
P0004	DC Link Voltage (Ud)	0	307	0	999	0	V
P0006	Drive Status	0: Ready	0: Ready	0	5	0: Ready	
P0008	DI3 to DI1 Status	0	0	0	7	0	
P0009	DI101 to DI106 Status	0	0	0	63	0	
P0010	DI107 to DI112 Status	0	0	0	63	0	
P0011	DI201 to DI206 Status	0	0	0	63	0	
P0012	DI207 to DI212 Status	0	0	0	63	0	
P0013	DI301 to DI306 Status	0	0	0	63	0	
P0014	DI307 to DI312 Status	0	0	0	63	0	
P0015	DO1 Status	0	0	0	1	0	
P0016	DO101 to DO106 Status	0	0	0	63	0	
P0017	DO201 to DO206 Status	0	0	0	63	0	
P0018	DO301 to DO306 Status	0	0	0	63	0	
P0021	Internal Air Temperature	0	41	0	1000	0	°C
P0022	Dissipator Temperature	0	33	0	1000	0	°C
P0023	Software Version	9.99	1.40	0.00	655.35	9.99	
P0024	Bootloader Version	0.00	20.03	0.00	655.35	0.00	
P0025	FPGA Project Version	0.00	0.03	0.00	655.35	0.00	
P0030	Present Alarm	0	0	0	2000	0	
P0031	Last Alarm	0	826	0	2000	0	
P0032	Last alarm Day.Month	0.00	3.01	0.00	31.12	0.00	
P0033	Last alarm year	0	1586	0	4096	0	
P0034	Last alarm Hour.Min	0.00	17.01	0.00	23.59	0.00	
P0035	Present Fault	0	0	0	2000	0	
P0036	Last Fault	0	2	0	2000	0	
P0037	Last Fault Day.Month	0.00	3.01	0.00	31.12	0.00	
P0038	Last fault year	0	1594	0	4096	0	
P0039	Last fault Hour.Min	0.00	17.13	0.00	23.59	0.00	
P0040	Second fault	0	2	0	2000	0	
P0041	Second fault Day.Month	0.00	3.01	0.00	31.12	0.00	
P0042	Second fault year	0	1594	0	4096	0	
P0043	Second fault Hour.Min	0.00	17.12	0.00	23.59	0.00	
P0044	Third fault	0	32	0	2000	0	
P0045	Third fault Day.Month	0.00	1.01	0.00	31.12	0.00	
P0046	Third fault year	0	1577	0	4096	0	
P0047	Third fault Hour.Min	0.00		0.00	23.59	0.00	

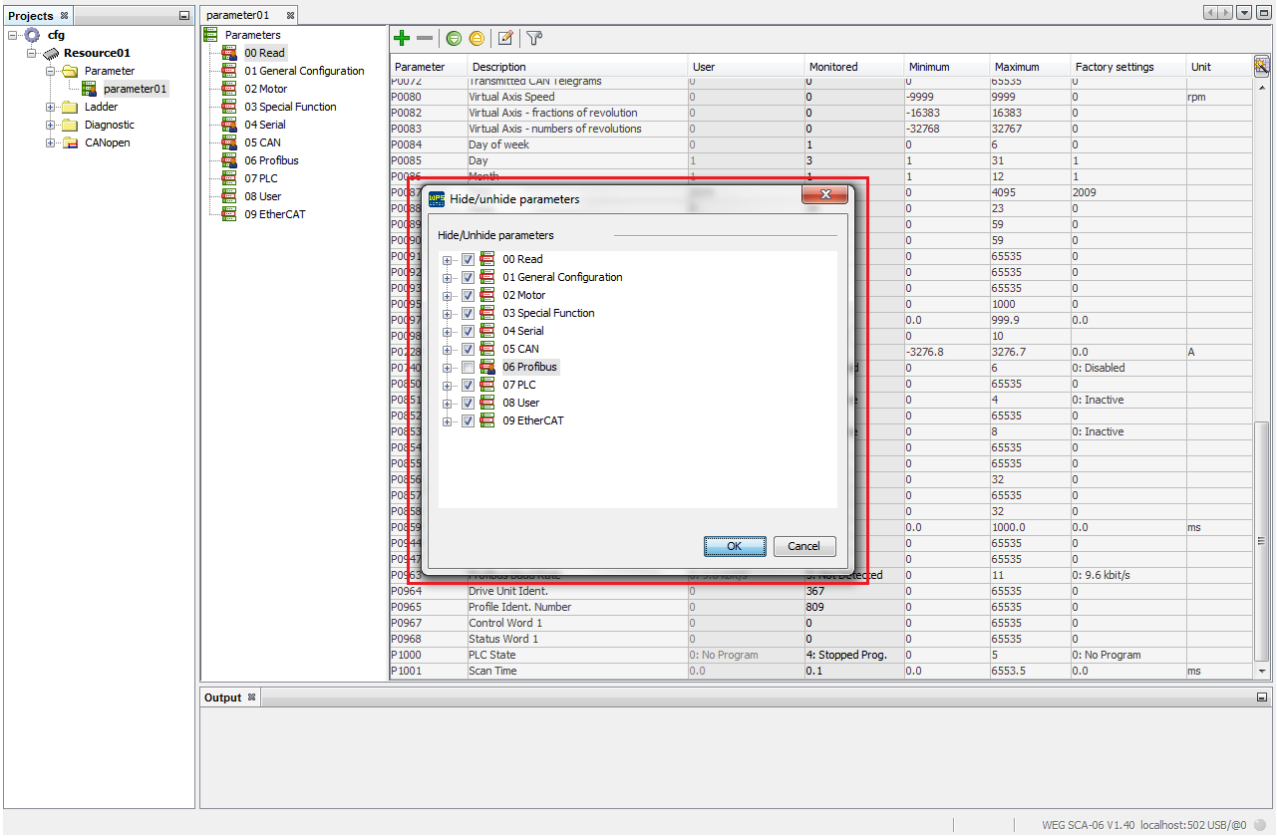

parameter01
Parameters
00 Read
01 General Configuration
02 Motor
04 Serial
05 CAN
06 Profibus
07 PLC
08 User
09 EtherCAT
Select groups
Select to unhide:
03 Special Function
OK
Cancel
Output
WEG SCA-06 V1.40 localhost:502 USB/@0



Para...	Description	User	M...	Minimum	Maxim...	Factory settings	Unit
P0002	Motor Speed	0		-9999	9999	0	rpm
P0003	Motor Current	0.0		-999.9	999.9	0.0	A
P0004	DC Link Voltage (Ud)	0		0	999	0	V
P0006	Drive Status	0: Ready		0	5	0: Ready	
P0008	DI3 to DI1 Status	0		0	7	0	
P0009	DI101 to DI106 Status	0		0	63	0	
P0010	DI107 to DI112 Status	0		0	63	0	
P0011	DI201 to DI206 Status	0		0	63	0	
P0012	DI207 to DI212 Status	0		0	63	0	
P0013	DI301 to DI306 Status	0		0	63	0	
P0014	DI307 to DI312 Status	0		0	63	0	
P0015	DO1 Status	0		0	1	0	
P0016	DO101 to DO106 Status	0		0	63	0	
P0017	DO201 to DO206 Status	0		0	63	0	
P0018	DO301 to DO306 Status	0		0	63	0	
P0021	Internal Air Temperature	0		0	1000	0	°C
P0022	Dissipator Temperature	0		0	1000	0	°C
P0023	Software Version	9.99		0.00	655.35	9.99	
P0024	Bootloader Version	0.00		0.00	655.35	0.00	
P0025	FPGA Project Version	0.00		0.00	655.35	0.00	
P0030	Present Alarm	0		0	2000	0	
P0031	Last Alarm	0		0	2000	0	
P0032	Last alarm Day.Month	0.00		0.00	31.12	0.00	
P0033	Last alarm year	0		0	4096	0	
P0034	Last alarm Hour.Min	0.00		0.00	23.59	0.00	
P0035	Present Fault	0		0	2000	0	
P0036	Last Fault	0		0	2000	0	
P0037	Last Fault Day.Month	0.00		0.00	31.12	0.00	
P0038	Last fault year	0		0	4096	0	
P0039	Last fault Hour.Min	0.00		0.00	23.59	0.00	
P0040	Second fault	0		0	2000	0	
P0041	Second fault Day.Month	0.00		0.00	31.12	0.00	
P0042	Second fault year	0		0	4096	0	
P0043	Second fault Hour.Min	0.00		0.00	23.59	0.00	
P0044	Third fault	0		0	2000	0	
P0045	Third fault Day.Month	0.00		0.00	31.12	0.00	
P0046	Third fault year	0		0	4096	0	

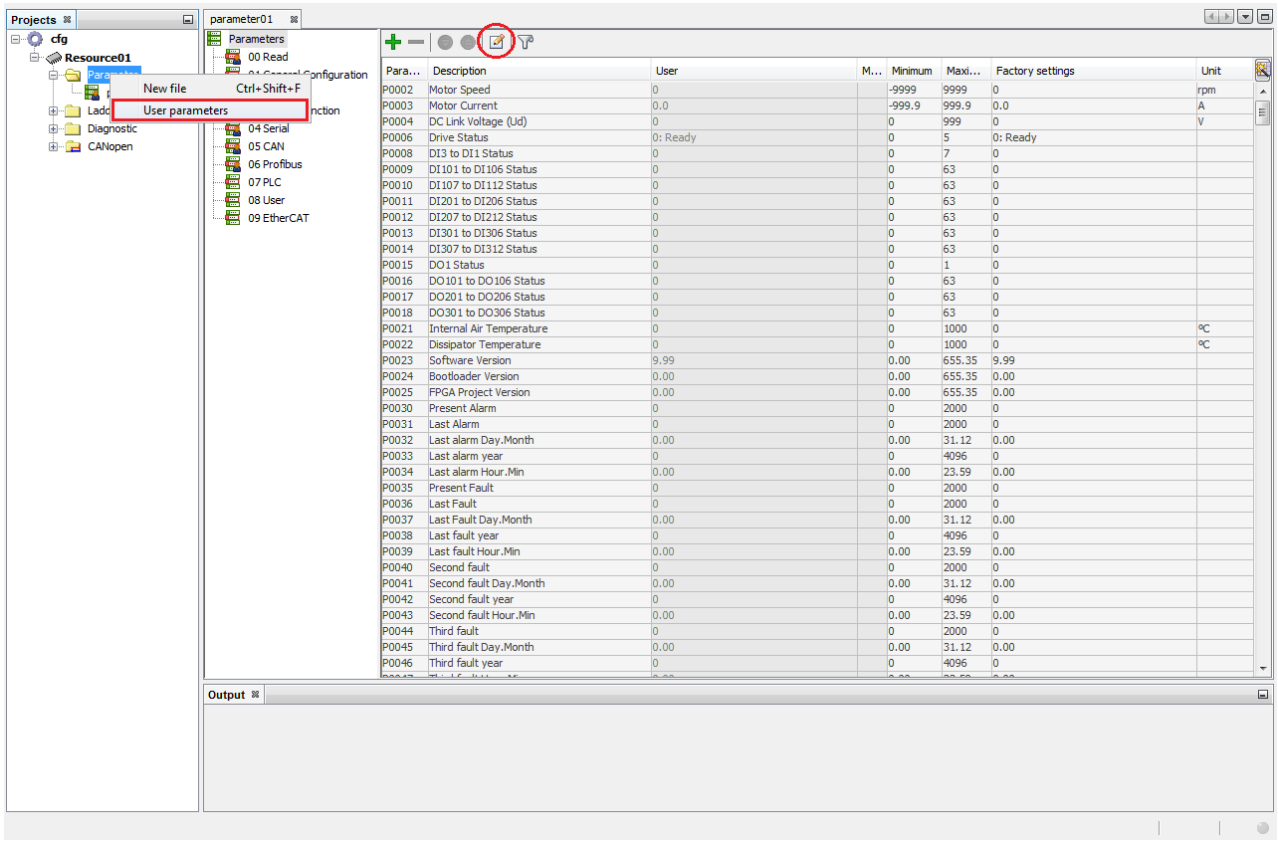
5. **Hide and Show Parameters and Groups of Parameters**. By means of this option, you have full control of the parameters and groups of parameters. It is possible to hide and unhide individual parameters, multiple parameters, individual groups and multiple groups in the same action.


Projects
cfg
Resource01
Parameter
parameter01
Ladder
Diagnostic
CANopen
Open
Hide/unhide parameters
Cut Ctrl+X
Copy Ctrl+C
Delete Delete
Rename...
parameter01
Parameters
00 Read
01 General Configuration
Parameter
Description
User
Monitored
Minimum
Maximum
Factory settings
Unit
P0080 Virtual Axis Speed
P0082 Virtual Axis - fractions of revolution
P0083 Virtual Axis - numbers of revolutions
P0084 Day of week
P0085 Day
P0086 Month
P0087 Year
P0088 Hous
P0089 Minutes
P0090 Seconds
P0091 Slot 1 ID
P0092 Slot 2 ID
P0093 Slot 3 ID
P0095 Opcional Card
P0097 Nominal Current
P0098 Line Rated Voltage
P0228 Iq RMS Cicle
P0740 Profibus Comm. Status
P0850 EtherCAT: FW Revision
P0851 ECAT: AccessoryStatus
P0852 ECAT: Link Status
P0853 ECAT: Slave Status
P0854 EtherCAT: Reserved
P0855 ECAT: TxPDO Config
P0856 ECAT: TxPDO Data Size
P0857 ECAT: RxPDO Config
P0858 ECAT: RxPDO Data Size
P0859 ECAT: Data Update
P0944 Fault Message Counter
P0947 Fault Number
P0963 Profibus Baud Rate
P0964 Drive Unit Ident.
P0965 Profile Ident. Number
P0967 Control Word 1
P0968 Status Word 1
P1000 PLC State
P1001 Scan Time
Output
WEG SCA-06 V1.40 localhost:502 USB/@0


Projects
cfg
Resource01
Parameter
parameter01
Ladder
Diagnostic
CANopen
parameter01
Parameters
00 Read
01 General Configuration
02 Motor
03 Special Function
04 Serial
05 CAN
06 Profibus
07 PLC
08 User
09 EtherCAT
Hide/unhide parameters
Hide/Unhide parameters
OK
Cancel
Output
WEG SCA-06 V1.40 localhost:502 USB/@0

11.3.2.5 User Parameters

In order to open the configuration screen of the user parameters, just click on the **User Parameters** option on the Parameter node of the project tree or click on the icon indicated on the tool bar of the parameter file.



Para...	Description	User	M...	Minimum	Maxi...	Factory settings	Unit
P0002	Motor Speed	0		-9999	9999	0	rpm
P0003	Motor Current	0.0		-999.9	999.9	0.0	A
P0004	DC Link Voltage (Ud)	0		0	999	0	V
P0006	Drive Status	0: Ready		0	5	0: Ready	
P0008	DI3 to DI1 Status	0		0	7	0	
P0009	DI101 to DI106 Status	0		0	63	0	
P0010	DI107 to DI112 Status	0		0	63	0	
P0011	DI201 to DI206 Status	0		0	63	0	
P0012	DI207 to DI212 Status	0		0	63	0	
P0013	DI301 to DI306 Status	0		0	63	0	
P0014	DI307 to DI312 Status	0		0	63	0	
P0015	DO1 Status	0		0	1	0	
P0016	DO101 to DO106 Status	0		0	63	0	
P0017	DO201 to DO206 Status	0		0	63	0	
P0018	DO301 to DO306 Status	0		0	63	0	
P0021	Internal Air Temperature	0		0	1000	0	°C
P0022	Dissipator Temperature	0		0	1000	0	°C
P0023	Software Version	9.99		0.00	655.35	9.99	
P0024	Bootloader Version	0.00		0.00	655.35	0.00	
P0025	FPGA Project Version	0.00		0.00	655.35	0.00	
P0030	Present Alarm	0		0	2000	0	
P0031	Last Alarm	0		0	2000	0	
P0032	Last alarm Day.Month	0.00		0.00	31.12	0.00	
P0033	Last alarm year	0		0	4096	0	
P0034	Last alarm Hour.Min	0.00		0.00	23.59	0.00	
P0035	Present Fault	0		0	2000	0	
P0036	Last Fault	0		0	2000	0	
P0037	Last Fault Day.Month	0.00		0.00	31.12	0.00	
P0038	Last fault year	0		0	4096	0	
P0039	Last fault Hour.Min	0.00		0.00	23.59	0.00	
P0040	Second fault	0		0	2000	0	
P0041	Second fault Day.Month	0.00		0.00	31.12	0.00	
P0042	Second fault year	0		0	4096	0	
P0043	Second fault Hour.Min	0.00		0.00	23.59	0.00	
P0044	Third fault	0		0	2000	0	
P0045	Third fault Day.Month	0.00		0.00	31.12	0.00	
P0046	Third fault year	0		0	4096	0	

Configuration Table.

On the user parameter configuration table, it is possible to define several attributes to the parameters, such as description, minimum and maximum values, unit, digits, data type, etc.

NOTE!
These settings will be automatically displayed in the parameter table. However, to be sent to the device, you need to *download* the resource.

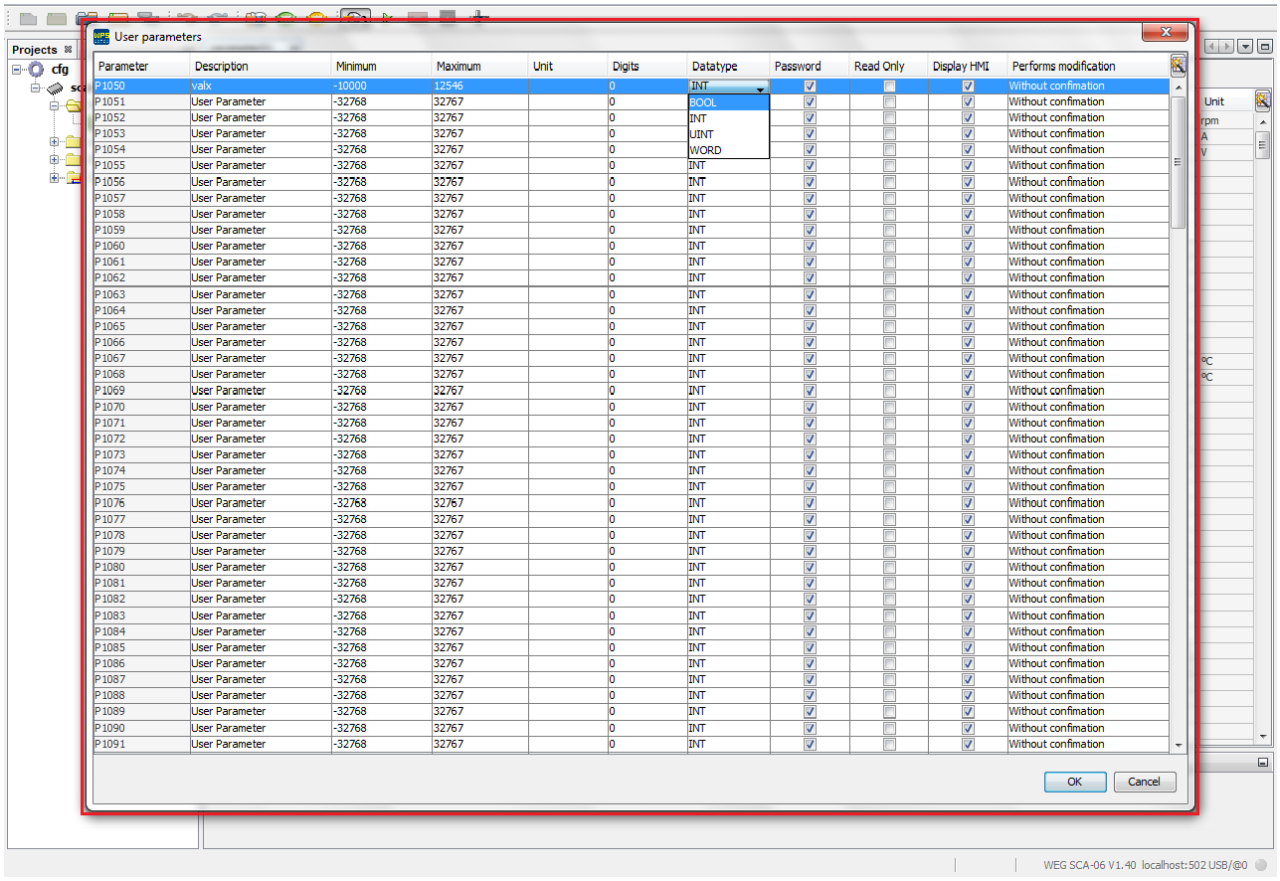


Table fields:

- Parameter: User parameter identification.
- Description: Description of the user parameter in the parameter table. On devices that have text-based HMIs, the description is sent to the machine and displayed on the HMI.
- Minimum: Minimum input value for parameter.
- Maximum: Maximum input value for parameter.
- Unit: Unit displayed on the device's HMI.
- Default: Value loaded when restore factory default is selected.
- Retentive: Retain value after rebooting devices.
- Hexadecimal: Displays the value in hexadecimal.
- Digits: Number of decimal digits for displaying value.
- Datatype: Parameter datatype used by the ladder application.
- Password: Enables password request by changing parameter value.

- Read only: It does not allow the writing of values in the parameter by the communication network or the HMI. Writing is done only by the ladder application.

- Display HMI: Displays the parameter in the HMI.

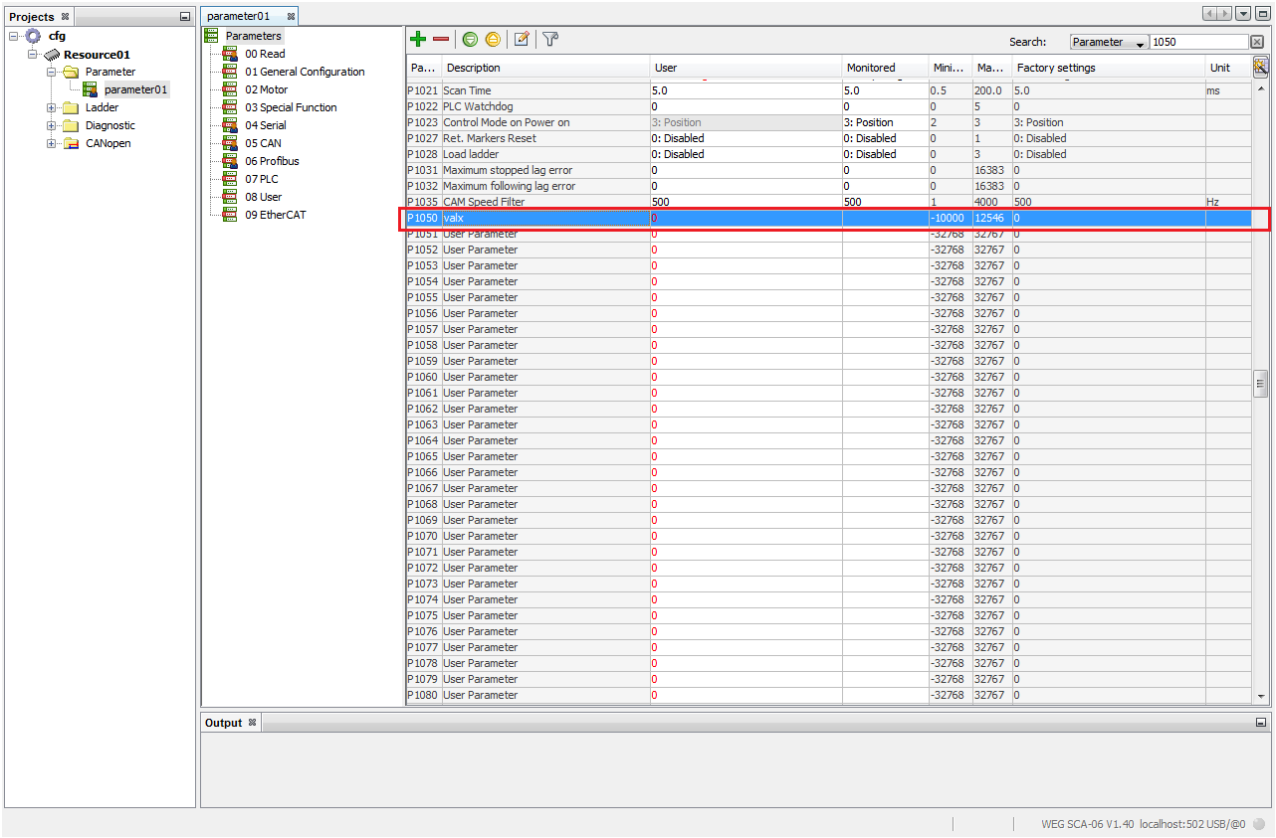
- Performs modification: Confirmation options when changing the parameter:
o No confirmation: Does not prompt for confirmation when changing parameter.
o With confirmation and engine stopped: Request confirmation and allow change only with engine stopped.
o With confirmation: Prompt for confirmation when changing parameter.

- Stopped motor: Perform change only with motor stopped.

- Help: On devices that have text-based HMI, you can edit a help text for the parameter.

View the user parameter

In the parameter table, the user parameters will be shown as they are configured on the configuration screen.



11.4 CFW501

Enter topic text here.

11.4.1 Description

With a modern design and power range of 0.25 to 30 hp, the CFW501 is a high-performance variable speed drive that assists in the speed and torque control of three-phase induction motors. The device also features sensorless vector, vector encoder or scalar, SoftPLC, which adds functions of PLC (Programmable Logic Controller), Pump Genius, which features dedicated functions for pumping and selectable plug-in modules, which provide a flexible and optimized for any application.

Refer to the user's manual of the CFW501 for further details about the product.

NOTE!

This product does not have the Ladder tool available in WPS.
You can use the WLP application if this feature is required.

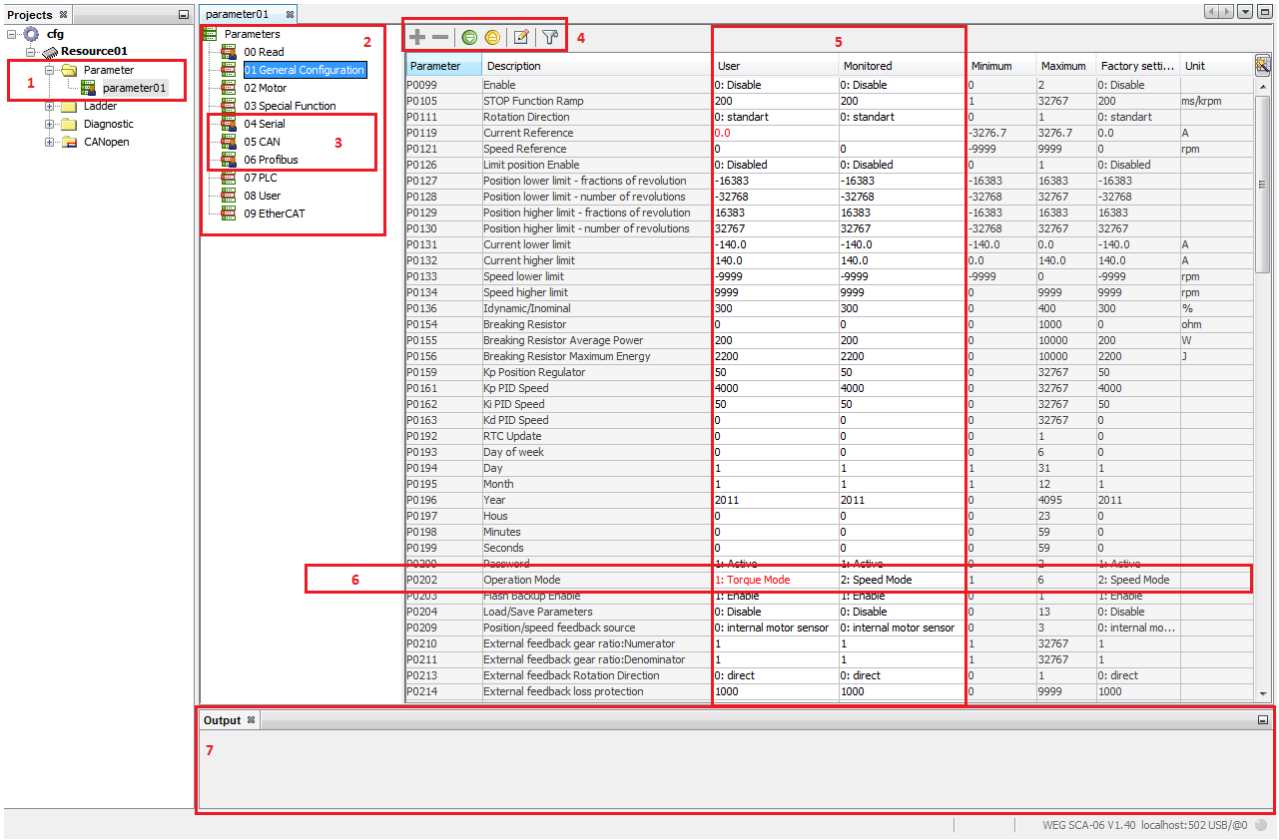
11.4.2 Parameters

11.4.2.1 Overview

The parameter configuration screen is used to configure and monitor all the parameters of the equipment, including the user parameters.

NOTE!
The reading and writing of such parameters is done on this screen; only the user parameter configuration must be sent the first time or whenever modified by means of the resource download routine.

Below is an overview of the parameter configuration screen.



Parameter	Description	User	Monitored	Minimum	Maximum	Factory setti...	Unit
P0099	Enable	0: Disable	0: Disable	0	2	0: Disable	
P0105	STOP Function Ramp	200	200	1	32767	200	ms/krpm
P0111	Rotation Direction	0: standart	0: standart	0	1	0: standart	
P0119	Current Reference	0.0		-3276.7	3276.7	0.0	A
P0121	Speed Reference	0	0	-9999	9999	0	rpm
P0126	Limit position Enable	0: Disabled	0: Disabled	0	1	0: Disabled	
P0127	Position lower limit - fractions of revolution	-16383	-16383	-16383	16383	-16383	
P0128	Position lower limit - number of revolutions	-32768	-32768	-32768	32767	-32768	
P0129	Position higher limit - fractions of revolution	16383	16383	-16383	16383	16383	
P0130	Position higher limit - number of revolutions	32767	32767	-32768	32767	32767	
P0131	Current lower limit	-140.0	-140.0	-140.0	0.0	-140.0	A
P0132	Current higher limit	140.0	140.0	0.0	140.0	140.0	A
P0133	Speed lower limit	-9999	-9999	-9999	0	-9999	rpm
P0134	Speed higher limit	9999	9999	0	9999	9999	rpm
P0136	Idynamic/Inominal	300	300	0	400	300	%
P0154	Breaking Resistor	0	0	0	1000	0	ohm
P0155	Breaking Resistor Average Power	200	200	0	10000	200	W
P0156	Breaking Resistor Maximum Energy	2200	2200	0	10000	2200	J
P0159	Kp Position Regulator	50	50	0	32767	50	
P0161	Kp PID Speed	4000	4000	0	32767	4000	
P0162	Ki PID Speed	50	50	0	32767	50	
P0163	Kd PID Speed	0	0	0	32767	0	
P0192	RTC Update	0	0	0	1	0	
P0193	Day of week	0	0	0	6	0	
P0194	Day	1	1	1	31	1	
P0195	Month	1	1	1	12	1	
P0196	Year	2011	2011	0	4095	2011	
P0197	Hous	0	0	0	23	0	
P0198	Minutes	0	0	0	59	0	
P0199	Seconds	0	0	0	59	0	
P0200	Password	1: Active	1: Active	0	2	1: Active	
P0202	Operation Mode	1: Torque Mode	2: Speed Mode	1	6	2: Speed Mode	
P0203	Flash Backup Enable	1: Enable	1: Enable	0	1	1: Enable	
P0204	Load/Save Parameters	0: Disable	0: Disable	0	13	0: Disable	
P0209	Position/speed feedback source	0: internal motor sensor	0: internal motor sensor	0	3	0: internal mo...	
P0210	External feedback gear ratio:Numerator	1	1	1	32767	1	
P0211	External feedback gear ratio:Denominator	1	1	1	32767	1	
P0213	External feedback Rotation Direction	0: direct	0: direct	0	1	0: direct	
P0214	External feedback loss protection	1000	1000	0	9999	1000	

1. **Parameter files**. In this part are all the parameter configuration files created by the user. Notice that when the file features a person figure on the table, it means this parameter table contains hidden parameters/ group of parameters.
2. **Group of parameters**. This tree shows all the group of parameters. Notice that the same parameter can be in more than one group, and when its value is modified, it will be modified in all the groups to which it belongs.
3. **Modified group of parameters**. Group of parameters which contain the figure of a person on the table means they have hidden parameters.
4. **Commands**. The commands are described below in the order they appear:
   - 4.1. **Unhide parameter**: In case some parameter has been hidden, this button allows making it visible again.
   - 4.2. **Hide parameter**: Just select one or more parameters on the table and trigger this command to hide them.
   - 4.3. **Save table**: It saves the values of the parameters shown on the equipment screen; the sent values are the ones in the User column. The flow is User -> Monitored (equipment)
   - 4.4. **Read table**: It reads the parameters of the equipment shown in the Monitored column and saves them in the parameter file in the User column. The flow is Monitored (equipment) -> User
   - 4.5. **User parameters**: It opens a screen to edit the user parameters.
   - 4.6. **Filter**: It opens a parameter filter option, and it can filter by parameter number or description.
   - 4.7. **User Parameters and Monitored Parameters**. These two columns show the off-line and on-line parameters, so to speak. The **User** column shows the values contained in the file located on the computer and the **Monitored** column shows the values that are effectively saved on the equipment. Whenever you use the **Save Parameter** option, the sent values will be from the **User** column to the **Monitored** column, that is, File -> Equipment. In case of reading, the flow is the opposite, from the **Monitored** column to the **User** column, that is, Equipment -> File. In case you wish to change the values directly on the equipment without changing it in the file, just click on the monitored column,

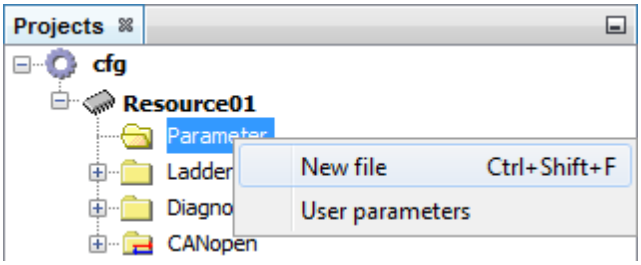
change the values and the modification will occur on-line.

5. **Modified parameters**: Whenever a parameter value in the **User** column is different from the **Monitored** column, it will be shown in red.
6. **Output**. This screen shows error information in case they occur during the writing or reading of the parameters.

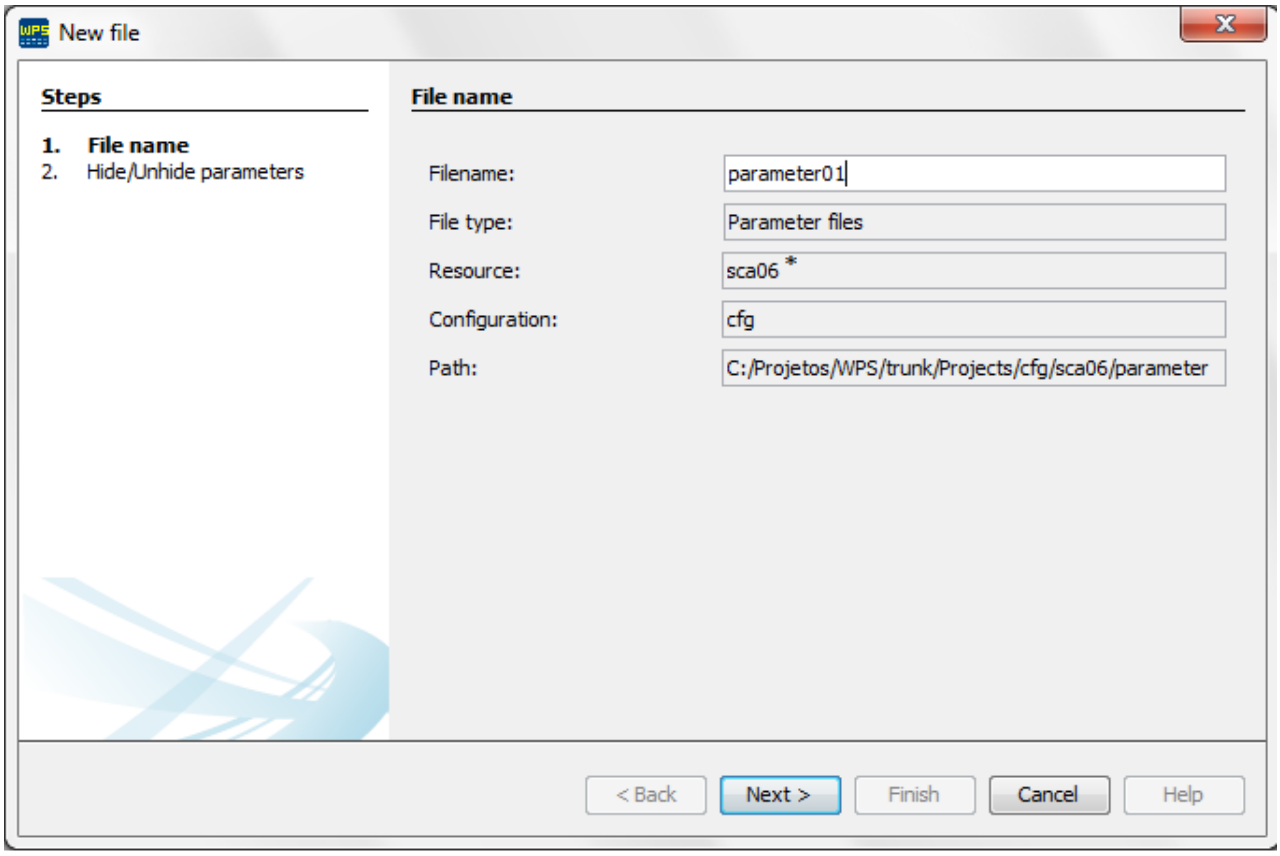
11.4.2.2 Configuration

Below is the list of the required steps to create a parameter file.

1. Create a new parameter file.

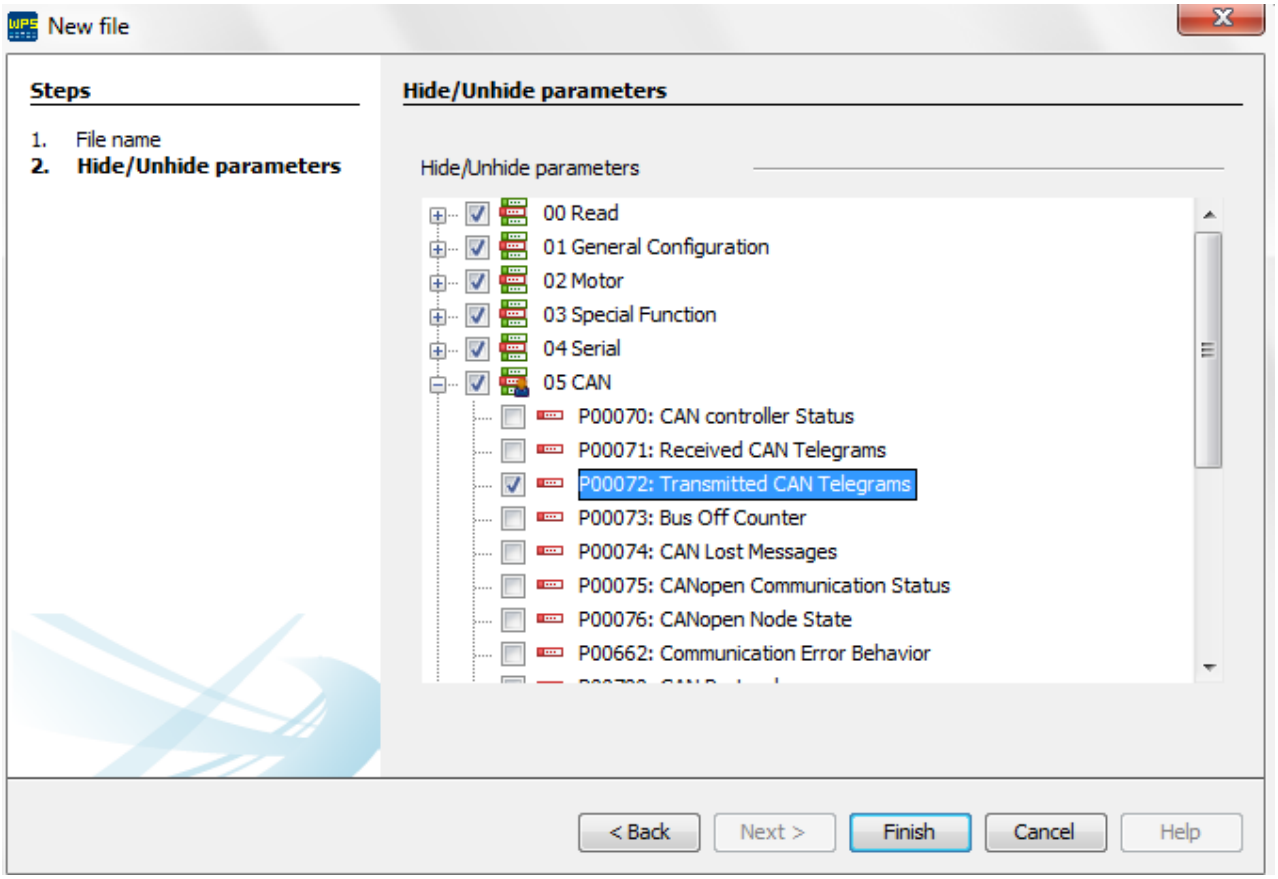


2. Define a name for the parameter file

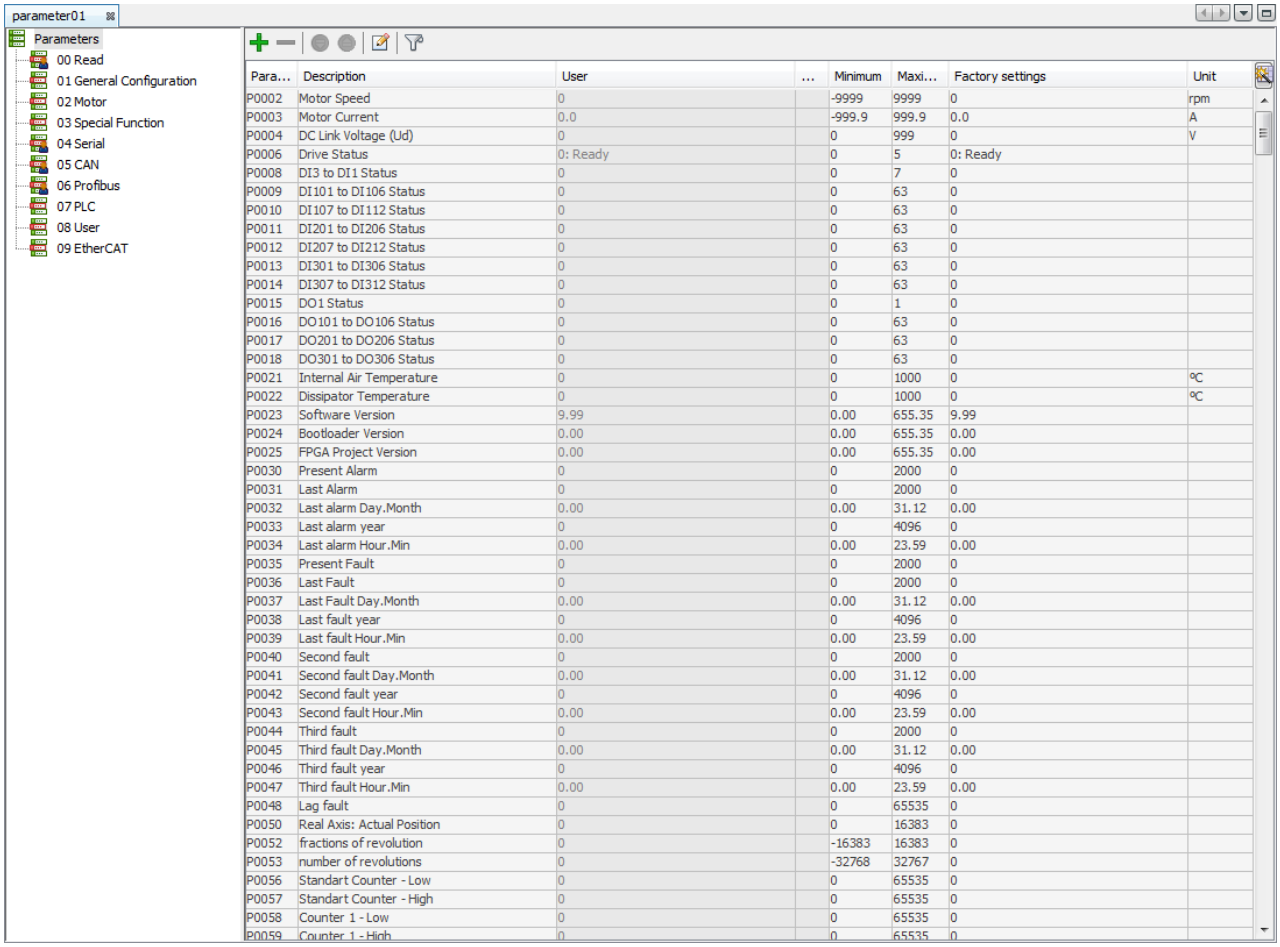


*** Resource: Resource01, SCA06, CFW300, etc.**

3. Configure which parameters you wish to view in your parameter table



4. After performing the steps above, the parameter file will be created and the equipment can be parameterized.

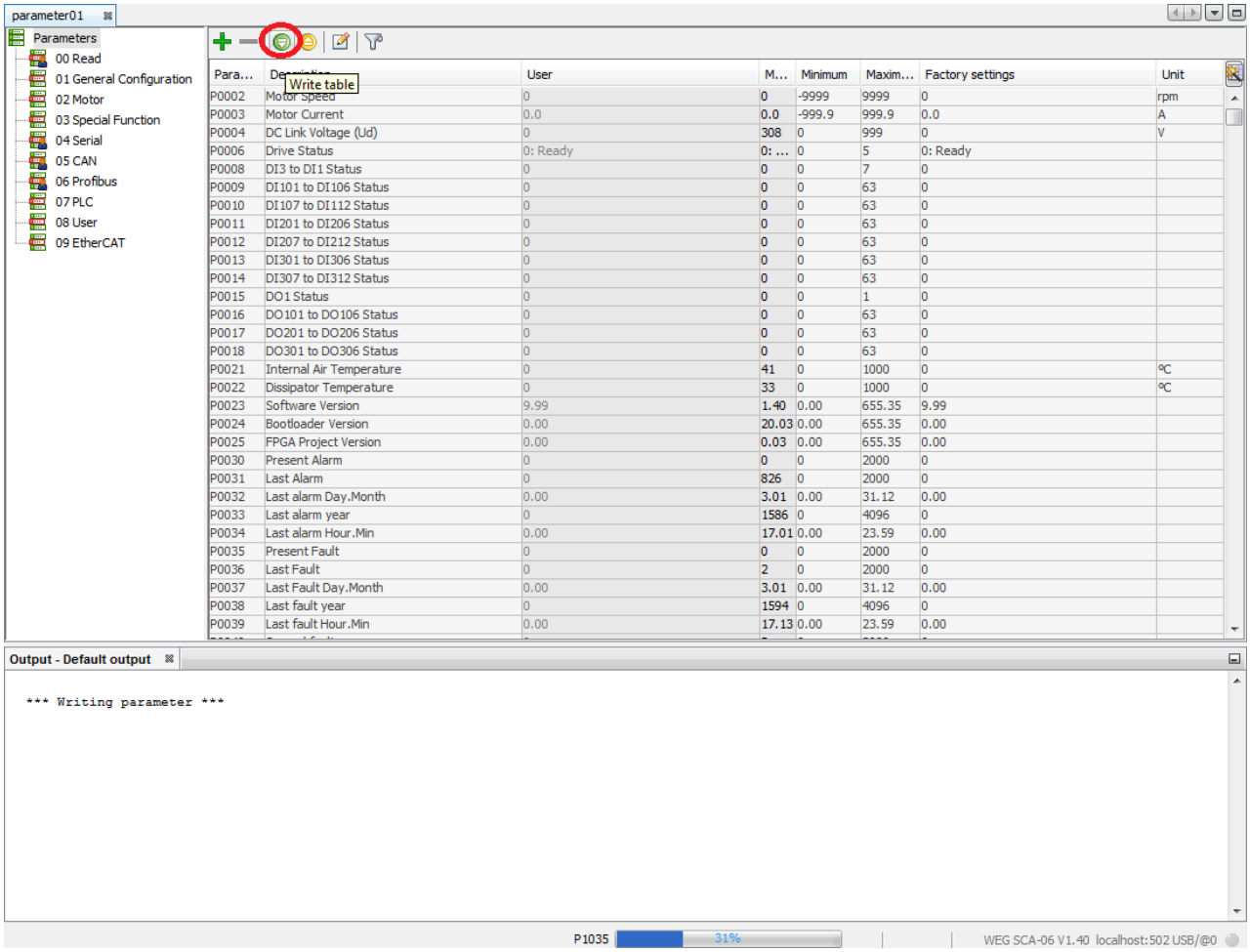


Para...	Description	User	...	Minimum	Maxi...	Factory settings	Unit
P0002	Motor Speed	0		-9999	9999	0	rpm
P0003	Motor Current	0.0		-999.9	999.9	0.0	A
P0004	DC Link Voltage (Ud)	0		0	999	0	V
P0006	Drive Status	0: Ready		0	5	0: Ready	
P0008	DI3 to DI1 Status	0		0	7	0	
P0009	DI101 to DI106 Status	0		0	63	0	
P0010	DI107 to DI112 Status	0		0	63	0	
P0011	DI201 to DI206 Status	0		0	63	0	
P0012	DI207 to DI212 Status	0		0	63	0	
P0013	DI301 to DI306 Status	0		0	63	0	
P0014	DI307 to DI312 Status	0		0	63	0	
P0015	DO1 Status	0		0	1	0	
P0016	DO101 to DO106 Status	0		0	63	0	
P0017	DO201 to DO206 Status	0		0	63	0	
P0018	DO301 to DO306 Status	0		0	63	0	
P0021	Internal Air Temperature	0		0	1000	0	ºC
P0022	Dissipator Temperature	0		0	1000	0	ºC
P0023	Software Version	9.99		0.00	655.35	9.99	
P0024	Bootloader Version	0.00		0.00	655.35	0.00	
P0025	FPGA Project Version	0.00		0.00	655.35	0.00	
P0030	Present Alarm	0		0	2000	0	
P0031	Last Alarm	0		0	2000	0	
P0032	Last alarm Day.Month	0.00		0.00	31.12	0.00	
P0033	Last alarm year	0		0	4096	0	
P0034	Last alarm Hour.Min	0.00		0.00	23.59	0.00	
P0035	Present Fault	0		0	2000	0	
P0036	Last Fault	0		0	2000	0	
P0037	Last Fault Day.Month	0.00		0.00	31.12	0.00	
P0038	Last fault year	0		0	4096	0	
P0039	Last fault Hour.Min	0.00		0.00	23.59	0.00	
P0040	Second fault	0		0	2000	0	
P0041	Second fault Day.Month	0.00		0.00	31.12	0.00	
P0042	Second fault year	0		0	4096	0	
P0043	Second fault Hour.Min	0.00		0.00	23.59	0.00	
P0044	Third fault	0		0	2000	0	
P0045	Third fault Day.Month	0.00		0.00	31.12	0.00	
P0046	Third fault year	0		0	4096	0	
P0047	Third fault Hour.Min	0.00		0.00	23.59	0.00	
P0048	Lag fault	0		0	65535	0	
P0050	Real Axis: Actual Position	0		0	16383	0	
P0052	fractions of revolution	0		-16383	16383	0	
P0053	number of revolutions	0		-32768	32767	0	
P0056	Standart Counter - Low	0		0	65535	0	
P0057	Standart Counter - High	0		0	65535	0	
P0058	Counter 1 - Low	0		0	65535	0	
P0059	Counter 1 - High	0		0	65535	0	

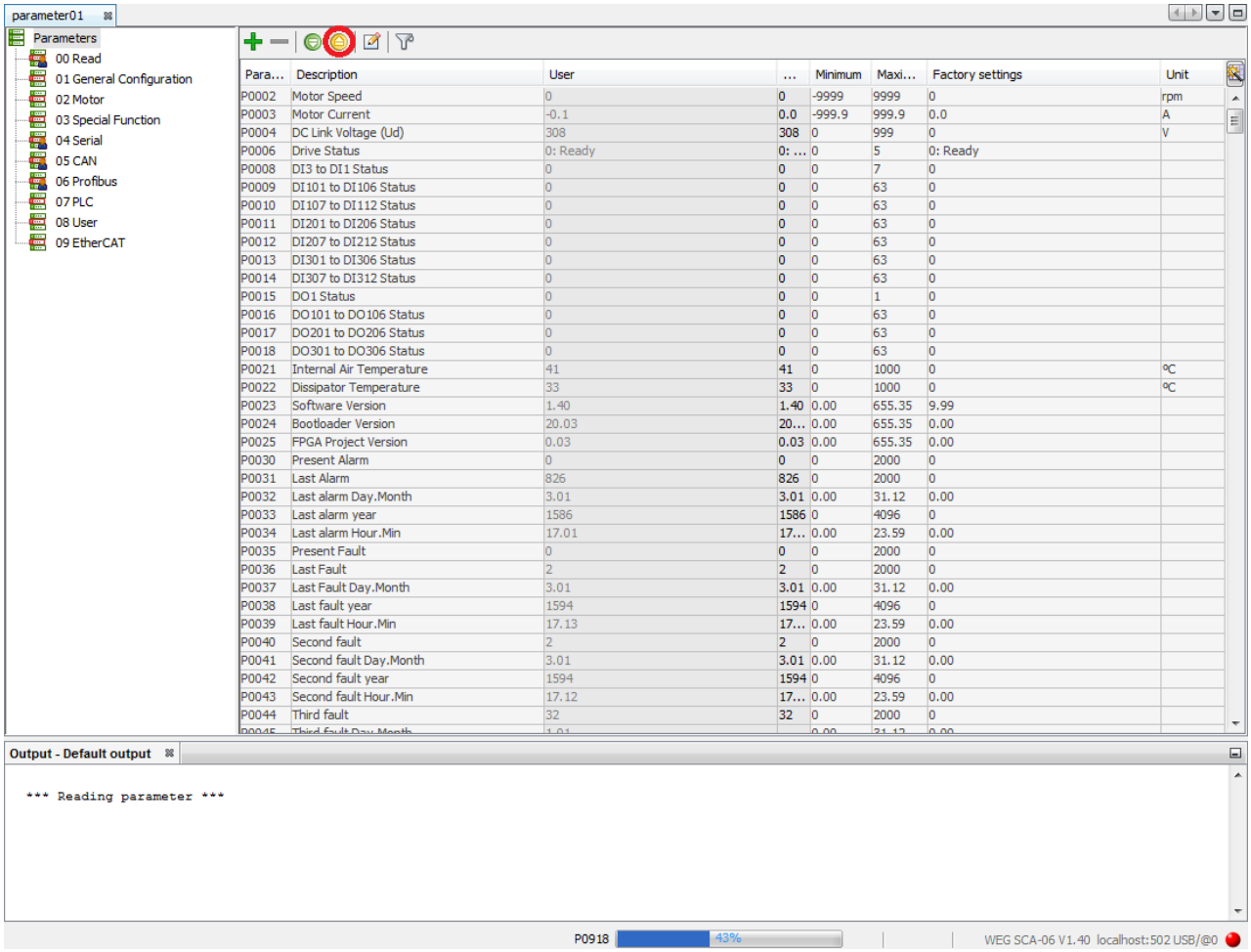
11.4.2.3 Read and Write of Parameters

There are 3 (three) ways to do the reading and writing of the parameters: by means of table, selection and group.

1. **Table writing**. The table writing command will send all visible parameters on the equipment screen. If and error occurs during the sending of some specific parameter, a message will be shown on the output window informing the error. It is important to notice that only visible parameters will be sent; therefore, it is necessary attention to which node of the group of parameters tree you are viewing. Example: If you wish to write all of them without filtering per group, just select the tree root.

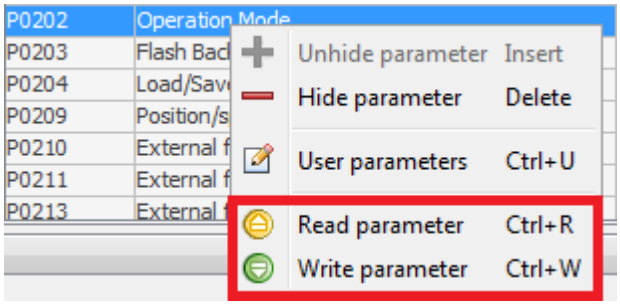


2. **Table reading**. The table reading command will read all the parameters of the equipment. If a error occurs during the reading of some specific parameter, a message will be shown on the output window informing the error. It is important to notice that only visible parameters will be read; therefore, it is necessary attention to which node of the group of parameters tree you are viewing. Example: If you wish to read all of them without filtering per group, just select the tree root.

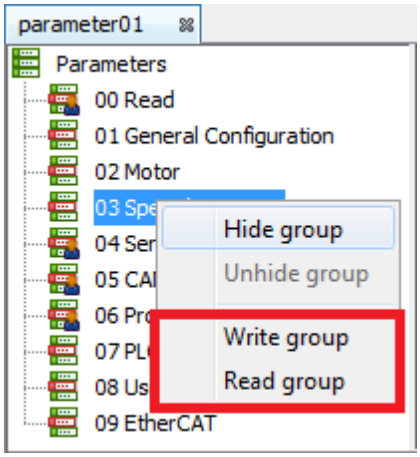


Para...	Description	User	...	Minimum	Maxi...	Factory settings	Unit
P0002	Motor Speed	0	0	-9999	9999	0	rpm
P0003	Motor Current	-0.1	0.0	-999.9	999.9	0.0	A
P0004	DC Link Voltage (Ud)	308	308	0	999	0	V
P0006	Drive Status	0: Ready	0: ...	0	5	0: Ready	
P0008	DI3 to DI1 Status	0	0	0	7	0	
P0009	DI101 to DI106 Status	0	0	0	63	0	
P0010	DI107 to DI112 Status	0	0	0	63	0	
P0011	DI201 to DI206 Status	0	0	0	63	0	
P0012	DI207 to DI212 Status	0	0	0	63	0	
P0013	DI301 to DI306 Status	0	0	0	63	0	
P0014	DI307 to DI312 Status	0	0	0	63	0	
P0015	DO1 Status	0	0	0	1	0	
P0016	DO101 to DO106 Status	0	0	0	63	0	
P0017	DO201 to DO206 Status	0	0	0	63	0	
P0018	DO301 to DO306 Status	0	0	0	63	0	
P0021	Internal Air Temperature	41	41	0	1000	0	°C
P0022	Dissipator Temperature	33	33	0	1000	0	°C
P0023	Software Version	1.40	1.40	0.00	655.35	9.99	
P0024	Bootloader Version	20.03	20...	0.00	655.35	0.00	
P0025	FPGA Project Version	0.03	0.03	0.00	655.35	0.00	
P0030	Present Alarm	0	0	0	2000	0	
P0031	Last Alarm	826	826	0	2000	0	
P0032	Last alarm Day.Month	3.01	3.01	0.00	31.12	0.00	
P0033	Last alarm year	1586	1586	0	4096	0	
P0034	Last alarm Hour.Min	17.01	17...	0.00	23.59	0.00	
P0035	Present Fault	0	0	0	2000	0	
P0036	Last Fault	2	2	0	2000	0	
P0037	Last Fault Day.Month	3.01	3.01	0.00	31.12	0.00	
P0038	Last fault year	1594	1594	0	4096	0	
P0039	Last fault Hour.Min	17.13	17...	0.00	23.59	0.00	
P0040	Second fault	2	2	0	2000	0	
P0041	Second fault Day.Month	3.01	3.01	0.00	31.12	0.00	
P0042	Second fault year	1594	1594	0	4096	0	
P0043	Second fault Hour.Min	17.12	17...	0.00	23.59	0.00	
P0044	Third fault	32	32	0	2000	0	

3. **Reading/writing of specific parameters**. In order to read/write one or more specific parameters, just select them on the table, right click and choose the desired option: read or write parameter.



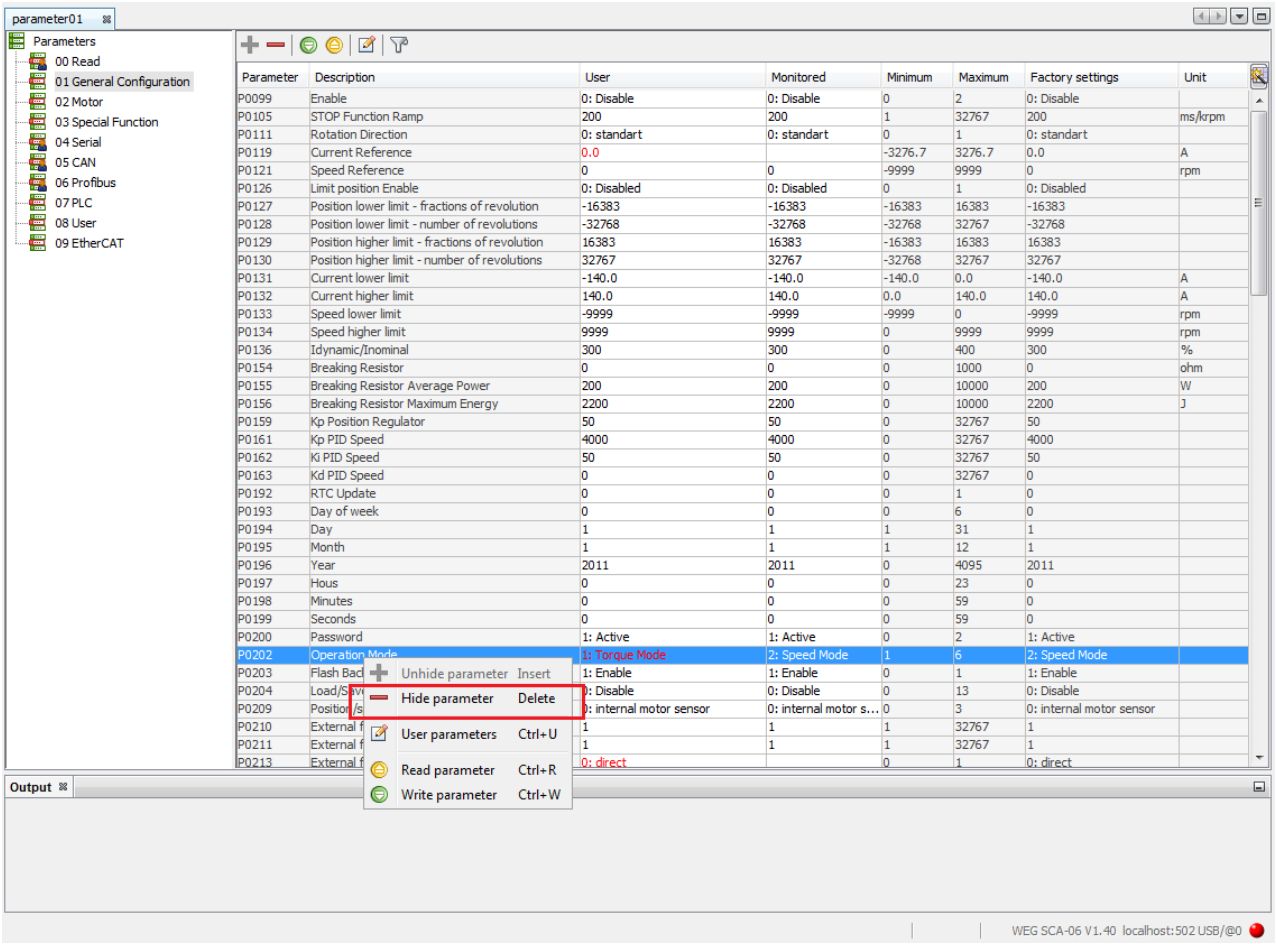
4. **Reading/writing of group of parameters**. In order to read/write only one group of parameters, just select it on the group tree, right click and choose the desired option: read or write group.



11.4.2.4 Hide/Unhide Parameters and Group of Parameters

The parameter can be hidden/unhidden in two ways: individually or in group.

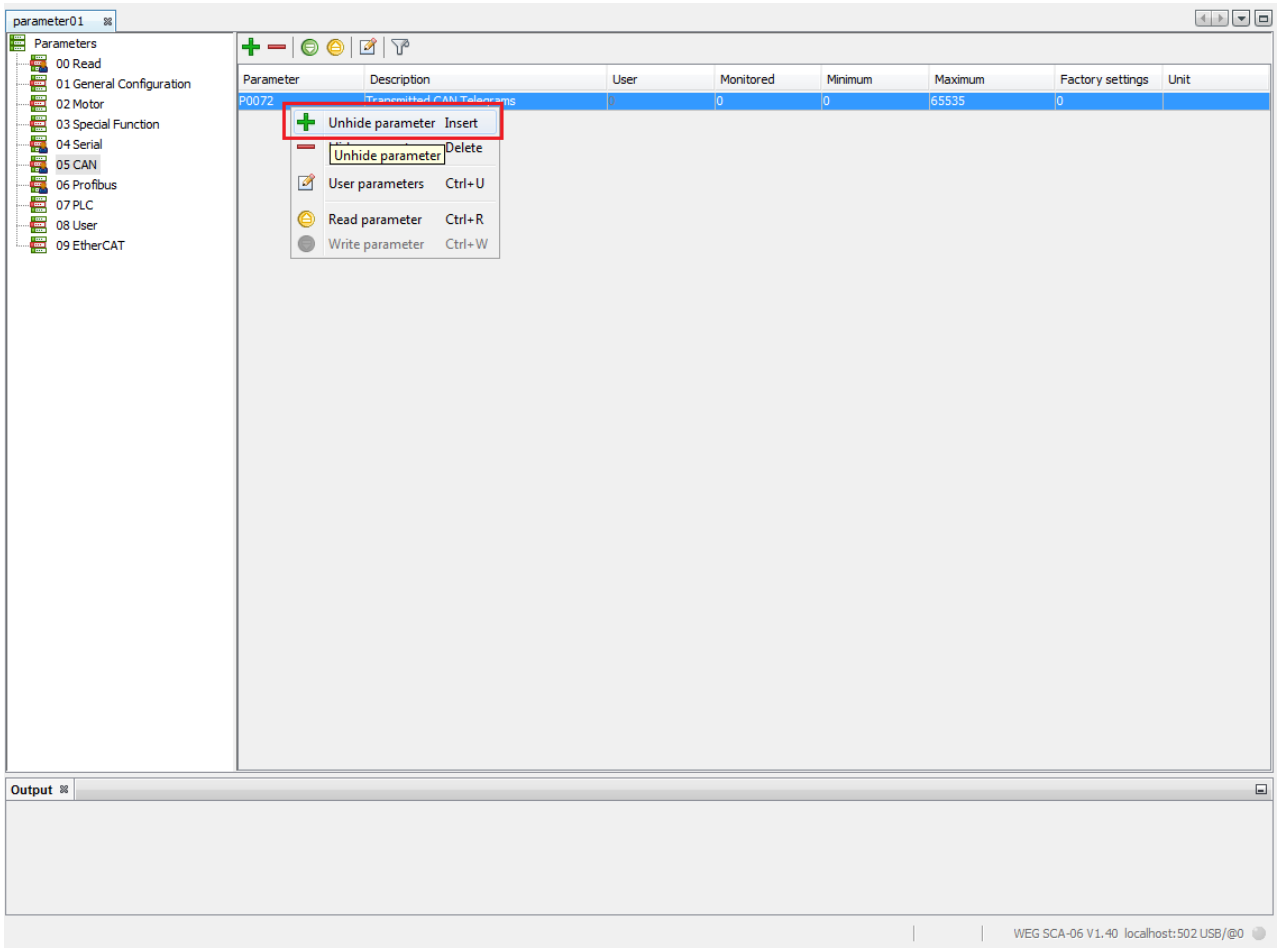
1. **Hide parameters**. In order to hide a parameter individually, just right click on the desired parameters and select the **Hide Parameter** option. You can also press the **Delete** key.

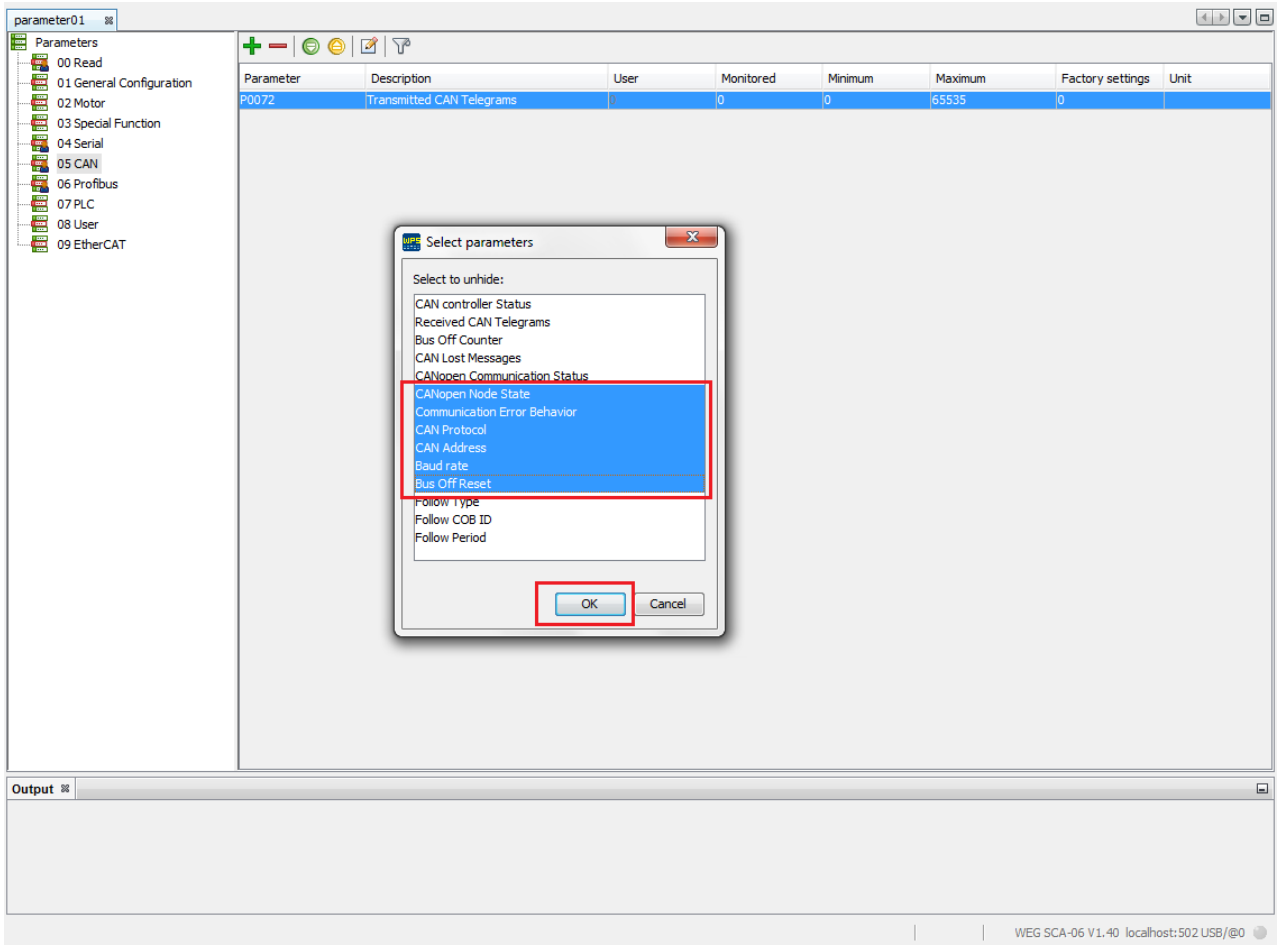


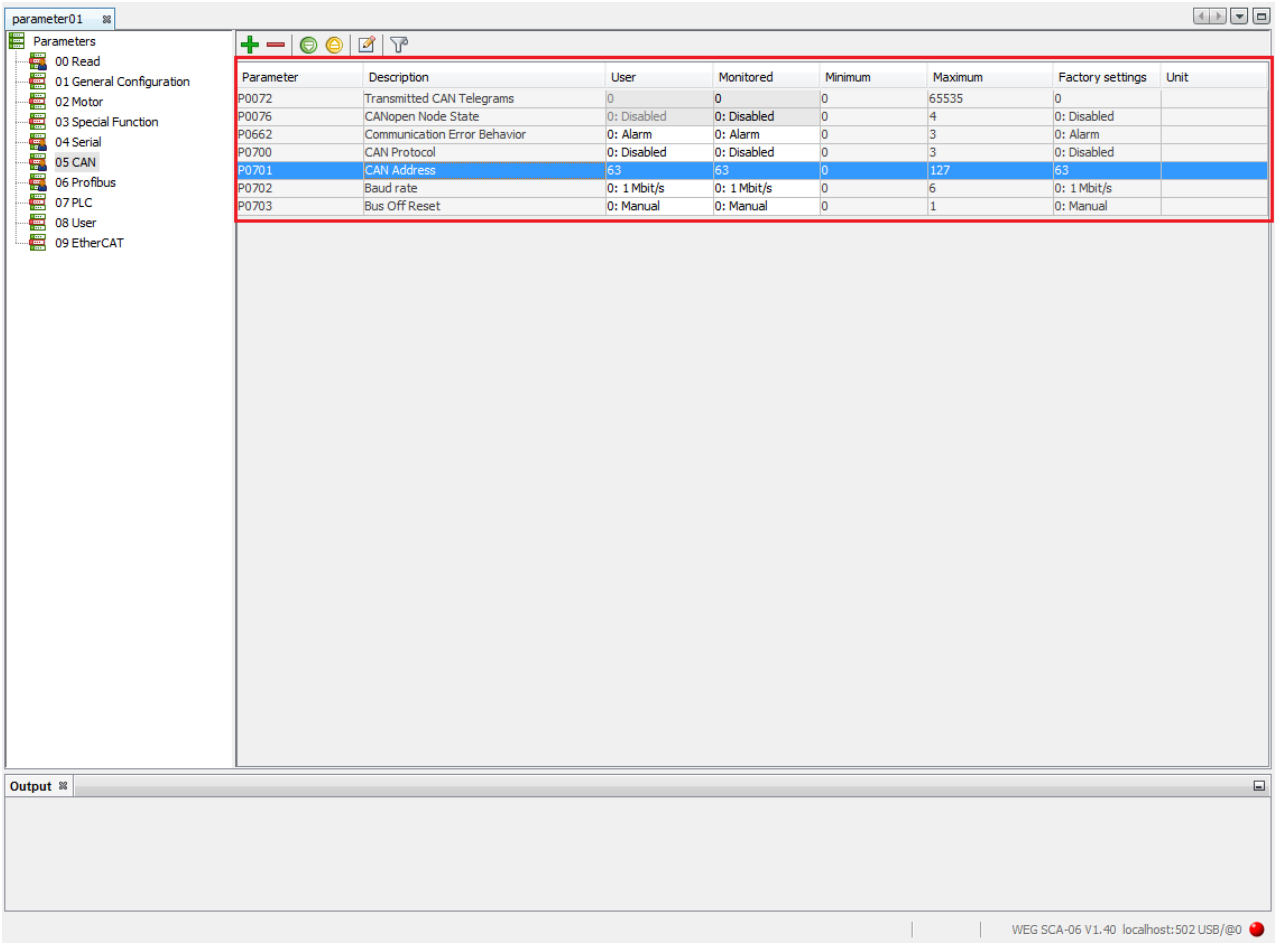
Parameter	Description	User	Monitored	Minimum	Maximum	Factory settings	Unit
P0099	Enable	0: Disable	0: Disable	0	2	0: Disable	
P0105	STOP Function Ramp	200	200	1	32767	200	ms/krpm
P0111	Rotation Direction	0: standart	0: standart	0	1	0: standart	
P0119	Current Reference	0.0		-3276.7	3276.7	0.0	A
P0121	Speed Reference	0	0	-9999	9999	0	rpm
P0126	Limit position Enable	0: Disabled	0: Disabled	0	1	0: Disabled	
P0127	Position lower limit - fractions of revolution	-16383	-16383	-16383	16383	-16383	
P0128	Position lower limit - number of revolutions	-32768	-32768	-32768	32767	-32768	
P0129	Position higher limit - fractions of revolution	16383	16383	-16383	16383	16383	
P0130	Position higher limit - number of revolutions	32767	32767	-32768	32767	32767	
P0131	Current lower limit	-140.0	-140.0	-140.0	0.0	-140.0	A
P0132	Current higher limit	140.0	140.0	0.0	140.0	140.0	A
P0133	Speed lower limit	-9999	-9999	-9999	0	-9999	rpm
P0134	Speed higher limit	9999	9999	0	9999	9999	rpm
P0136	Idynamic/Inominal	300	300	0	400	300	%
P0154	Breaking Resistor	0	0	0	1000	0	ohm
P0155	Breaking Resistor Average Power	200	200	0	10000	200	W
P0156	Breaking Resistor Maximum Energy	2200	2200	0	10000	2200	J
P0159	Kp Position Regulator	50	50	0	32767	50	
P0161	Kp PID Speed	4000	4000	0	32767	4000	
P0162	Ki PID Speed	50	50	0	32767	50	
P0163	Kd PID Speed	0	0	0	32767	0	
P0192	RTC Update	0	0	0	1	0	
P0193	Day of week	0	0	0	6	0	
P0194	Day	1	1	1	31	1	
P0195	Month	1	1	1	12	1	
P0196	Year	2011	2011	0	4095	2011	
P0197	Hous	0	0	0	23	0	
P0198	Minutes	0	0	0	59	0	
P0199	Seconds	0	0	0	59	0	
P0200	Password	1: Active	1: Active	0	2	1: Active	
P0202	Operation Mode	1: Torque Mode	2: Speed Mode	1	6	2: Speed Mode	
P0203	Flash Bac	1: Enable	1: Enable	0	1	1: Enable	
P0204	Load/S	0: Disable	0: Disable	0	13	0: Disable	
P0209	Position/s	0: internal motor sensor	0: internal motor s...	0	3	0: internal motor sensor	
P0210	External f	1	1	1	32767	1	
P0211	External f	1	1	1	32767	1	
P0213	External f	0: direct		0	1	0: direct	

2. **Unhide Parameters**. In order to show hidden parameters, right click and choose the **Unhide Parameters**

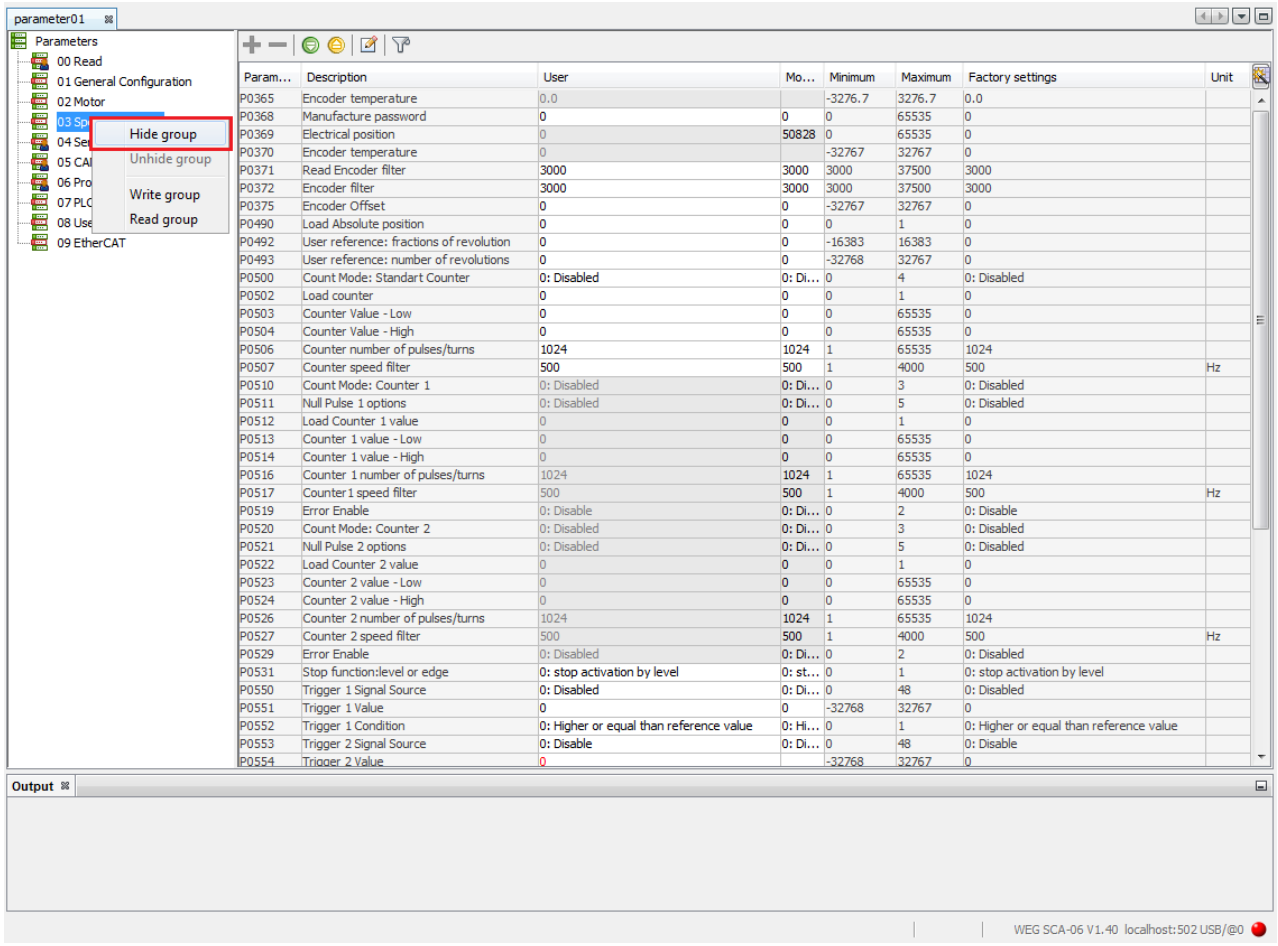
or press the **Insert** key. Then, a window will open and show the hidden parameters. Now, you just have to select the desired parameters and confirm.
Note: The parameters shown on this new window are only those which belong to the current filter according to the selection on the parameter group tree. In the figures below, the CAN group is selected; that means that only the hidden parameters of this group will be shown.

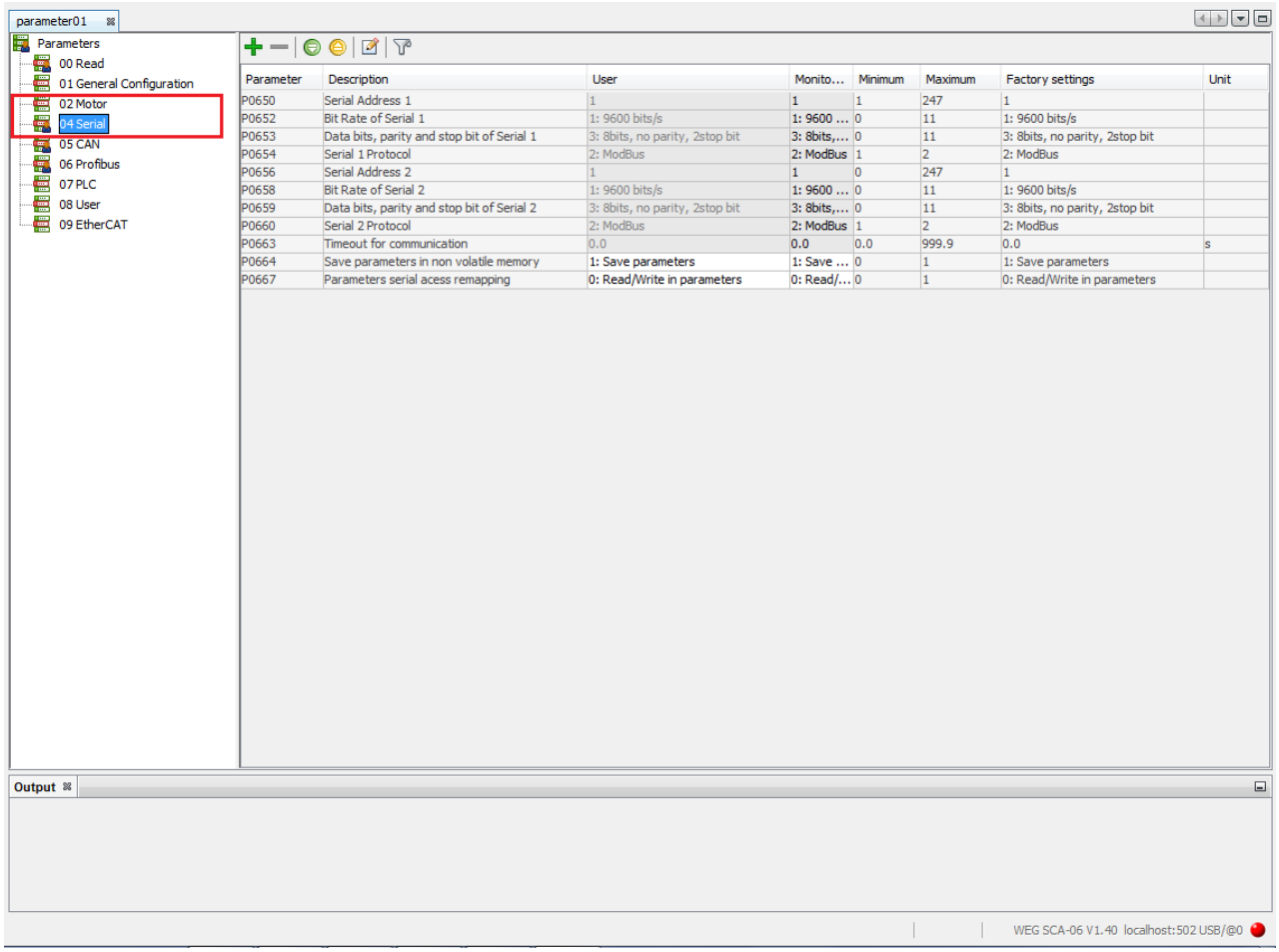



parameter01
Parameters
00 Read
01 General Configuration
02 Motor
03 Special Function
04 Serial
05 CAN
06 Profibus
07 PLC
08 User
09 EtherCAT
Parameter
Description
User
Monitored
Minimum
Maximum
Factory settings
Unit
P0072
Transmitted CAN Telegrams
0
0
65535
0
Select parameters
Select to unhide:
CAN controller Status
Received CAN Telegrams
Bus Off Counter
CAN Lost Messages
CANopen Communication Status
CANopen Node State
Communication Error Behavior
CAN Protocol
CAN Address
Baud rate
Bus Off Reset
Follow Type
Follow COB ID
Follow Period
OK
Cancel
Output
WEG SCA-06 V1.40 localhost:502 USB/@0

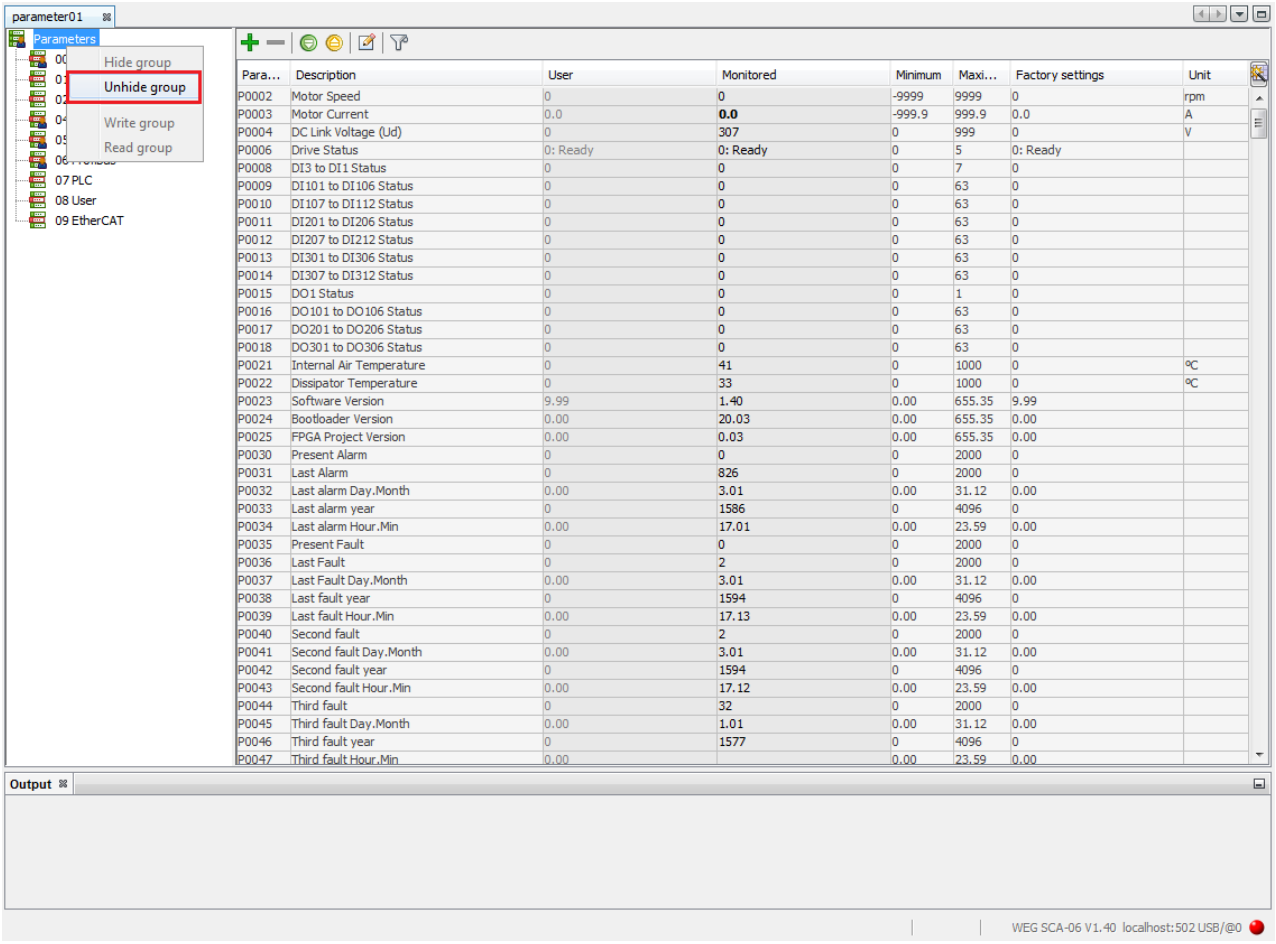


3. **Hide Group of Parameters**. In order to hide a group of parameters, just select the group on the tree and use the **Hide Group** option.

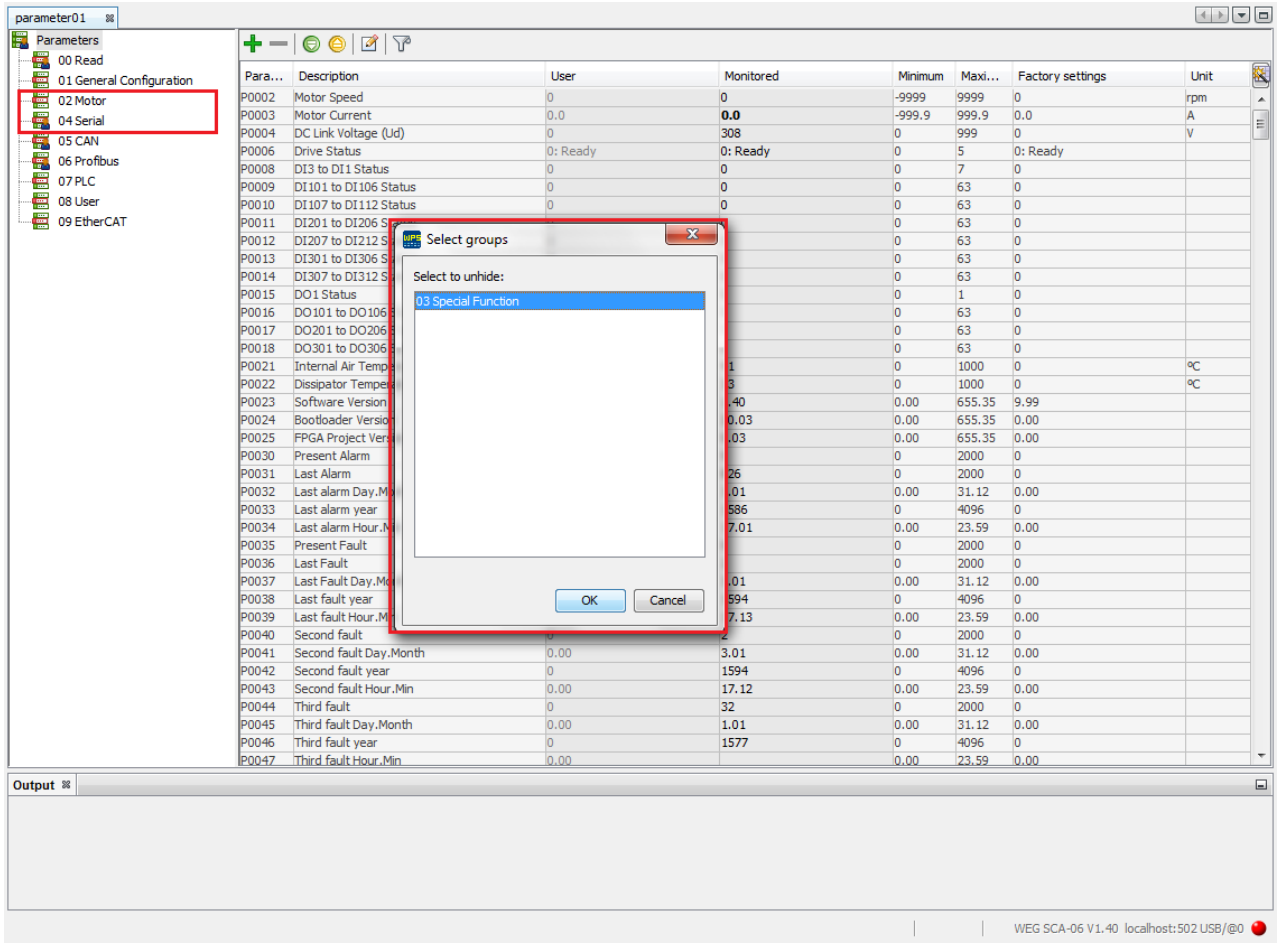

parameter01
Parameters
00 Read
01 General Configuration
02 Motor
03 Sp
04 Se
05 CA
06 Pro
07 PLC
08 Use
09 EtherCAT
Hide group
Unhide group
Write group
Read group
Param...
Description
User
Mo...
Minimum
Maximum
Factory settings
Unit
P0365 Encoder temperature 0.0 -3276.7 3276.7 0.0
P0368 Manufacture password 0 0 0 65535 0
P0369 Electrical position 0 50828 0 65535 0
P0370 Encoder temperature 0 -32767 32767 0
P0371 Read Encoder filter 3000 3000 3000 37500 3000
P0372 Encoder filter 3000 3000 3000 37500 3000
P0375 Encoder Offset 0 0 -32767 32767 0
P0490 Load Absolute position 0 0 0 1 0
P0492 User reference: fractions of revolution 0 0 -16383 16383 0
P0493 User reference: number of revolutions 0 0 -32768 32767 0
P0500 Count Mode: Standart Counter 0: Disabled 0: Di... 0 4 0: Disabled
P0502 Load counter 0 0 0 1 0
P0503 Counter Value - Low 0 0 0 65535 0
P0504 Counter Value - High 0 0 0 65535 0
P0506 Counter number of pulses/turns 1024 1024 1 65535 1024
P0507 Counter speed filter 500 500 1 4000 500 Hz
P0510 Count Mode: Counter 1 0: Disabled 0: Di... 0 3 0: Disabled
P0511 Null Pulse 1 options 0: Disabled 0: Di... 0 5 0: Disabled
P0512 Load Counter 1 value 0 0 0 1 0
P0513 Counter 1 value - Low 0 0 0 65535 0
P0514 Counter 1 value - High 0 0 0 65535 0
P0516 Counter 1 number of pulses/turns 1024 1024 1 65535 1024
P0517 Counter1 speed filter 500 500 1 4000 500 Hz
P0519 Error Enable 0: Disable 0: Di... 0 2 0: Disable
P0520 Count Mode: Counter 2 0: Disabled 0: Di... 0 3 0: Disabled
P0521 Null Pulse 2 options 0: Disabled 0: Di... 0 5 0: Disabled
P0522 Load Counter 2 value 0 0 0 1 0
P0523 Counter 2 value - Low 0 0 0 65535 0
P0524 Counter 2 value - High 0 0 0 65535 0
P0526 Counter 2 number of pulses/turns 1024 1024 1 65535 1024
P0527 Counter 2 speed filter 500 500 1 4000 500 Hz
P0529 Error Enable 0: Disabled 0: Di... 0 2 0: Disabled
P0531 Stop function:level or edge 0: stop activation by level 0: st... 0 1 0: stop activation by level
P0550 Trigger 1 Signal Source 0: Disabled 0: Di... 0 48 0: Disabled
P0551 Trigger 1 Value 0 0 -32768 32767 0
P0552 Trigger 1 Condition 0: Higher or equal than reference value 0: Hi... 0 1 0: Higher or equal than reference value
P0553 Trigger 2 Signal Source 0: Disable 0: Di... 0 48 0: Disable
P0554 Trigger 2 Value 0 -32768 32767 0
Output
WEG SCA-06 V1.40 localhost:502 USB/@0

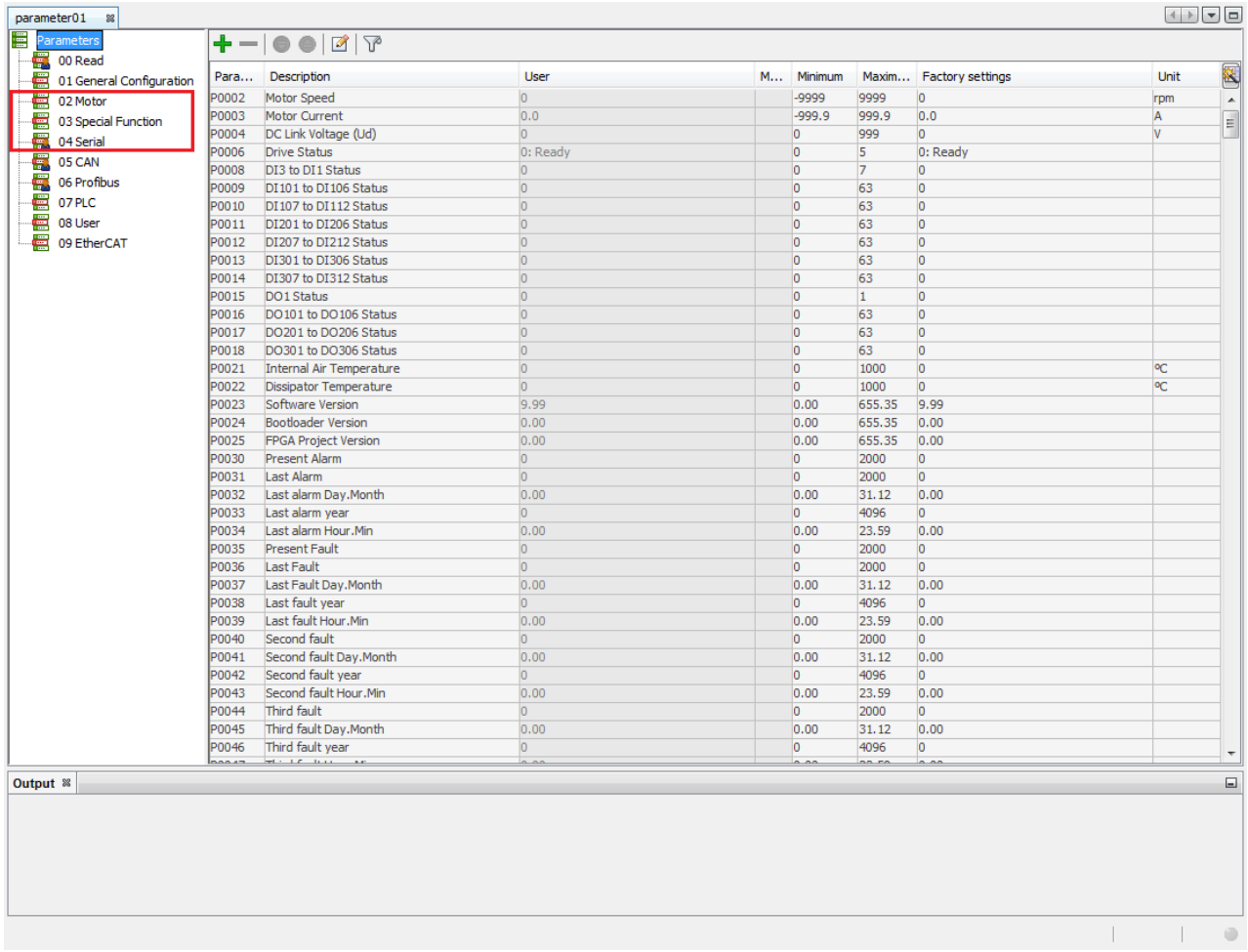


4. **Unhide Group of Parameters**. In order to show a hidden group of parameters, just select the root of the group tree and select the **Unhide Group** option. A window will open showing the groups that are hidden; then just select the group you wish to unhide.



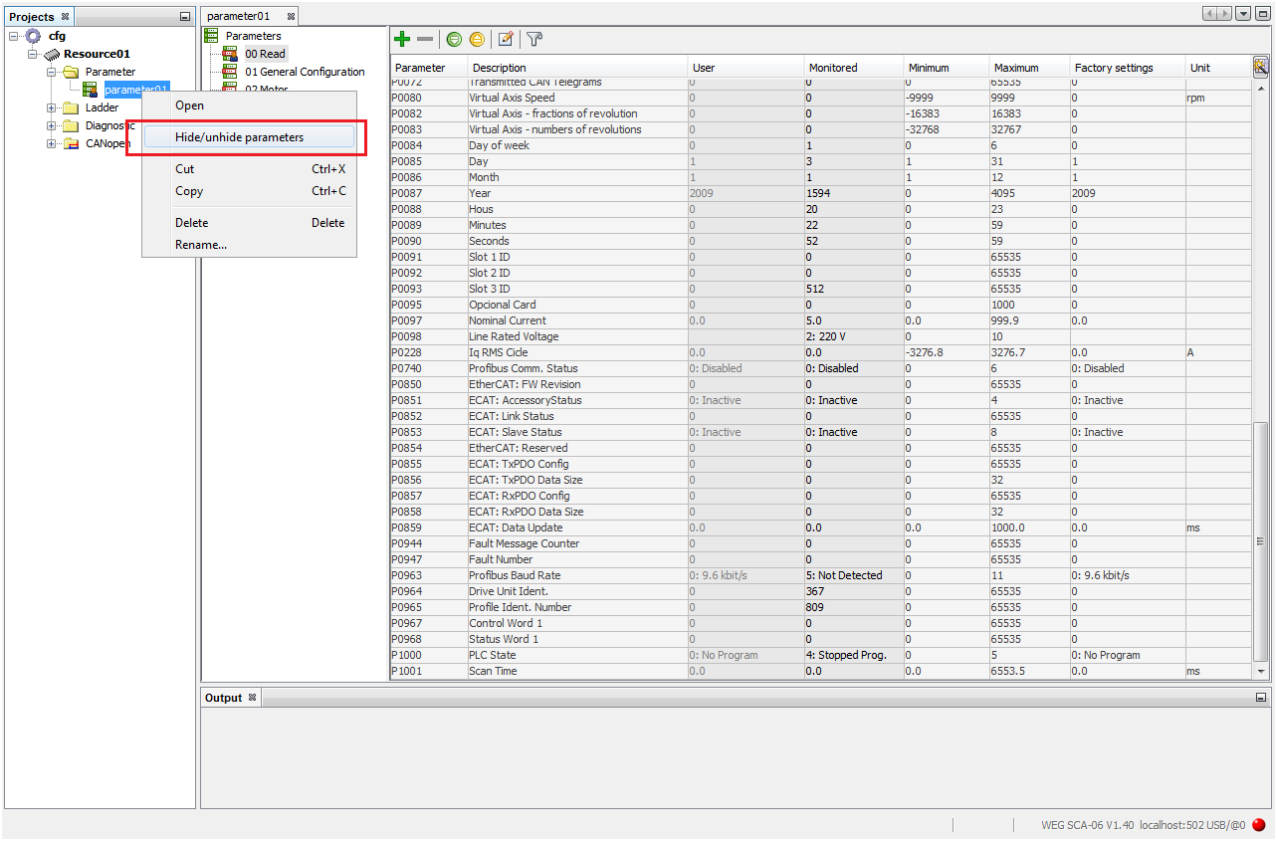
Para...	Description	User	Monitored	Minimum	Maxi...	Factory settings	Unit
P0002	Motor Speed	0	0	-9999	9999	0	rpm
P0003	Motor Current	0.0	**0.0**	-999.9	999.9	0.0	A
P0004	DC Link Voltage (Ud)	0	307	0	999	0	V
P0006	Drive Status	0: Ready	0: Ready	0	5	0: Ready	
P0008	DI3 to DI1 Status	0	0	0	7	0	
P0009	DI101 to DI106 Status	0	0	0	63	0	
P0010	DI107 to DI112 Status	0	0	0	63	0	
P0011	DI201 to DI206 Status	0	0	0	63	0	
P0012	DI207 to DI212 Status	0	0	0	63	0	
P0013	DI301 to DI306 Status	0	0	0	63	0	
P0014	DI307 to DI312 Status	0	0	0	63	0	
P0015	DO1 Status	0	0	0	1	0	
P0016	DO101 to DO106 Status	0	0	0	63	0	
P0017	DO201 to DO206 Status	0	0	0	63	0	
P0018	DO301 to DO306 Status	0	0	0	63	0	
P0021	Internal Air Temperature	0	41	0	1000	0	°C
P0022	Dissipator Temperature	0	33	0	1000	0	°C
P0023	Software Version	9.99	1.40	0.00	655.35	9.99	
P0024	Bootloader Version	0.00	20.03	0.00	655.35	0.00	
P0025	FPGA Project Version	0.00	0.03	0.00	655.35	0.00	
P0030	Present Alarm	0	0	0	2000	0	
P0031	Last Alarm	0	826	0	2000	0	
P0032	Last alarm Day.Month	0.00	3.01	0.00	31.12	0.00	
P0033	Last alarm year	0	1586	0	4096	0	
P0034	Last alarm Hour.Min	0.00	17.01	0.00	23.59	0.00	
P0035	Present Fault	0	0	0	2000	0	
P0036	Last Fault	0	2	0	2000	0	
P0037	Last Fault Day.Month	0.00	3.01	0.00	31.12	0.00	
P0038	Last fault year	0	1594	0	4096	0	
P0039	Last fault Hour.Min	0.00	17.13	0.00	23.59	0.00	
P0040	Second fault	0	2	0	2000	0	
P0041	Second fault Day.Month	0.00	3.01	0.00	31.12	0.00	
P0042	Second fault year	0	1594	0	4096	0	
P0043	Second fault Hour.Min	0.00	17.12	0.00	23.59	0.00	
P0044	Third fault	0	32	0	2000	0	
P0045	Third fault Day.Month	0.00	1.01	0.00	31.12	0.00	
P0046	Third fault year	0	1577	0	4096	0	
P0047	Third fault Hour.Min	0.00		0.00	23.59	0.00	

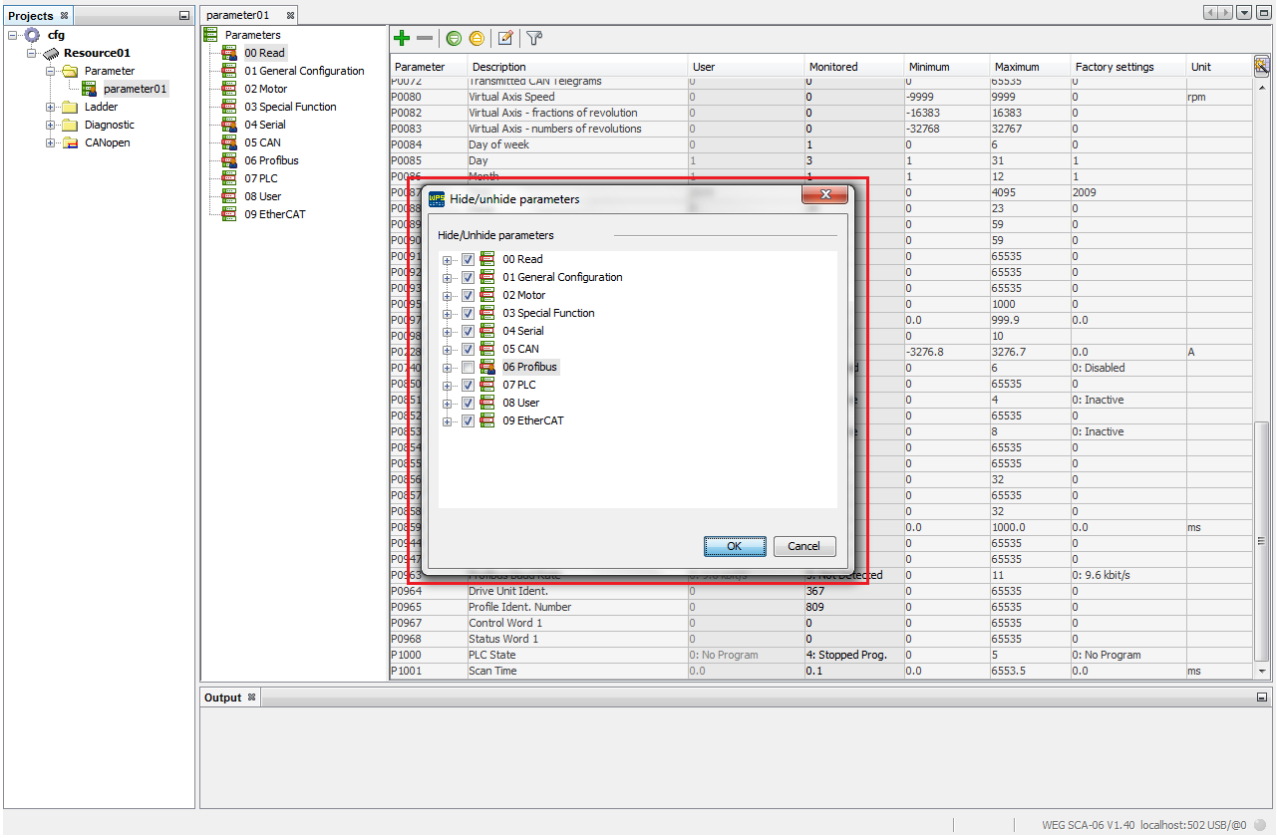

parameter01
Parameters
00 Read
01 General Configuration
02 Motor
04 Serial
05 CAN
06 Profibus
07 PLC
08 User
09 EtherCAT
Select groups
Select to unhide:
03 Special Function
OK
Cancel
Output
WEG SCA-06 V1.40 localhost:502 USB/@0



Para...	Description	User	M...	Minimum	Maxim...	Factory settings	Unit
P0002	Motor Speed	0		-9999	9999	0	rpm
P0003	Motor Current	0.0		-999.9	999.9	0.0	A
P0004	DC Link Voltage (Ud)	0		0	999	0	V
P0006	Drive Status	0: Ready		0	5	0: Ready	
P0008	DI3 to DI1 Status	0		0	7	0	
P0009	DI101 to DI106 Status	0		0	63	0	
P0010	DI107 to DI112 Status	0		0	63	0	
P0011	DI201 to DI206 Status	0		0	63	0	
P0012	DI207 to DI212 Status	0		0	63	0	
P0013	DI301 to DI306 Status	0		0	63	0	
P0014	DI307 to DI312 Status	0		0	63	0	
P0015	DO1 Status	0		0	1	0	
P0016	DO101 to DO106 Status	0		0	63	0	
P0017	DO201 to DO206 Status	0		0	63	0	
P0018	DO301 to DO306 Status	0		0	63	0	
P0021	Internal Air Temperature	0		0	1000	0	°C
P0022	Dissipator Temperature	0		0	1000	0	°C
P0023	Software Version	9.99		0.00	655.35	9.99	
P0024	Bootloader Version	0.00		0.00	655.35	0.00	
P0025	FPGA Project Version	0.00		0.00	655.35	0.00	
P0030	Present Alarm	0		0	2000	0	
P0031	Last Alarm	0		0	2000	0	
P0032	Last alarm Day.Month	0.00		0.00	31.12	0.00	
P0033	Last alarm year	0		0	4096	0	
P0034	Last alarm Hour.Min	0.00		0.00	23.59	0.00	
P0035	Present Fault	0		0	2000	0	
P0036	Last Fault	0		0	2000	0	
P0037	Last Fault Day.Month	0.00		0.00	31.12	0.00	
P0038	Last fault year	0		0	4096	0	
P0039	Last fault Hour.Min	0.00		0.00	23.59	0.00	
P0040	Second fault	0		0	2000	0	
P0041	Second fault Day.Month	0.00		0.00	31.12	0.00	
P0042	Second fault year	0		0	4096	0	
P0043	Second fault Hour.Min	0.00		0.00	23.59	0.00	
P0044	Third fault	0		0	2000	0	
P0045	Third fault Day.Month	0.00		0.00	31.12	0.00	
P0046	Third fault year	0		0	4096	0	

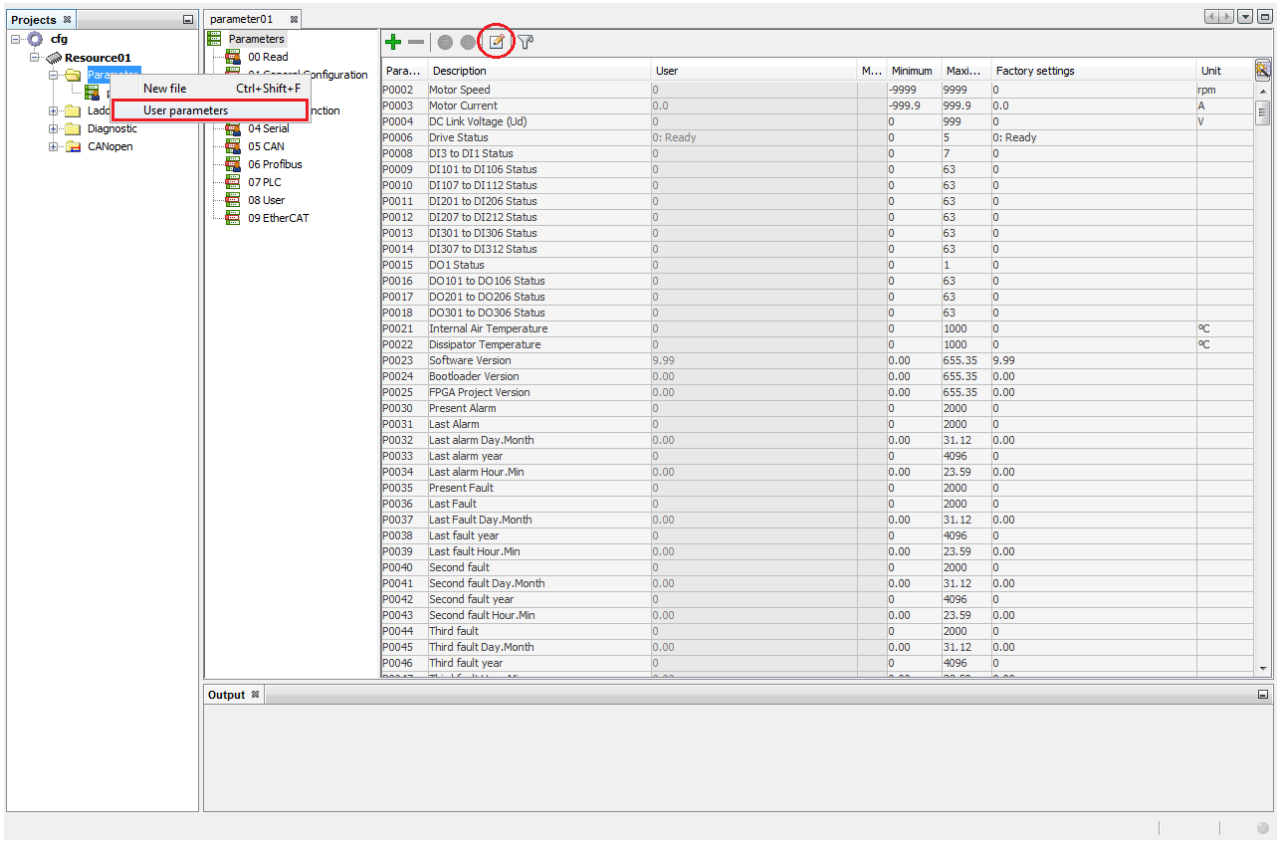
5. **Hide and Show Parameters and Groups of Parameters**. By means of this option, you have full control of the parameters and groups of parameters. It is possible to hide and unhide individual parameters, multiple parameters, individual groups and multiple groups in the same action.


Projects
cfg
Resource01
Parameter
parameter01
Ladder
Diagnostic
CANopen
parameter01
Parameters
00 Read
01 General Configuration
Open
Hide/unhide parameters
Cut Ctrl+X
Copy Ctrl+C
Delete Delete
Rename...
Parameter
Description
User
Monitored
Minimum
Maximum
Factory settings
Unit
P0080 Virtual Axis Speed 0 0 -9999 9999 0 rpm
P0082 Virtual Axis - fractions of revolution 0 0 -16383 16383 0
P0083 Virtual Axis - numbers of revolutions 0 0 -32768 32767 0
P0084 Day of week 0 1 0 6 0
P0085 Day 1 3 1 31 1
P0086 Month 1 1 1 12 1
P0087 Year 2009 1594 0 4095 2009
P0088 Hous 0 20 0 23 0
P0089 Minutes 0 22 0 59 0
P0090 Seconds 0 52 0 59 0
P0091 Slot 1 ID 0 0 0 65535 0
P0092 Slot 2 ID 0 0 0 65535 0
P0093 Slot 3 ID 0 512 0 65535 0
P0095 Opcional Card 0 0 0 1000 0
P0097 Nominal Current 0.0 5.0 0.0 999.9 0.0
P0098 Line Rated Voltage 2: 220 V 0 10
P0228 Iq RMS Cicle 0.0 0.0 -3276.8 3276.7 0.0 A
P0740 Profibus Comm. Status 0: Disabled 0: Disabled 0 6 0: Disabled
P0850 EtherCAT: FW Revision 0 0 0 65535 0
P0851 ECAT: AccessoryStatus 0: Inactive 0: Inactive 0 4 0: Inactive
P0852 ECAT: Link Status 0 0 0 65535 0
P0853 ECAT: Slave Status 0: Inactive 0: Inactive 0 8 0: Inactive
P0854 EtherCAT: Reserved 0 0 0 65535 0
P0855 ECAT: TxPDO Config 0 0 0 65535 0
P0856 ECAT: TxPDO Data Size 0 0 0 32 0
P0857 ECAT: RxPDO Config 0 0 0 65535 0
P0858 ECAT: RxPDO Data Size 0 0 0 32 0
P0859 ECAT: Data Update 0.0 0.0 0.0 1000.0 0.0 ms
P0944 Fault Message Counter 0 0 0 65535 0
P0947 Fault Number 0 0 0 65535 0
P0963 Profibus Baud Rate 0: 9.6 kbit/s 5: Not Detected 0 11 0: 9.6 kbit/s
P0964 Drive Unit Ident. 0 367 0 65535 0
P0965 Profile Ident. Number 0 809 0 65535 0
P0967 Control Word 1 0 0 0 65535 0
P0968 Status Word 1 0 0 0 65535 0
P1000 PLC State 0: No Program 4: Stopped Prog. 0 5 0: No Program
P1001 Scan Time 0.0 0.0 0.0 6553.5 0.0 ms
Output
WEG SCA-06 V1.40 localhost:502 USB/@0


Projects
cfg
Resource01
Parameter
parameter01
Ladder
Diagnostic
CANopen
parameter01
Parameters
00 Read
01 General Configuration
02 Motor
03 Special Function
04 Serial
05 CAN
06 Profibus
07 PLC
08 User
09 EtherCAT
Hide/unhide parameters
Hide/Unhide parameters
00 Read
01 General Configuration
02 Motor
03 Special Function
04 Serial
05 CAN
06 Profibus
07 PLC
08 User
09 EtherCAT
OK
Cancel
Output
WEG SCA-06 V1.40 localhost:502 USB/@0

11.4.2.5 User Parameters

In order to open the configuration screen of the user parameters, just click on the **User Parameters** option on the Parameter node of the project tree or click on the icon indicated on the tool bar of the parameter file.



Para...	Description	User	M...	Minimum	Maxi...	Factory settings	Unit
P0002	Motor Speed	0		-9999	9999	0	rpm
P0003	Motor Current	0.0		-999.9	999.9	0.0	A
P0004	DC Link Voltage (Ud)	0		0	999	0	V
P0006	Drive Status	0: Ready		0	5	0: Ready	
P0008	DI3 to DI1 Status	0		0	7	0	
P0009	DI101 to DI106 Status	0		0	63	0	
P0010	DI107 to DI112 Status	0		0	63	0	
P0011	DI201 to DI206 Status	0		0	63	0	
P0012	DI207 to DI212 Status	0		0	63	0	
P0013	DI301 to DI306 Status	0		0	63	0	
P0014	DI307 to DI312 Status	0		0	63	0	
P0015	DO1 Status	0		0	1	0	
P0016	DO101 to DO106 Status	0		0	63	0	
P0017	DO201 to DO206 Status	0		0	63	0	
P0018	DO301 to DO306 Status	0		0	63	0	
P0021	Internal Air Temperature	0		0	1000	0	°C
P0022	Dissipator Temperature	0		0	1000	0	°C
P0023	Software Version	9.99		0.00	655.35	9.99	
P0024	Bootloader Version	0.00		0.00	655.35	0.00	
P0025	FPGA Project Version	0.00		0.00	655.35	0.00	
P0030	Present Alarm	0		0	2000	0	
P0031	Last Alarm	0		0	2000	0	
P0032	Last alarm Day.Month	0.00		0.00	31.12	0.00	
P0033	Last alarm year	0		0	4096	0	
P0034	Last alarm Hour.Min	0.00		0.00	23.59	0.00	
P0035	Present Fault	0		0	2000	0	
P0036	Last Fault	0		0	2000	0	
P0037	Last Fault Day.Month	0.00		0.00	31.12	0.00	
P0038	Last fault year	0		0	4096	0	
P0039	Last fault Hour.Min	0.00		0.00	23.59	0.00	
P0040	Second fault	0		0	2000	0	
P0041	Second fault Day.Month	0.00		0.00	31.12	0.00	
P0042	Second fault year	0		0	4096	0	
P0043	Second fault Hour.Min	0.00		0.00	23.59	0.00	
P0044	Third fault	0		0	2000	0	
P0045	Third fault Day.Month	0.00		0.00	31.12	0.00	
P0046	Third fault year	0		0	4096	0	

Configuration Table.

On the user parameter configuration table, it is possible to define several attributes to the parameters, such as description, minimum and maximum values, unit, digits, data type, etc.

NOTE!
These settings will be automatically displayed in the parameter table. However, to be sent to the device, you need to *download* the resource.

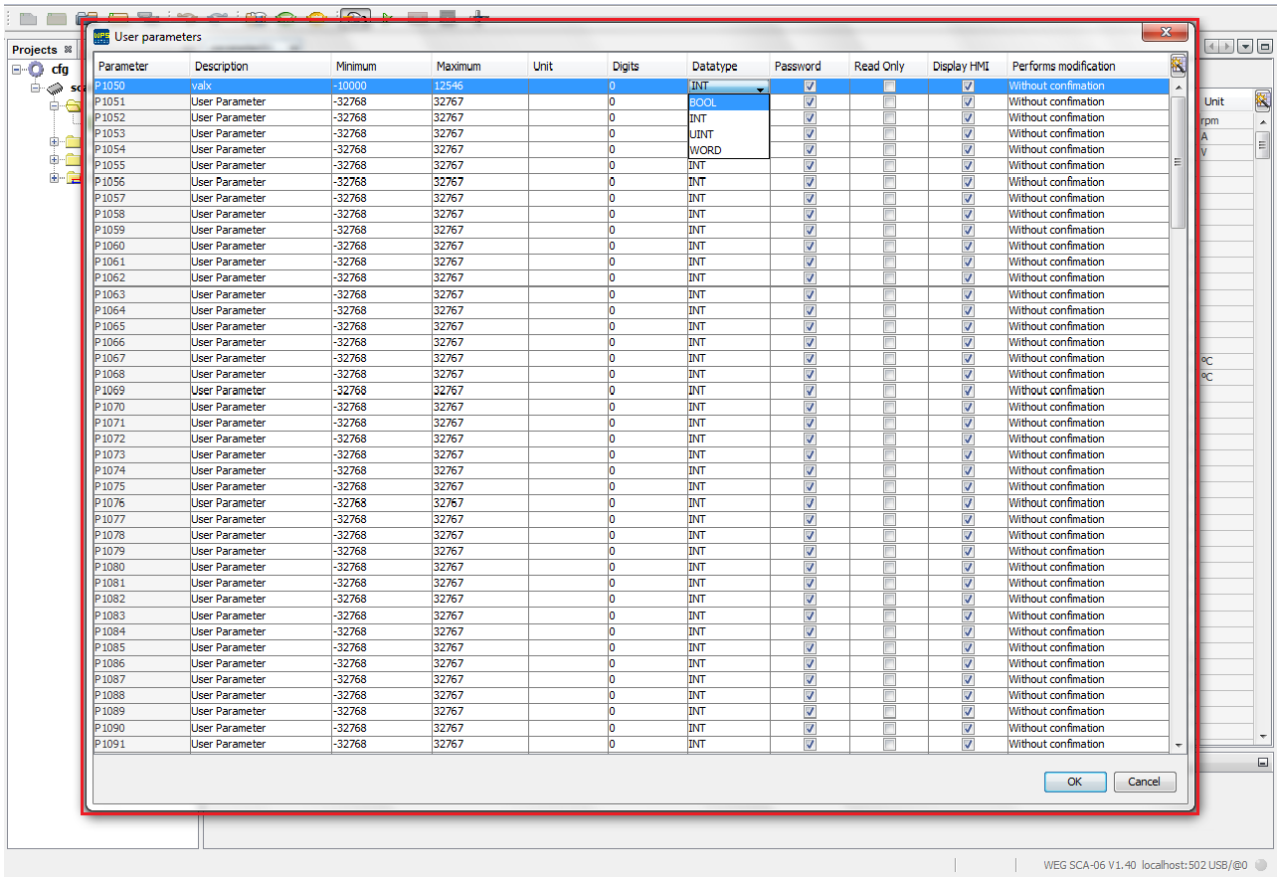


Table fields:

- Parameter: User parameter identification.
- Description: Description of the user parameter in the parameter table. On devices that have text-based HMIs, the description is sent to the machine and displayed on the HMI.
- Minimum: Minimum input value for parameter.
- Maximum: Maximum input value for parameter.
- Unit: Unit displayed on the device's HMI.
- Default: Value loaded when restore factory default is selected.
- Retentive: Retain value after rebooting devices.
- Hexadecimal: Displays the value in hexadecimal.
- Digits: Number of decimal digits for displaying value.
- Datatype: Parameter datatype used by the ladder application.
- Password: Enables password request by changing parameter value.

- Read only: It does not allow the writing of values in the parameter by the communication network or the HMI. Writing is done only by the ladder application.

- Display HMI: Displays the parameter in the HMI.

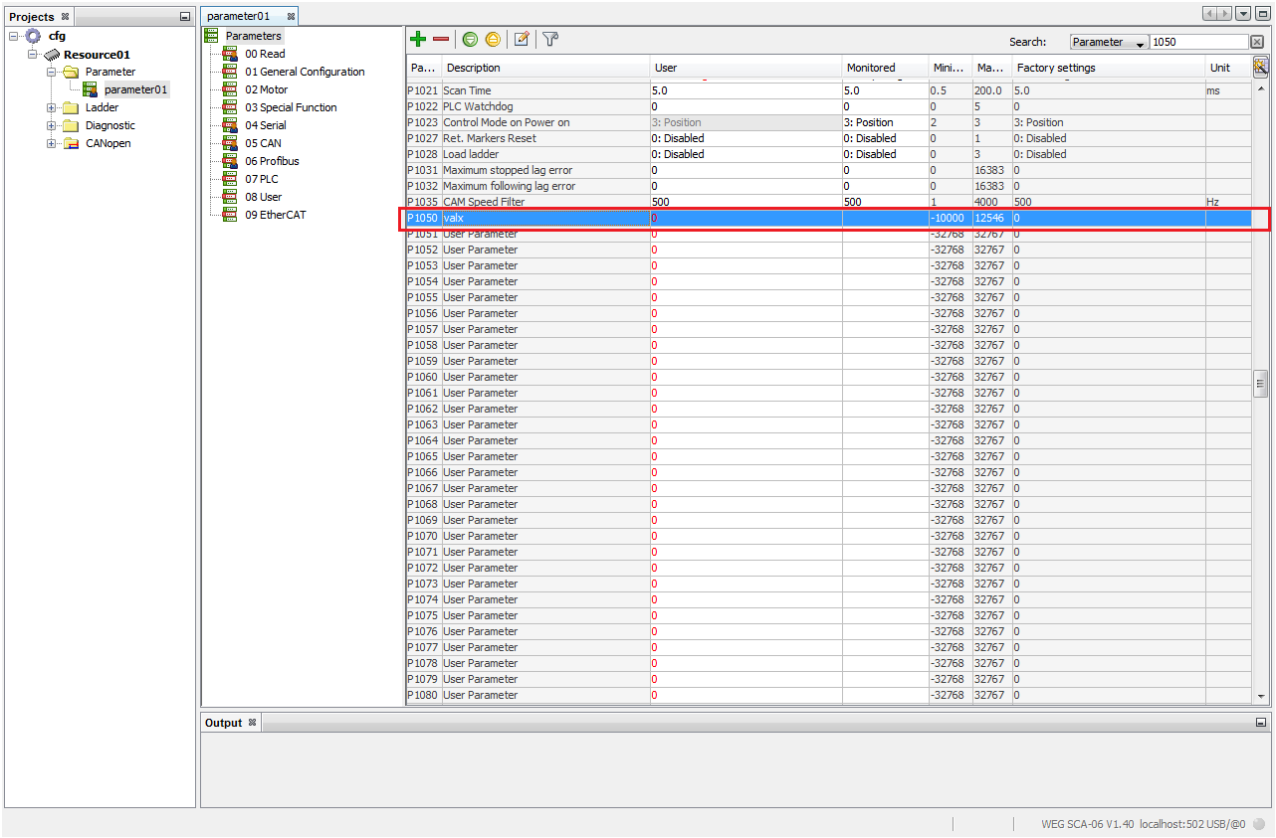
- Performs modification: Confirmation options when changing the parameter:
o No confirmation: Does not prompt for confirmation when changing parameter.
o With confirmation and engine stopped: Request confirmation and allow change only with engine stopped.
o With confirmation: Prompt for confirmation when changing parameter.

- Stopped motor: Perform change only with motor stopped.

- Help: On devices that have text-based HMI, you can edit a help text for the parameter.

View the user parameter

In the parameter table, the user parameters will be shown as they are configured on the configuration screen.



11.5 CFW-11

Enter topic text here.

11.5.1 Description

Refer to the user's manual of the CFW-11 for further details about the product.

NOTE!

This product does not have the Ladder tool available in WPS.
You can use the WLP application if this feature is required.

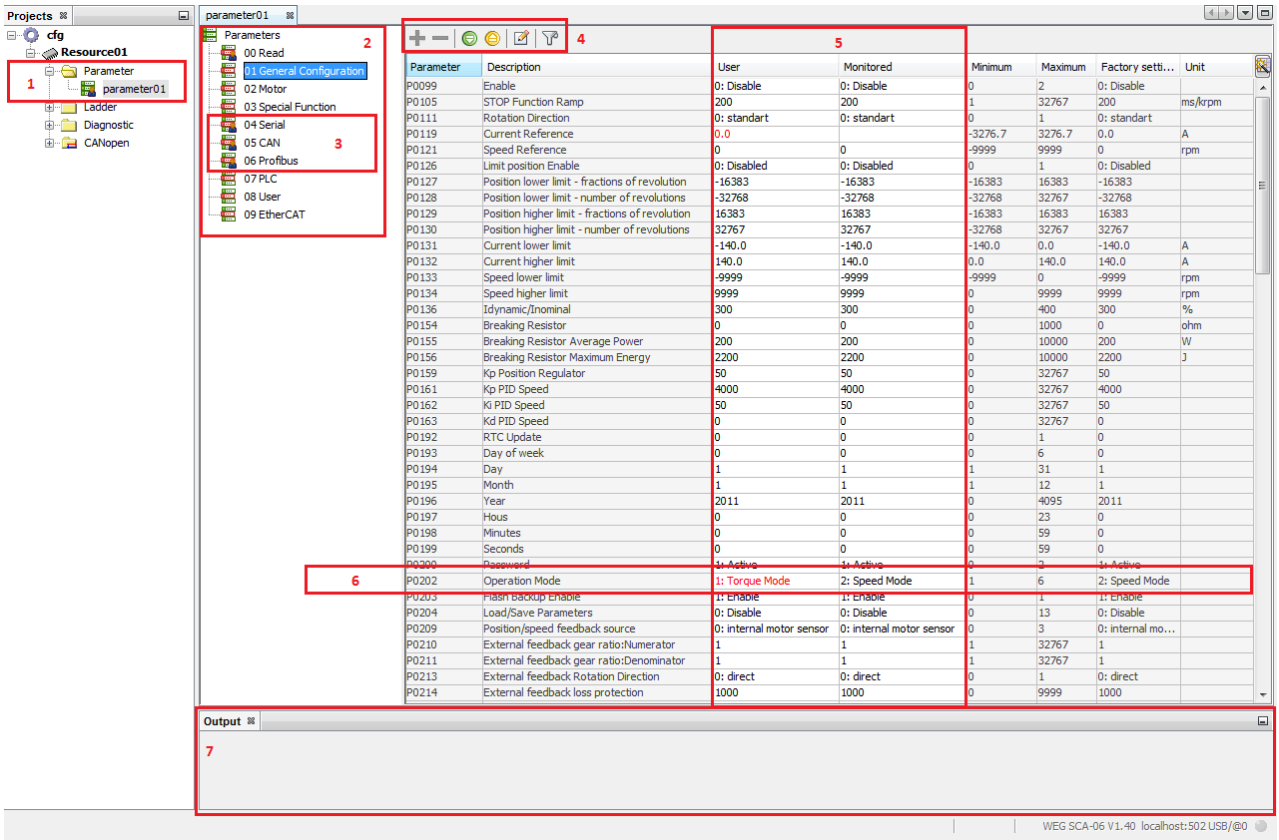
11.5.2 Parameters

11.5.2.1 Overview

The parameter configuration screen is used to configure and monitor all the parameters of the equipment, including the user parameters.

NOTE!
The reading and writing of such parameters is done on this screen; only the user parameter configuration must be sent the first time or whenever modified by means of the resource download routine.

Below is an overview of the parameter configuration screen.



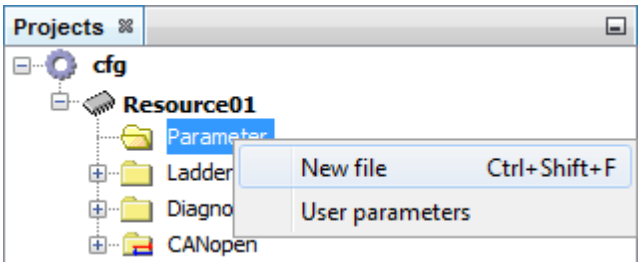
Parameter	Description	User	Monitored	Minimum	Maximum	Factory setti...	Unit
P0099	Enable	0: Disable	0: Disable	0	2	0: Disable	
P0105	STOP Function Ramp	200	200	1	32767	200	ms/krpm
P0111	Rotation Direction	0: standart	0: standart	0	1	0: standart	
P0119	Current Reference	0.0		-3276.7	3276.7	0.0	A
P0121	Speed Reference	0	0	-9999	9999	0	rpm
P0126	Limit position Enable	0: Disabled	0: Disabled	0	1	0: Disabled	
P0127	Position lower limit - fractions of revolution	-16383	-16383	-16383	16383	-16383	
P0128	Position lower limit - number of revolutions	-32768	-32768	-32768	32767	-32768	
P0129	Position higher limit - fractions of revolution	16383	16383	-16383	16383	16383	
P0130	Position higher limit - number of revolutions	32767	32767	-32768	32767	32767	
P0131	Current lower limit	-140.0	-140.0	-140.0	0.0	-140.0	A
P0132	Current higher limit	140.0	140.0	0.0	140.0	140.0	A
P0133	Speed lower limit	-9999	-9999	-9999	0	-9999	rpm
P0134	Speed higher limit	9999	9999	0	9999	9999	rpm
P0136	Idynamic/Inominal	300	300	0	400	300	%
P0154	Breaking Resistor	0	0	0	1000	0	ohm
P0155	Breaking Resistor Average Power	200	200	0	10000	200	W
P0156	Breaking Resistor Maximum Energy	2200	2200	0	10000	2200	J
P0159	Kp Position Regulator	50	50	0	32767	50	
P0161	Kp PID Speed	4000	4000	0	32767	4000	
P0162	Ki PID Speed	50	50	0	32767	50	
P0163	Kd PID Speed	0	0	0	32767	0	
P0192	RTC Update	0	0	0	1	0	
P0193	Day of week	0	0	0	6	0	
P0194	Day	1	1	1	31	1	
P0195	Month	1	1	1	12	1	
P0196	Year	2011	2011	0	4095	2011	
P0197	Hous	0	0	0	23	0	
P0198	Minutes	0	0	0	59	0	
P0199	Seconds	0	0	0	59	0	
P0200	Password	1: Active	1: Active	0	3	1: Active	
P0202	Operation Mode	1: Torque Mode	2: Speed Mode	1	6	2: Speed Mode	
P0203	Flash Backup Enable	1: Enable	1: Enable	0	1	1: Enable	
P0204	Load/Save Parameters	0: Disable	0: Disable	0	13	0: Disable	
P0209	Position/speed feedback source	0: internal motor sensor	0: internal motor sensor	0	3	0: internal mo...	
P0210	External feedback gear ratio:Numerator	1	1	1	32767	1	
P0211	External feedback gear ratio:Denominator	1	1	1	32767	1	
P0213	External feedback Rotation Direction	0: direct	0: direct	0	1	0: direct	
P0214	External feedback loss protection	1000	1000	0	9999	1000	

1. **Parameter files**. In this part are all the parameter configuration files created by the user. Notice that when the file features a person figure on the table, it means this parameter table contains hidden parameters/ group of parameters.
2. **Group of parameters**. This tree shows all the group of parameters. Notice that the same parameter can be in more than one group, and when its value is modified, it will be modified in all the groups to which it belongs.
3. **Modified group of parameters**. Group of parameters which contain the figure of a person on the table means they have hidden parameters.
4. **Commands**. The commands are described below in the order they appear:
   4.1. **Unhide parameter**: In case some parameter has been hidden, this button allows making it visible again.
   4.2. **Hide parameter**: Just select one or more parameters on the table and trigger this command to hide them.
   4.3. **Save table**: It saves the values of the parameters shown on the equipment screen; the sent values are the ones in the User column. The flow is User -> Monitored (equipment)
   4.4. **Read table**: It reads the parameters of the equipment shown in the Monitored column and saves them in the parameter file in the User column. The flow is Monitored (equipment) -> User
   4.5. **User parameters**: It opens a screen to edit the user parameters.
   4.6. **Filter**: It opens a parameter filter option, and it can filter by parameter number or description.
   4.7. **User Parameters and Monitored Parameters**. These two columns show the off-line and on-line parameters, so to speak. The **User** column shows the values contained in the file located on the computer and the **Monitored** column shows the values that are effectively saved on the equipment. Whenever you use the **Save Parameter** option, the sent values will be from the **User** column to the **Monitored** column, that is, File -> Equipment. In case of reading, the flow is the opposite, from the **Monitored** column to the **User** column, that is, Equipment -> File. In case you wish to change the values directly on the equipment without changing it in the file, just click on the monitored column, change the values and the modification will occur on-line.
5. **Modified parameters**: Whenever a parameter value in the **User** column is different from the **Monitored** column, it will be shown in red.
6. **Output**. This screen shows error information in case they occur during the writing or reading of the parameters.

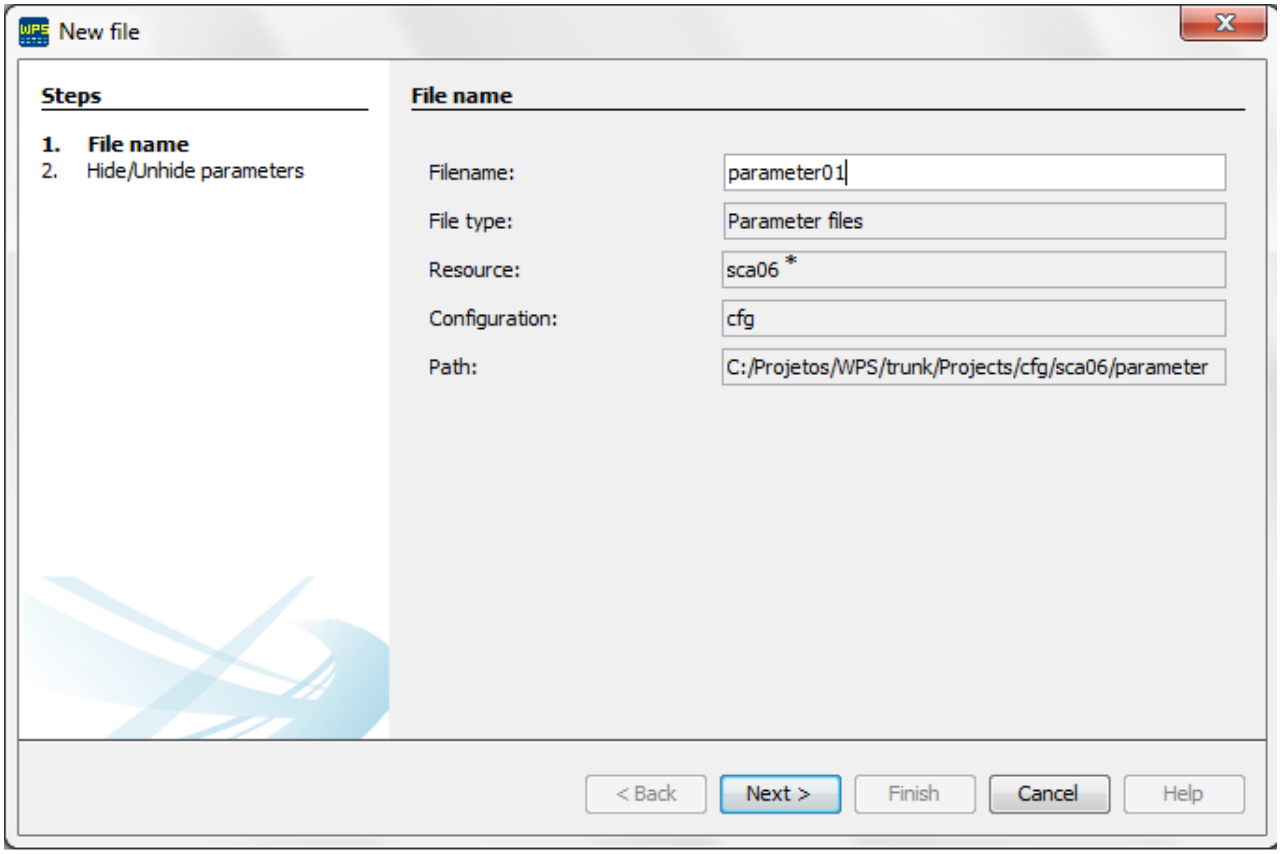
11.5.2.2 Configuration

Below is the list of the required steps to create a parameter file.

1. Create a new parameter file.

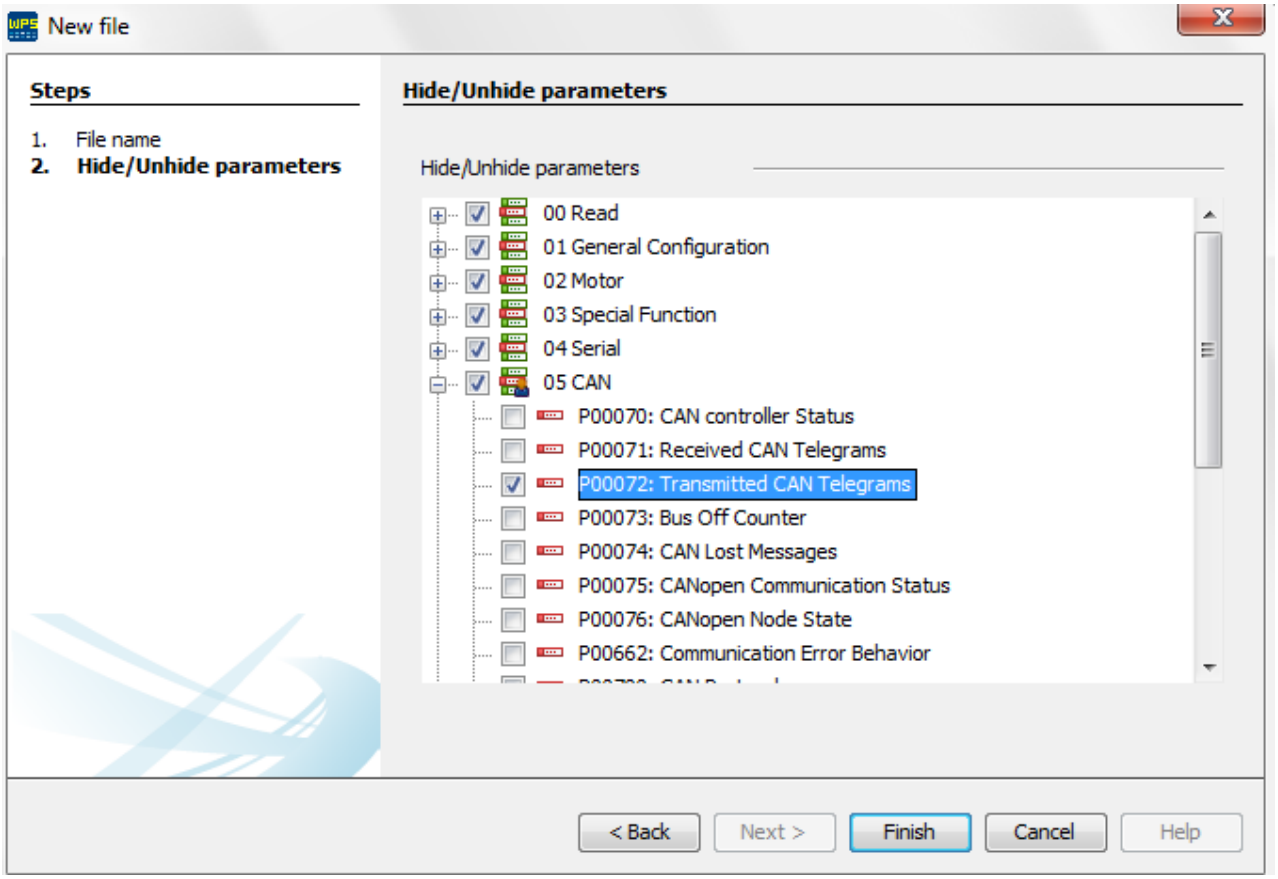


2. Define a name for the parameter file

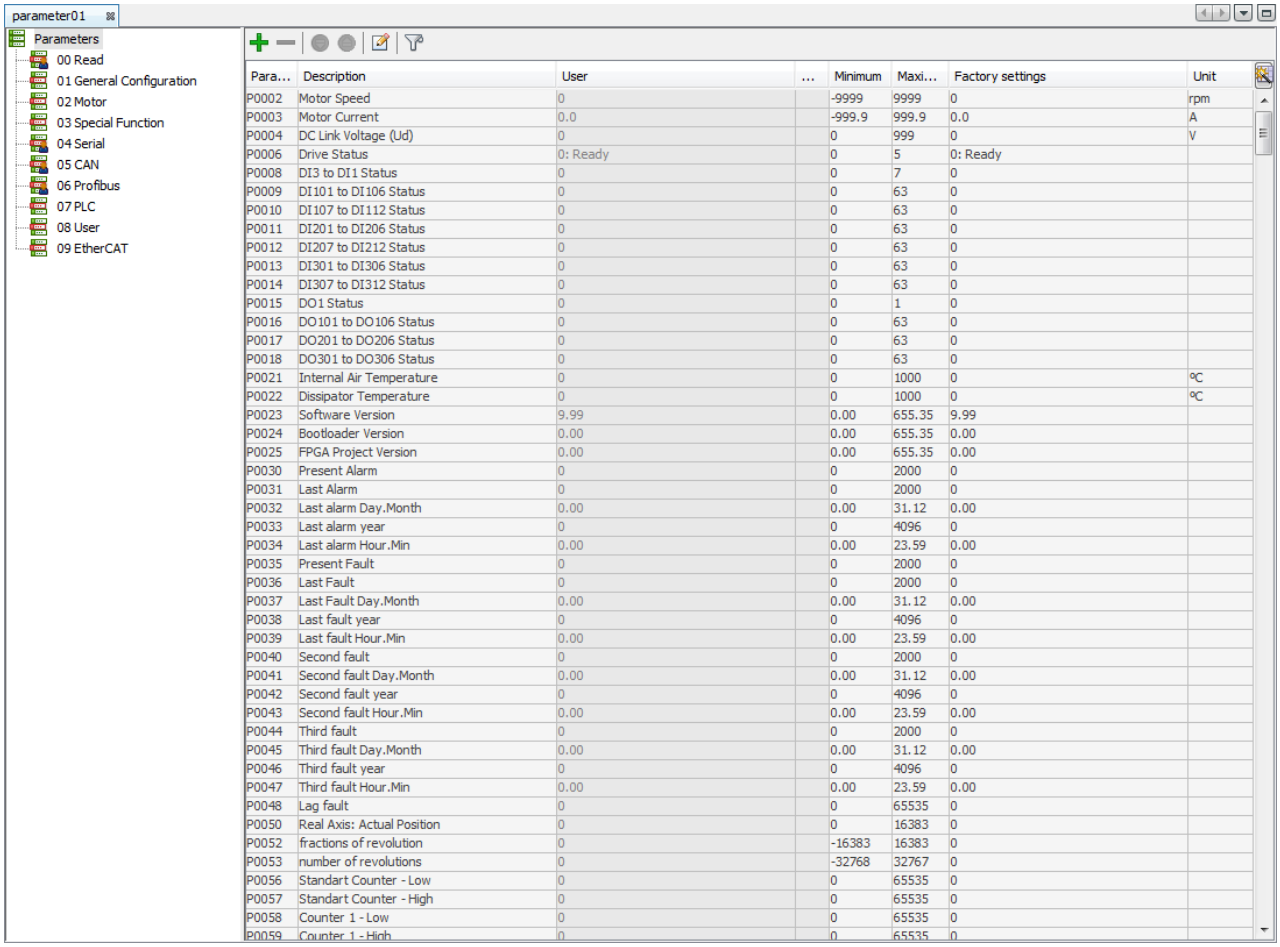


*** Resource: Resource01, SCA06, CFW300, etc.**

3. Configure which parameters you wish to view in your parameter table



4. After performing the steps above, the parameter file will be created and the equipment can be parameterized.

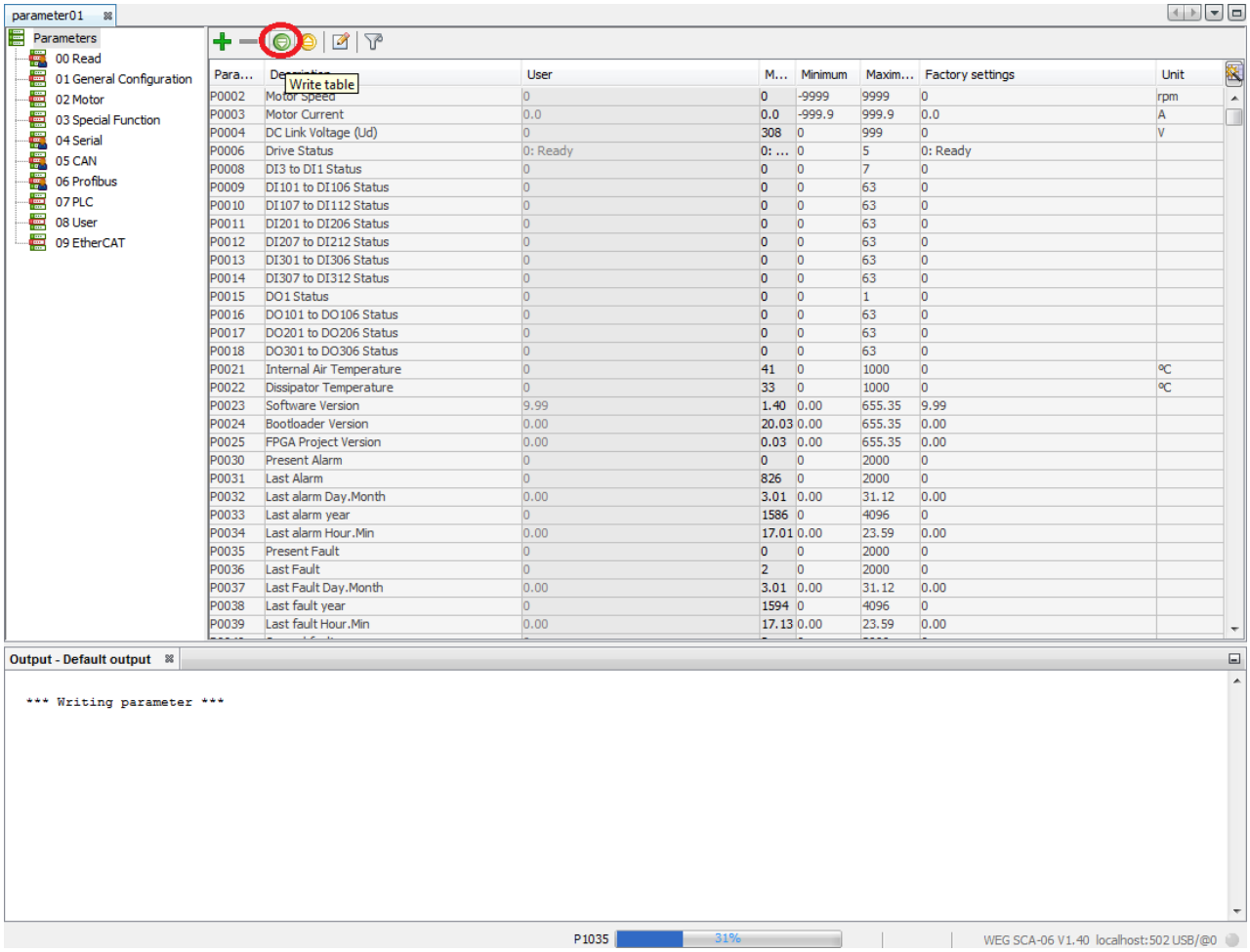


Para...	Description	User	...	Minimum	Maxi...	Factory settings	Unit
P0002	Motor Speed	0		-9999	9999	0	rpm
P0003	Motor Current	0.0		-999.9	999.9	0.0	A
P0004	DC Link Voltage (Ud)	0		0	999	0	V
P0006	Drive Status	0: Ready		0	5	0: Ready	
P0008	DI3 to DI1 Status	0		0	7	0	
P0009	DI101 to DI106 Status	0		0	63	0	
P0010	DI107 to DI112 Status	0		0	63	0	
P0011	DI201 to DI206 Status	0		0	63	0	
P0012	DI207 to DI212 Status	0		0	63	0	
P0013	DI301 to DI306 Status	0		0	63	0	
P0014	DI307 to DI312 Status	0		0	63	0	
P0015	DO1 Status	0		0	1	0	
P0016	DO101 to DO106 Status	0		0	63	0	
P0017	DO201 to DO206 Status	0		0	63	0	
P0018	DO301 to DO306 Status	0		0	63	0	
P0021	Internal Air Temperature	0		0	1000	0	ºC
P0022	Dissipator Temperature	0		0	1000	0	ºC
P0023	Software Version	9.99		0.00	655.35	9.99	
P0024	Bootloader Version	0.00		0.00	655.35	0.00	
P0025	FPGA Project Version	0.00		0.00	655.35	0.00	
P0030	Present Alarm	0		0	2000	0	
P0031	Last Alarm	0		0	2000	0	
P0032	Last alarm Day.Month	0.00		0.00	31.12	0.00	
P0033	Last alarm year	0		0	4096	0	
P0034	Last alarm Hour.Min	0.00		0.00	23.59	0.00	
P0035	Present Fault	0		0	2000	0	
P0036	Last Fault	0		0	2000	0	
P0037	Last Fault Day.Month	0.00		0.00	31.12	0.00	
P0038	Last fault year	0		0	4096	0	
P0039	Last fault Hour.Min	0.00		0.00	23.59	0.00	
P0040	Second fault	0		0	2000	0	
P0041	Second fault Day.Month	0.00		0.00	31.12	0.00	
P0042	Second fault year	0		0	4096	0	
P0043	Second fault Hour.Min	0.00		0.00	23.59	0.00	
P0044	Third fault	0		0	2000	0	
P0045	Third fault Day.Month	0.00		0.00	31.12	0.00	
P0046	Third fault year	0		0	4096	0	
P0047	Third fault Hour.Min	0.00		0.00	23.59	0.00	
P0048	Lag fault	0		0	65535	0	
P0050	Real Axis: Actual Position	0		0	16383	0	
P0052	fractions of revolution	0		-16383	16383	0	
P0053	number of revolutions	0		-32768	32767	0	
P0056	Standart Counter - Low	0		0	65535	0	
P0057	Standart Counter - High	0		0	65535	0	
P0058	Counter 1 - Low	0		0	65535	0	
P0059	Counter 1 - High	0		0	65535	0	

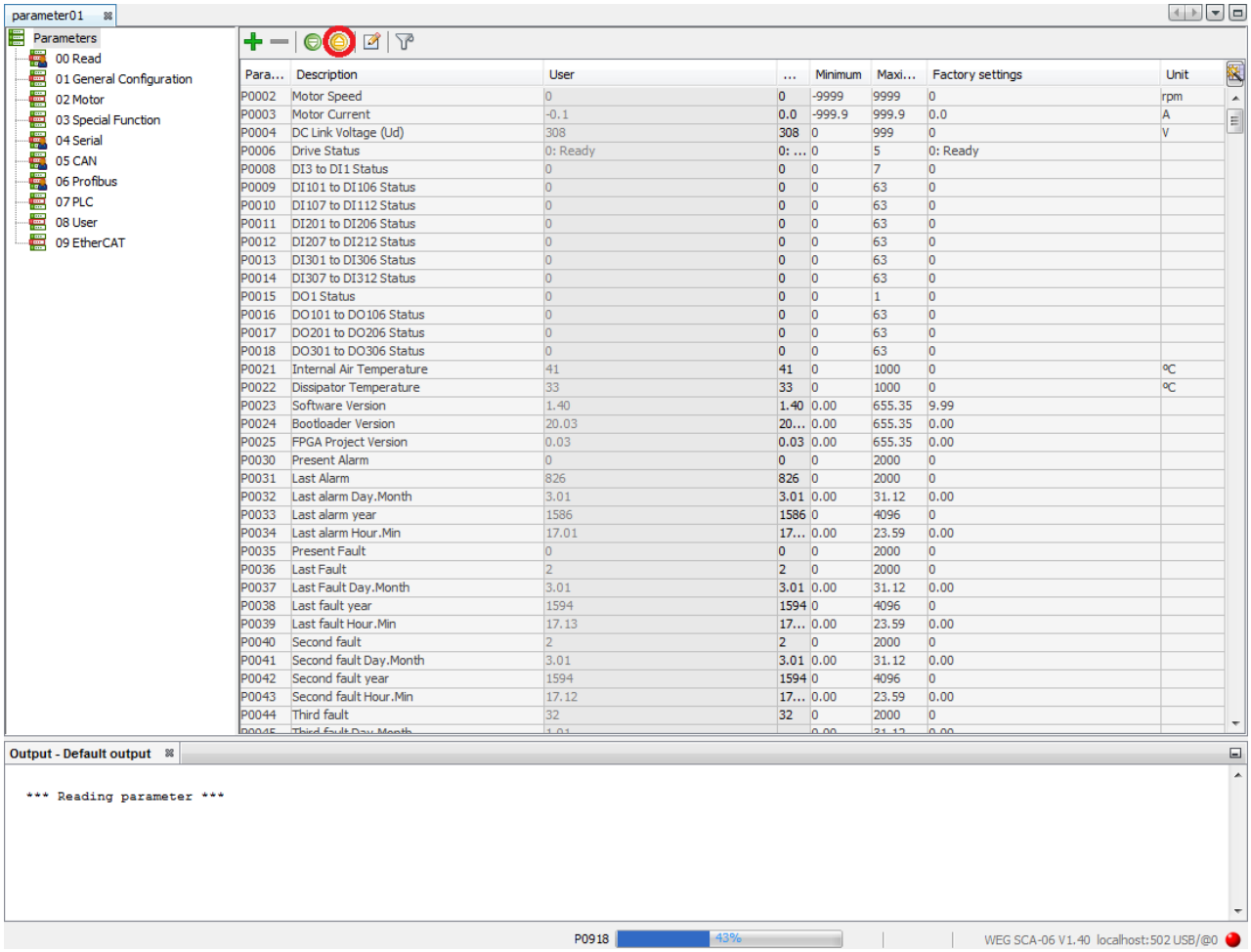
11.5.2.3 Read and Write of Parameters

There are 3 (three) ways to do the reading and writing of the parameters: by means of table, selection and group.

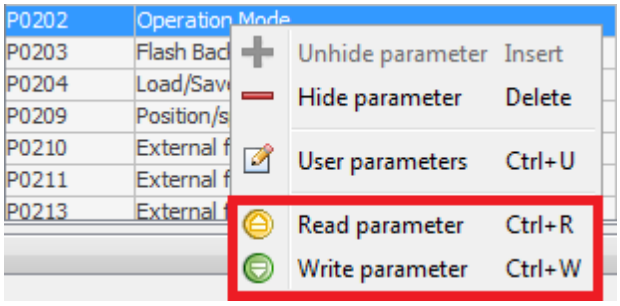
1. **Table writing**. The table writing command will send all visible parameters on the equipment screen. If and error occurs during the sending of some specific parameter, a message will be shown on the output window informing the error. It is important to notice that only visible parameters will be sent; therefore, it is necessary attention to which node of the group of parameters tree you are viewing. Example: If you wish to write all of them without filtering per group, just select the tree root.



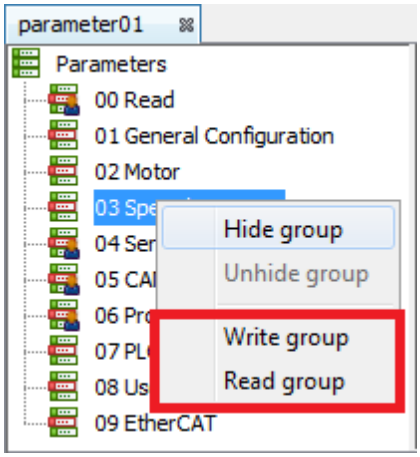
2. **Table reading**. The table reading command will read all the parameters of the equipment. If a error occurs during the reading of some specific parameter, a message will be shown on the output window informing the error. It is important to notice that only visible parameters will be read; therefore, it is necessary attention to which node of the group of parameters tree you are viewing. Example: If you wish to read all of them without filtering per group, just select the tree root.



3. **Reading/writing of specific parameters**. In order to read/write one or more specific parameters, just select them on the table, right click and choose the desired option: read or write parameter.



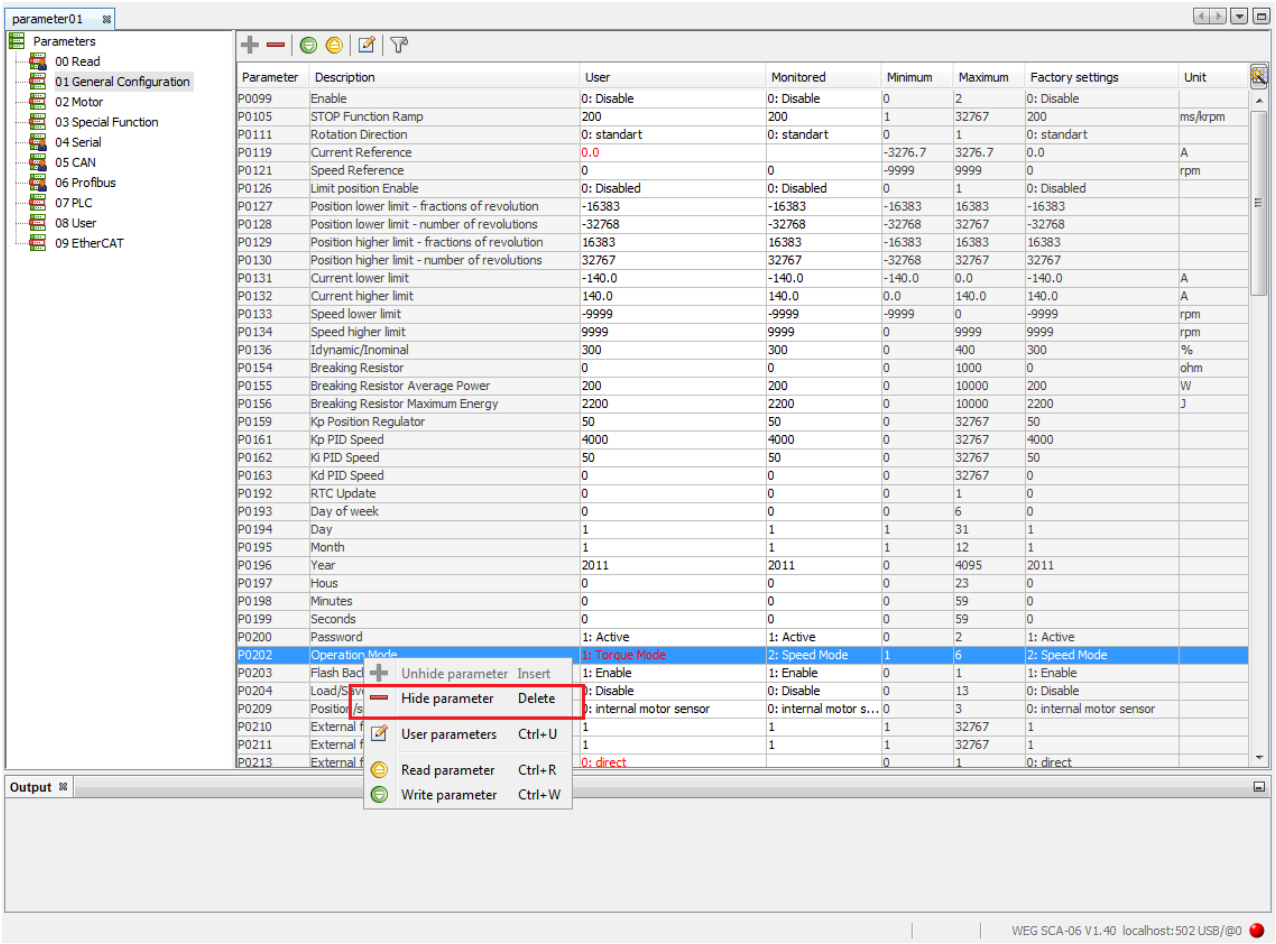
4. **Reading/writing of group of parameters**. In order to read/write only one group of parameters, just select it on the group tree, right click and choose the desired option: read or write group.



11.5.2.4 Hide/Unhide Parameters and Group of Parameters

The parameter can be hidden/unhidden in two ways: individually or in group.

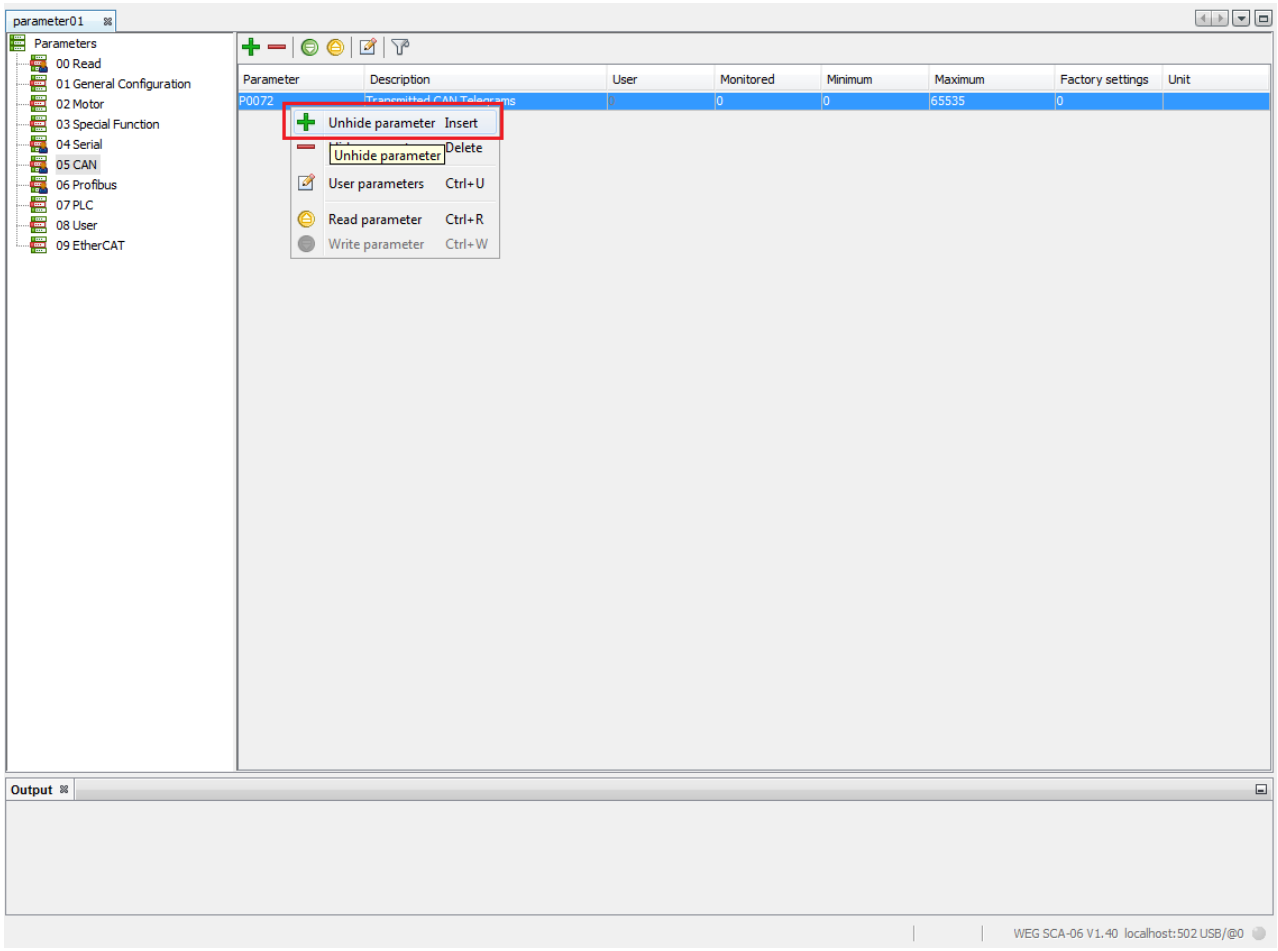
1. **Hide parameters**. In order to hide a parameter individually, just right click on the desired parameters and select the **Hide Parameter** option. You can also press the **Delete** key.

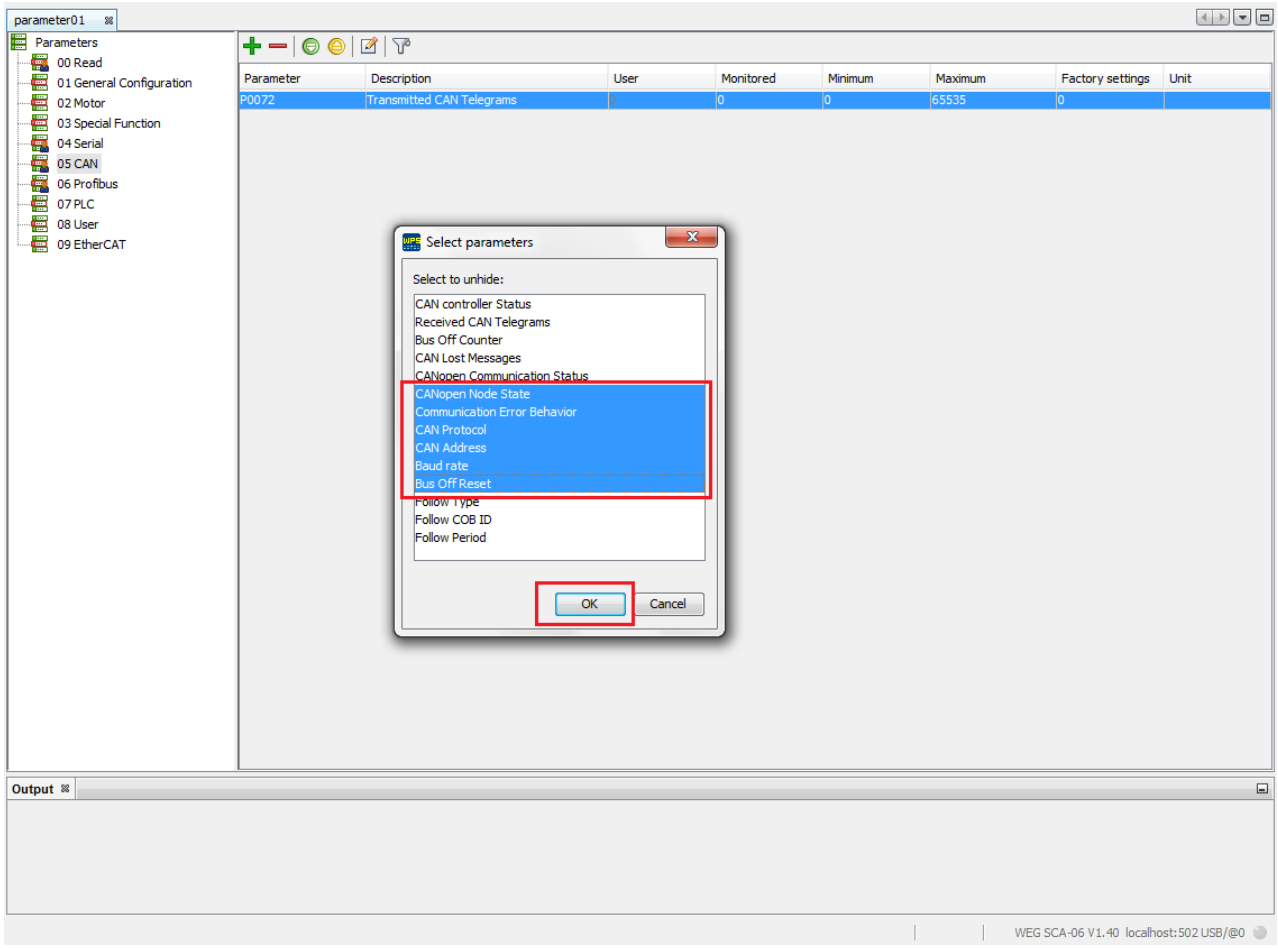


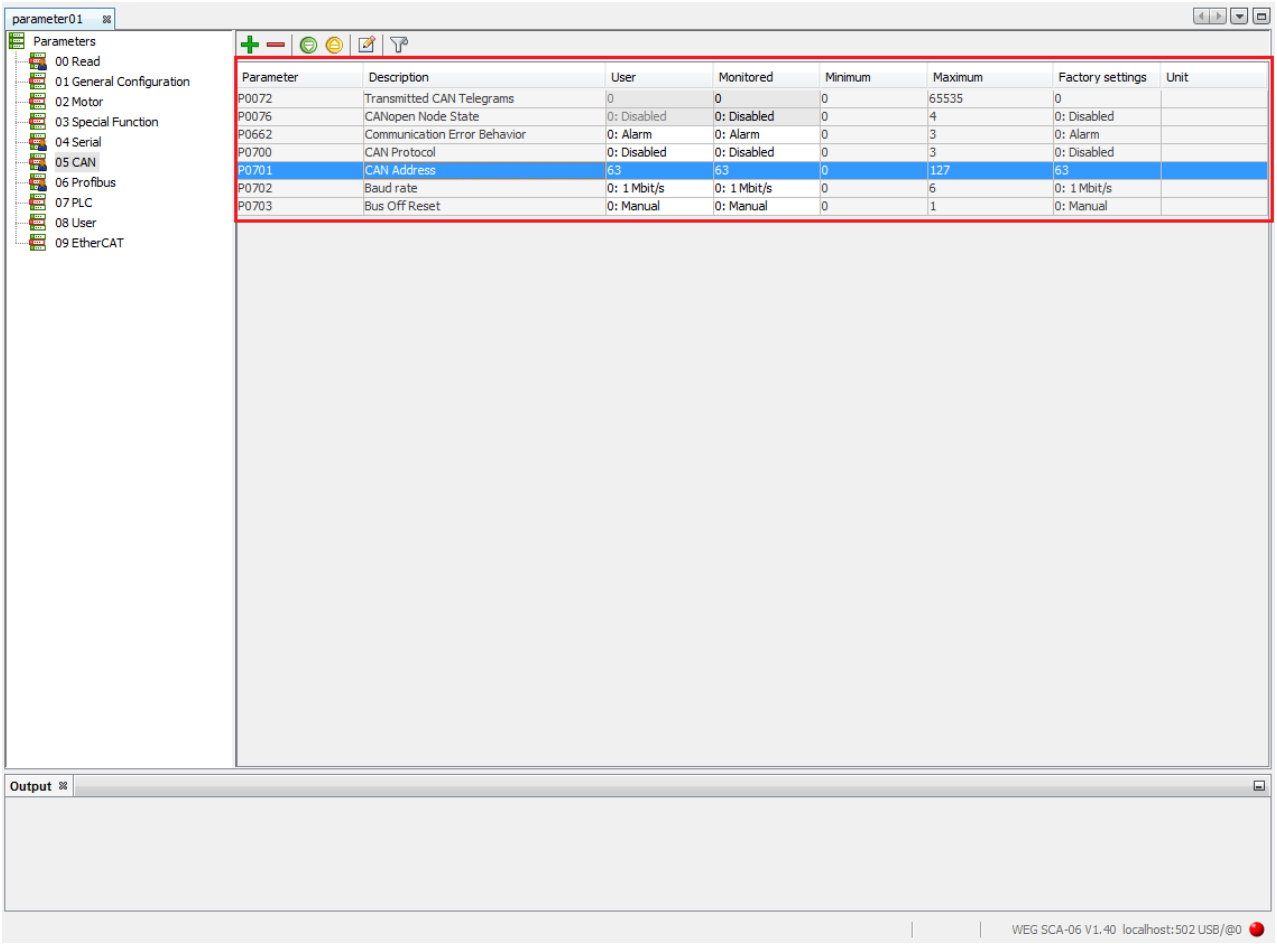
Parameter	Description	User	Monitored	Minimum	Maximum	Factory settings	Unit
P0099	Enable	0: Disable	0: Disable	0	2	0: Disable	
P0105	STOP Function Ramp	200	200	1	32767	200	ms/krpm
P0111	Rotation Direction	0: standart	0: standart	0	1	0: standart	
P0119	Current Reference	0.0		-3276.7	3276.7	0.0	A
P0121	Speed Reference	0	0	-9999	9999	0	rpm
P0126	Limit position Enable	0: Disabled	0: Disabled	0	1	0: Disabled	
P0127	Position lower limit - fractions of revolution	-16383	-16383	-16383	16383	-16383	
P0128	Position lower limit - number of revolutions	-32768	-32768	-32768	32767	-32768	
P0129	Position higher limit - fractions of revolution	16383	16383	-16383	16383	16383	
P0130	Position higher limit - number of revolutions	32767	32767	-32768	32767	32767	
P0131	Current lower limit	-140.0	-140.0	-140.0	0.0	-140.0	A
P0132	Current higher limit	140.0	140.0	0.0	140.0	140.0	A
P0133	Speed lower limit	-9999	-9999	-9999	0	-9999	rpm
P0134	Speed higher limit	9999	9999	0	9999	9999	rpm
P0136	Idynamic/Inominal	300	300	0	400	300	%
P0154	Breaking Resistor	0	0	0	1000	0	ohm
P0155	Breaking Resistor Average Power	200	200	0	10000	200	W
P0156	Breaking Resistor Maximum Energy	2200	2200	0	10000	2200	J
P0159	Kp Position Regulator	50	50	0	32767	50	
P0161	Kp PID Speed	4000	4000	0	32767	4000	
P0162	Ki PID Speed	50	50	0	32767	50	
P0163	Kd PID Speed	0	0	0	32767	0	
P0192	RTC Update	0	0	0	1	0	
P0193	Day of week	0	0	0	6	0	
P0194	Day	1	1	1	31	1	
P0195	Month	1	1	1	12	1	
P0196	Year	2011	2011	0	4095	2011	
P0197	Hous	0	0	0	23	0	
P0198	Minutes	0	0	0	59	0	
P0199	Seconds	0	0	0	59	0	
P0200	Password	1: Active	1: Active	0	2	1: Active	
P0202	Operation Mode	1: Torque Mode	2: Speed Mode	1	6	2: Speed Mode	
P0203	Flash Bac	1: Enable	1: Enable	0	1	1: Enable	
P0204	Load/S	0: Disable	0: Disable	0	13	0: Disable	
P0209	Position/s	0: internal motor sensor	0: internal motor s...	0	3	0: internal motor sensor	
P0210	External f	1	1	1	32767	1	
P0211	External f	1	1	1	32767	1	
P0213	External f	0: direct		0	1	0: direct	

2. **Unhide Parameters**. In order to show hidden parameters, right click and choose the **Unhide Parameters**

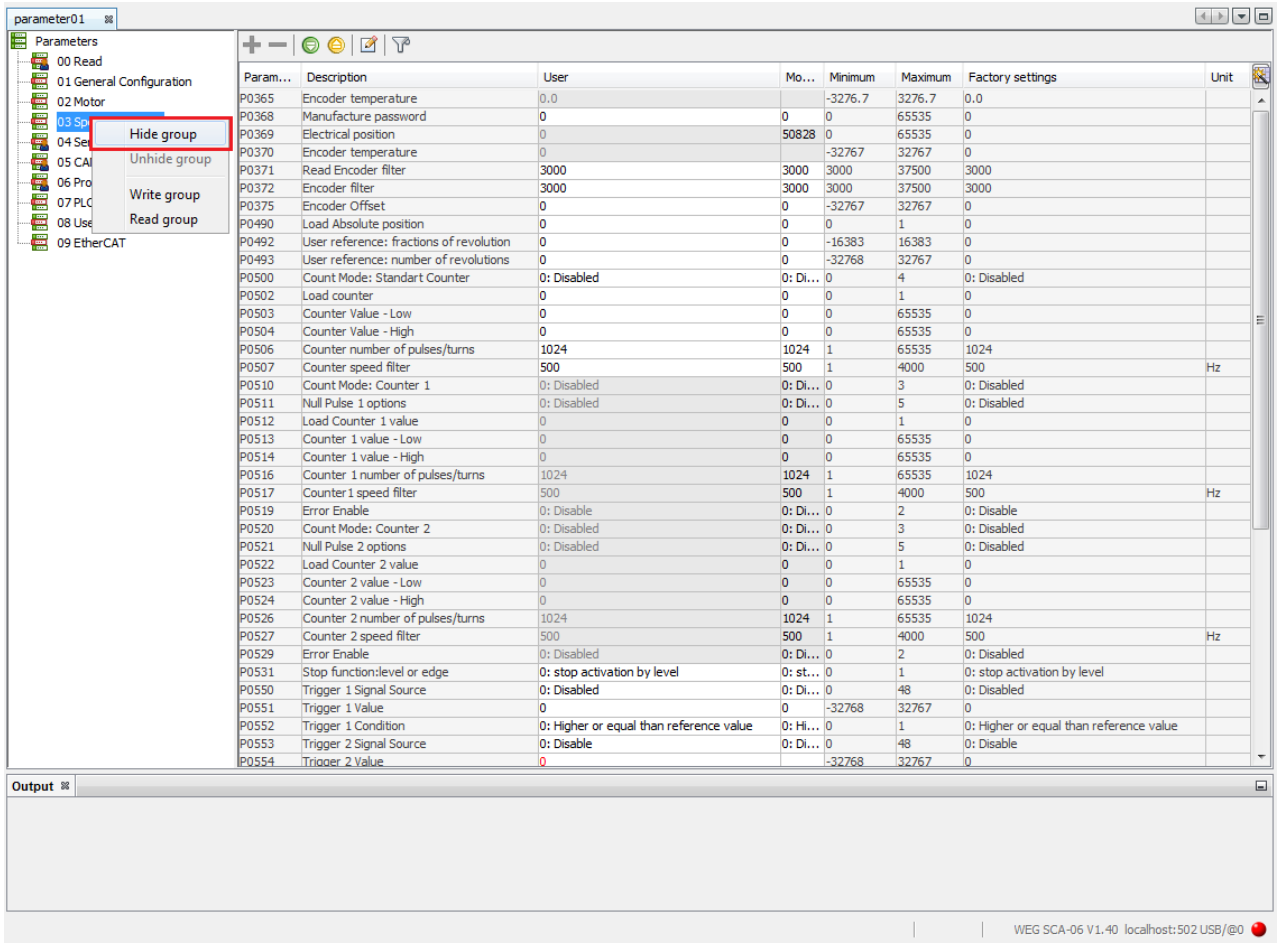
or press the **Insert** key. Then, a window will open and show the hidden parameters. Now, you just have to select the desired parameters and confirm.
Note: The parameters shown on this new window are only those which belong to the current filter according to the selection on the parameter group tree. In the figures below, the CAN group is selected; that means that only the hidden parameters of this group will be shown.

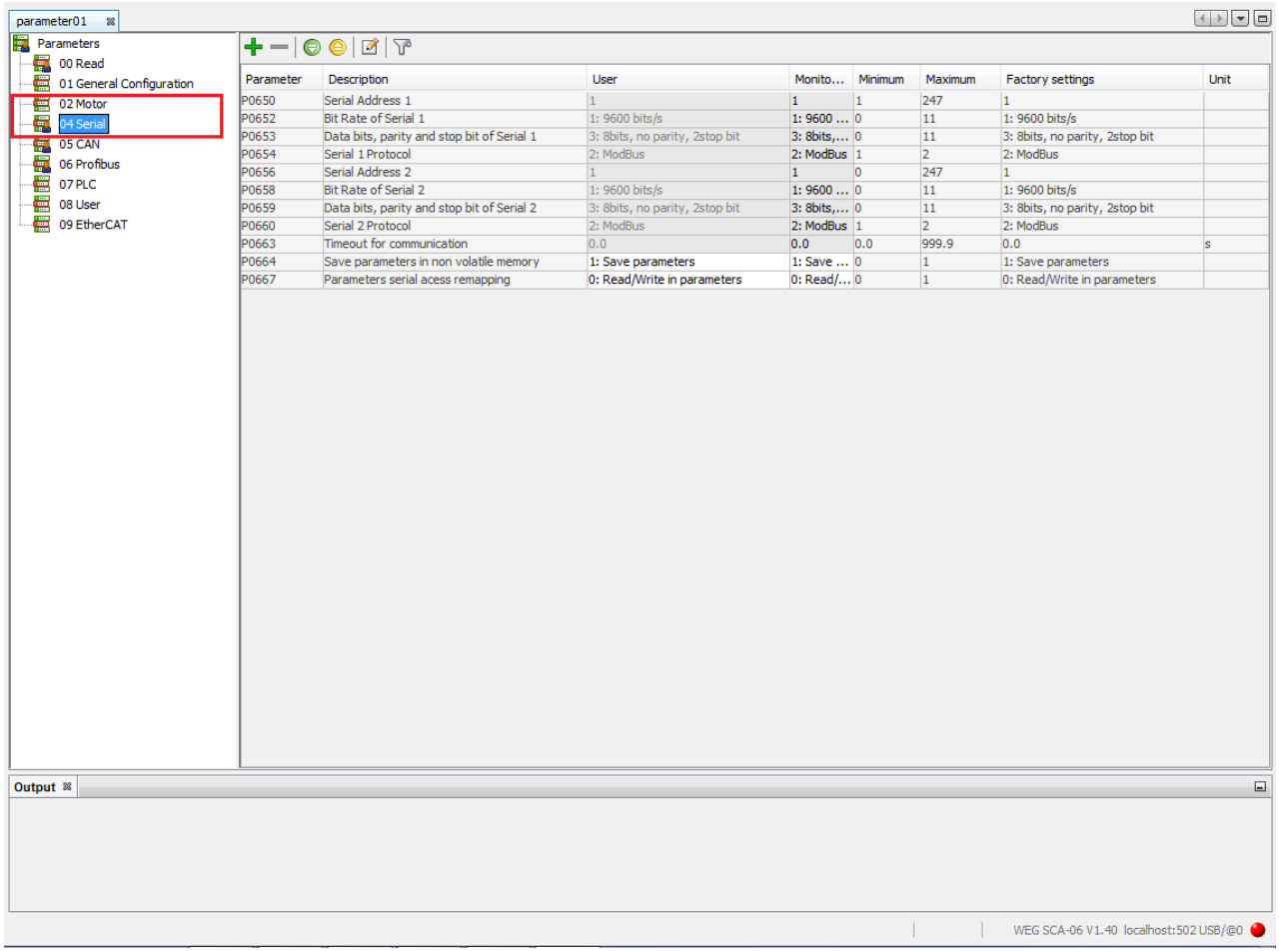



parameter01
Parameters
00 Read
01 General Configuration
02 Motor
03 Special Function
04 Serial
05 CAN
06 Profibus
07 PLC
08 User
09 EtherCAT
Parameter
Description
User
Monitored
Minimum
Maximum
Factory settings
Unit
P0072
Transmitted CAN Telegrams
0
0
0
65535
0
Select parameters
Select to unhide:
CAN controller Status
Received CAN Telegrams
Bus Off Counter
CAN Lost Messages
CANopen Communication Status
CANopen Node State
Communication Error Behavior
CAN Protocol
CAN Address
Baud rate
Bus Off Reset
Follow Type
Follow COB ID
Follow Period
OK
Cancel
Output
WEG SCA-06 V1.40 localhost:502 USB/@0

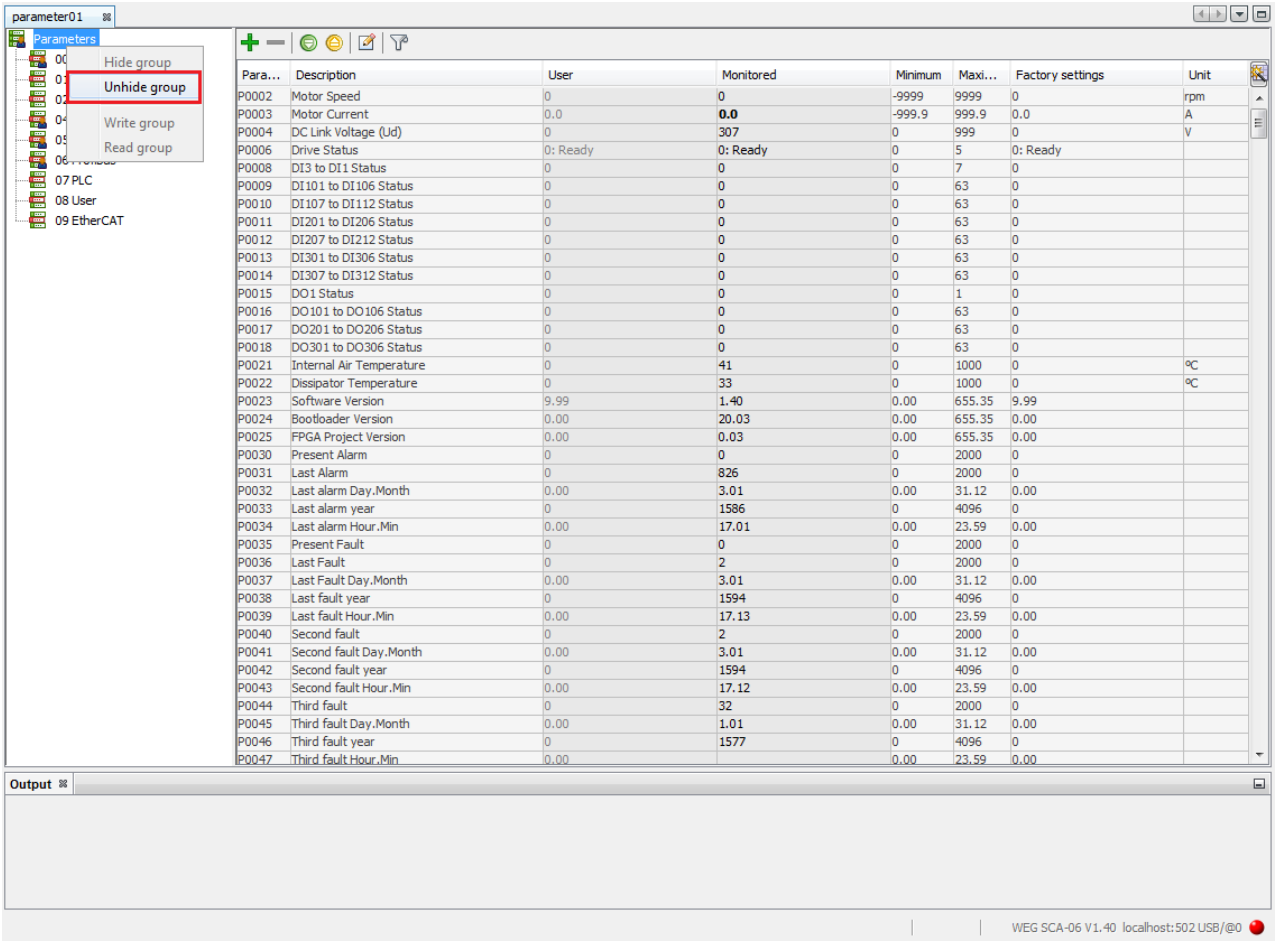


3. **Hide Group of Parameters**. In order to hide a group of parameters, just select the group on the tree and use the **Hide Group** option.

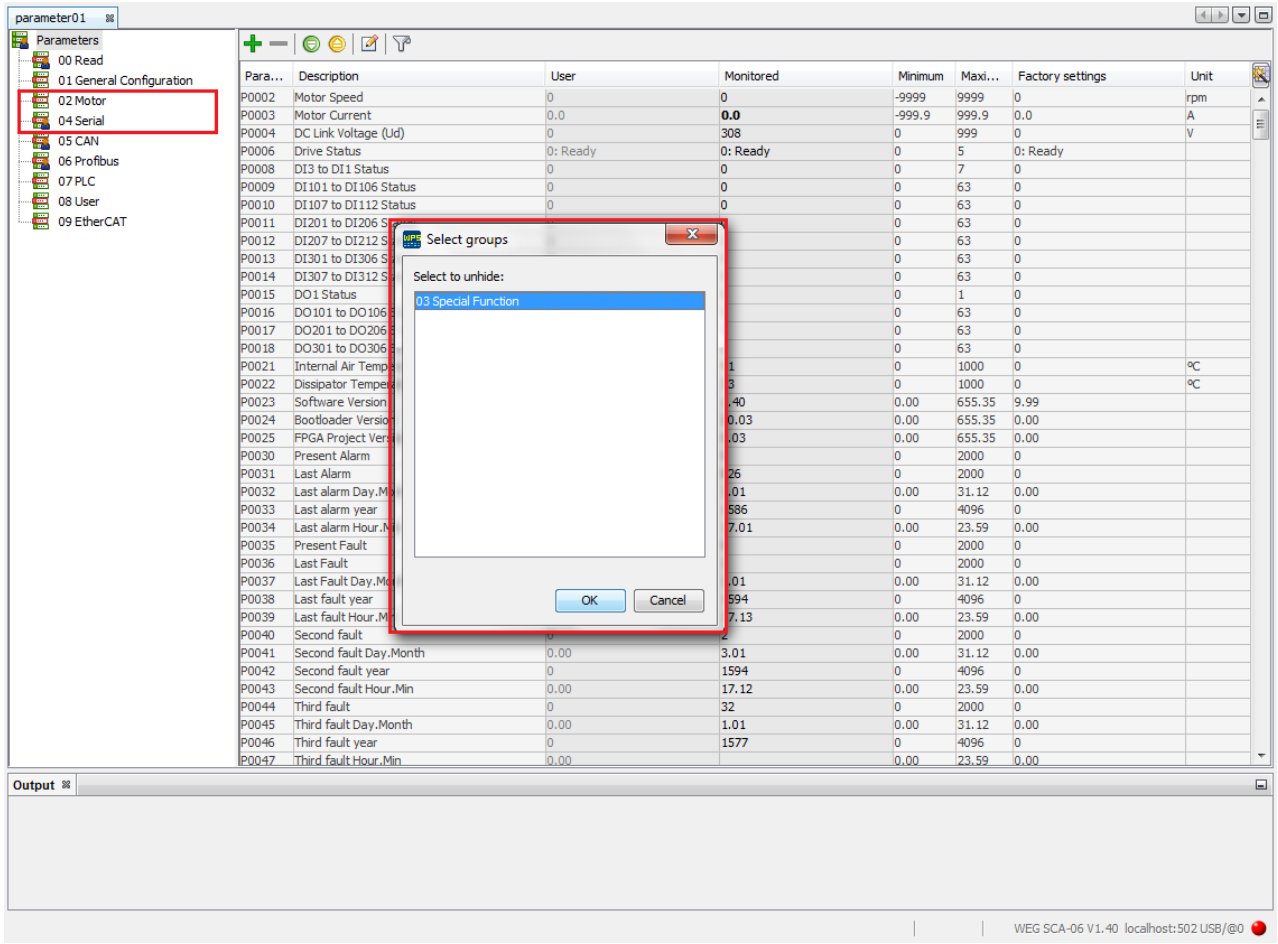

parameter01
Parameters
00 Read
01 General Configuration
02 Motor
03 Sp
04 Se
05 CA
06 Pro
07 PLC
08 Use
09 EtherCAT
Hide group
Unhide group
Write group
Read group
Param...
Description
User
Mo...
Minimum
Maximum
Factory settings
Unit
P0365 Encoder temperature 0.0 -3276.7 3276.7 0.0
P0368 Manufacture password 0 0 0 65535 0
P0369 Electrical position 0 50828 0 65535 0
P0370 Encoder temperature 0 -32767 32767 0
P0371 Read Encoder filter 3000 3000 3000 37500 3000
P0372 Encoder filter 3000 3000 3000 37500 3000
P0375 Encoder Offset 0 0 -32767 32767 0
P0490 Load Absolute position 0 0 0 1 0
P0492 User reference: fractions of revolution 0 0 -16383 16383 0
P0493 User reference: number of revolutions 0 0 -32768 32767 0
P0500 Count Mode: Standart Counter 0: Disabled 0: Di... 0 4 0: Disabled
P0502 Load counter 0 0 0 1 0
P0503 Counter Value - Low 0 0 0 65535 0
P0504 Counter Value - High 0 0 0 65535 0
P0506 Counter number of pulses/turns 1024 1024 1 65535 1024
P0507 Counter speed filter 500 500 1 4000 500 Hz
P0510 Count Mode: Counter 1 0: Disabled 0: Di... 0 3 0: Disabled
P0511 Null Pulse 1 options 0: Disabled 0: Di... 0 5 0: Disabled
P0512 Load Counter 1 value 0 0 0 1 0
P0513 Counter 1 value - Low 0 0 0 65535 0
P0514 Counter 1 value - High 0 0 0 65535 0
P0516 Counter 1 number of pulses/turns 1024 1024 1 65535 1024
P0517 Counter1 speed filter 500 500 1 4000 500 Hz
P0519 Error Enable 0: Disable 0: Di... 0 2 0: Disable
P0520 Count Mode: Counter 2 0: Disabled 0: Di... 0 3 0: Disabled
P0521 Null Pulse 2 options 0: Disabled 0: Di... 0 5 0: Disabled
P0522 Load Counter 2 value 0 0 0 1 0
P0523 Counter 2 value - Low 0 0 0 65535 0
P0524 Counter 2 value - High 0 0 0 65535 0
P0526 Counter 2 number of pulses/turns 1024 1024 1 65535 1024
P0527 Counter 2 speed filter 500 500 1 4000 500 Hz
P0529 Error Enable 0: Disabled 0: Di... 0 2 0: Disabled
P0531 Stop function:level or edge 0: stop activation by level 0: st... 0 1 0: stop activation by level
P0550 Trigger 1 Signal Source 0: Disabled 0: Di... 0 48 0: Disabled
P0551 Trigger 1 Value 0 0 -32768 32767 0
P0552 Trigger 1 Condition 0: Higher or equal than reference value 0: Hi... 0 1 0: Higher or equal than reference value
P0553 Trigger 2 Signal Source 0: Disable 0: Di... 0 48 0: Disable
P0554 Trigger 2 Value 0 -32768 32767 0
Output
WEG SCA-06 V1.40 localhost:502 USB/@0

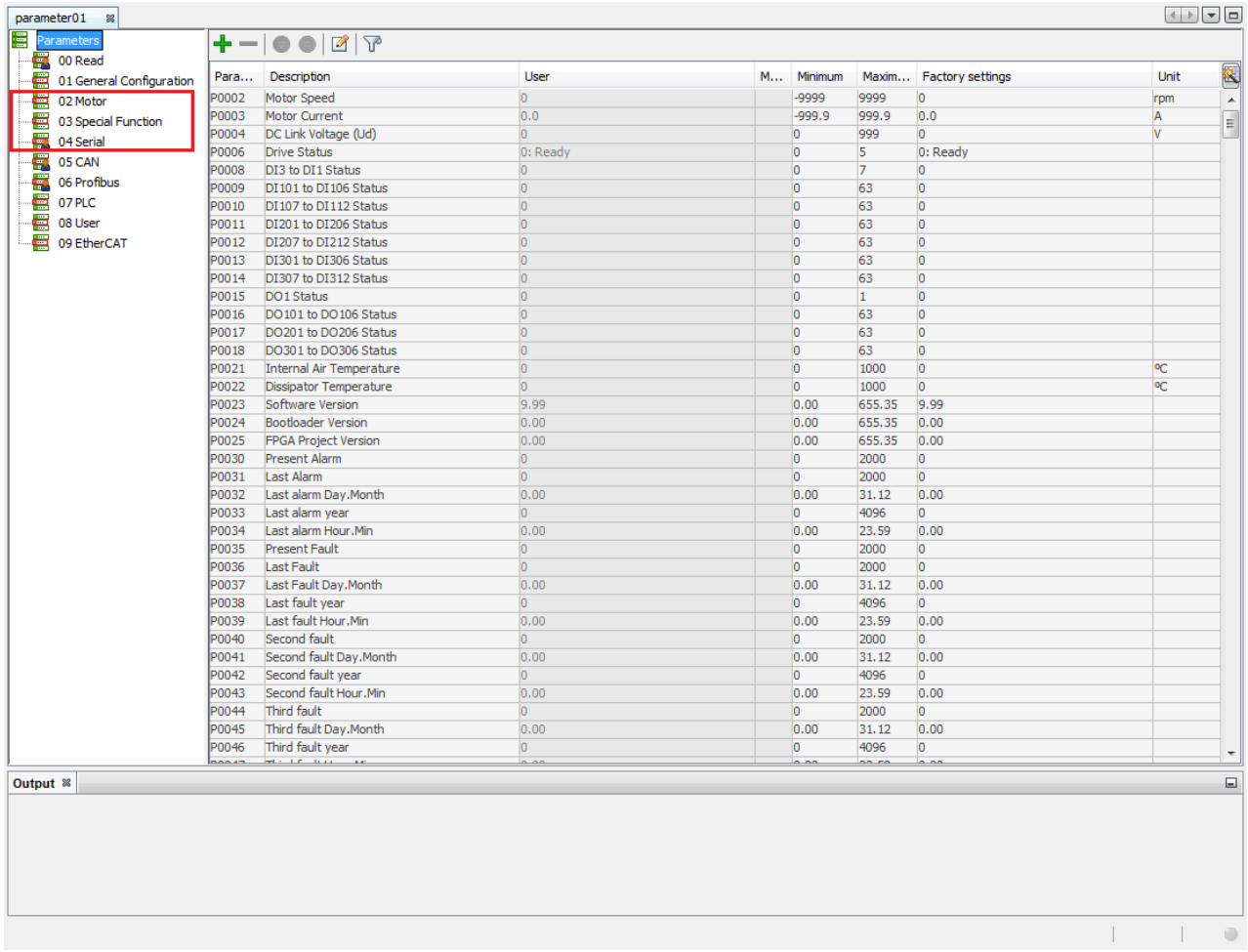


4. **Unhide Group of Parameters**. In order to show a hidden group of parameters, just select the root of the group tree and select the **Unhide Group** option. A window will open showing the groups that are hidden; then just select the group you wish to unhide.



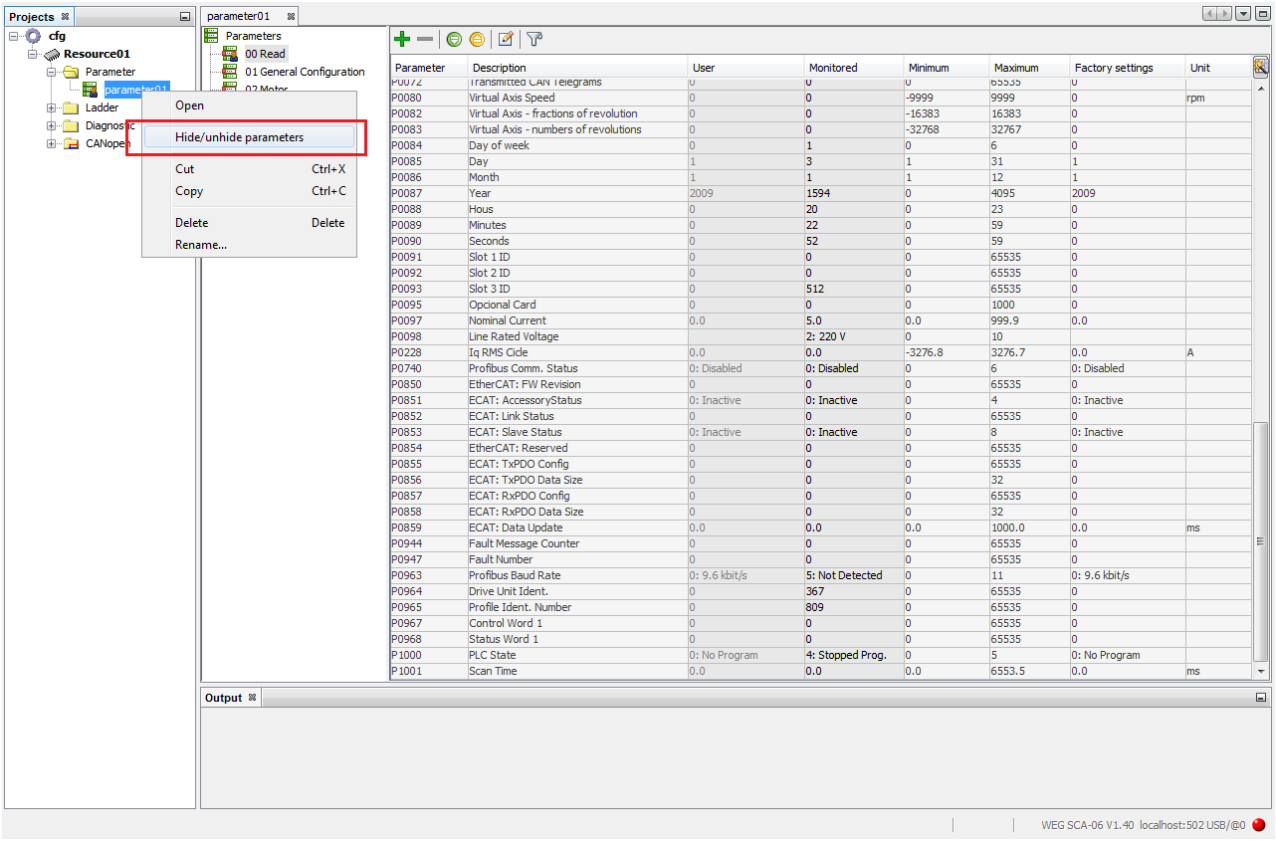
Para...	Description	User	Monitored	Minimum	Maxi...	Factory settings	Unit
P0002	Motor Speed	0	0	-9999	9999	0	rpm
P0003	Motor Current	0.0	**0.0**	-999.9	999.9	0.0	A
P0004	DC Link Voltage (Ud)	0	307	0	999	0	V
P0006	Drive Status	0: Ready	0: Ready	0	5	0: Ready	
P0008	DI3 to DI1 Status	0	0	0	7	0	
P0009	DI101 to DI106 Status	0	0	0	63	0	
P0010	DI107 to DI112 Status	0	0	0	63	0	
P0011	DI201 to DI206 Status	0	0	0	63	0	
P0012	DI207 to DI212 Status	0	0	0	63	0	
P0013	DI301 to DI306 Status	0	0	0	63	0	
P0014	DI307 to DI312 Status	0	0	0	63	0	
P0015	DO1 Status	0	0	0	1	0	
P0016	DO101 to DO106 Status	0	0	0	63	0	
P0017	DO201 to DO206 Status	0	0	0	63	0	
P0018	DO301 to DO306 Status	0	0	0	63	0	
P0021	Internal Air Temperature	0	41	0	1000	0	°C
P0022	Dissipator Temperature	0	33	0	1000	0	°C
P0023	Software Version	9.99	1.40	0.00	655.35	9.99	
P0024	Bootloader Version	0.00	20.03	0.00	655.35	0.00	
P0025	FPGA Project Version	0.00	0.03	0.00	655.35	0.00	
P0030	Present Alarm	0	0	0	2000	0	
P0031	Last Alarm	0	826	0	2000	0	
P0032	Last alarm Day.Month	0.00	3.01	0.00	31.12	0.00	
P0033	Last alarm year	0	1586	0	4096	0	
P0034	Last alarm Hour.Min	0.00	17.01	0.00	23.59	0.00	
P0035	Present Fault	0	0	0	2000	0	
P0036	Last Fault	0	2	0	2000	0	
P0037	Last Fault Day.Month	0.00	3.01	0.00	31.12	0.00	
P0038	Last fault year	0	1594	0	4096	0	
P0039	Last fault Hour.Min	0.00	17.13	0.00	23.59	0.00	
P0040	Second fault	0	2	0	2000	0	
P0041	Second fault Day.Month	0.00	3.01	0.00	31.12	0.00	
P0042	Second fault year	0	1594	0	4096	0	
P0043	Second fault Hour.Min	0.00	17.12	0.00	23.59	0.00	
P0044	Third fault	0	32	0	2000	0	
P0045	Third fault Day.Month	0.00	1.01	0.00	31.12	0.00	
P0046	Third fault year	0	1577	0	4096	0	
P0047	Third fault Hour.Min	0.00		0.00	23.59	0.00	

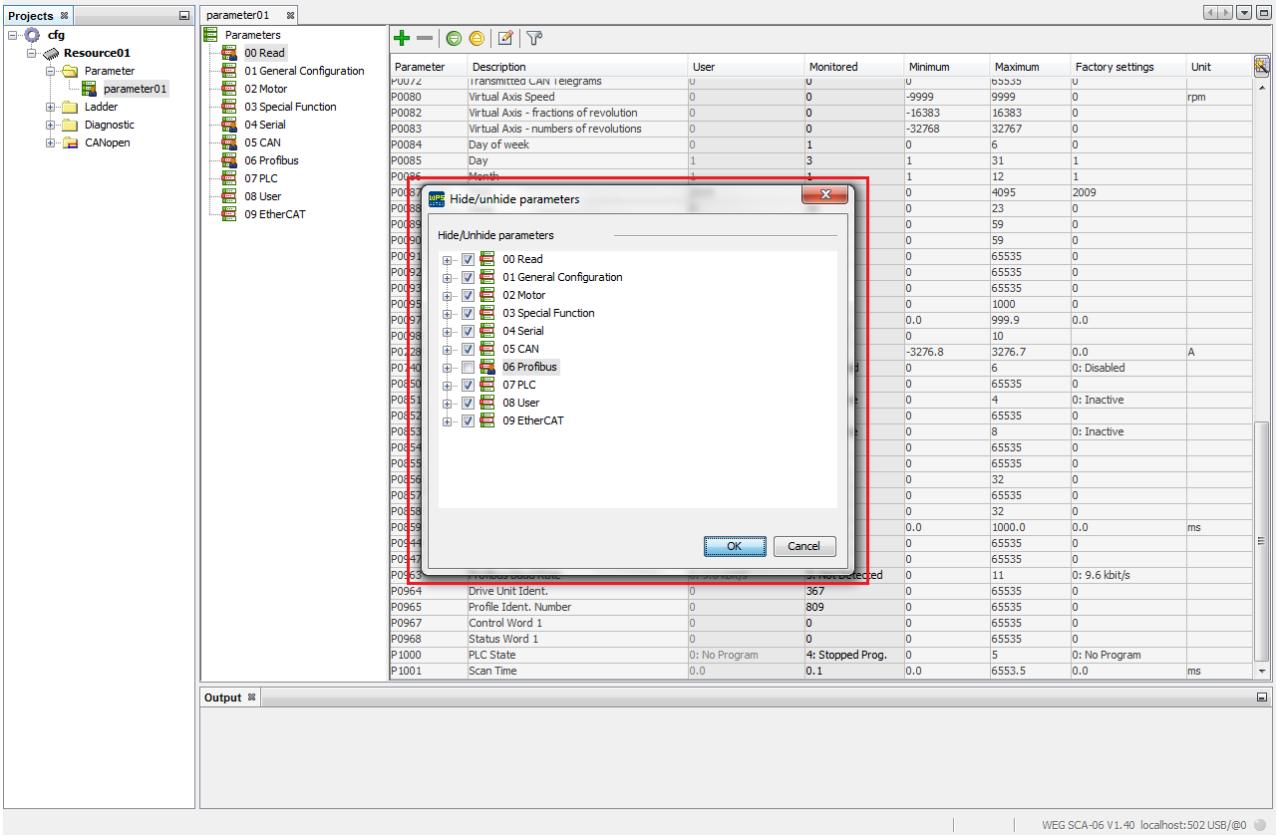

parameter01
Parameters
00 Read
01 General Configuration
02 Motor
04 Serial
05 CAN
06 Profibus
07 PLC
08 User
09 EtherCAT
Select groups
Select to unhide:
03 Special Function
OK
Cancel
Output
WEG SCA-06 V1.40 localhost:502 USB/@0



Para...	Description	User	M...	Minimum	Maxim...	Factory settings	Unit
P0002	Motor Speed	0		-9999	9999	0	rpm
P0003	Motor Current	0.0		-999.9	999.9	0.0	A
P0004	DC Link Voltage (Ud)	0		0	999	0	V
P0006	Drive Status	0: Ready		0	5	0: Ready	
P0008	DI3 to DI1 Status	0		0	7	0	
P0009	DI101 to DI106 Status	0		0	63	0	
P0010	DI107 to DI112 Status	0		0	63	0	
P0011	DI201 to DI206 Status	0		0	63	0	
P0012	DI207 to DI212 Status	0		0	63	0	
P0013	DI301 to DI306 Status	0		0	63	0	
P0014	DI307 to DI312 Status	0		0	63	0	
P0015	DO1 Status	0		0	1	0	
P0016	DO101 to DO106 Status	0		0	63	0	
P0017	DO201 to DO206 Status	0		0	63	0	
P0018	DO301 to DO306 Status	0		0	63	0	
P0021	Internal Air Temperature	0		0	1000	0	°C
P0022	Dissipator Temperature	0		0	1000	0	°C
P0023	Software Version	9.99		0.00	655.35	9.99	
P0024	Bootloader Version	0.00		0.00	655.35	0.00	
P0025	FPGA Project Version	0.00		0.00	655.35	0.00	
P0030	Present Alarm	0		0	2000	0	
P0031	Last Alarm	0		0	2000	0	
P0032	Last alarm Day.Month	0.00		0.00	31.12	0.00	
P0033	Last alarm year	0		0	4096	0	
P0034	Last alarm Hour.Min	0.00		0.00	23.59	0.00	
P0035	Present Fault	0		0	2000	0	
P0036	Last Fault	0		0	2000	0	
P0037	Last Fault Day.Month	0.00		0.00	31.12	0.00	
P0038	Last fault year	0		0	4096	0	
P0039	Last fault Hour.Min	0.00		0.00	23.59	0.00	
P0040	Second fault	0		0	2000	0	
P0041	Second fault Day.Month	0.00		0.00	31.12	0.00	
P0042	Second fault year	0		0	4096	0	
P0043	Second fault Hour.Min	0.00		0.00	23.59	0.00	
P0044	Third fault	0		0	2000	0	
P0045	Third fault Day.Month	0.00		0.00	31.12	0.00	
P0046	Third fault year	0		0	4096	0	

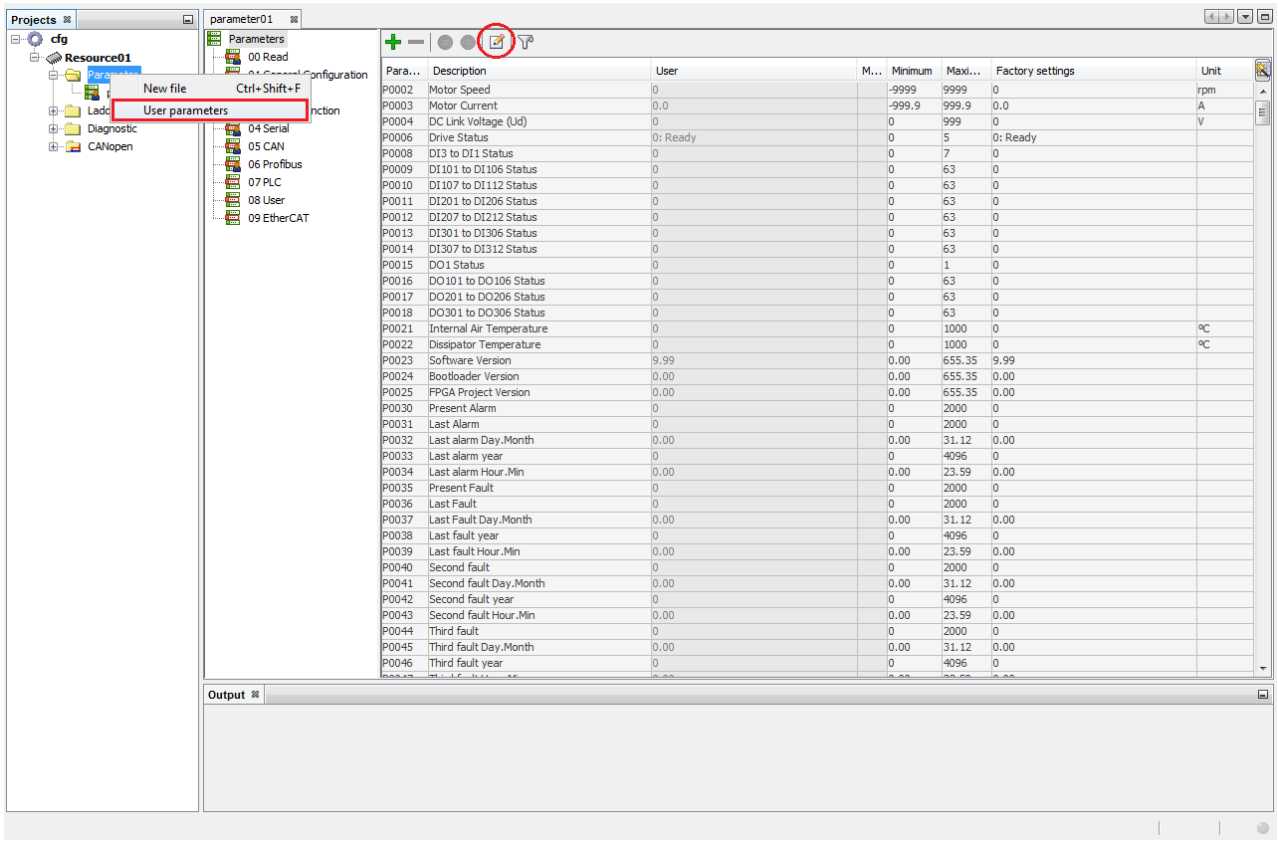
5. **Hide and Show Parameters and Groups of Parameters**. By means of this option, you have full control of the parameters and groups of parameters. It is possible to hide and unhide individual parameters, multiple parameters, individual groups and multiple groups in the same action.


Projects
cfg
Resource01
Parameter
parameter01
Ladder
Diagnostic
CANopen
Open
Hide/unhide parameters
Cut Ctrl+X
Copy Ctrl+C
Delete Delete
Rename...
parameter01
Parameters
00 Read
01 General Configuration
Parameter
Description
User
Monitored
Minimum
Maximum
Factory settings
Unit
P0080 Virtual Axis Speed
P0082 Virtual Axis - fractions of revolution
P0083 Virtual Axis - numbers of revolutions
P0084 Day of week
P0085 Day
P0086 Month
P0087 Year
P0088 Hous
P0089 Minutes
P0090 Seconds
P0091 Slot 1 ID
P0092 Slot 2 ID
P0093 Slot 3 ID
P0095 Opcional Card
P0097 Nominal Current
P0098 Line Rated Voltage
P0228 Iq RMS Cicle
P0740 Profibus Comm. Status
P0850 EtherCAT: FW Revision
P0851 ECAT: AccessoryStatus
P0852 ECAT: Link Status
P0853 ECAT: Slave Status
P0854 EtherCAT: Reserved
P0855 ECAT: TxPDO Config
P0856 ECAT: TxPDO Data Size
P0857 ECAT: RxPDO Config
P0858 ECAT: RxPDO Data Size
P0859 ECAT: Data Update
P0944 Fault Message Counter
P0947 Fault Number
P0963 Profibus Baud Rate
P0964 Drive Unit Ident.
P0965 Profile Ident. Number
P0967 Control Word 1
P0968 Status Word 1
P1000 PLC State
P1001 Scan Time
Output
WEG SCA-06 V1.40 localhost:502 USB/@0


Projects
cfg
Resource01
Parameter
parameter01
Ladder
Diagnostic
CANopen
parameter01
Parameters
00 Read
01 General Configuration
02 Motor
03 Special Function
04 Serial
05 CAN
06 Profibus
07 PLC
08 User
09 EtherCAT
Hide/unhide parameters
Hide/Unhide parameters
OK
Cancel
Output
WEG SCA-06 V1.40 localhost:502 USB/@0

11.5.2.5 User Parameters

In order to open the configuration screen of the user parameters, just click on the **User Parameters** option on the Parameter node of the project tree or click on the icon indicated on the tool bar of the parameter file.



Para...	Description	User	M...	Minimum	Maxi...	Factory settings	Unit
P0002	Motor Speed	0		-9999	9999	0	rpm
P0003	Motor Current	0.0		-999.9	999.9	0.0	A
P0004	DC Link Voltage (Ud)	0		0	999	0	V
P0006	Drive Status	0: Ready		0	5	0: Ready	
P0008	DI3 to DI1 Status	0		0	7	0	
P0009	DI101 to DI106 Status	0		0	63	0	
P0010	DI107 to DI112 Status	0		0	63	0	
P0011	DI201 to DI206 Status	0		0	63	0	
P0012	DI207 to DI212 Status	0		0	63	0	
P0013	DI301 to DI306 Status	0		0	63	0	
P0014	DI307 to DI312 Status	0		0	63	0	
P0015	DO1 Status	0		0	1	0	
P0016	DO101 to DO106 Status	0		0	63	0	
P0017	DO201 to DO206 Status	0		0	63	0	
P0018	DO301 to DO306 Status	0		0	63	0	
P0021	Internal Air Temperature	0		0	1000	0	°C
P0022	Dissipator Temperature	0		0	1000	0	°C
P0023	Software Version	9.99		0.00	655.35	9.99	
P0024	Bootloader Version	0.00		0.00	655.35	0.00	
P0025	FPGA Project Version	0.00		0.00	655.35	0.00	
P0030	Present Alarm	0		0	2000	0	
P0031	Last Alarm	0		0	2000	0	
P0032	Last alarm Day.Month	0.00		0.00	31.12	0.00	
P0033	Last alarm year	0		0	4096	0	
P0034	Last alarm Hour.Min	0.00		0.00	23.59	0.00	
P0035	Present Fault	0		0	2000	0	
P0036	Last Fault	0		0	2000	0	
P0037	Last Fault Day.Month	0.00		0.00	31.12	0.00	
P0038	Last fault year	0		0	4096	0	
P0039	Last fault Hour.Min	0.00		0.00	23.59	0.00	
P0040	Second fault	0		0	2000	0	
P0041	Second fault Day.Month	0.00		0.00	31.12	0.00	
P0042	Second fault year	0		0	4096	0	
P0043	Second fault Hour.Min	0.00		0.00	23.59	0.00	
P0044	Third fault	0		0	2000	0	
P0045	Third fault Day.Month	0.00		0.00	31.12	0.00	
P0046	Third fault year	0		0	4096	0	

Configuration Table.

On the user parameter configuration table, it is possible to define several attributes to the parameters, such as description, minimum and maximum values, unit, digits, data type, etc.

NOTE!
These settings will be automatically displayed in the parameter table. However, to be sent to the device, you need to *download* the resource.

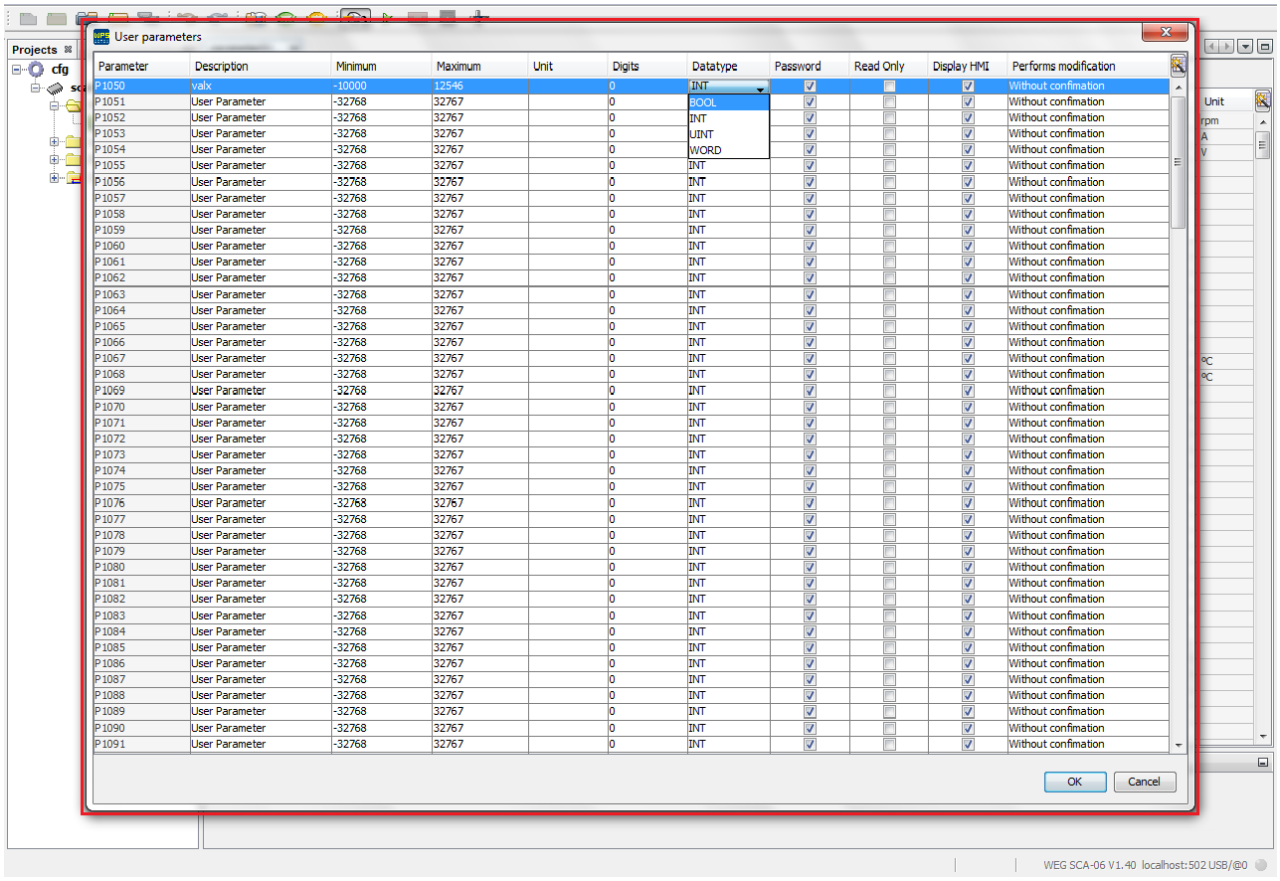


Table fields:

- Parameter: User parameter identification.
- Description: Description of the user parameter in the parameter table. On devices that have text-based HMIs, the description is sent to the machine and displayed on the HMI.
- Minimum: Minimum input value for parameter.
- Maximum: Maximum input value for parameter.
- Unit: Unit displayed on the device's HMI.
- Default: Value loaded when restore factory default is selected.
- Retentive: Retain value after rebooting devices.
- Hexadecimal: Displays the value in hexadecimal.
- Digits: Number of decimal digits for displaying value.
- Datatype: Parameter datatype used by the ladder application.
- Password: Enables password request by changing parameter value.

- Read only: It does not allow the writing of values in the parameter by the communication network or the HMI. Writing is done only by the ladder application.

- Display HMI: Displays the parameter in the HMI.

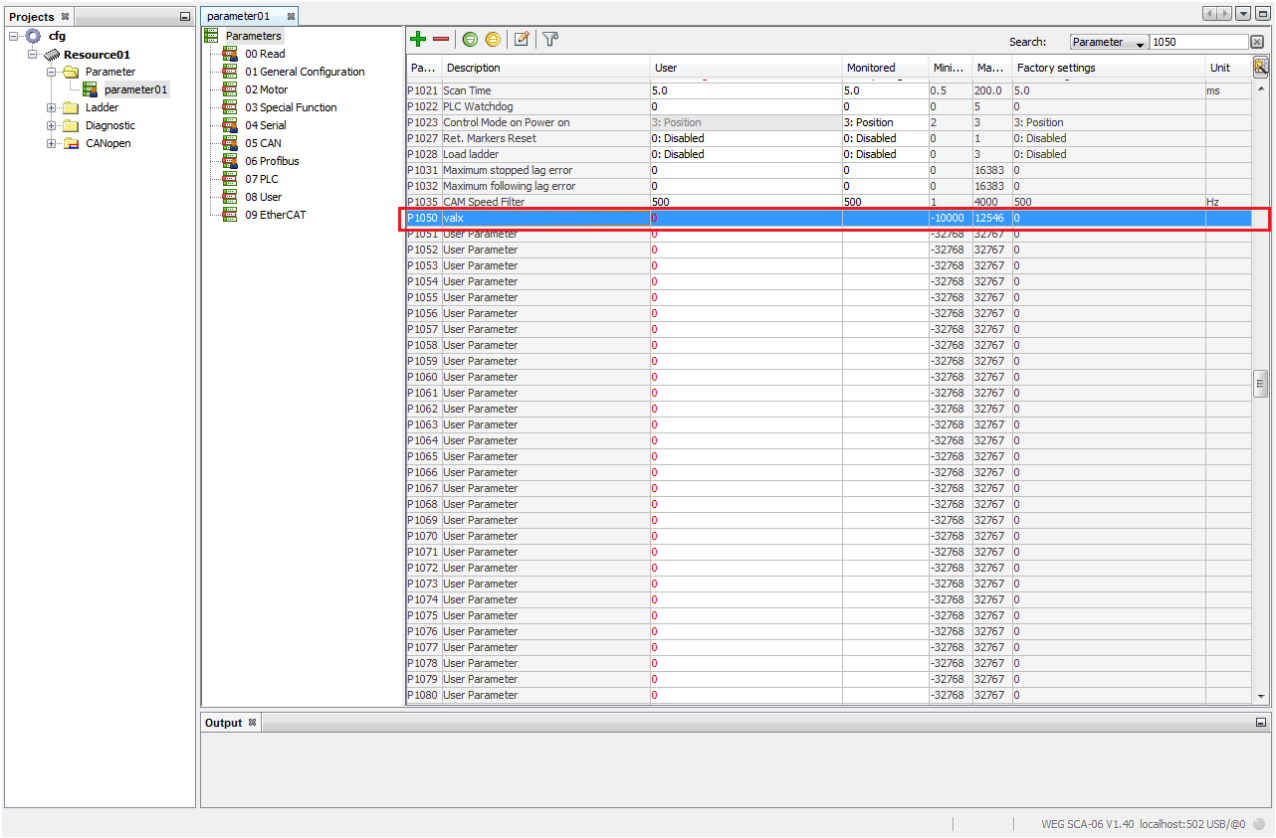
- Performs modification: Confirmation options when changing the parameter:
o No confirmation: Does not prompt for confirmation when changing parameter.
o With confirmation and engine stopped: Request confirmation and allow change only with engine stopped.
o With confirmation: Prompt for confirmation when changing parameter.

- Stopped motor: Perform change only with motor stopped.

- Help: On devices that have text-based HMI, you can edit a help text for the parameter.

View the user parameter

In the parameter table, the user parameters will be shown as they are configured on the configuration screen.



Pa...	Description	User	Monitored	Mini...	Ma...	Factory settings	Unit
P1021	Scan Time	5.0	5.0	0.5	200.0	5.0	ms
P1022	PLC Watchdog	0	0	0	5	0	
P1023	Control Mode on Power on	3: Position	3: Position	2	3	3: Position	
P1027	Ret. Markers Reset	0: Disabled	0: Disabled	0	1	0: Disabled	
P1028	Load ladder	0: Disabled	0: Disabled	0	3	0: Disabled	
P1031	Maximum stopped lag error	0	0	0	16383	0	
P1032	Maximum following lag error	0	0	0	16383	0	
P1035	CAM Speed Filter	500	500	1	4000	500	Hz
P1050	valx	0		-10000	12546	0	
P1051	User Parameter	0		-32768	32767	0	
P1052	User Parameter	0		-32768	32767	0	
P1053	User Parameter	0		-32768	32767	0	
P1054	User Parameter	0		-32768	32767	0	
P1055	User Parameter	0		-32768	32767	0	
P1056	User Parameter	0		-32768	32767	0	
P1057	User Parameter	0		-32768	32767	0	
P1058	User Parameter	0		-32768	32767	0	
P1059	User Parameter	0		-32768	32767	0	
P1060	User Parameter	0		-32768	32767	0	
P1061	User Parameter	0		-32768	32767	0	
P1062	User Parameter	0		-32768	32767	0	
P1063	User Parameter	0		-32768	32767	0	
P1064	User Parameter	0		-32768	32767	0	
P1065	User Parameter	0		-32768	32767	0	
P1066	User Parameter	0		-32768	32767	0	
P1067	User Parameter	0		-32768	32767	0	
P1068	User Parameter	0		-32768	32767	0	
P1069	User Parameter	0		-32768	32767	0	
P1070	User Parameter	0		-32768	32767	0	
P1071	User Parameter	0		-32768	32767	0	
P1072	User Parameter	0		-32768	32767	0	
P1073	User Parameter	0		-32768	32767	0	
P1074	User Parameter	0		-32768	32767	0	
P1075	User Parameter	0		-32768	32767	0	
P1076	User Parameter	0		-32768	32767	0	
P1077	User Parameter	0		-32768	32767	0	
P1078	User Parameter	0		-32768	32767	0	
P1079	User Parameter	0		-32768	32767	0	
P1080	User Parameter	0		-32768	32767	0	

11.5.3 Diagnostic

11.5.3.1 Trace

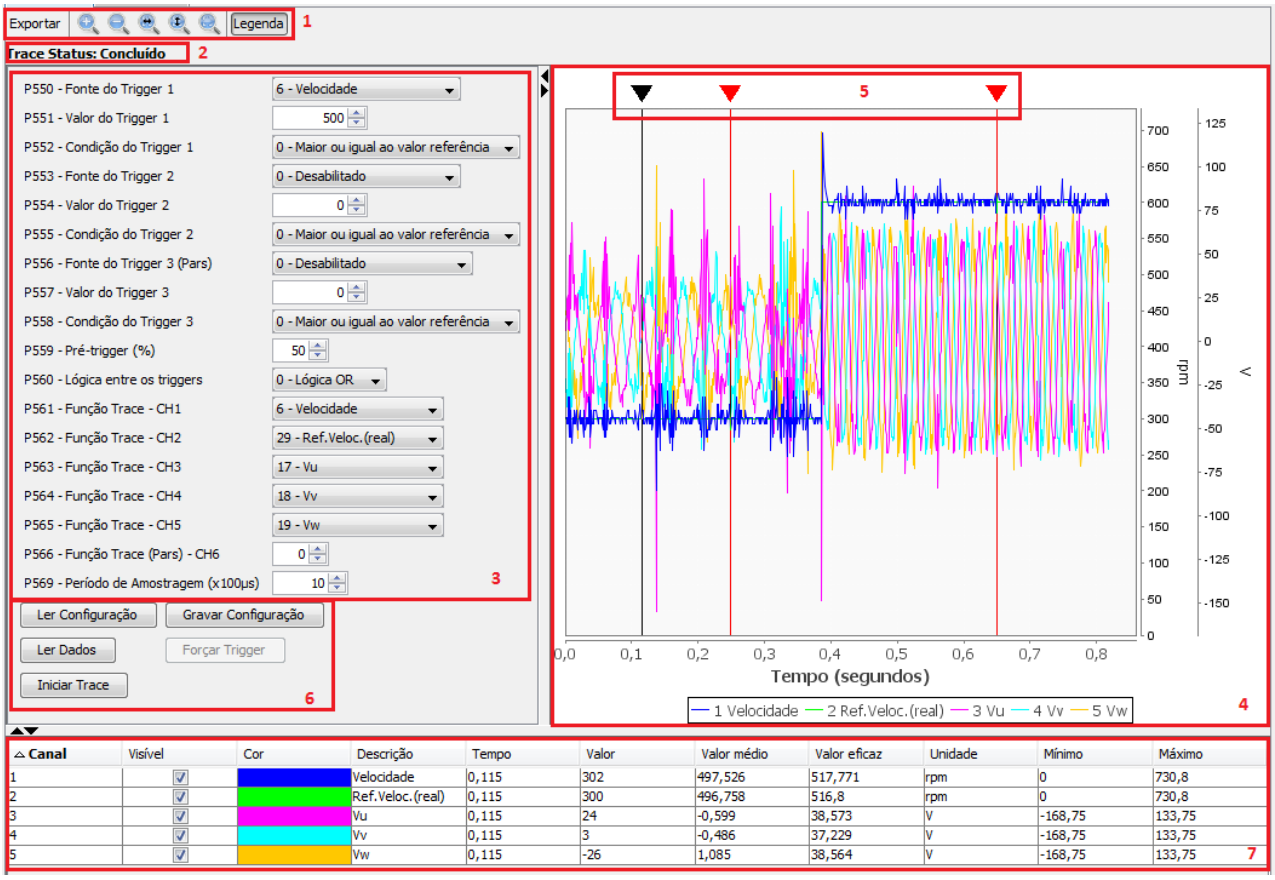
11.5.3.1.1 Overview

The trace function is used to register variables* of interest of the device (such as current, voltage, speed, etc.) when a certain event occurs in the system. Since it triggers the storage of the variables, in the system this

event is called trigger, and the user can define up to three trigger conditions and the logic to be used in them (AND or OR logic).
The stored variables can be seen as graphics by using the WPS running on a PC connected via USB or via serial to the device.

NOTE: Up to 6 (six) channels using SCA06; Up to 4 (four) channels using CFW-11.

Below is an overview of the configuration screen of the trace function (example using SCA06).



1. **Graphic Zoom**. This bar contains the options to control the graphic, such as export to image file, zoom in, zoom out, set width, set height, se all and show or not show the graphic lettering.
2. **Trace Status**. This item shows the present status of the trace function: not started, trigger occurred and concluded.
3. **Parameters**. In this part are all the parameters that can be configured in the trace routine, such as triggers, conditions, channels to be monitored and sampling period.
4. **Graphic**. In this area is the graphic after the conclusion of trace. In the lower part is the time line and on the right are the values separated by unit of measurement.
5. **Markers**. The markers are within the graphic area. After the graphic is set, just click on the black marker to create red markers (fixed). It is possible to add two fixed markers. Those fixed markers are used to calculate the average and effective values between the two points.
6. **Trace command**. Below is the description of the command functions:
   6.1. **Read configuration**: It reads the trace configuration parameters and updates the parameters on the screen (item 3).
   6.2. **Save configuration**: It sends the trace configuration parameters (item 3) to the equipment.

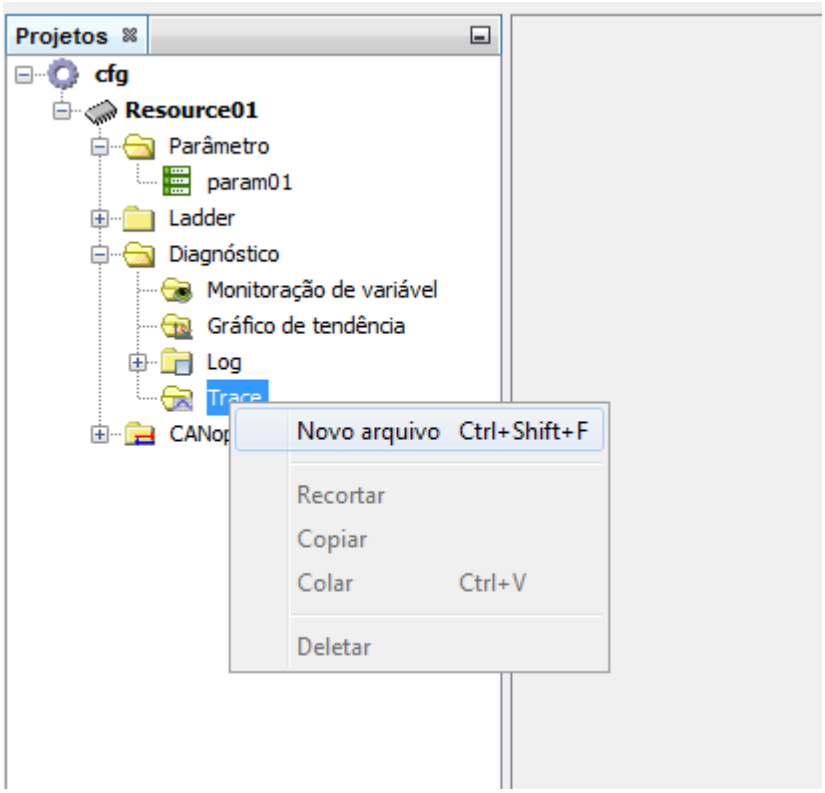
6.3. **Read Data**: Command used only when the trace status is concluded, that is, there is already a concluded trace on the equipment, and you just wish to download the data without starting a new trace.
6.4. **Force Trigger**: Forces the trigger regardless the conditions.
6.5. **Start Trace**: It starts the trace function.

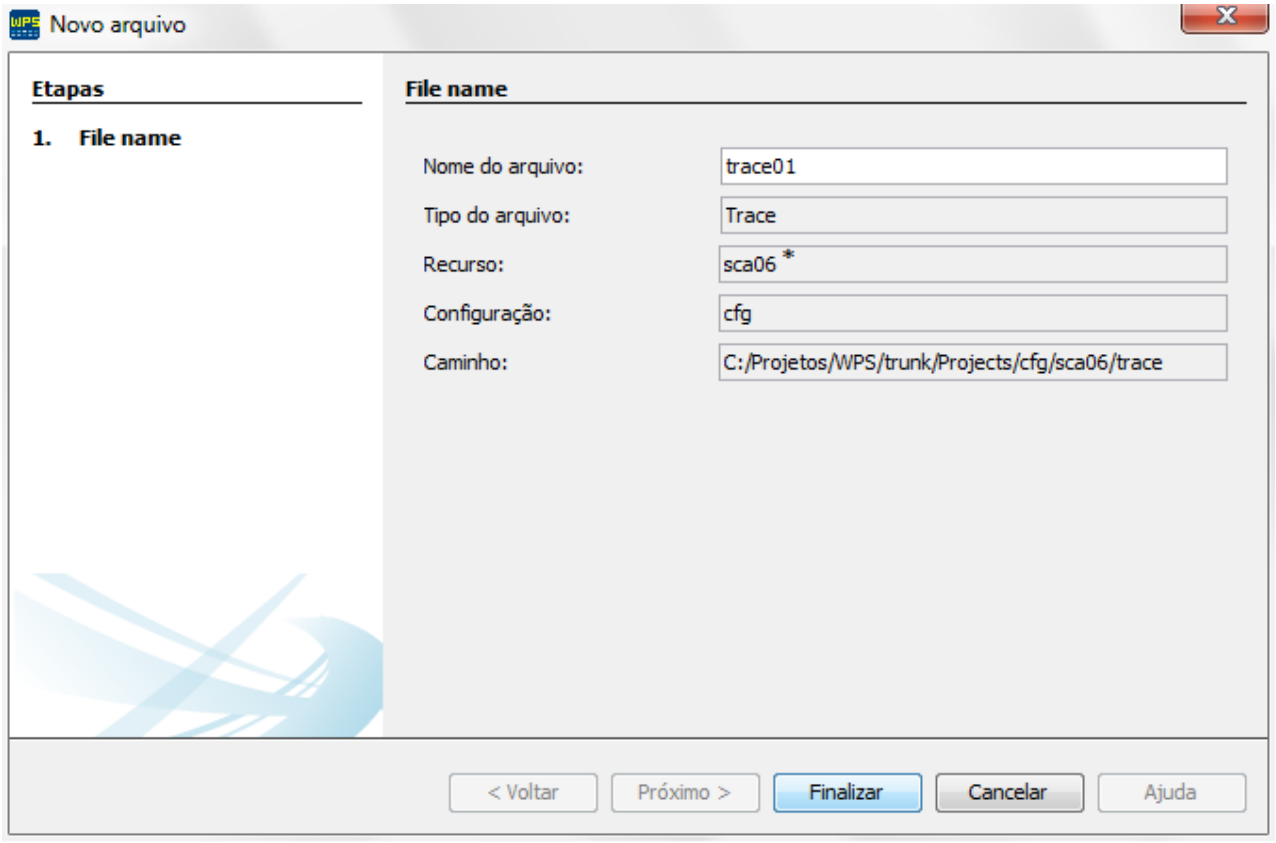
7. **Channel Table**. This table shows the data of the chosen channels, besides the possibility to hide channels (Visible), change the channel color (Color) and set the graphic limits per unit of measurement (Maximum).

11.5.3.1.2 Configuration

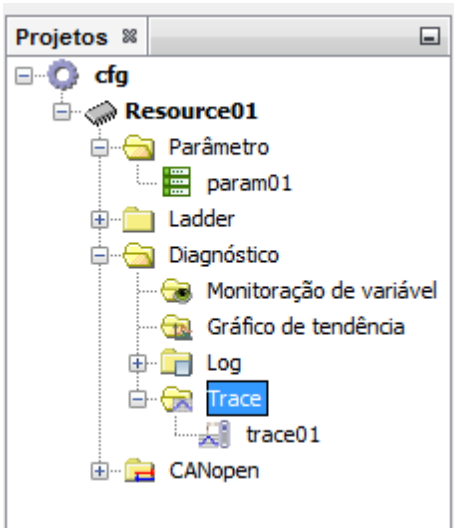
Below is a list of the necessary steps to create a trace configuration:

1. Creation of a new trace file.

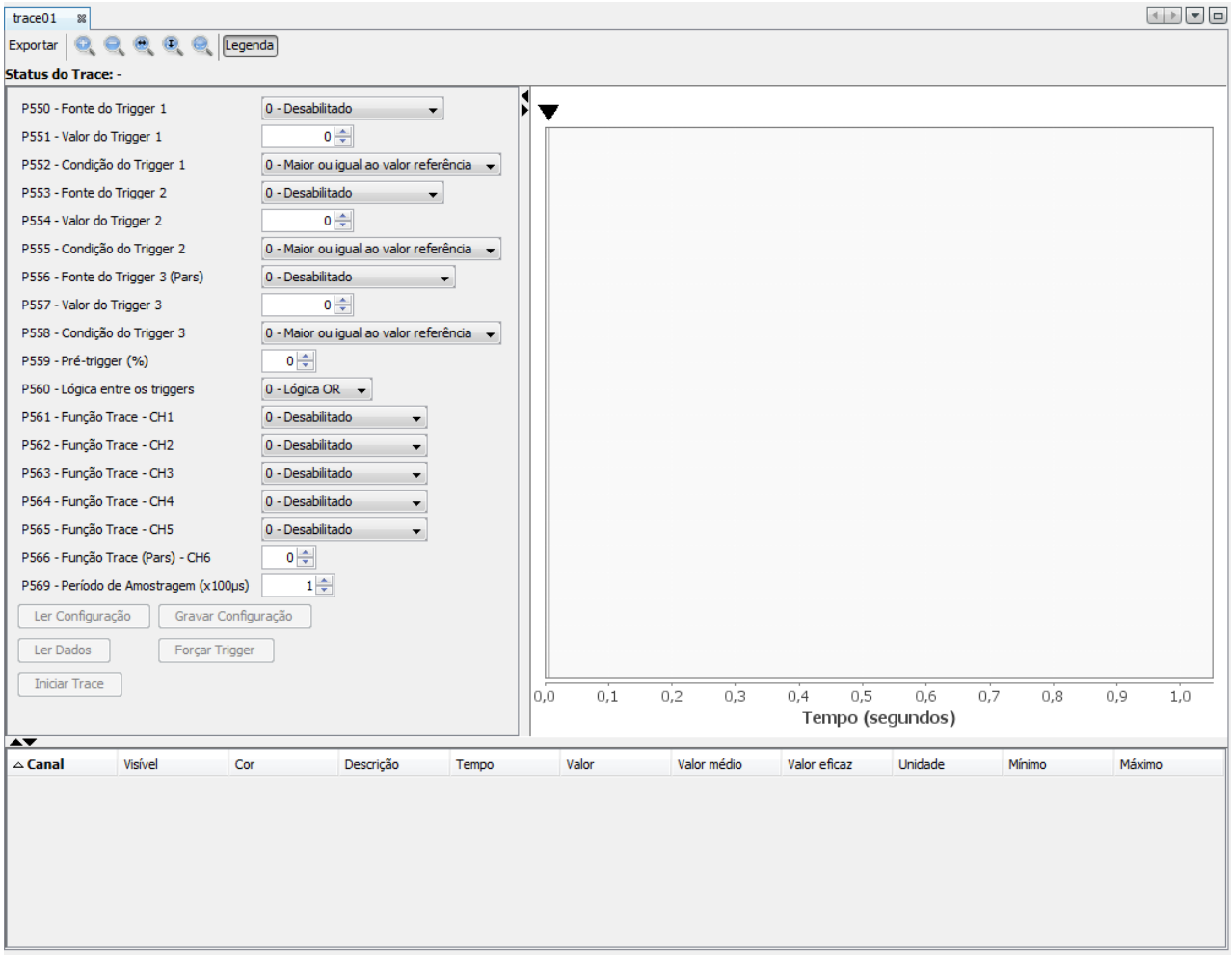




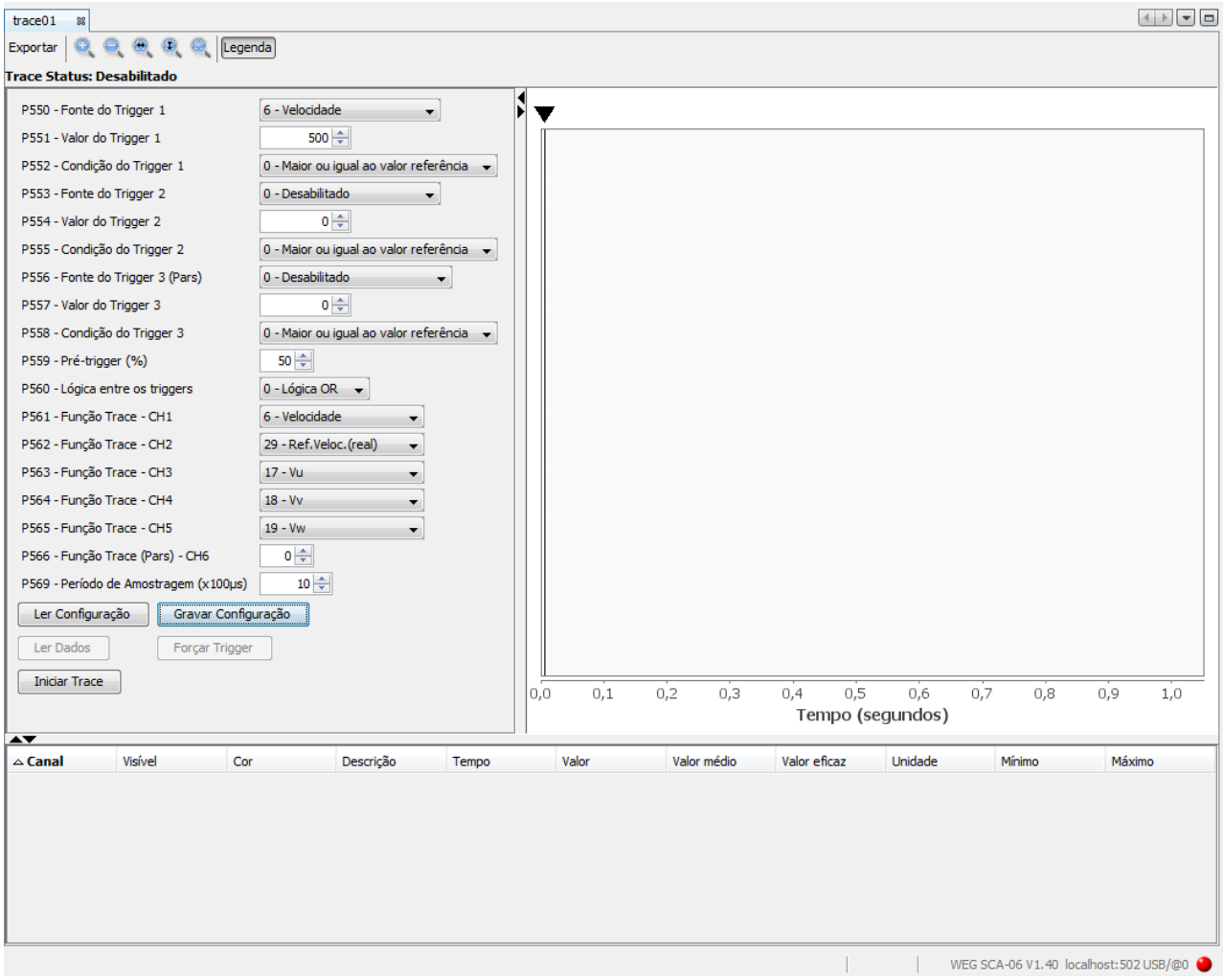
* **Resource:** Resource01, SCA06, CFW300, etc.



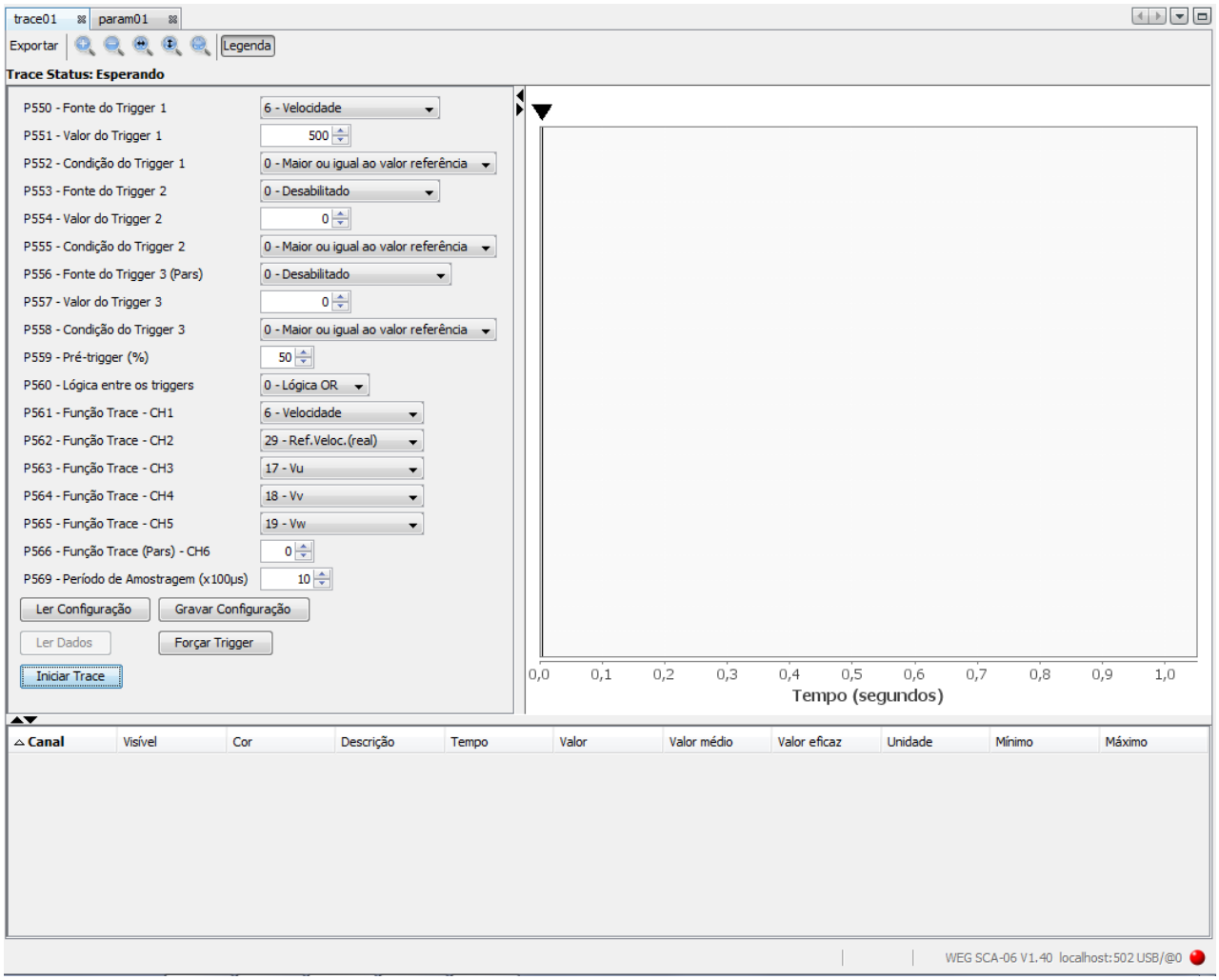
2. After the creation of the trace file, it is necessary to set the desired configurations in the part of parameters.



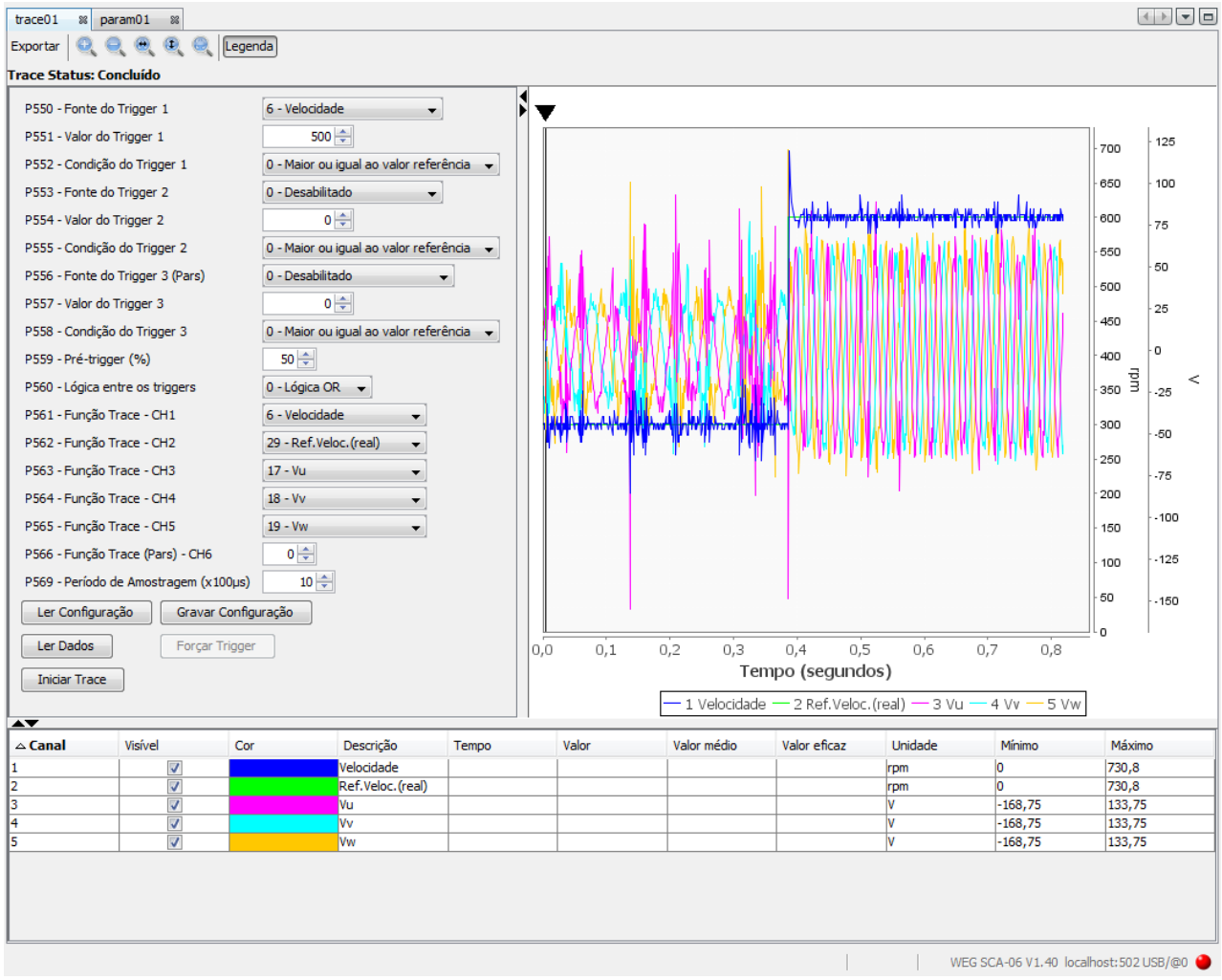
3. After making the desired configurations, just click on save configuration to send them to the equipment. Notice that it is necessary to be connected to the equipment with the option **Connect Device** of the WPS.



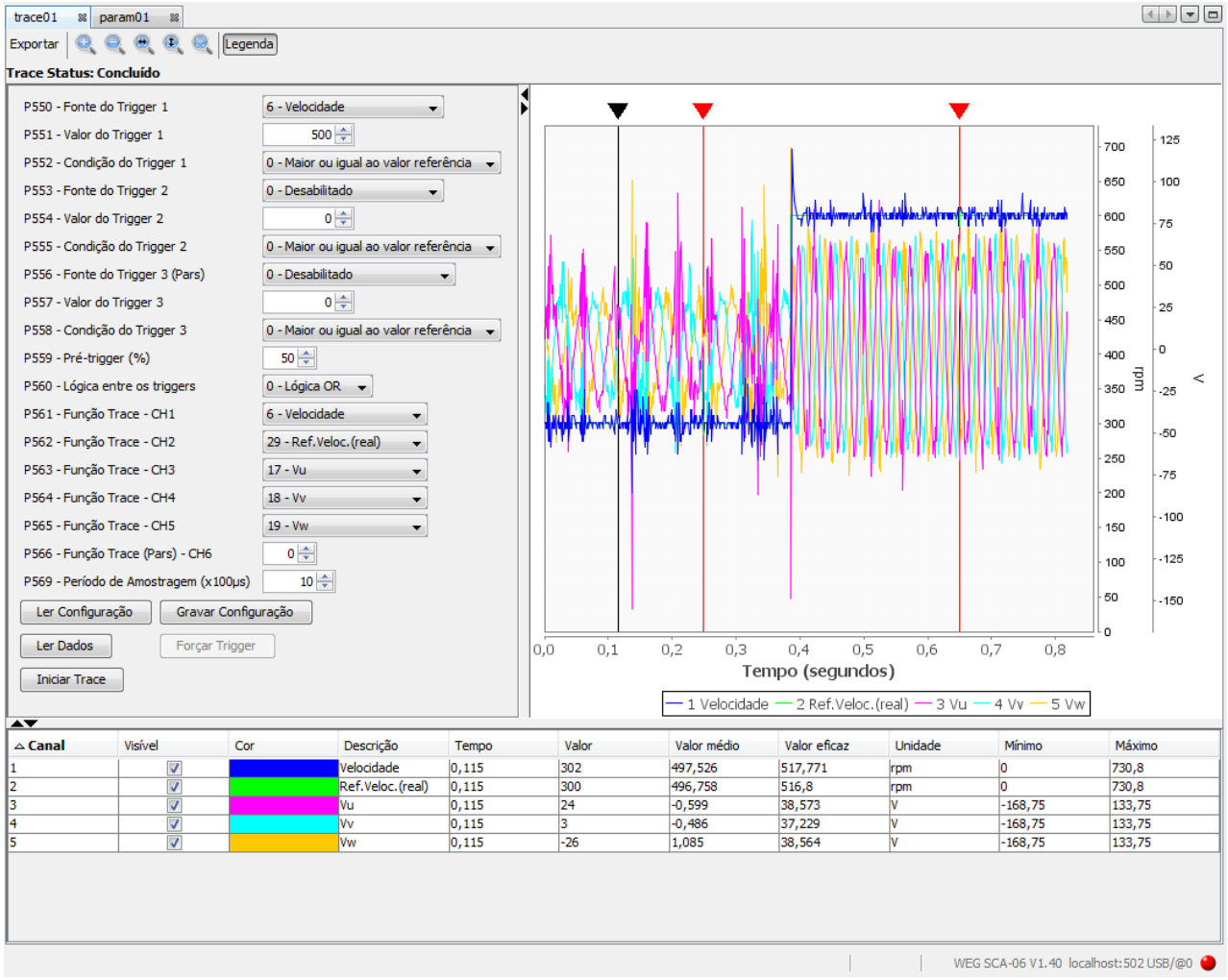
4. After the configurations are saved, just click on **Start Trace**. Notice that the status of the trace function changed to **Waiting**, that is, the tool is now waiting for the trigger execution to set the graphic and show the trace values.



5. After trigger occurs, the graphic and the values will be shown in the table and the trace function status will be **Concluded**.



6. If you wish, you can click on the black cursor of the graphic and add fixed cursors so that the calculation of the average and effective values will be performed for the channels in the defined ranges.



Canal	Visível	Cor	Descrição	Tempo	Valor	Valor médio	Valor eficaz	Unidade	Mínimo	Máximo
1	☑		Velocidade	0,115	302	497,526	517,771	rpm	0	730,8
2	☑		Ref.Veloc.(real)	0,115	300	496,758	516,8	rpm	0	730,8
3	☑		Vu	0,115	24	-0,599	38,573	V	-168,75	133,75
4	☑		Vv	0,115	3	-0,486	37,229	V	-168,75	133,75
5	☑		Vw	0,115	-26	1,085	38,564	V	-168,75	133,75

11.6 LDW900

11.6.1 Description

The LDW900 is a high performance product directed to the lifts market for speed control, torque and PM motor position of sinusoidal alternating current.

Its main feature is the energy regeneration option to the power grid at the time of engine braking, reducing the consumption of electricity.

Another feature of the product is the high performance and precision control of motor shaft movement due to operation in closed loop through position feedback provided by an encoder.

It has operator interface with six-digit LED's display for control, adjustment and display of all parameters, connected to expansion accessories, PLC function, positioning blocks, free programming software and CANopen communication included in the standard product.

Refer to the user manual and LDW900 programming manual for more information about the product.

11.6.2 System Markers

The following variables contained in the **GLOBAL_SYSTEM** group of the variables table, have the fixed tag. The tag of system markers were divided into groups and subgroups, where:

Groups:
- LDW: reading and writing variables of the LDW900 lift drive;
- CO: reading and writing variables of the CANopen network.

Subgroups:
- STS: reading variable (status);
- CMD: writing variable (command).

Reading System Markers (Status)

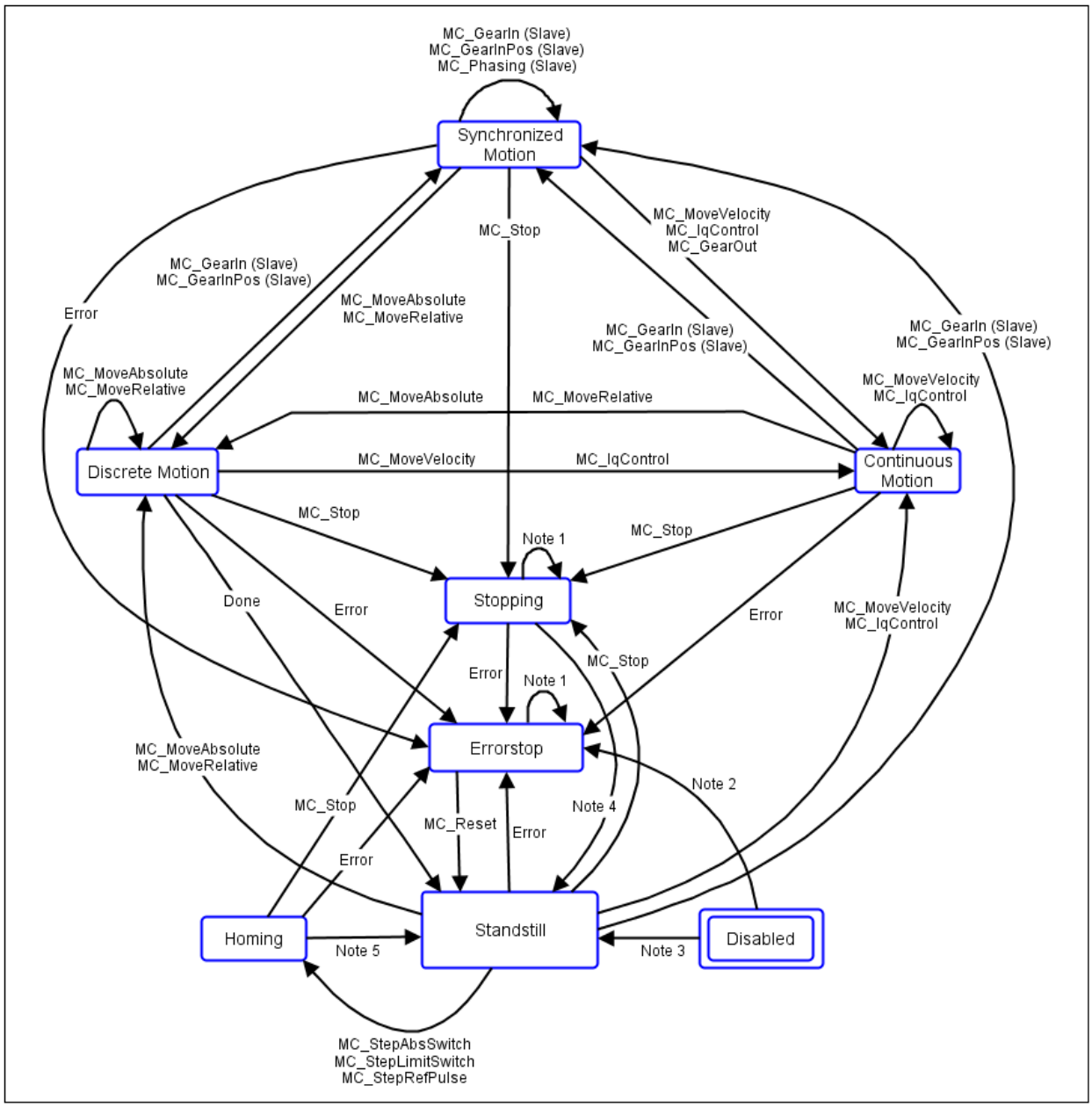
Address	Bit	Modbus	Tag	Description
Ladder				
%SB6000	0	0	FREQ_2HZ	Oscillator with frequency of 2 Hz
%SB6000	1	1	PULSE_1SCAN	Pulse during the first scan cycle
%SB6000	2	2	FALSE	Always in 0
%SB6000	3	3	TRUE	Always in 1
%SW6002	--	3001	ELAPSED_SCAN_CYCLES	Elapsed scan cycles
Real Axis				
%SW6004	--	3002	LDW_STS_REAL_AXIS_STATUS	Real axis status (see note)
%SD6008	--	3004	LDW_STS_REAL_AXIS_VELOCITY	Real axis velocity
%SL6024	--	3012	LDW_STS_REAL_AXIS_POSITION	Real axis position
Virtual Axis				
%SW6006	--	3003	LDW_STS_VIRTUAL_AXIS_STATUS	Virtual axis status (see note)
%SD6012	--	3006	LDW_STS_VIRTUAL_AXIS_VELOCITY	Virtual axis velocity
%SL6032	--	3016	LDW_STS_VIRTUAL_AXIS_POSITION	Virtual axis position
Current				
%SD6016	--	3008	LDW_STS_MOTOR_CURRENT	Motor current
Position in the DI´s transition				
%SD6040	--	3020	LDW_STS_DI1_POSITION_STORED	Position stored in the DI1 transition
%SD6048	--	3024	LDW_STS_DI2_POSITION_STORED	Position stored in the DI2 transition
%SD6056	--	3028	LDW_STS_DI3_POSITION_STORED	Position stored in the DI3 transition
Counters				
%SD6064	--	3032	LDW_STS_BUILT_IN_COUNTER	Built-in counter value
%SD6068	--	3034	LDW_STS_BUILT_IN_COUNTER_DI3	Built-in counter stored in the DI3 transition
%SD6072	--	3036	LDW_STS_ENC1_COUNTER	Encoder 1 counter value
%SD6076	--	3038	LDW_STS_ENC2_COUNTER	Encoder 2 counter value
%SD6080	--	3040	LDW_STS_ENC_COUNTER_Z1	Encoder counter stored in the Z1 transition as defined in P00511
%SD6084	--	3042	LDW_STS_ENC_COUNTER_Z2	Encoder counter stored in the Z2 transition as defined in P00521
CANopen				
%SB6100	0	800	CO_STS_MASTER_CONTACTED	The CANopen master contacted all the slaves
%SB6100	1	801	CO_STS_MASTER_CONFIG_OK	The CANopen master downloaded the configurations of the slaves
%SB6100	2	802	CO_STS_MASTER_ERROR_CTRL_OK	Error control protocol (node guarding/heartbeat) initiated with the slaves
%SB6100	3	803	CO_STS_MASTER_INIT_FINISHED	The CANopen master initialized all the slaves
%SB6100	4	804	CO_STS_MASTER_INIT_ERROR	A slave presented an initialization error
%SB6100	5	805	CO_STS_MASTER_ERROR_CTRL	The CANopen master detected a fault in a slave through the error detection protocol
%SB6100	6	806	CO_STS_MASTER_EMCY	A slave reported EMCY
%SB6101	0	808	CO_STS_MASTER_NMT_TOGGLE	NMT command toggle bit feedback
%SB6101	5	813	CO_STS_MASTER_BUS_OFF	The CANopen master is in bus off

Writing / Reading System Markers (Command)

Address	Bit	Modbus	Tag	Description
CANopen				
%CB6000	--	3000	CO_CMD_NMT_COMMAND	NMT command transmission by the CANopen master: command code
%CB6001	0	8	CO_CMD_NMT_TOGGLE	NMT command transmission by the CANopen master: toggle bit
%CB6001	7	15	CO_CMD_DISABLE	Disables the CANopen communication
%CB6002	--	3001	CO_CMD_NMT_SLAVE_ADDR	NMT command transmission by the CANopen master: slave address

NOTE!
Below description of the real axis and virtual status:

0. Disabled.
1. Errorstop.
2. Standstill.
3. Stopping.
4. Homing.
5. Continuous Motion.
6. Discrete Motion.
7. Synchronized Motion.

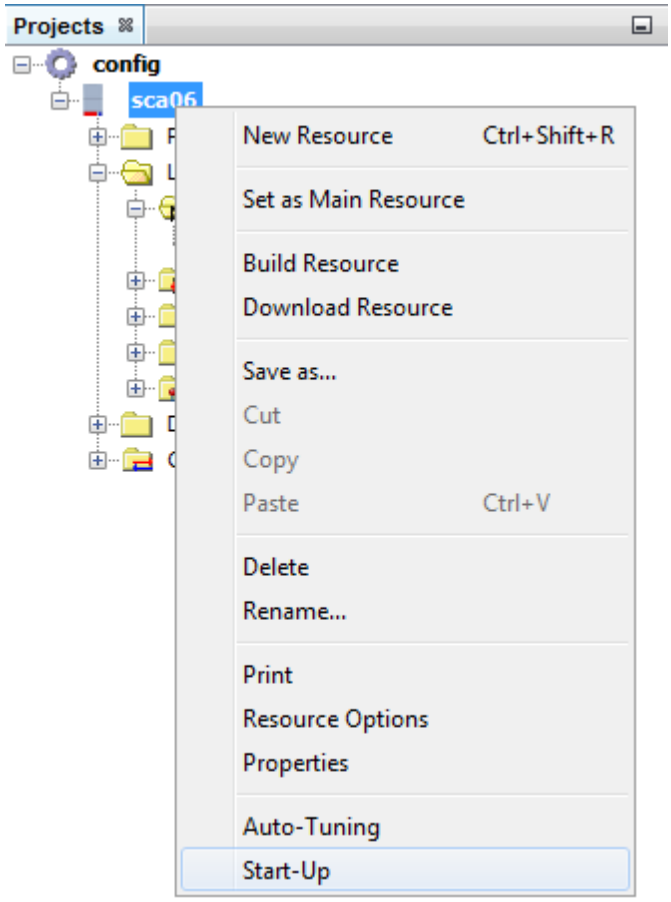


Note 1: When the drive is in "Stopping" on "Errorstop" every block can be called, but only MC_Reset block is executed;
Note 2: Attempt to enable the drive, but the drive is in fault;
Note 3: Enabling the drive and the drive is not in fault;
Note 4: MC_Stop.Done is true and MC_Stop.Execute is false;
Note 5: MC_StepDirect, MC_StepRefPulse or MC_FinishHoming.

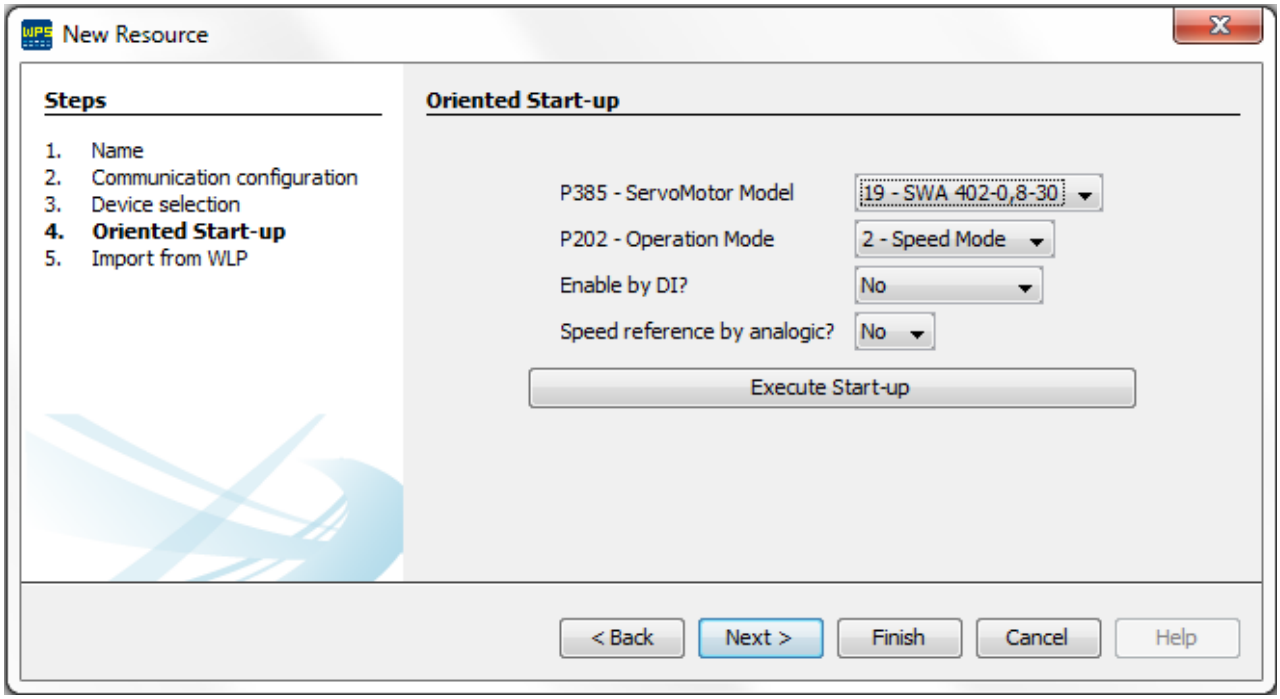
11.6.3 Oriented Start-Up

The function Oriented Start-Up is utilized to realize the minimal required configuration to put SCA-06 into operation.

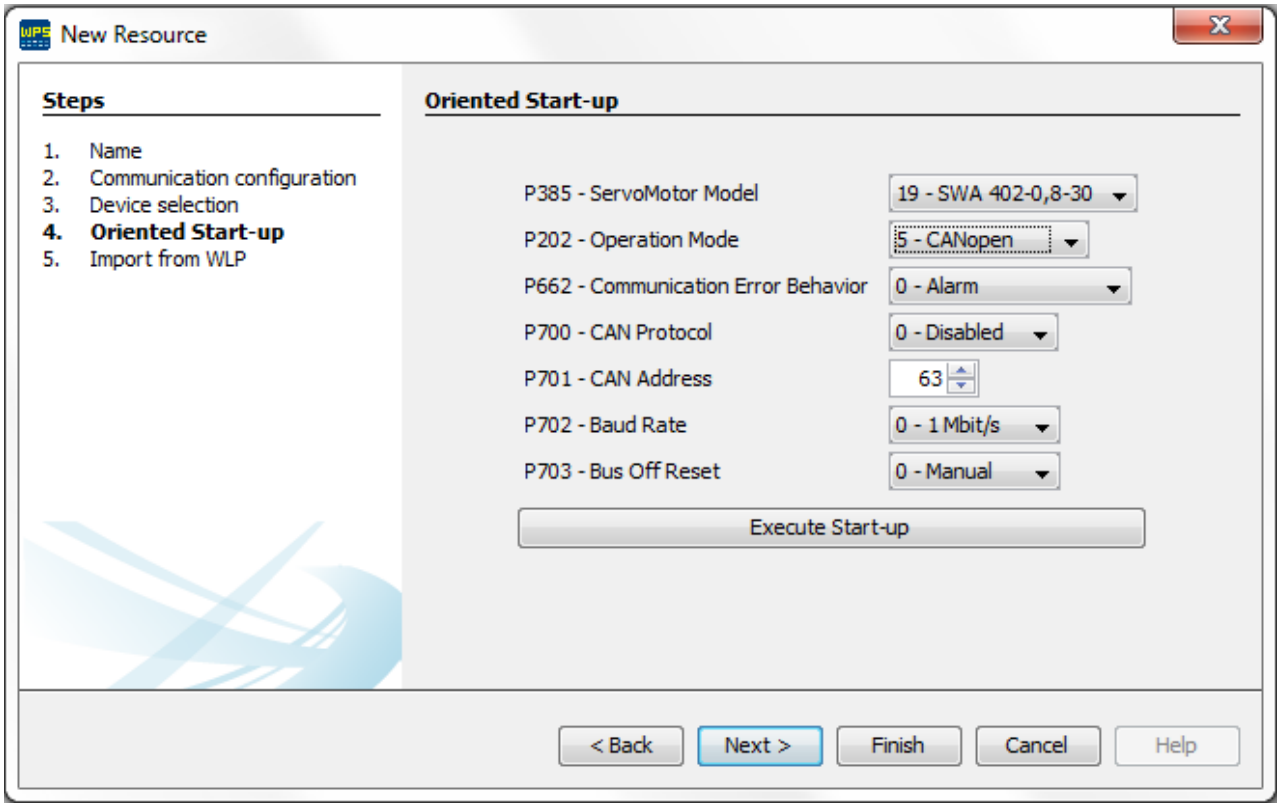
The Oriented Start-Up can be executed during the resource creation, or throught the context menu off the resource by selecting the option **Oriented Start-Up**.



1. On the Start-Up screen the main options that require configuration are the parameters P385 (Servomotor Model) and P202 (Operation Mode).

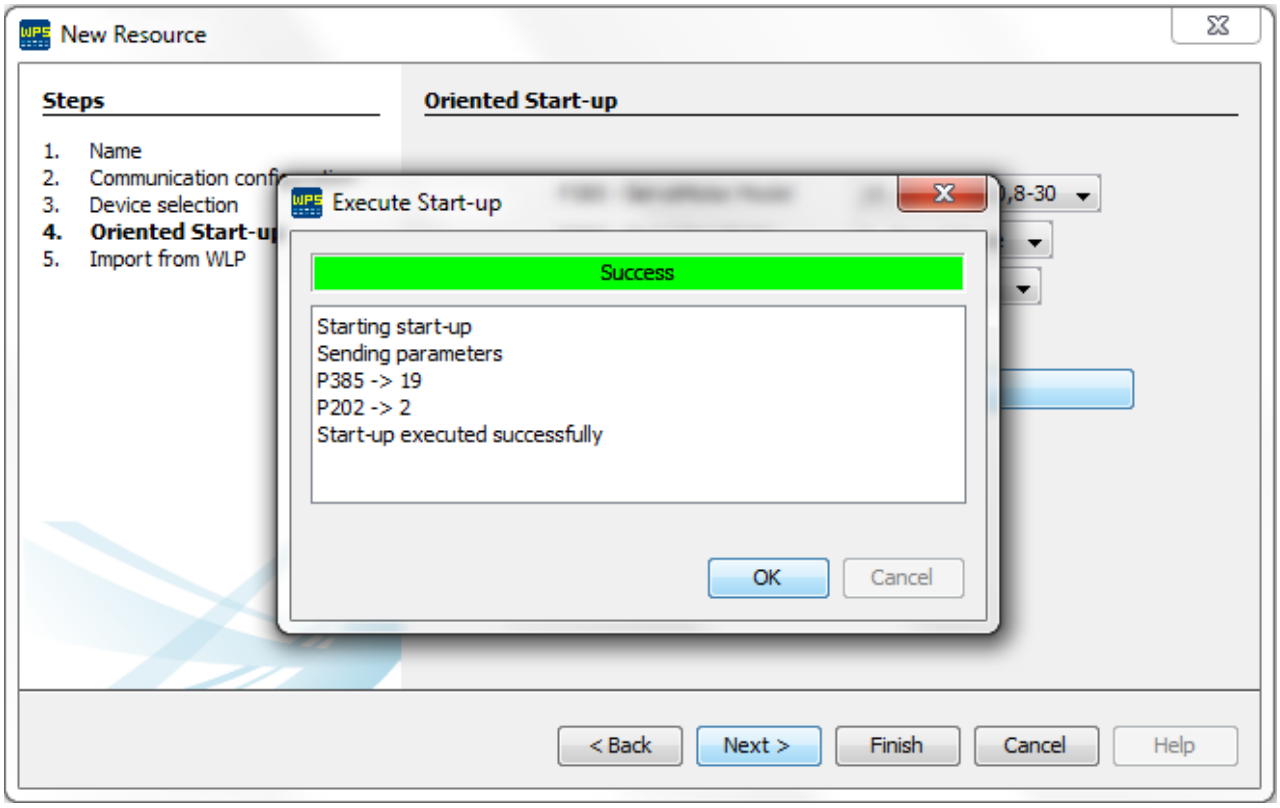


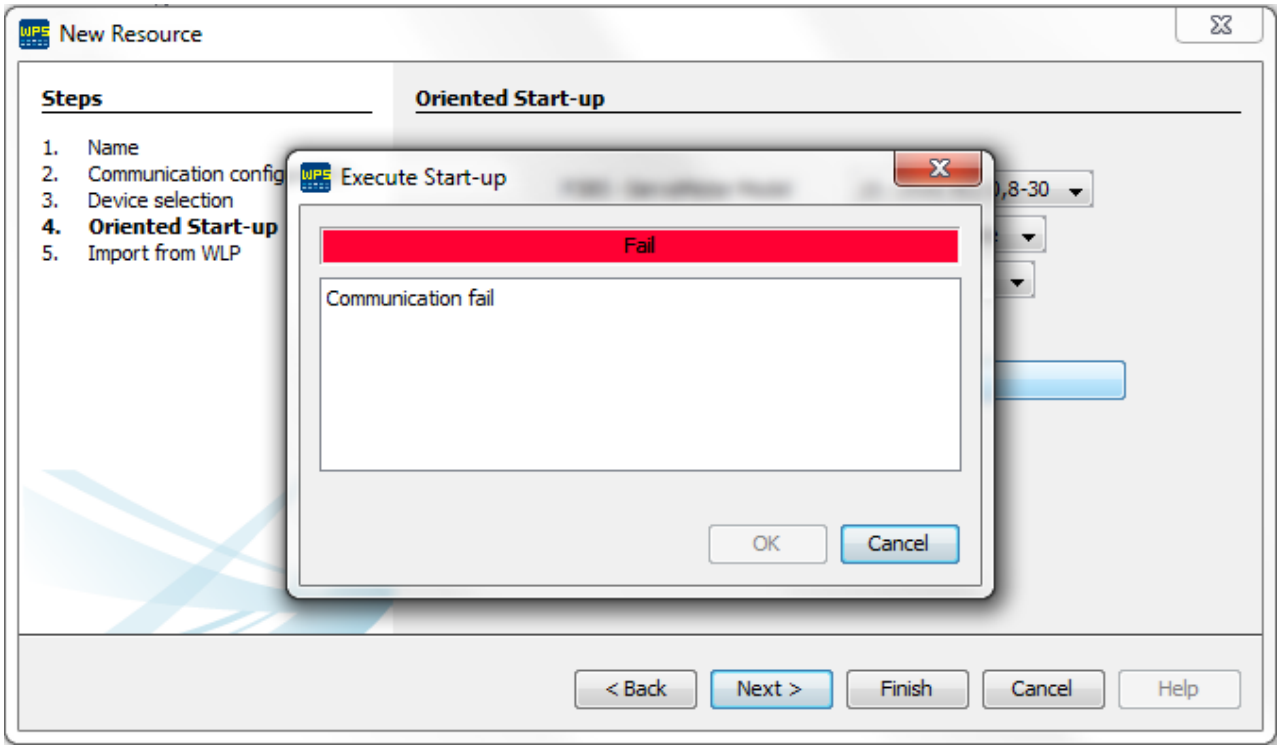
2. According to the operation mode selected the configurator will be adjusted enabling or disabling other options, below is a example of the options enabled when the operation mode is **5 - CANopen**.



3. After the definition of the operation mode and other options, is only necessary to click on **Execute Start-Up** , if it is executed properly a message informing success will be displayed otherwise a message informing

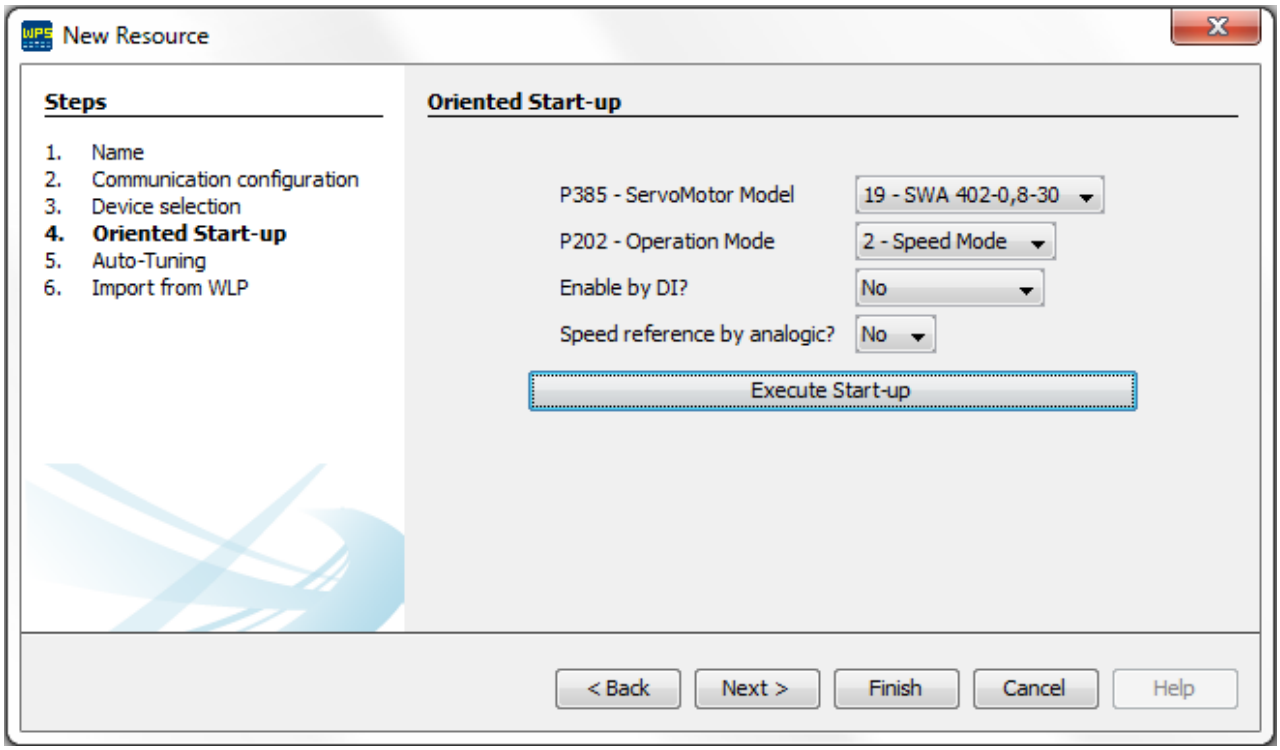
fail will be displayed.





4. After the successfull execution of the Start-Up during the resource creation, the system will enable a step

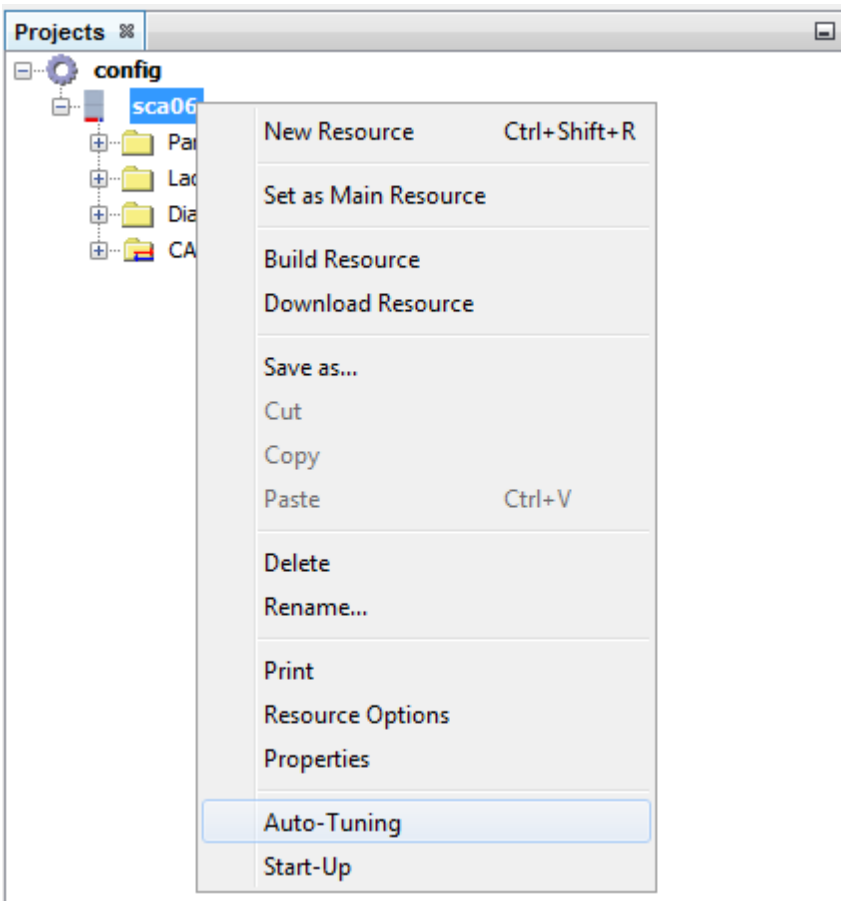
named Auto-Tuning in the wizard, this step is up to the user to perform or not.



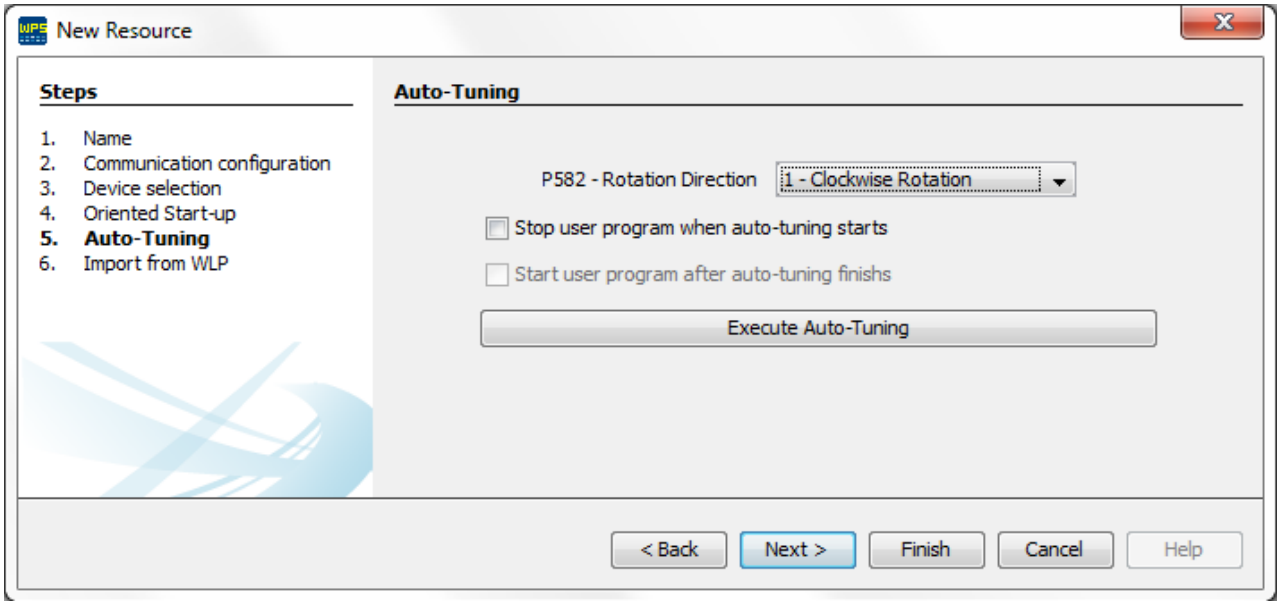
11.6.4 Auto-Tuning

The function Auto-Tuning is utilized to realize automatic adjusts on SCA-06 to obtain a better performance of the equipment.

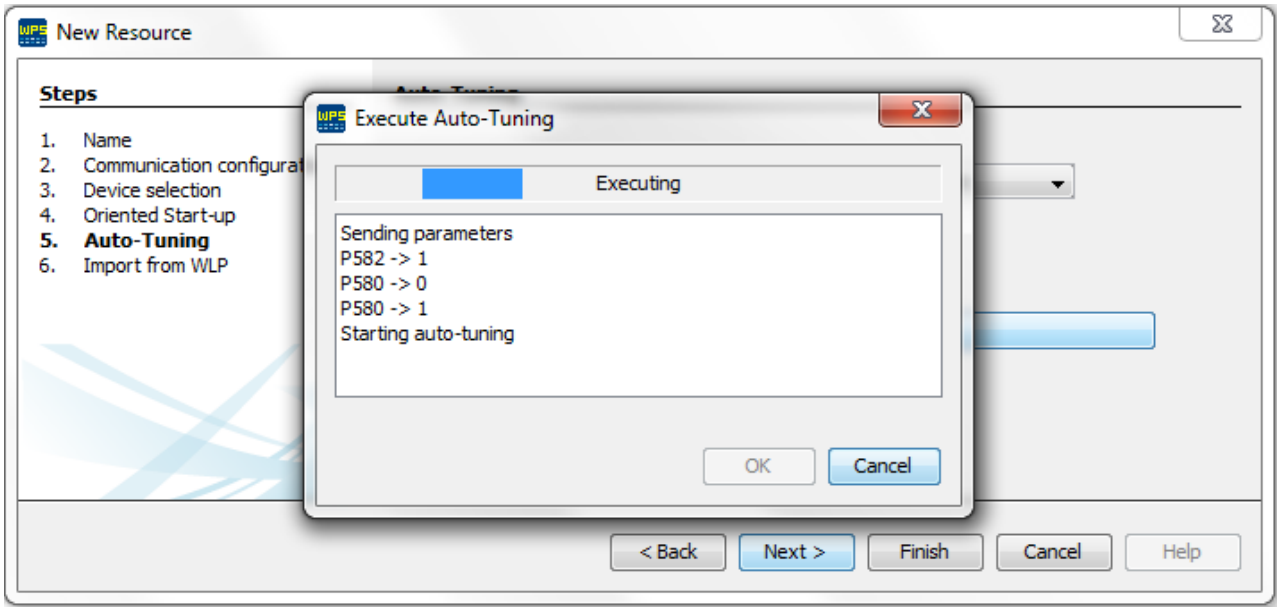
The Auto-Tuning can be executed during the resource creation, or throught the resource context menu by selecting the option **Auto-Tuning**.

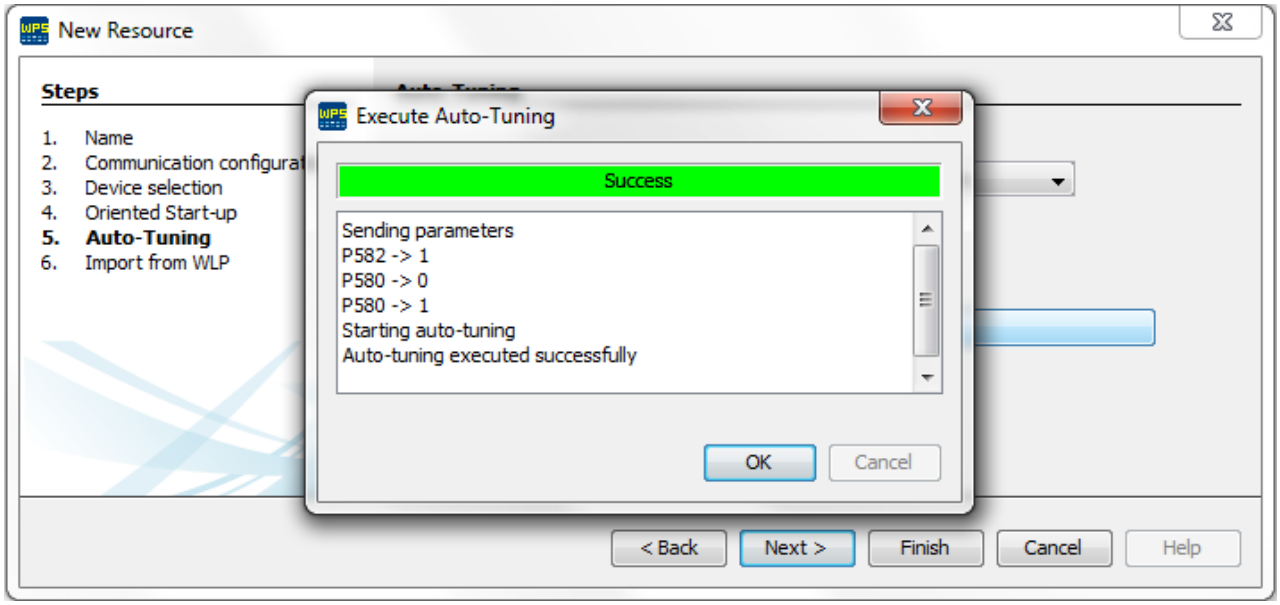


1. On the Auto-Tuning screen the options that require configuration are P582 (Rotation Direction) and if the user program should be stopped before execution and started again after the Auto-Tuning conclusion.



2. After choosing the options is only necessary to click on **Execute Auto-Tuning** to start the process that takes less than a minute. if it is executed properly a message informing success will be displayed otherwise a message informing fail will be displayed.

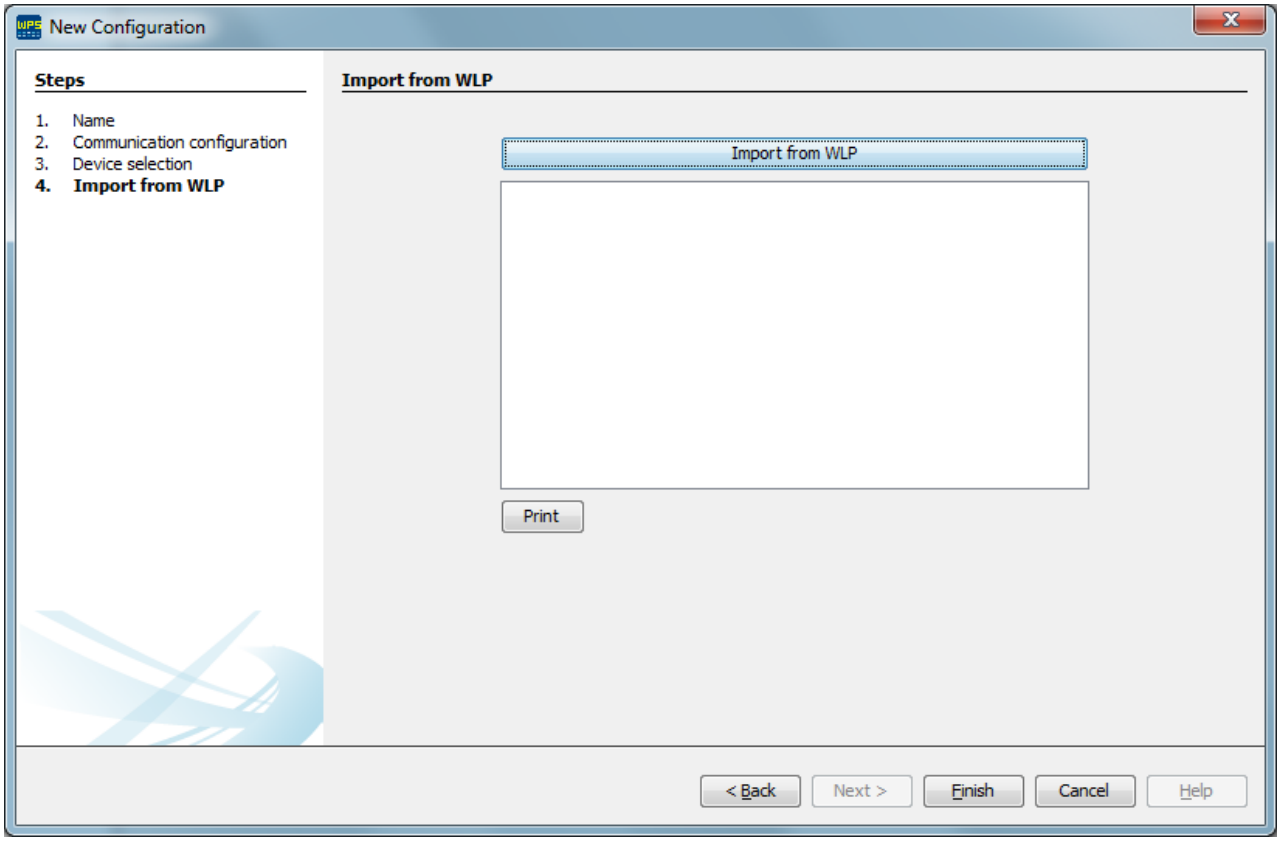




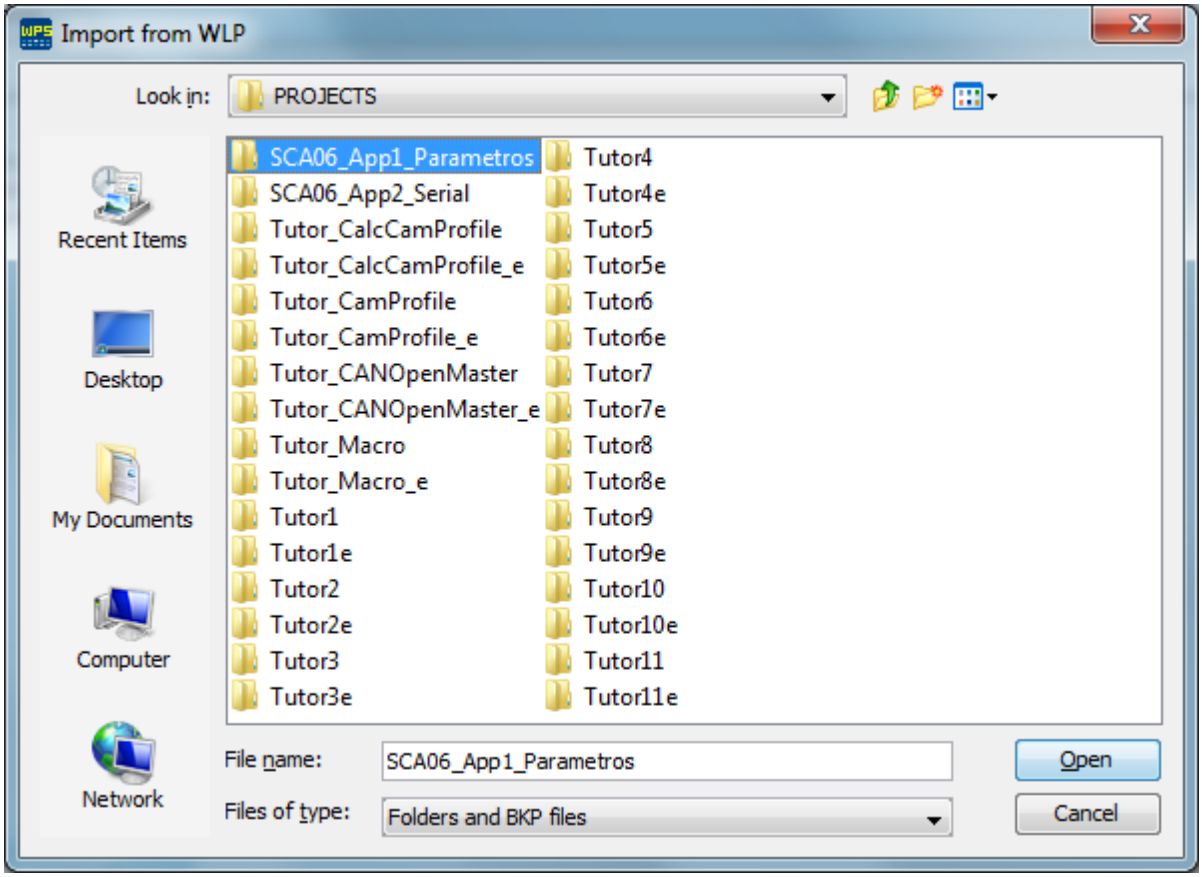
11.6.5 Import from WLP

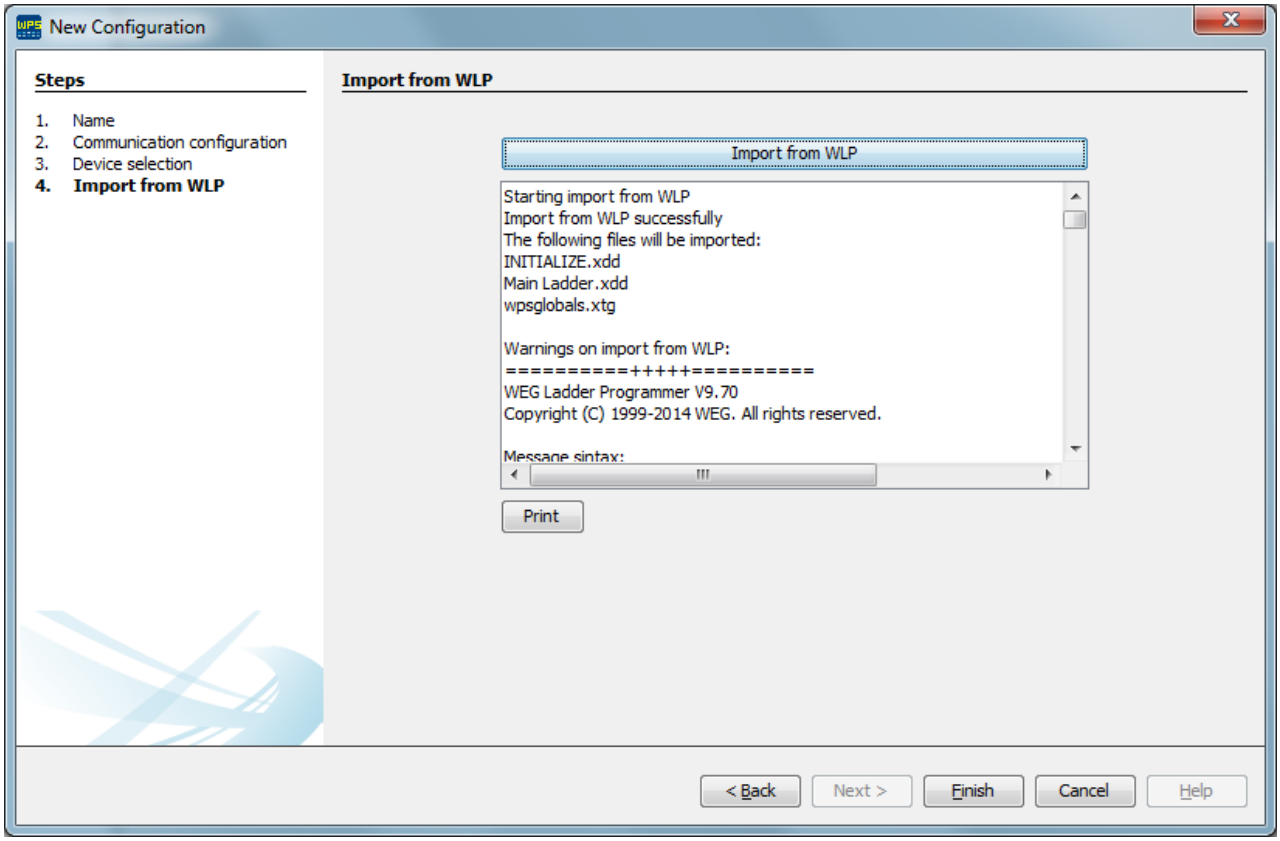
The function import from WLP is utilized to import Ladder developed on WLP software to equipment (device).

The import from WLP can be executed during the resource creation.



1. To execute the import WLP function click the Import from WLP button and select the WLP project folder or the WLP BKP file.

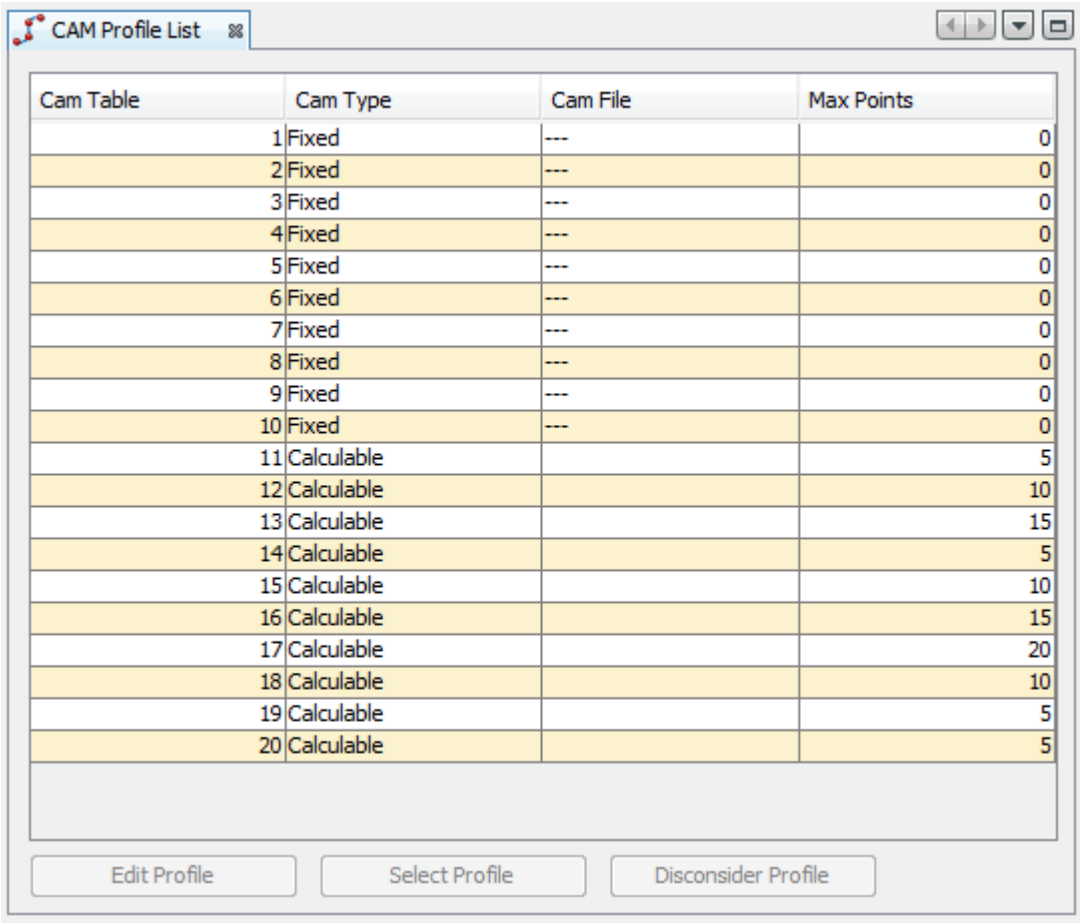

Import from WLP
Look in:
PROJECTS
SCA06_App1_Parametros
SCA06_App2_Serial
Tutor_CalcCamProfile
Tutor_CalcCamProfile_e
Tutor_CamProfile
Tutor_CamProfile_e
Tutor_CANOpenMaster
Tutor_CANOpenMaster_e
Tutor_Macro
Tutor_Macro_e
Tutor1
Tutor1e
Tutor2
Tutor2e
Tutor3
Tutor3e
Tutor4
Tutor4e
Tutor5
Tutor5e
Tutor6
Tutor6e
Tutor7
Tutor7e
Tutor8
Tutor8e
Tutor9
Tutor9e
Tutor10
Tutor10e
Tutor11
Tutor11e
Recent Items
Desktop
My Documents
Computer
Network
File name:
SCA06_App1_Parametros
Files of type:
Folders and BKP files
Open
Cancel



2. After import from WLP completed successfully click the Finish button to copy the imported files to new resource.

11.6.8 Cam Profiles

It allows loading and editing the cam table of the CAM curves.
Accessed by the **CAM Profile List** command with the right button of the mouse in the **CAM Profiles** folder of the resource.

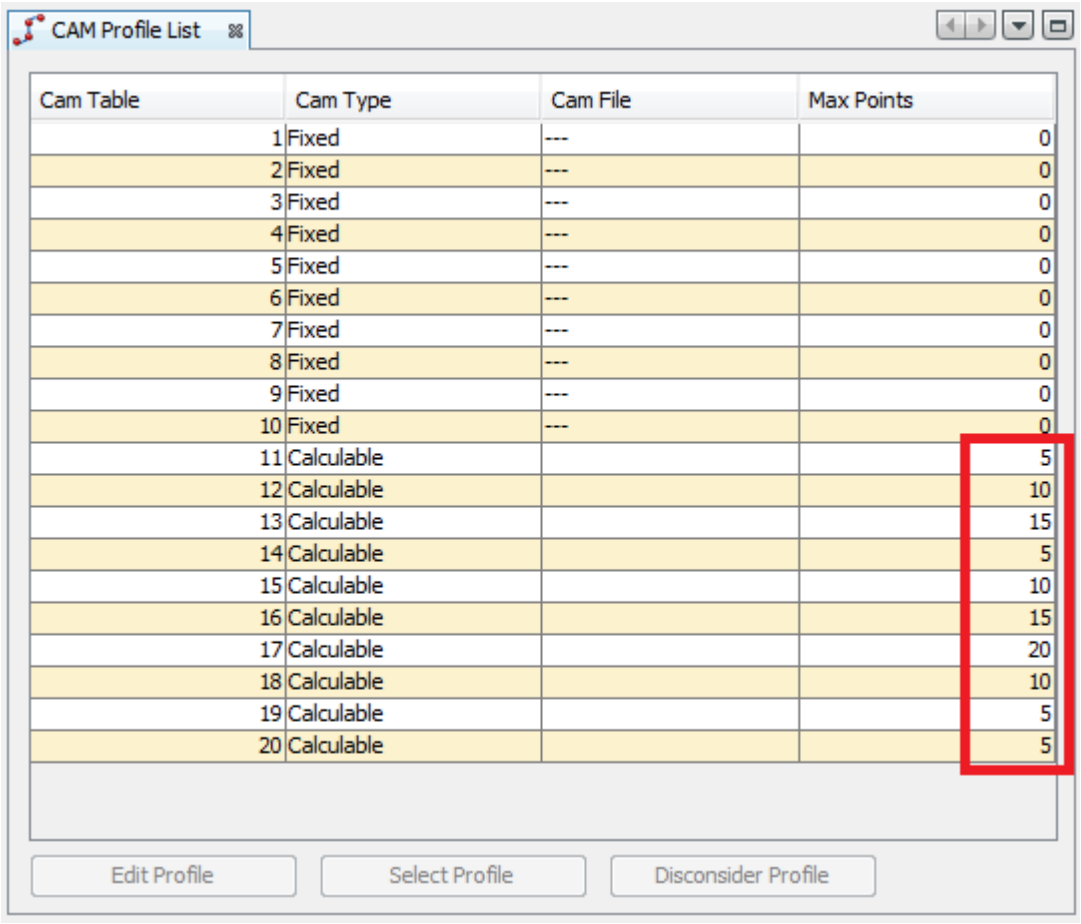


Cam Table	Cam Type	Cam File	Max Points
1	Fixed	---	0
2	Fixed	---	0
3	Fixed	---	0
4	Fixed	---	0
5	Fixed	---	0
6	Fixed	---	0
7	Fixed	---	0
8	Fixed	---	0
9	Fixed	---	0
10	Fixed	---	0
11	Calculable		5
12	Calculable		10
13	Calculable		15
14	Calculable		5
15	Calculable		10
16	Calculable		15
17	Calculable		20
18	Calculable		10
19	Calculable		5
20	Calculable		5

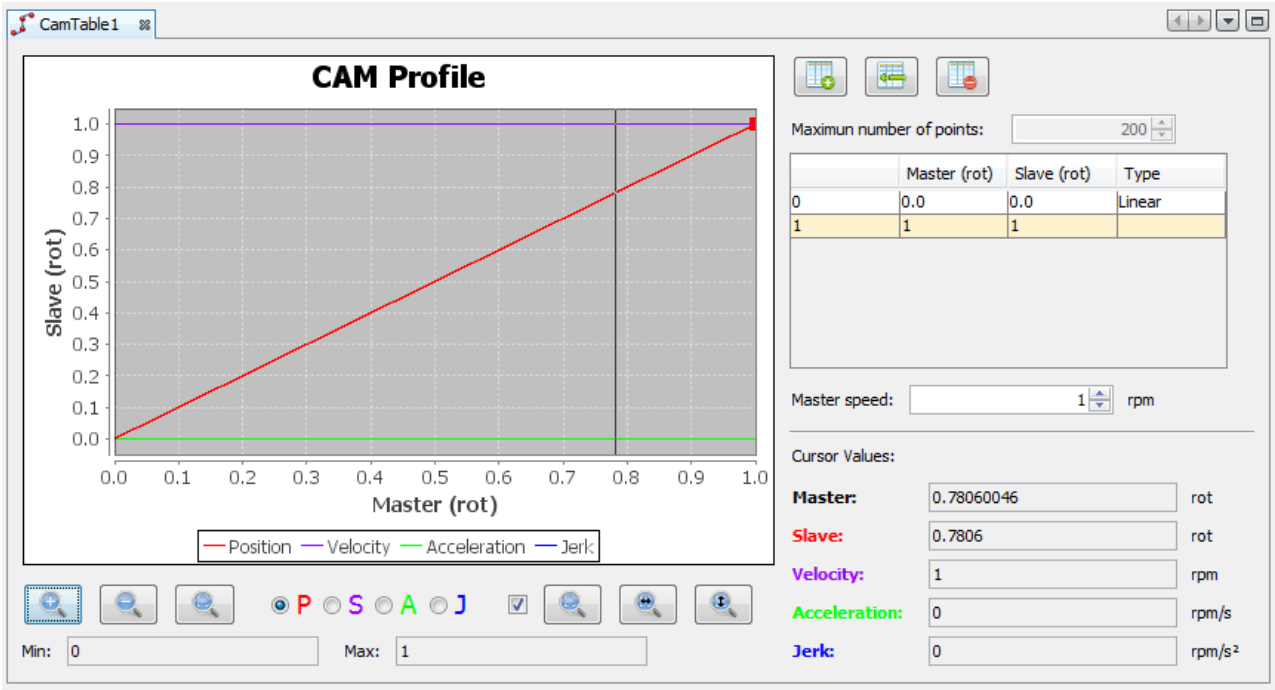
Description

The cam tables from 1 to 10 are tables of fixed points, which are transmitted at the moment of the download of the application. In order to use the tables 1 to 10, first the MC_CamTableSelect block must be executed with the desired table and then the MC_CamIn block.
The cam tables 11 to 20 are tables of variable points. In order to use the tables 11 to 20, first the MC_CamCalc block must be executed with the desired table and then the MC_CamIn block.

For the SCA06 equipment, it is allowed programming at most 200 fixed points and 100 variable points, seeing that the maximum number of variable points of each table must be configured in the Max Points column, as shown below:

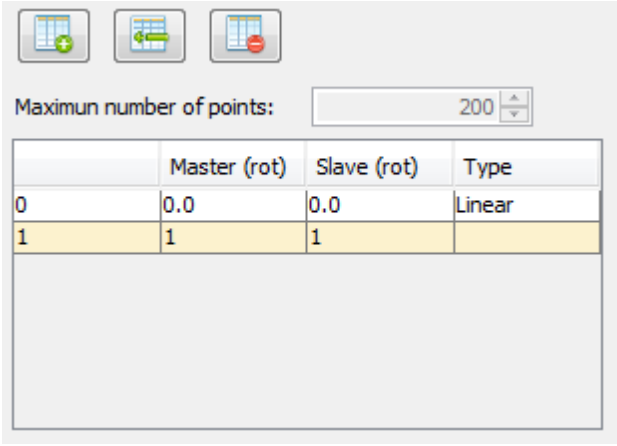


In order to edit the cam table, click on the **Edit** button, and the cam profile editor will open, as in the figure below:



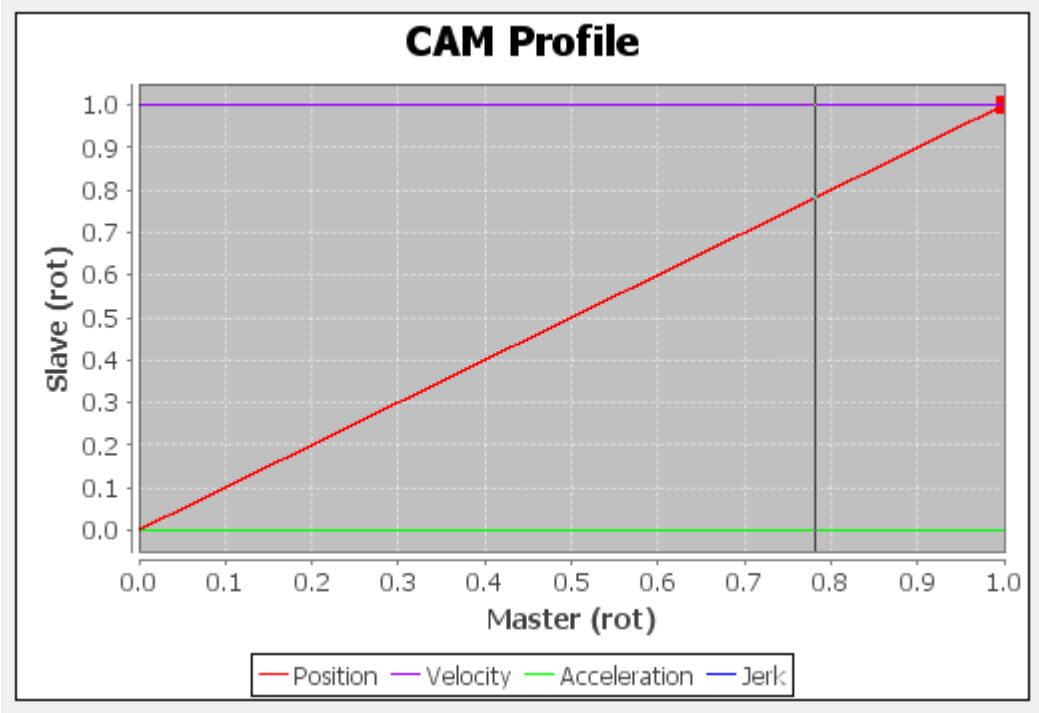
This window has the following controls:

Cam table:

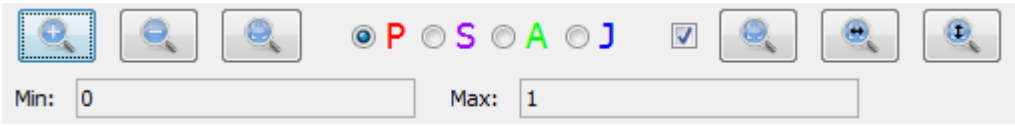


NOTE!
The CAM block is always relative, so the first point of the cam table will always be master= 0 and slave = 0.

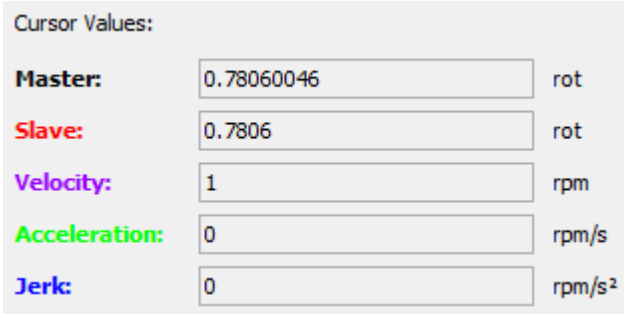
Graphic of the profile:



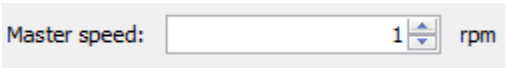
Graphic control tools:



Values of the cursor:
Values relative to the selected point of the cursor.



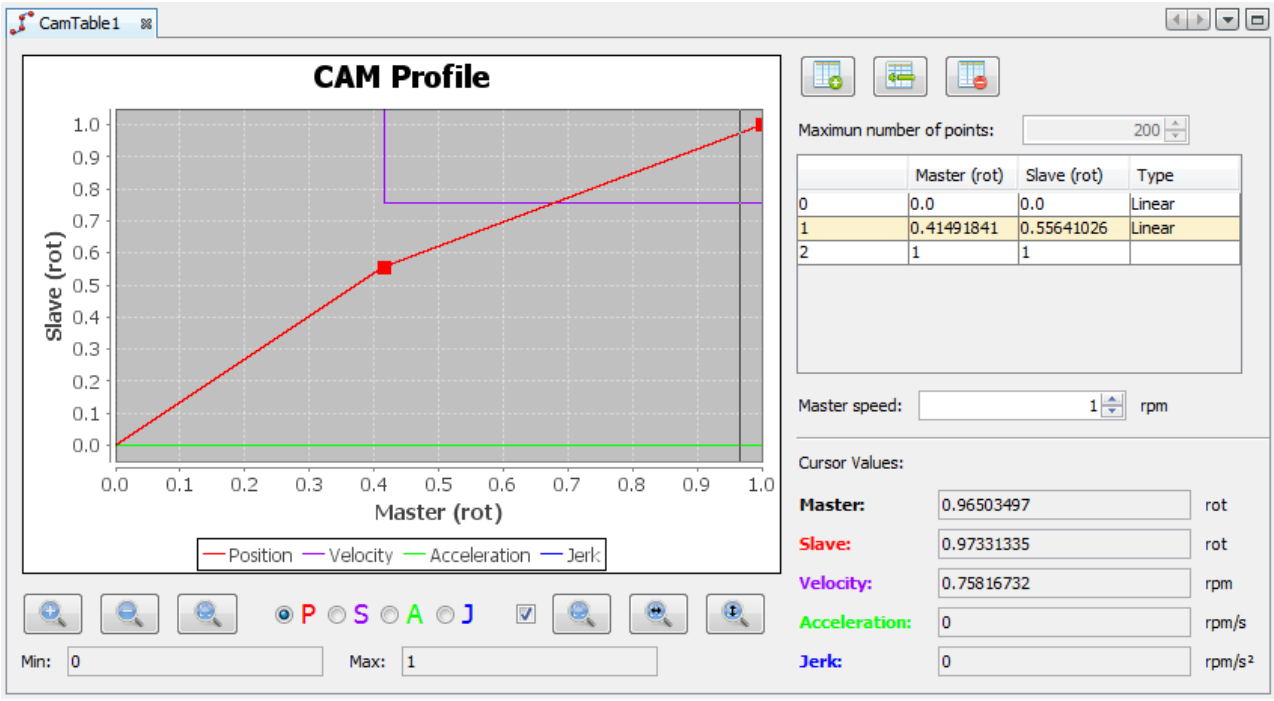
Master speed:
Speed used to calculate the speed, acceleration and jerk of the slave.



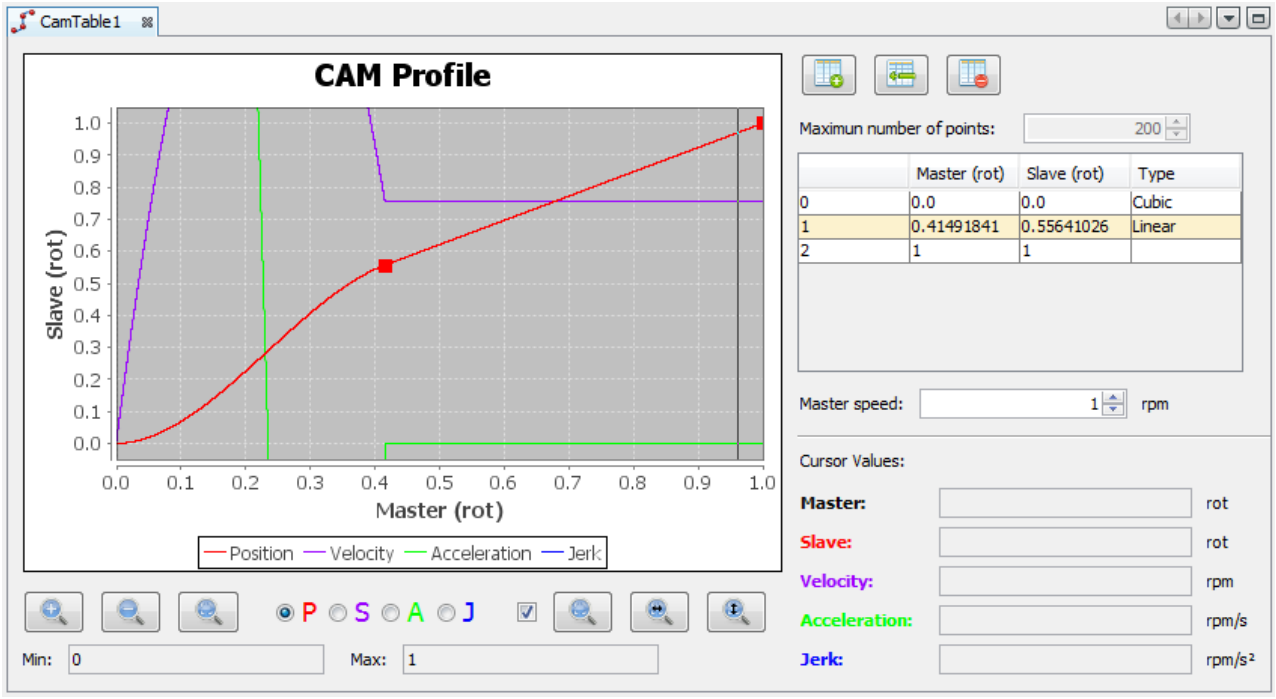
NOTE!
The speed, acceleration and jerk of the slave must be used as reference to develop the cam profile, where they are calculated numerically, not taking into account load, inertia, torque and dynamics of the drive.

Adding a new point to the cam profile

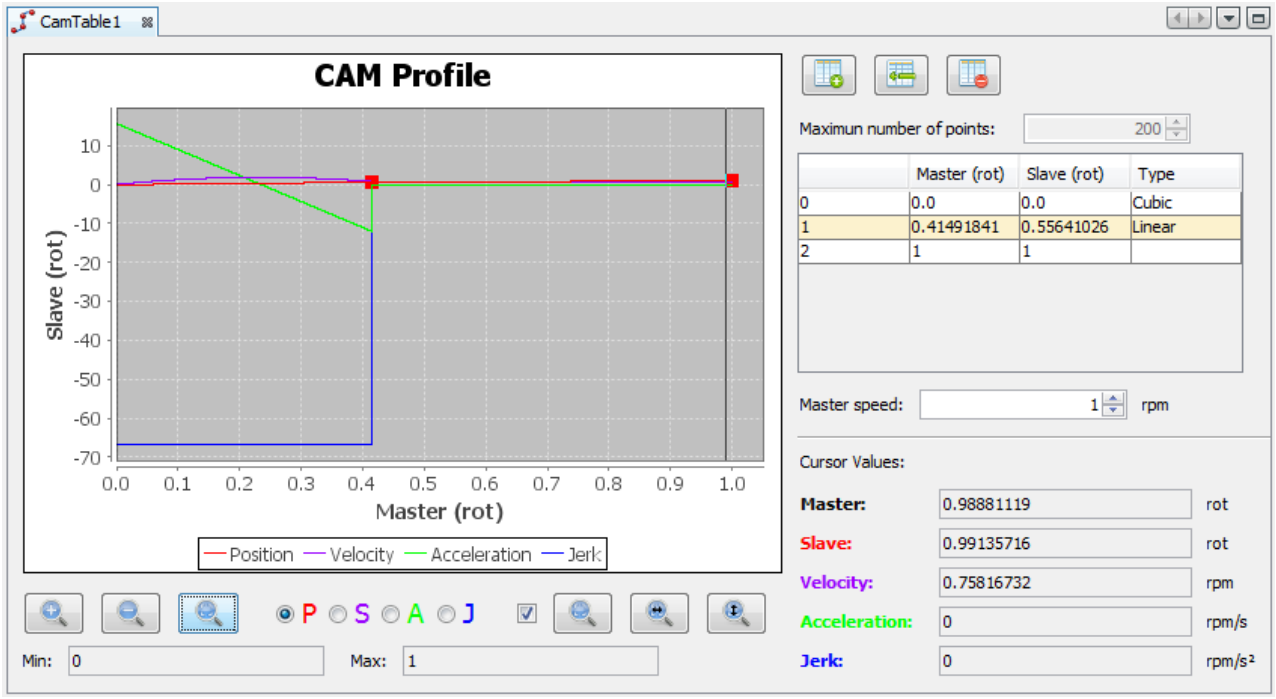
A point can be added by means of the add or insert point buttons or by double clicking the graphic in the position where you wish to add the point. You can double click any region of the graphic. In case an interpolation already exists in this region, the editor will insert this point between the two points of the interpolation.
The point is always added as linear interpolation.
When a point is added or inserted by means of the respective buttons, the master and slave values come zeroed. In case of point insertion, that may cause an interruption of the profile, because the master position must always grow in relation to the origin; therefore, the value of the master and slave must be edited by clicking on their cells in the cam table.
On the figure below, a point was inserted by double clicking:



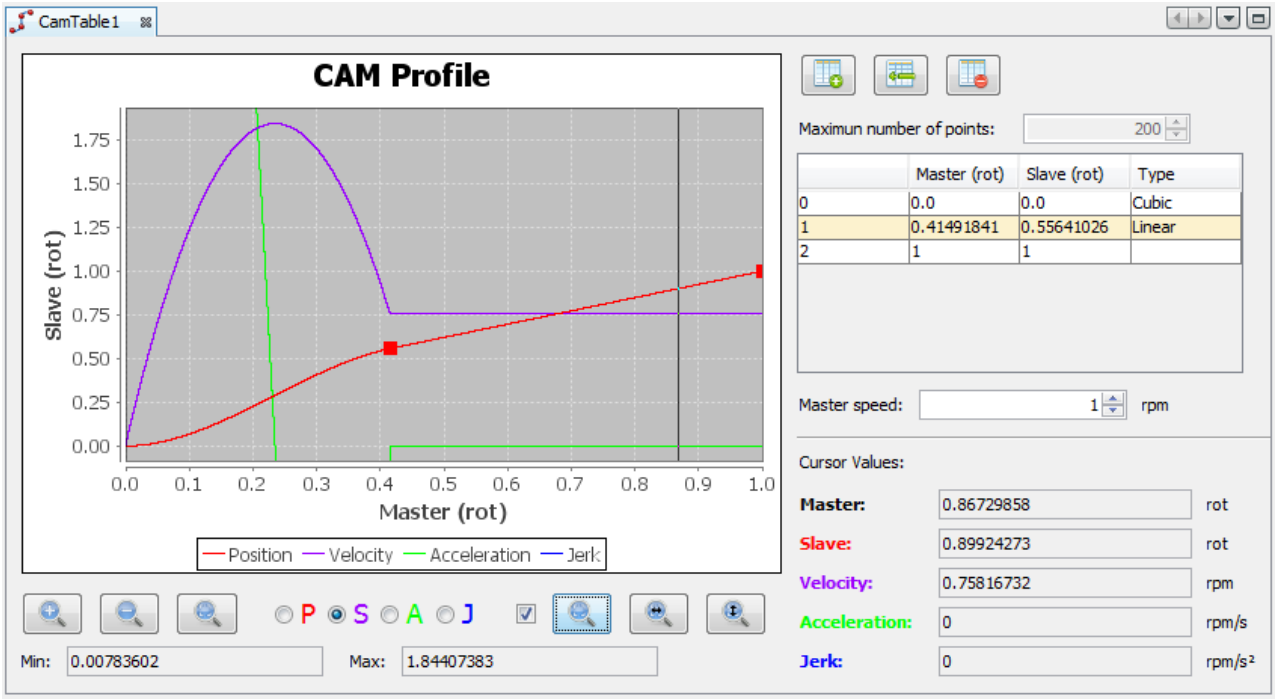
In order to change the type of interpolation, click on the type cell in the line corresponding to the origin of the interpolation and select the desired type.
In the figure below, the point was changed for cubic interpolation.



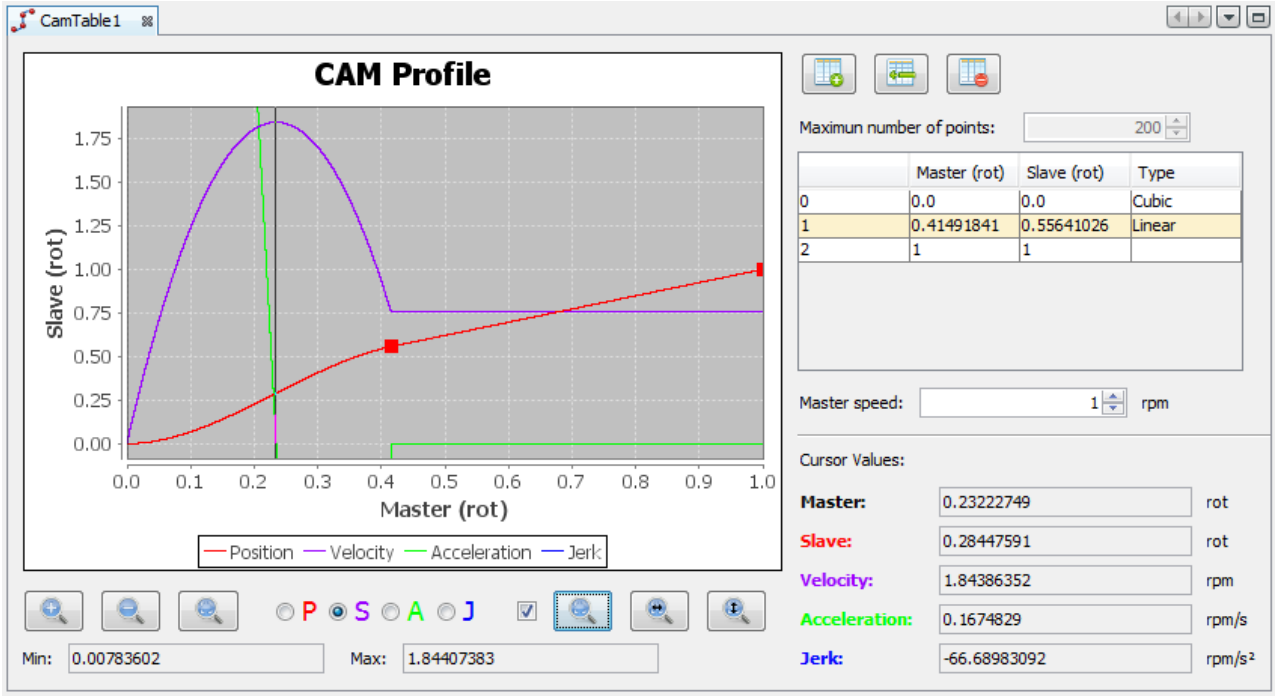
Now, in this curve, it is already possible to see other magnitudes besides the position, such as speed, acceleration and jerk. For a better view of all magnitudes, we can use the **Set Zoom All** button according to the figure below.



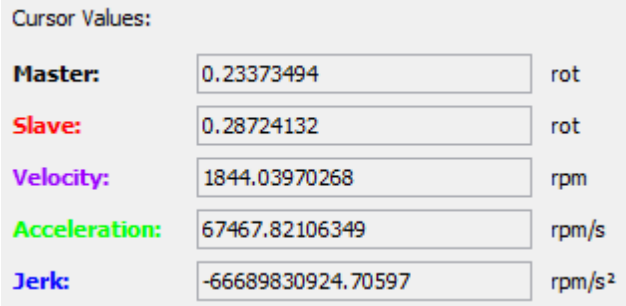
The same way, we can choose one of the magnitudes and use the **Apply Selected Zoom** button. In the example below, a zoom was applied to the speed.



Another interesting tool is the cursor. In the example below, we will place the cursor in point of maximum speed.

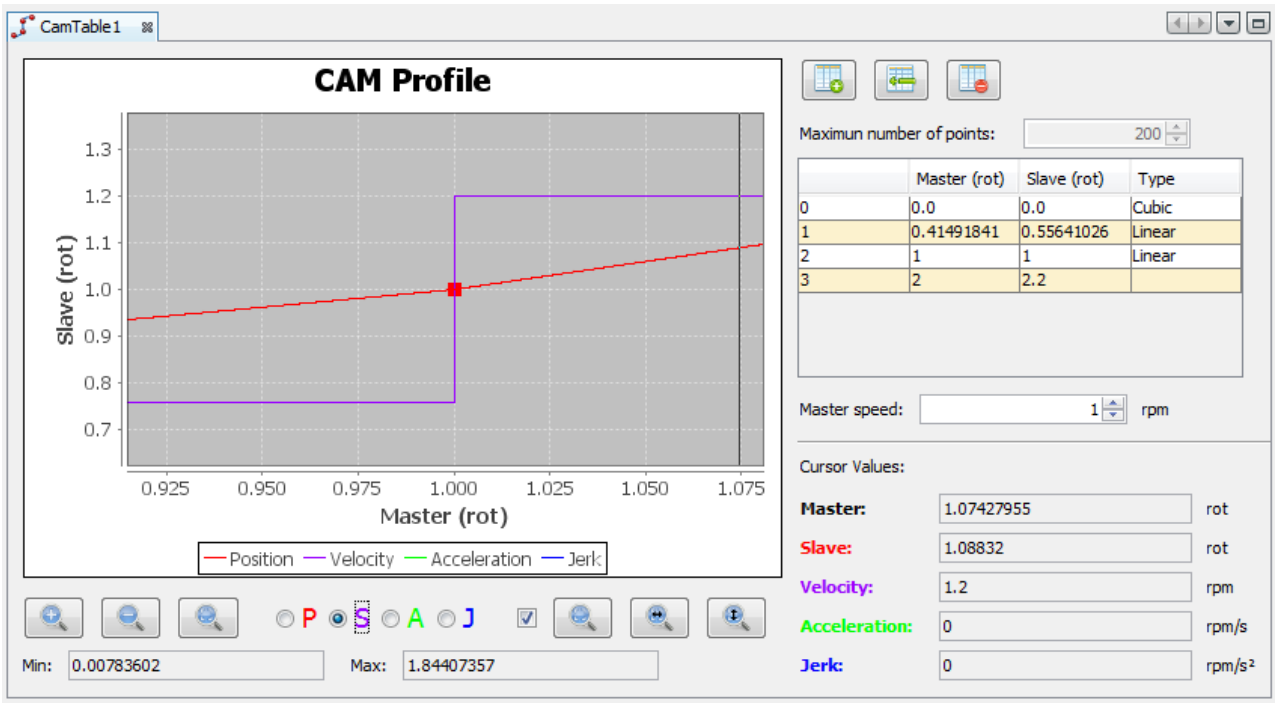


You must bear in mind that the speed, acceleration and jerk of the slave depend on the master speed; therefore, it is interesting to change them so as to simulate something really close to the effective values. In the figure below, the master speed will be changed to 1000 rpm and we will analyze the same position of the cursor.



During the project of the cam profile, all those magnitudes must be observed, because they may be accomplished or not due to mechanical, electrical and electronic limitations of the involved equipment.

Since the acceleration and jerk graphics are calculated taking into account the interpolation between two points, the acceleration and jerk will be shown as equal to zero in the junctions between linear interpolations. Although in theory we know that in a speed step the acceleration and jerk are infinite, in practice the acceleration and jerk at this moment will also depend on the mechanical, electrical and electronic limitations of the involved equipment. Those speed steps must be observed and considered in the project of the cam profile. The figure below shows an example of this situation.



The CAM block offers two types of interpolation: linear and cubic. The following equations are used:

Linear:

$$pe = pie * \left(\frac{pfm - pm}{pfm - pim} \right) + pfe * \left(\frac{pm - pim}{pfm - pim} \right)$$

$$ve = \left(\frac{-pie}{pfm - pim} + \frac{pfe}{pfm - pim} \right) * vm$$

$$ae = 0$$
$$je = 0$$

Cubic:

$$pe = a*(pm - pim)^3 + b*(pm - pim)^2 + c*(pm - pim) + pie$$
$$ve = (3*a*(pm - pim)^2 + 2*b*(pm - pim) + c)*vm$$
$$ae = (6*a*(pm - pim) + 2*b)*vm^2$$
$$je = 6*a*vm^3$$

where:

pe = slave position
ve = slave speed
ae = slave acceleration
je = slave jerk
pm = master position
vm = master speed
pim = master initial position
pfm = master final position
pie = slave initial position
pfe = slave final position
a = coefficient calculated by the CAM editor
b = coefficient calculated by the CAM editor
c = coefficient calculated by the CAM editor

Changing a point in the cam profile

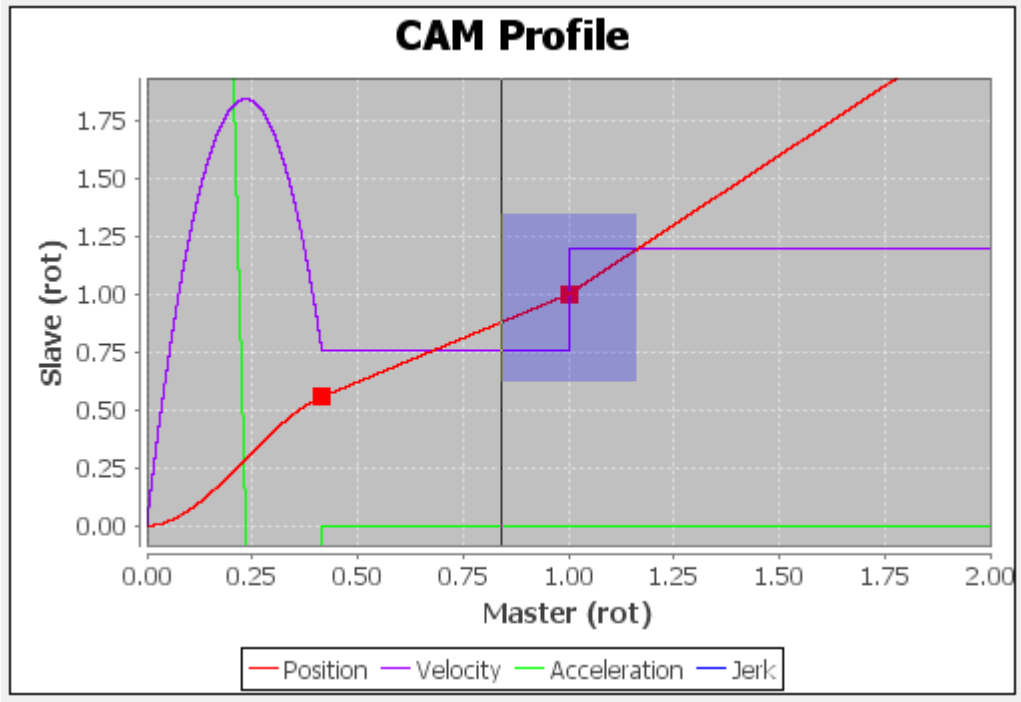
A point can be changed by means of the cam table by using direct edition or moving the point in the graphic. In order to move the point in the graphic, place the cursor on the point, which is marked with a red square, click and hold it, and drag it to the new position.
When you click on the point, the cam table will move to this point, selecting the related cell.
The operation of moving the point in the graphic is interactive and calculates all the profile each time the point is changed. The new point can be seen in the cam table.

Removing a point from the cam profile

The point is removed directly in the cam table. In order to do so, select one of the cells referring to the point and click on the **Remove Point** button.

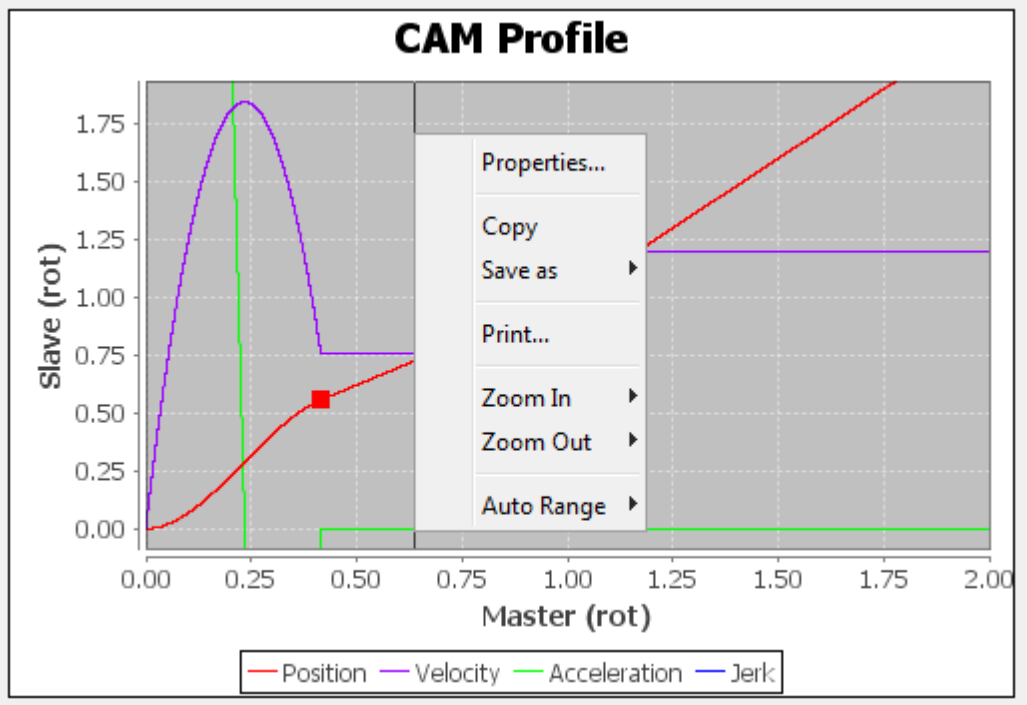
Zoom of a certain area of the graphic

Click on one of the corners of the region you wish to zoom and hold it, and move the mouse so as to mark a region. Then a rectangle will show on the graphic; release the button. The figure below shows an example of this zoom.

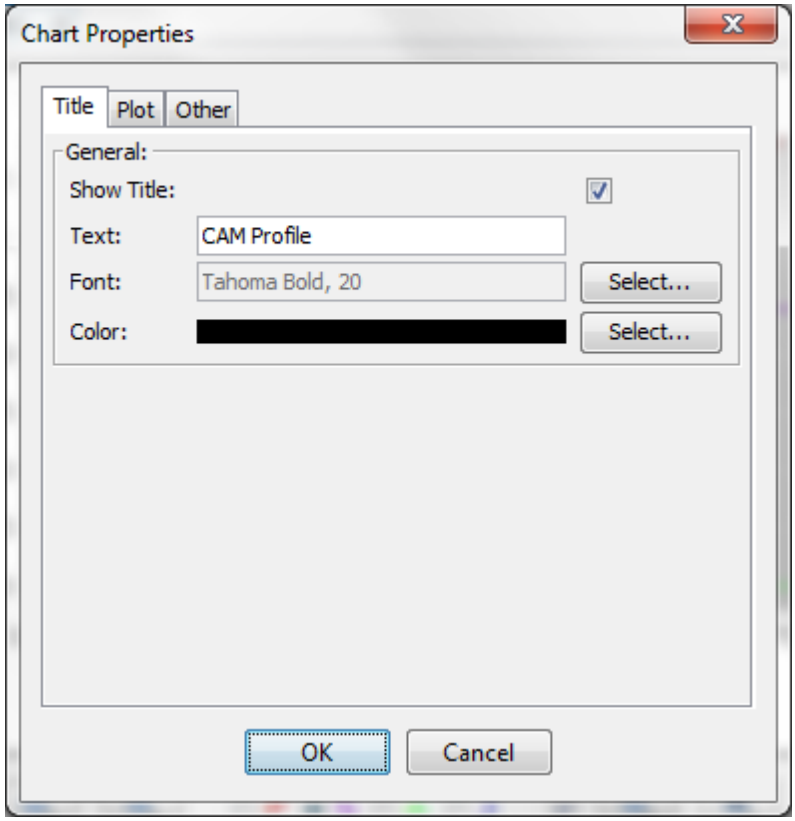


Graphic menu

In order to access the graphic menu, right click on the graphic area, and the following menu will show.



The figure below shows the graphic property box.



11.6.9 Structures

Structure is a data grouping used to define a recipe or an object.

In the Ladder program, it is possible to create variables of the structure type and use them in the blocks. To access the internal members of the structure, the '.' is used followed by its respective member.

Creating a structure

1. With the right button of the mouse on the folder **Structure**, click on **New file**.

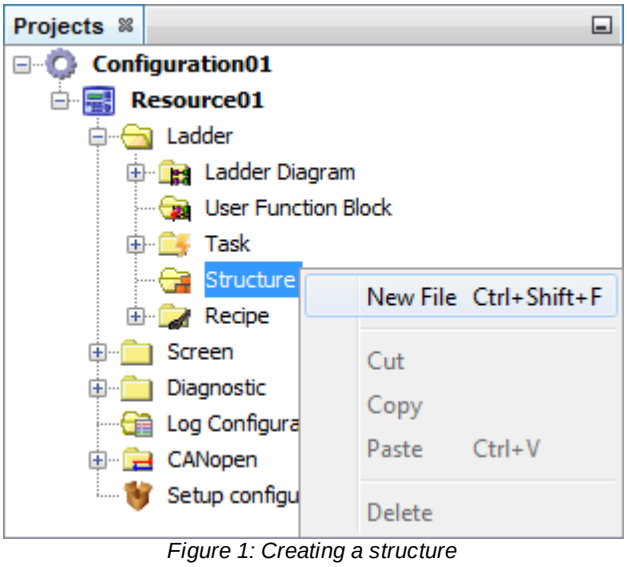


Figure 1: Creating a structure

2. Define the file name and press the **Next** button.

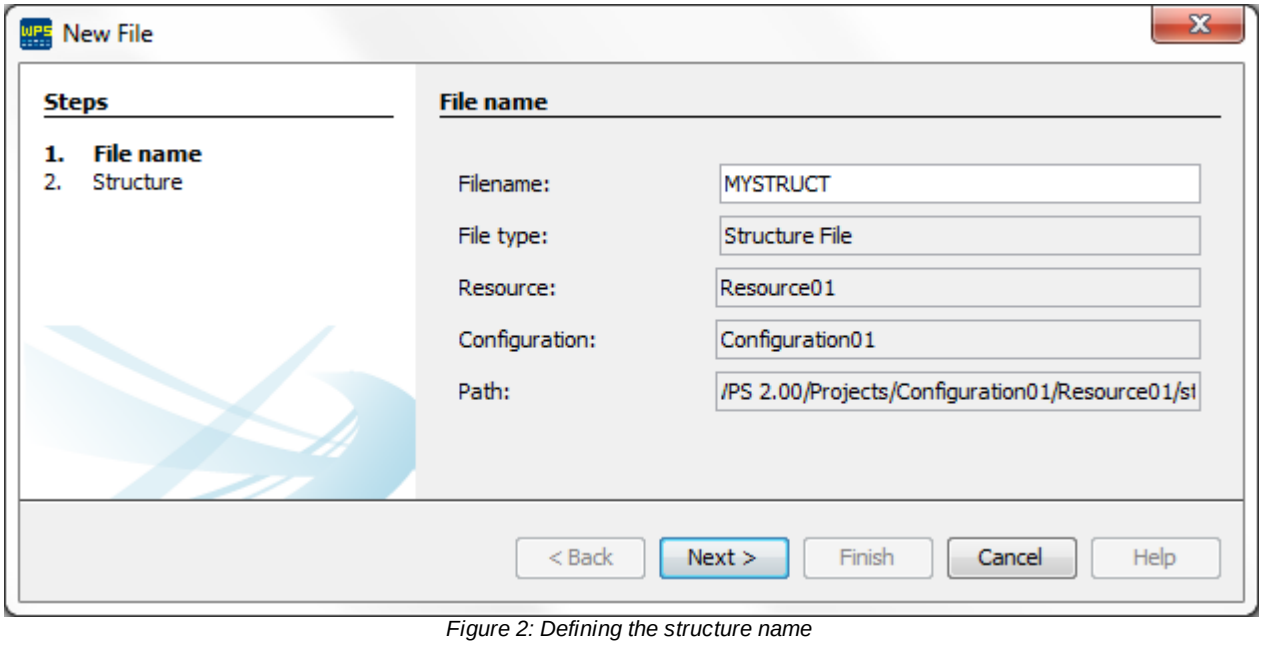


Figure 2: Defining the structure name

3. Configure the structure using the buttons presented in the figure below.

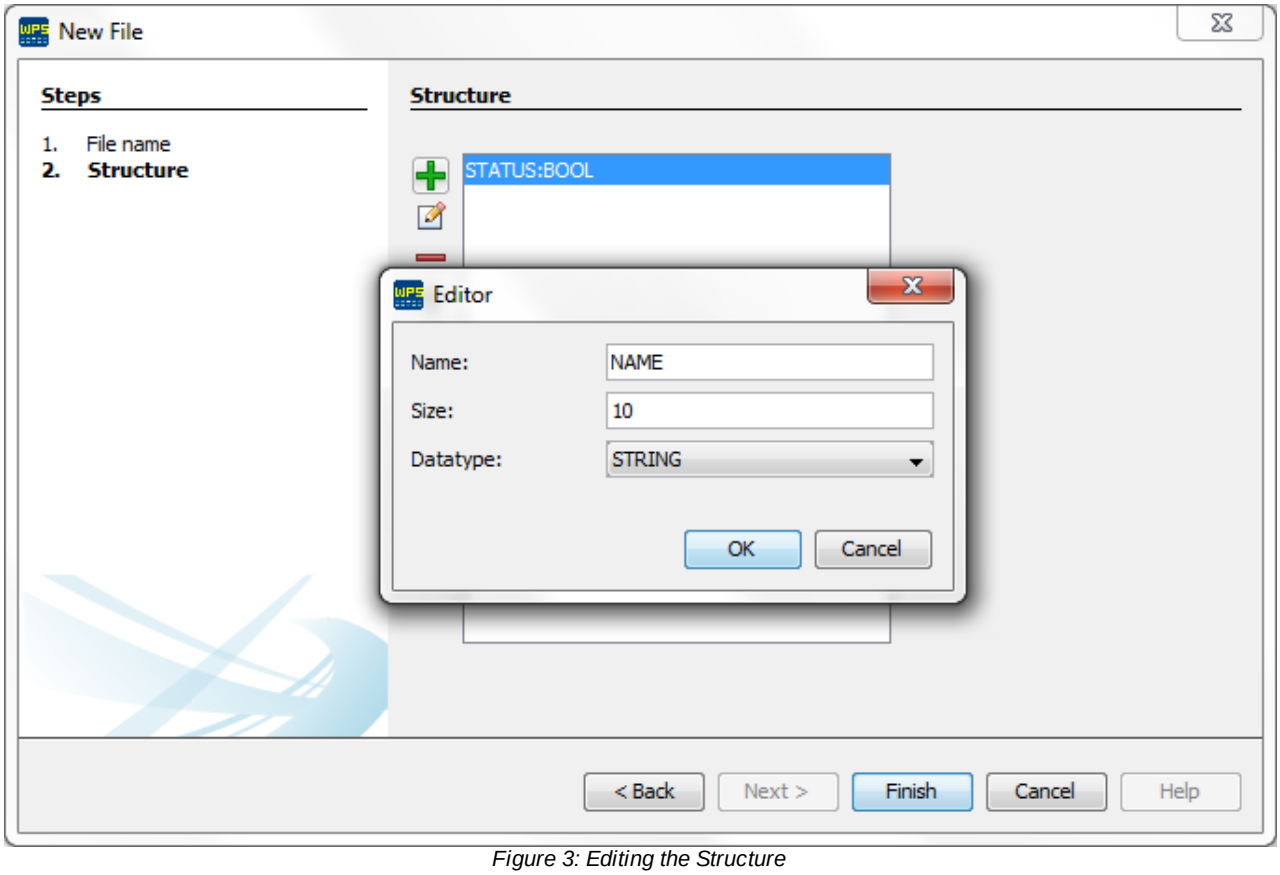


Figure 3: Editing the Structure

4. After finishing the edition of the structure, click on the button **Finish**.

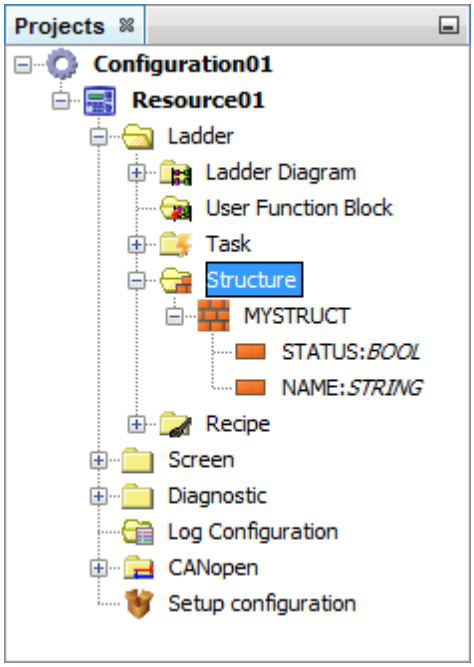


Figure 4: Structure created in the project

Editing a structure

Just double click on the desired structure, as shown in figure 4, and a window will open as shown in figure 3, allowing to insert new data, erase or move the position of the data.

11.7 MW500

11.7.1 Description

The MW500 is a high-performance variable speed drive for controlling three-phase induction motors with dedicated functions and high degree of protection IP66 / NEMA4X that allow their use in applications requiring a high level of precision and robustness. In addition, the MW500 has excellent flexibility because it can be installed directly on the wall or mounted on the motor, reducing costs of wiring and panels.

Refer to the user's manual of the MW500 for further details about the product.

NOTE!

This product does not have the Ladder tool available in WPS.
You can use the WLP application if this feature is required.

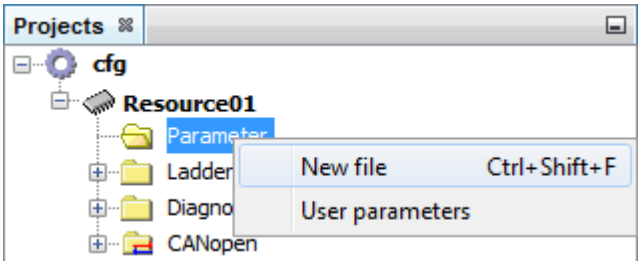
11.7.2 Parameters

11.7.2.2 Configuration

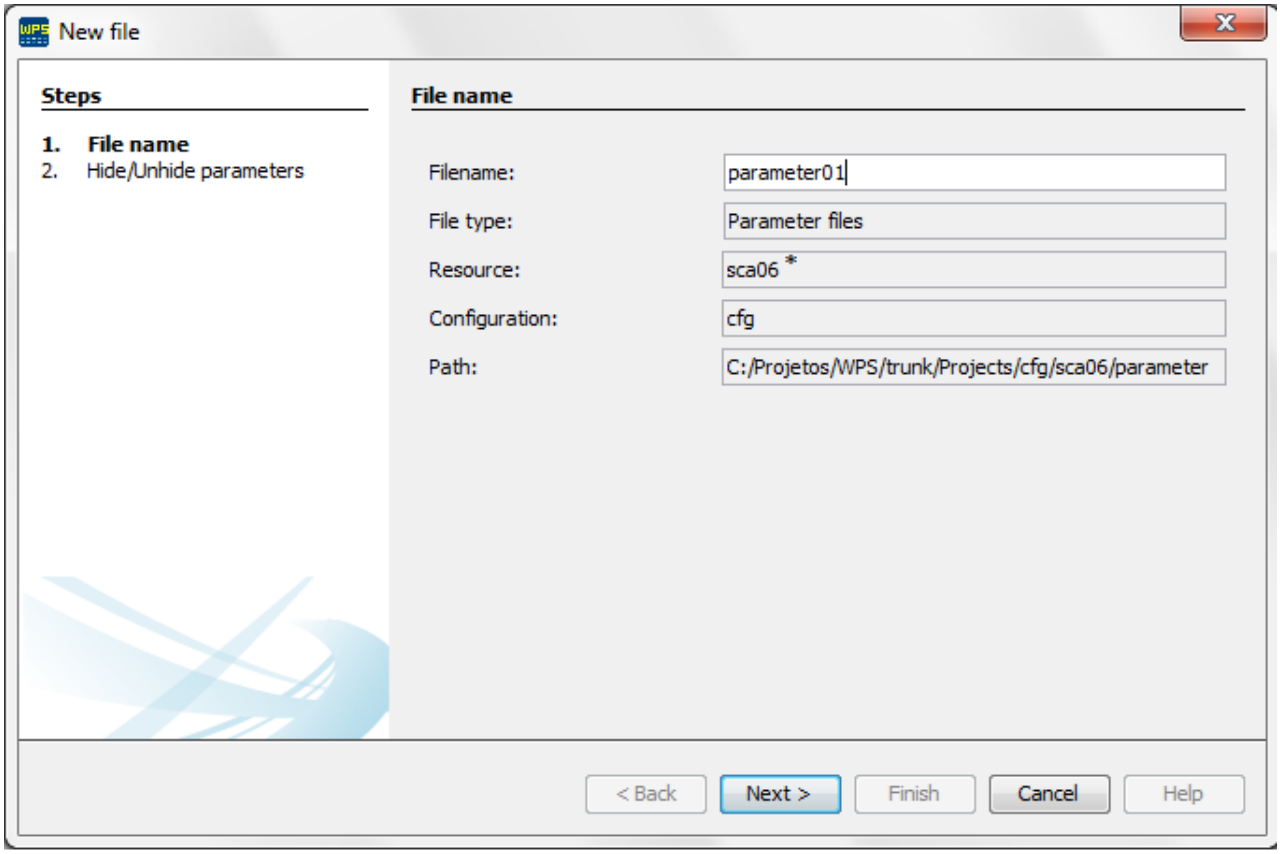
11.7.2.2 Configuration

Below is the list of the required steps to create a parameter file.

1. Create a new parameter file.

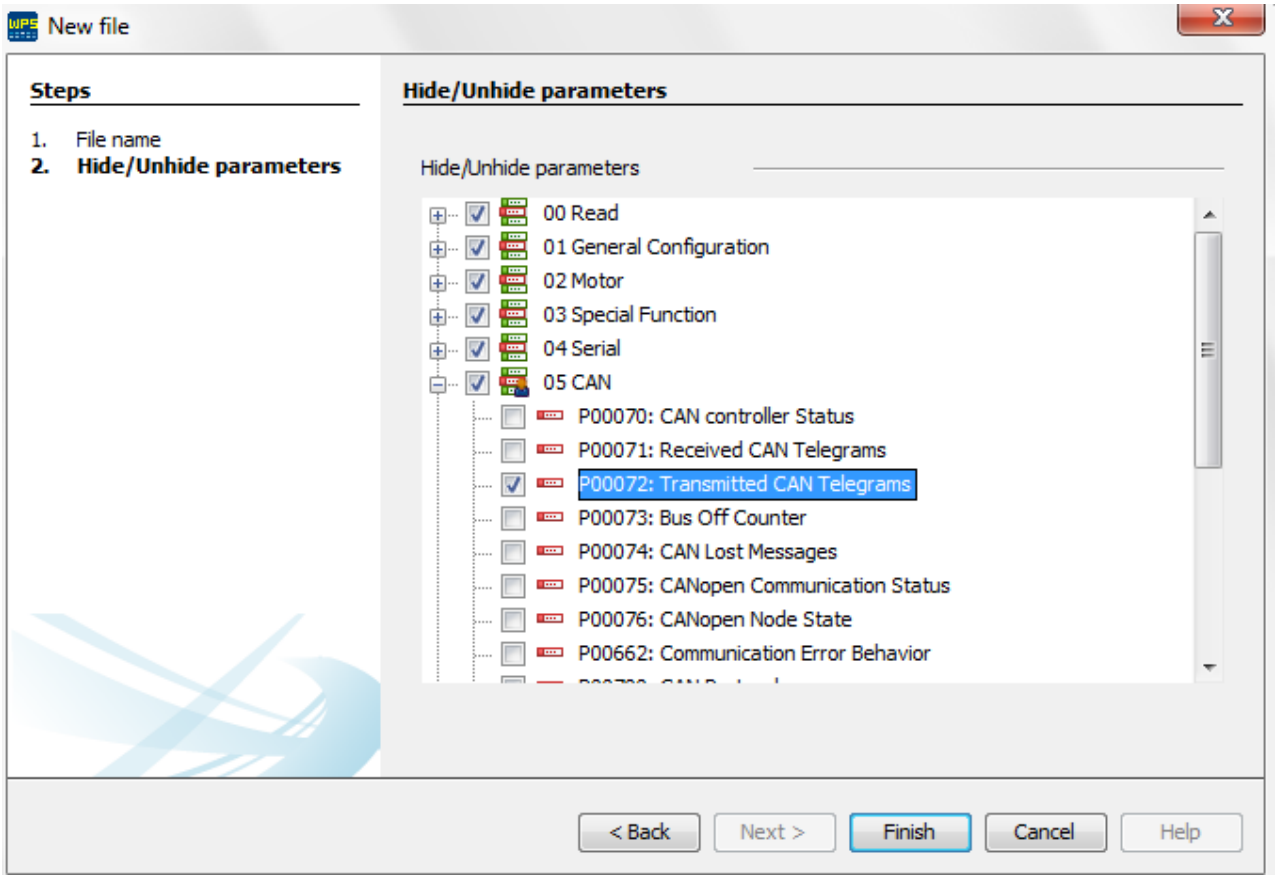


2. Define a name for the parameter file

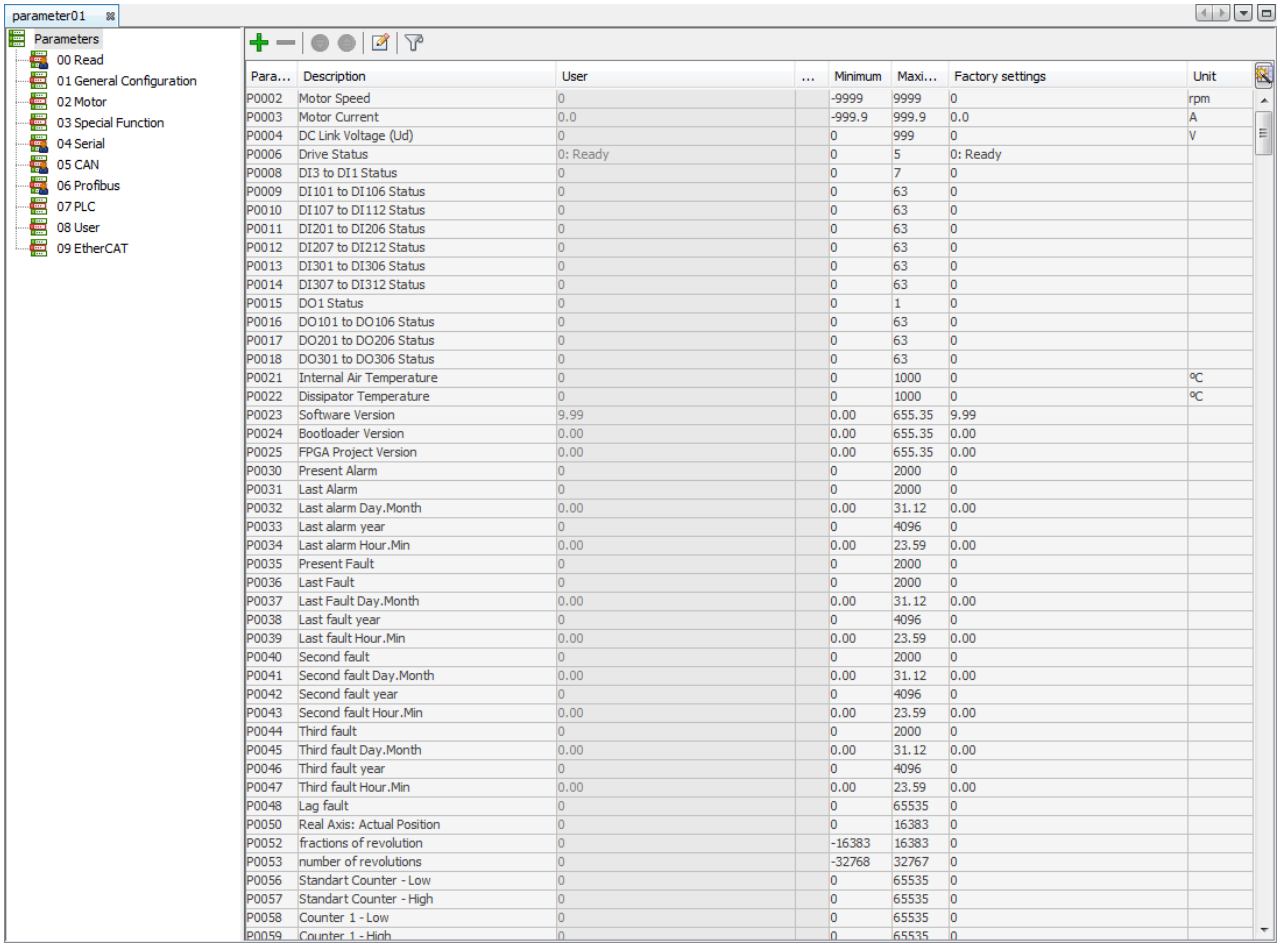


*** Resource: Resource01, SCA06, CFW300, etc.**

3. Configure which parameters you wish to view in your parameter table



4. After performing the steps above, the parameter file will be created and the equipment can be parameterized.

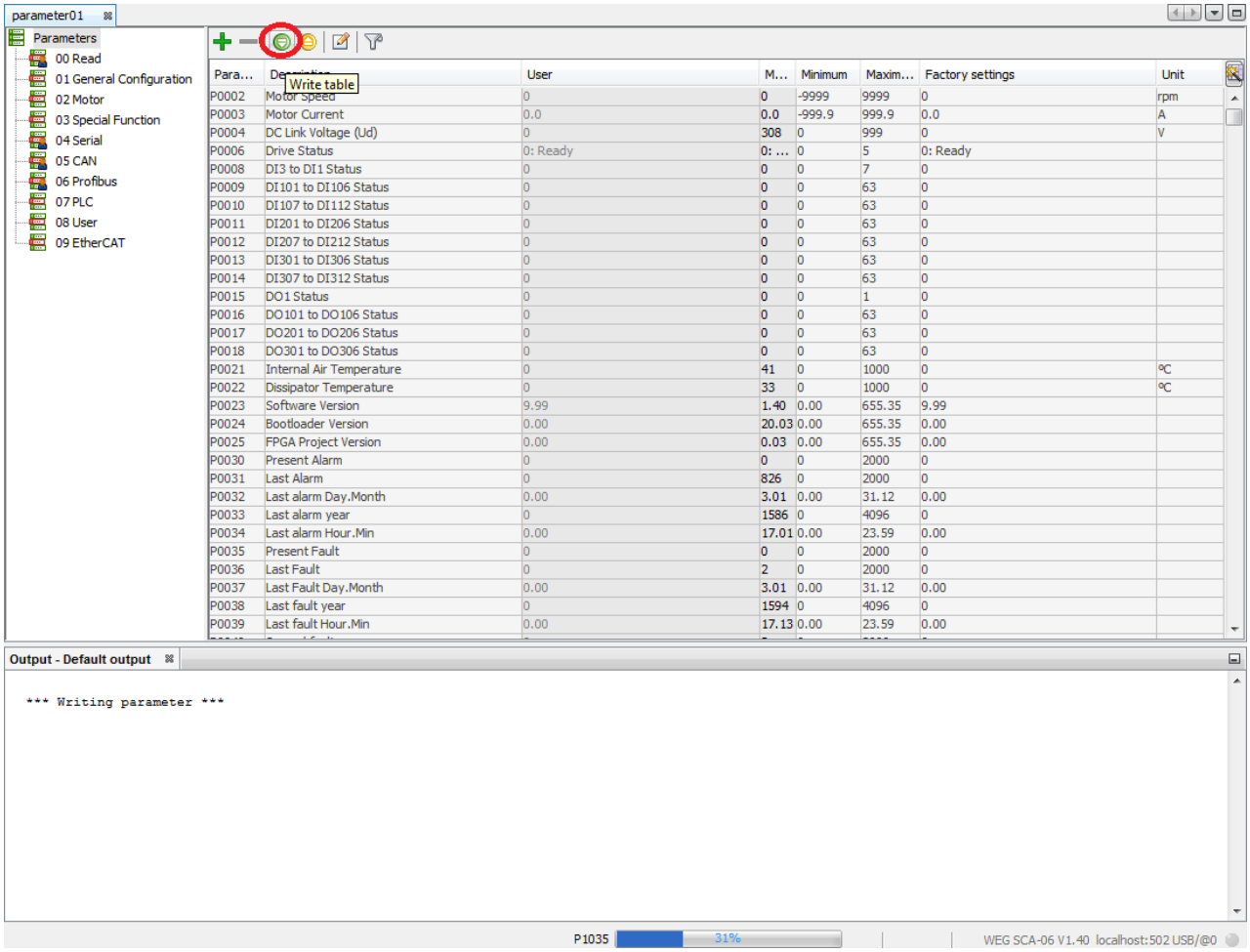


Para...	Description	User	...	Minimum	Maxi...	Factory settings	Unit
P0002	Motor Speed	0		-9999	9999	0	rpm
P0003	Motor Current	0.0		-999.9	999.9	0.0	A
P0004	DC Link Voltage (Ud)	0		0	999	0	V
P0006	Drive Status	0: Ready		0	5	0: Ready	
P0008	DI3 to DI1 Status	0		0	7	0	
P0009	DI101 to DI106 Status	0		0	63	0	
P0010	DI107 to DI112 Status	0		0	63	0	
P0011	DI201 to DI206 Status	0		0	63	0	
P0012	DI207 to DI212 Status	0		0	63	0	
P0013	DI301 to DI306 Status	0		0	63	0	
P0014	DI307 to DI312 Status	0		0	63	0	
P0015	DO1 Status	0		0	1	0	
P0016	DO101 to DO106 Status	0		0	63	0	
P0017	DO201 to DO206 Status	0		0	63	0	
P0018	DO301 to DO306 Status	0		0	63	0	
P0021	Internal Air Temperature	0		0	1000	0	ºC
P0022	Dissipator Temperature	0		0	1000	0	ºC
P0023	Software Version	9.99		0.00	655.35	9.99	
P0024	Bootloader Version	0.00		0.00	655.35	0.00	
P0025	FPGA Project Version	0.00		0.00	655.35	0.00	
P0030	Present Alarm	0		0	2000	0	
P0031	Last Alarm	0		0	2000	0	
P0032	Last alarm Day.Month	0.00		0.00	31.12	0.00	
P0033	Last alarm year	0		0	4096	0	
P0034	Last alarm Hour.Min	0.00		0.00	23.59	0.00	
P0035	Present Fault	0		0	2000	0	
P0036	Last Fault	0		0	2000	0	
P0037	Last Fault Day.Month	0.00		0.00	31.12	0.00	
P0038	Last fault year	0		0	4096	0	
P0039	Last fault Hour.Min	0.00		0.00	23.59	0.00	
P0040	Second fault	0		0	2000	0	
P0041	Second fault Day.Month	0.00		0.00	31.12	0.00	
P0042	Second fault year	0		0	4096	0	
P0043	Second fault Hour.Min	0.00		0.00	23.59	0.00	
P0044	Third fault	0		0	2000	0	
P0045	Third fault Day.Month	0.00		0.00	31.12	0.00	
P0046	Third fault year	0		0	4096	0	
P0047	Third fault Hour.Min	0.00		0.00	23.59	0.00	
P0048	Lag fault	0		0	65535	0	
P0050	Real Axis: Actual Position	0		0	16383	0	
P0052	fractions of revolution	0		-16383	16383	0	
P0053	number of revolutions	0		-32768	32767	0	
P0056	Standart Counter - Low	0		0	65535	0	
P0057	Standart Counter - High	0		0	65535	0	
P0058	Counter 1 - Low	0		0	65535	0	
P0059	Counter 1 - High	0		0	65535	0	

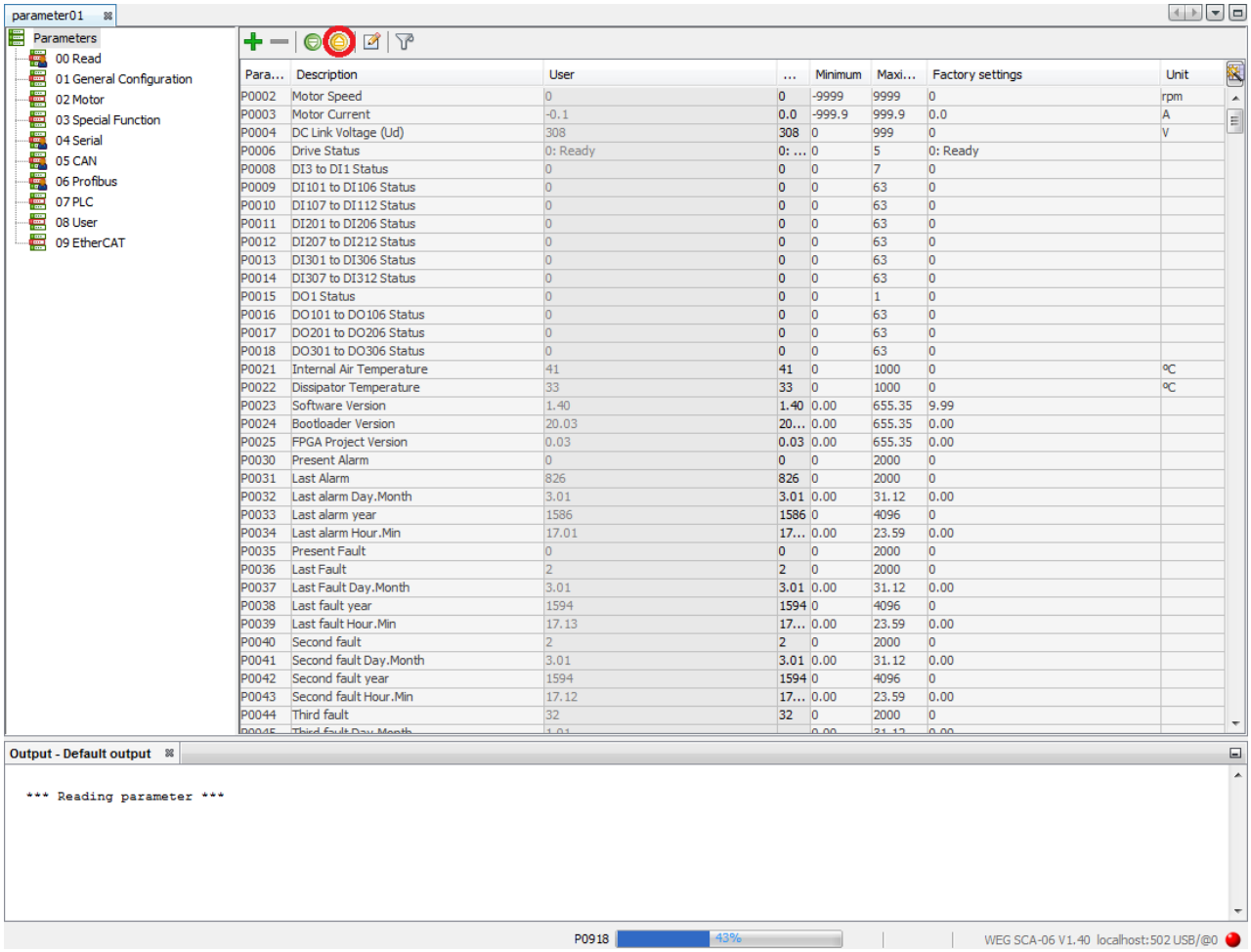
11.7.2.3 Read and Write of Parameters

There are 3 (three) ways to do the reading and writing of the parameters: by means of table, selection and group.

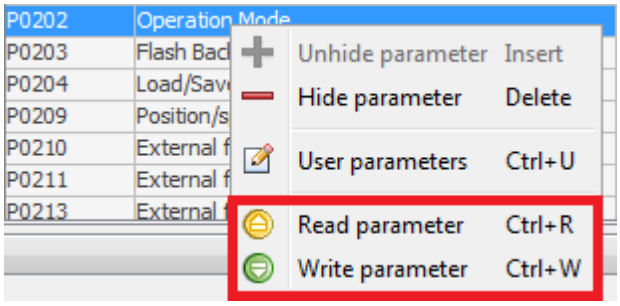
1. **Table writing**. The table writing command will send all visible parameters on the equipment screen. If and error occurs during the sending of some specific parameter, a message will be shown on the output window informing the error. It is important to notice that only visible parameters will be sent; therefore, it is necessary attention to which node of the group of parameters tree you are viewing. Example: If you wish to write all of them without filtering per group, just select the tree root.



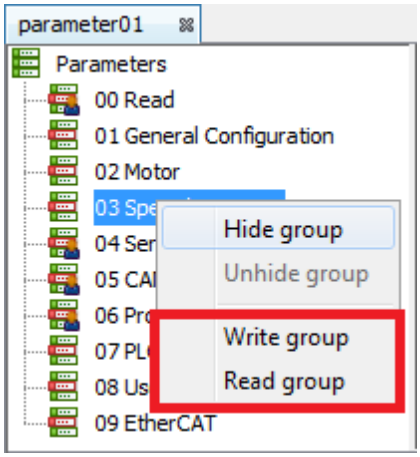
2. **Table reading**. The table reading command will read all the parameters of the equipment. If a error occurs during the reading of some specific parameter, a message will be shown on the output window informing the error. It is important to notice that only visible parameters will be read; therefore, it is necessary attention to which node of the group of parameters tree you are viewing. Example: If you wish to read all of them without filtering per group, just select the tree root.



3. **Reading/writing of specific parameters**. In order to read/write one or more specific parameters, just select them on the table, right click and choose the desired option: read or write parameter.



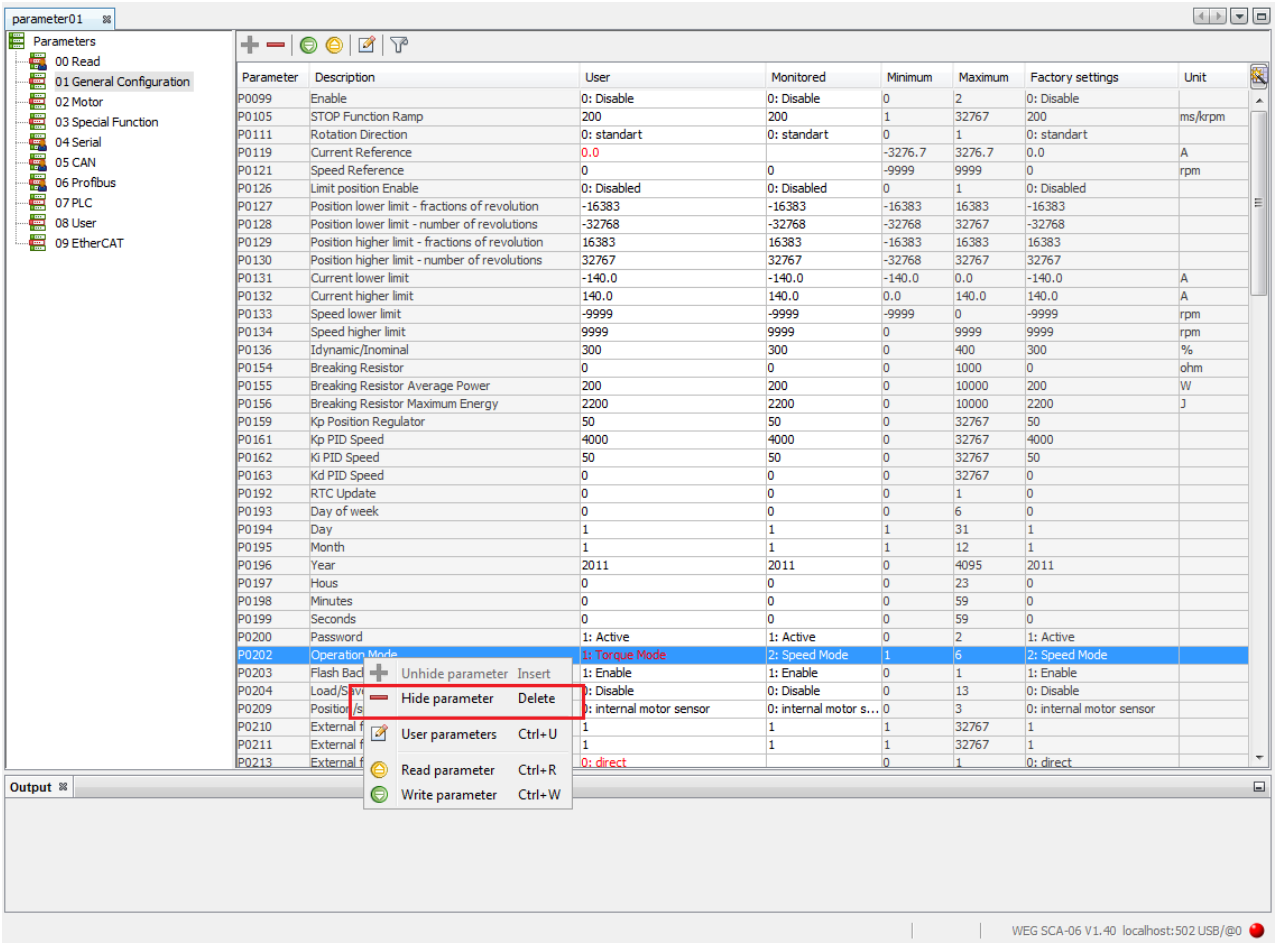
4. **Reading/writing of group of parameters**. In order to read/write only one group of parameters, just select it on the group tree, right click and choose the desired option: read or write group.



11.7.2.4 Hide/Unhide Parameters and Group of Parameters

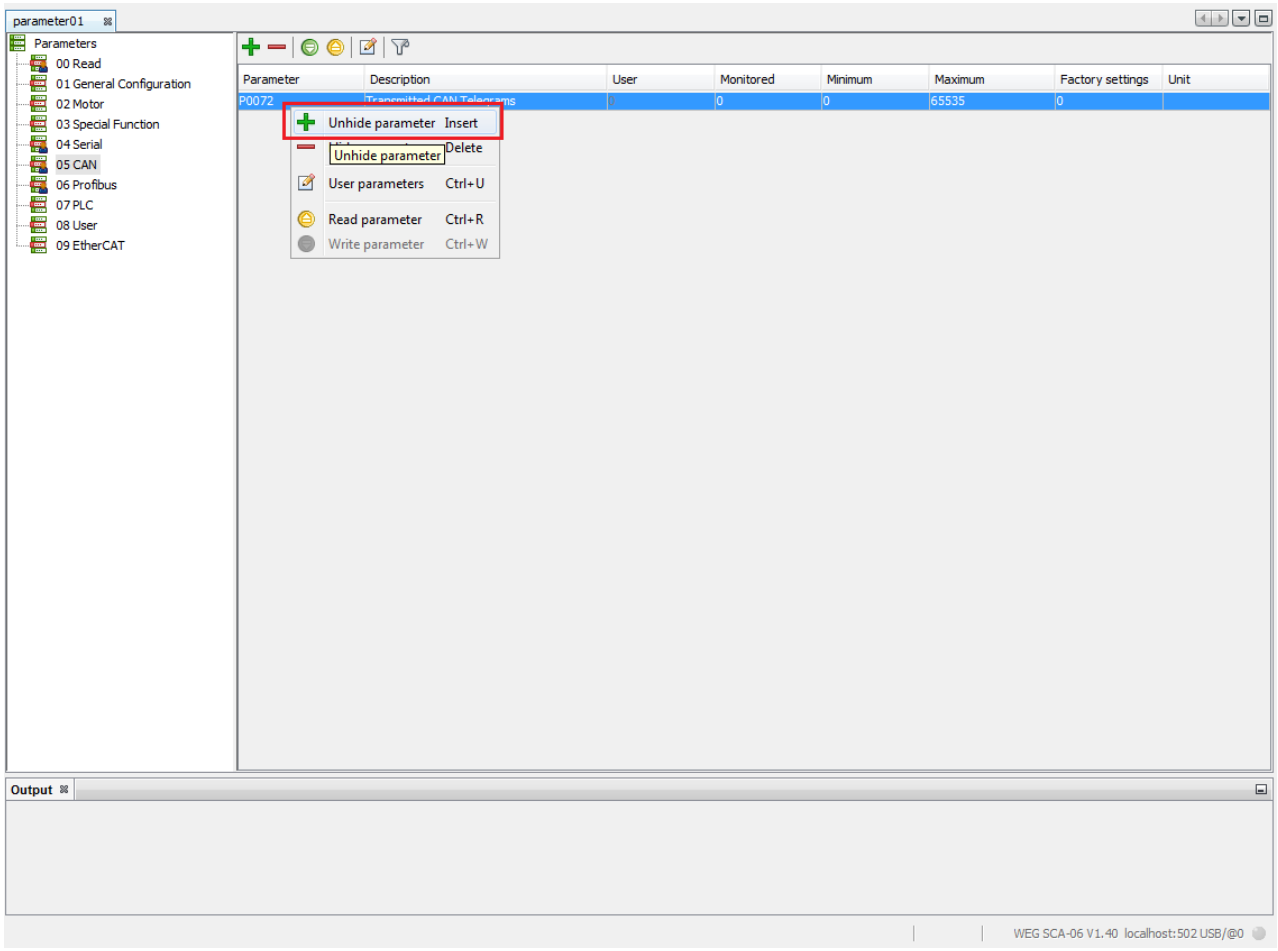
The parameter can be hidden/unhidden in two ways: individually or in group.

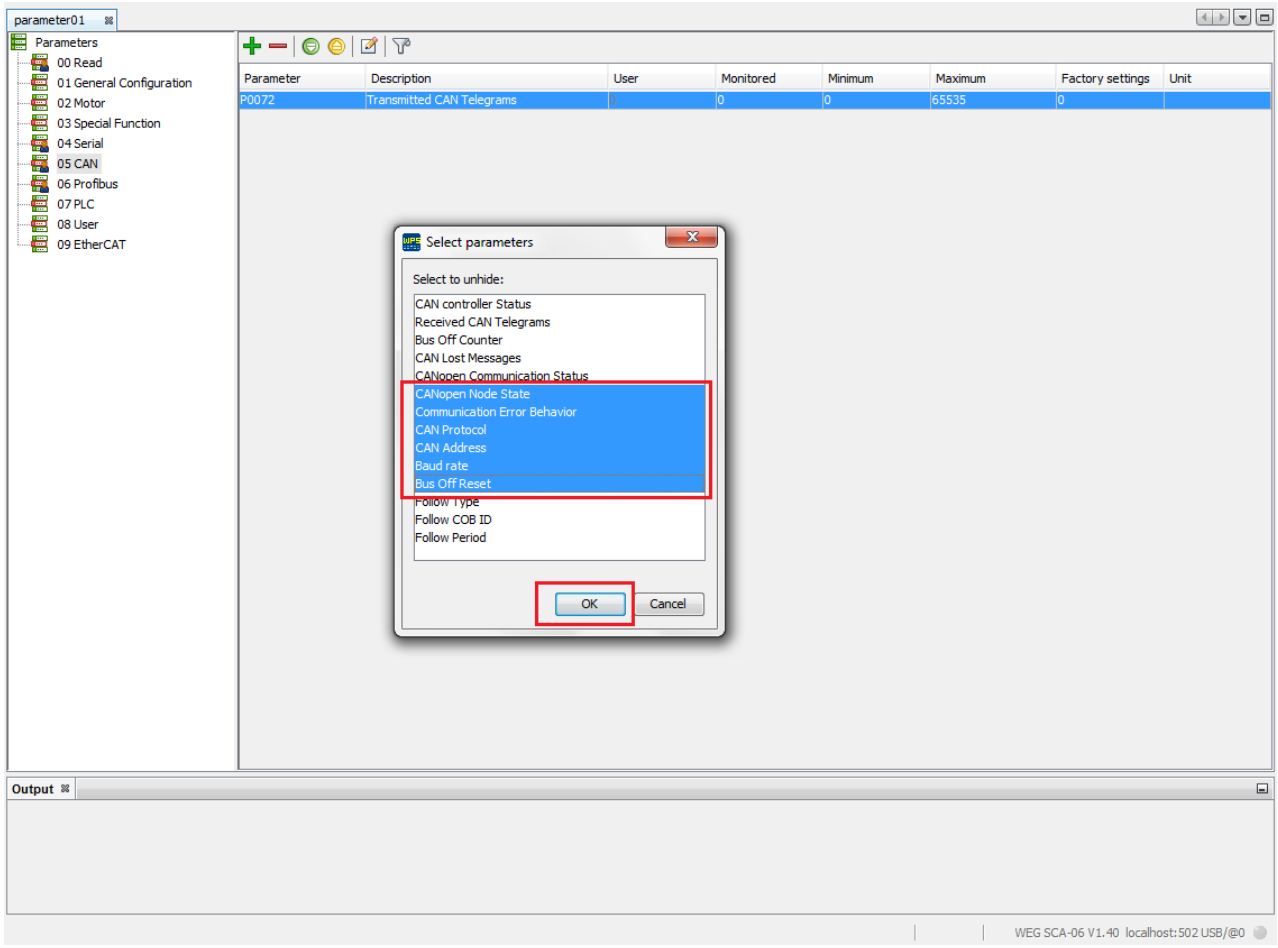
1. **Hide parameters**. In order to hide a parameter individually, just right click on the desired parameters and select the **Hide Parameter** option. You can also press the **Delete** key.

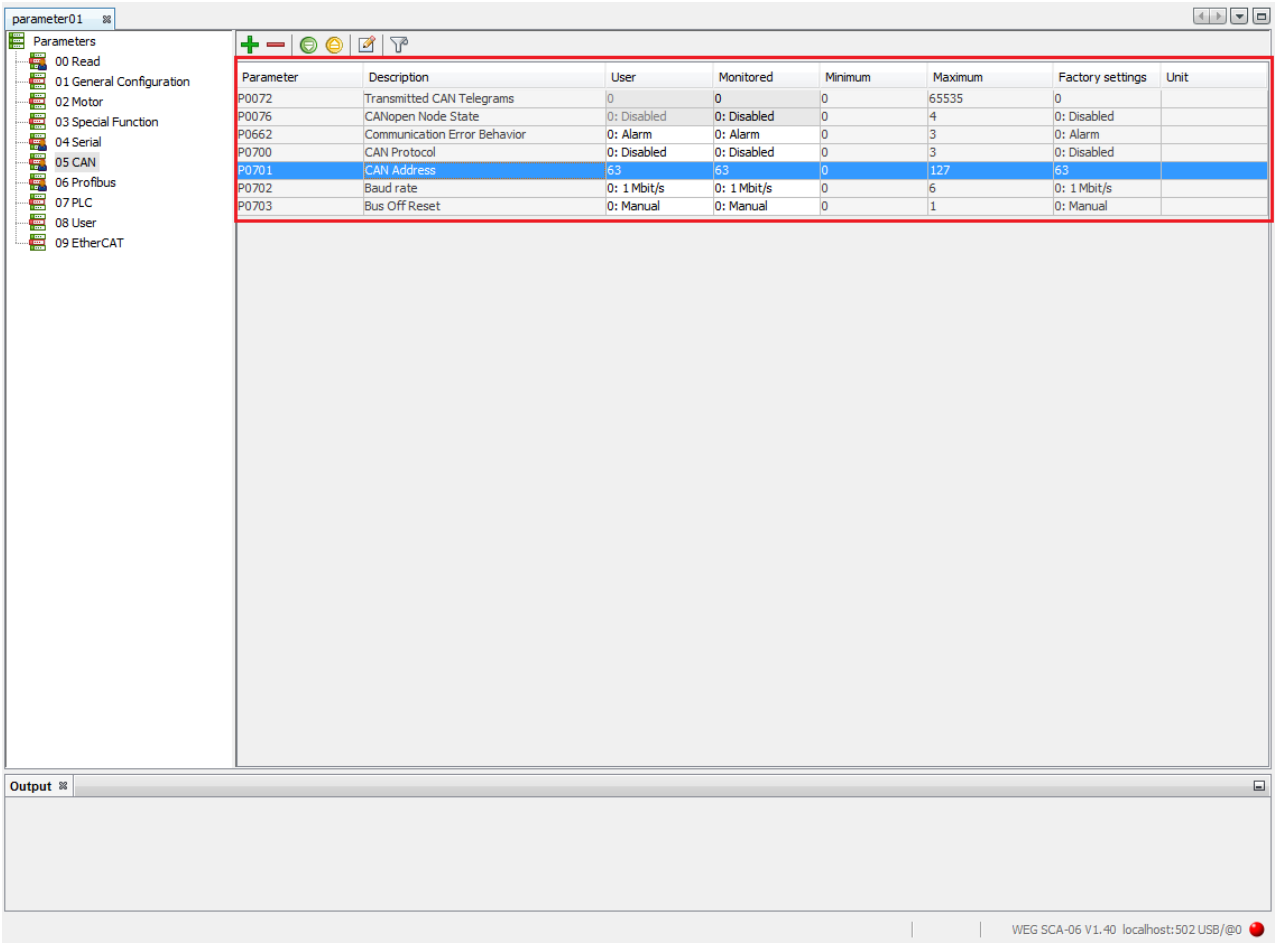


2. **Unhide Parameters**. In order to show hidden parameters, right click and choose the **Unhide Parameters**

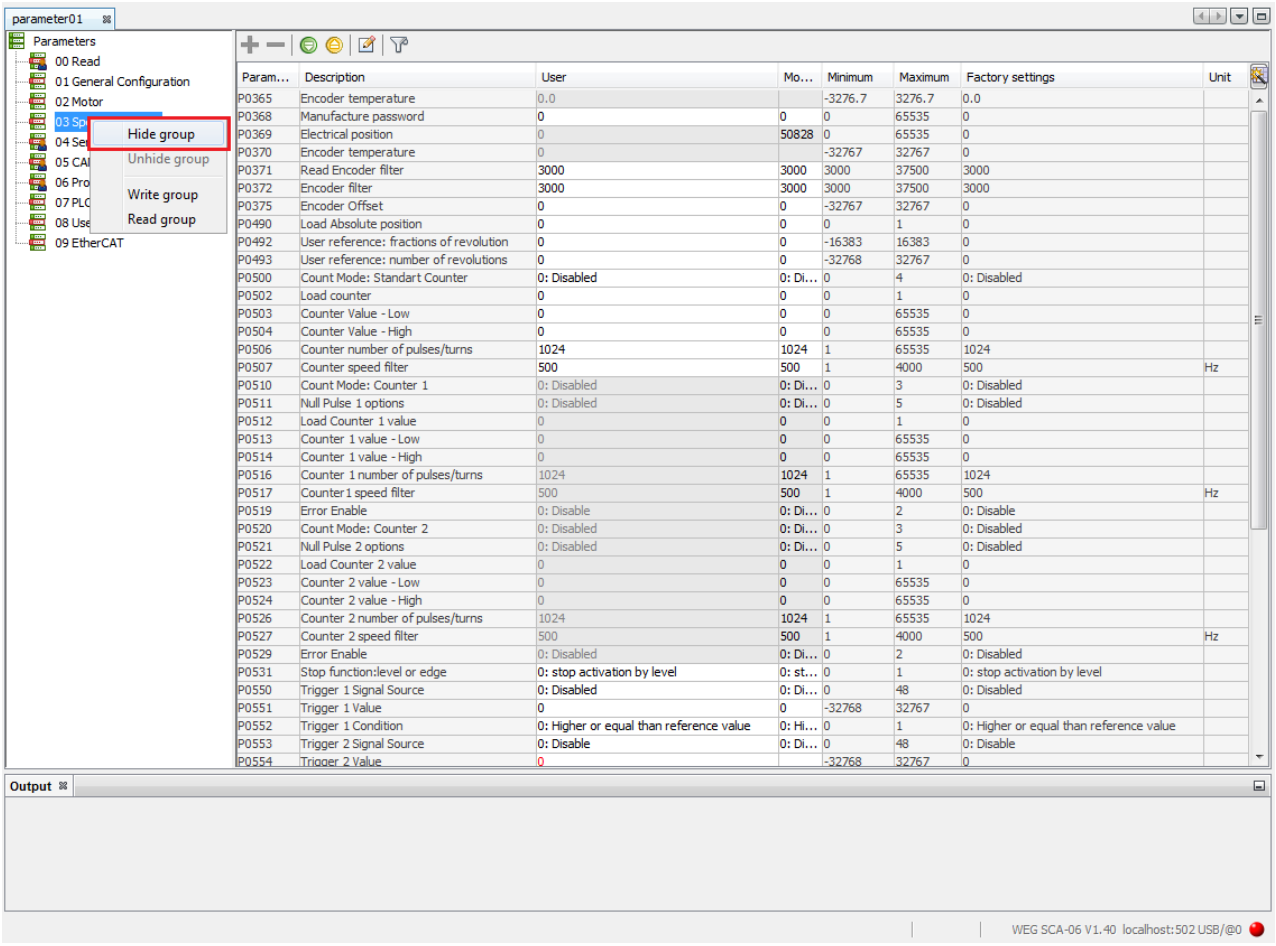
or press the **Insert** key. Then, a window will open and show the hidden parameters. Now, you just have to select the desired parameters and confirm.
Note: The parameters shown on this new window are only those which belong to the current filter according to the selection on the parameter group tree. In the figures below, the CAN group is selected; that means that only the hidden parameters of this group will be shown.

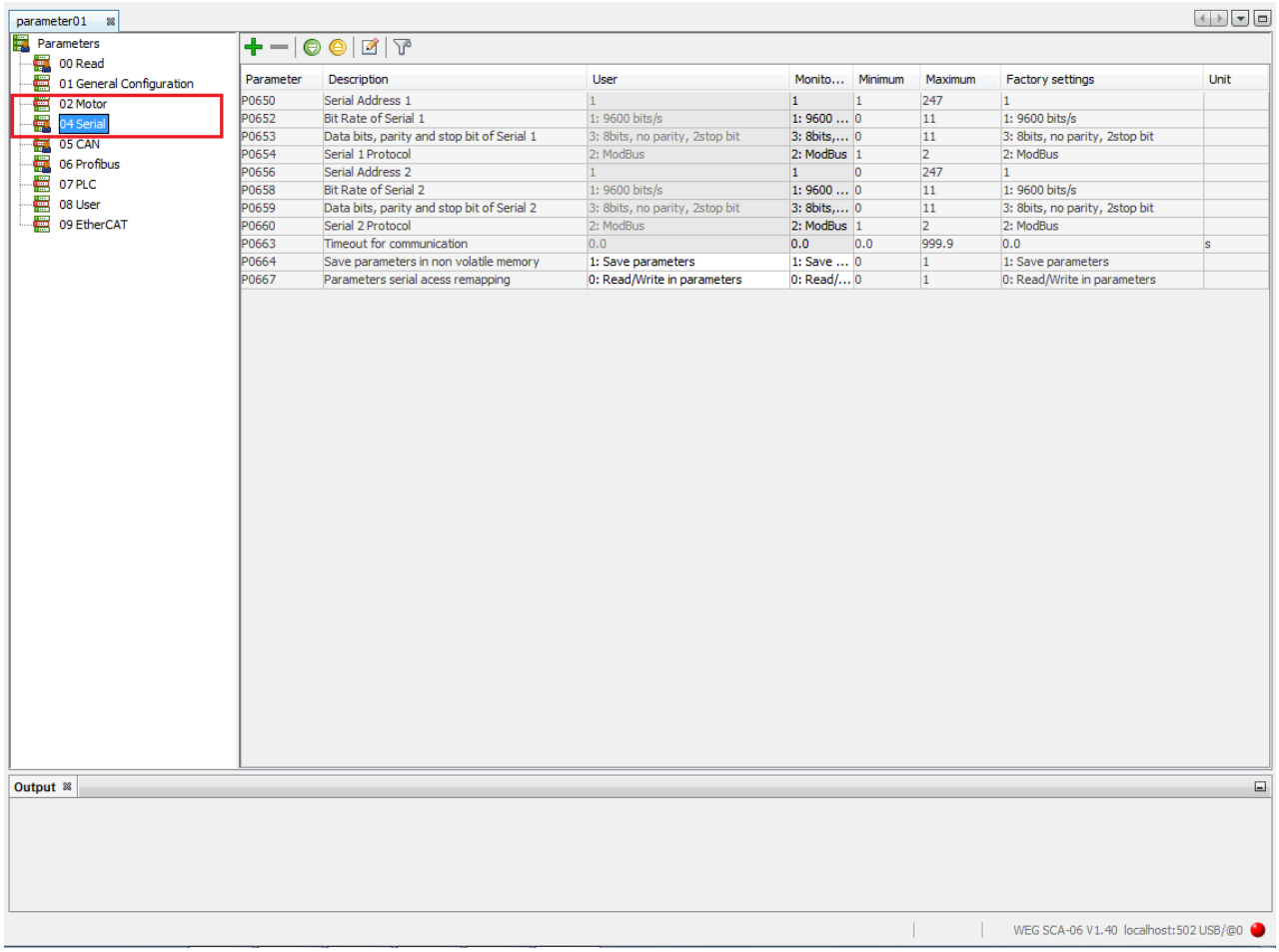



parameter01
Parameters
00 Read
01 General Configuration
02 Motor
03 Special Function
04 Serial
05 CAN
06 Profibus
07 PLC
08 User
09 EtherCAT
Parameter
Description
User
Monitored
Minimum
Maximum
Factory settings
Unit
P0072
Transmitted CAN Telegrams
0
0
65535
0
Select parameters
Select to unhide:
CAN controller Status
Received CAN Telegrams
Bus Off Counter
CAN Lost Messages
CANopen Communication Status
CANopen Node State
Communication Error Behavior
CAN Protocol
CAN Address
Baud rate
Bus Off Reset
Follow Type
Follow COB ID
Follow Period
OK
Cancel
Output
WEG SCA-06 V1.40 localhost:502 USB/@0

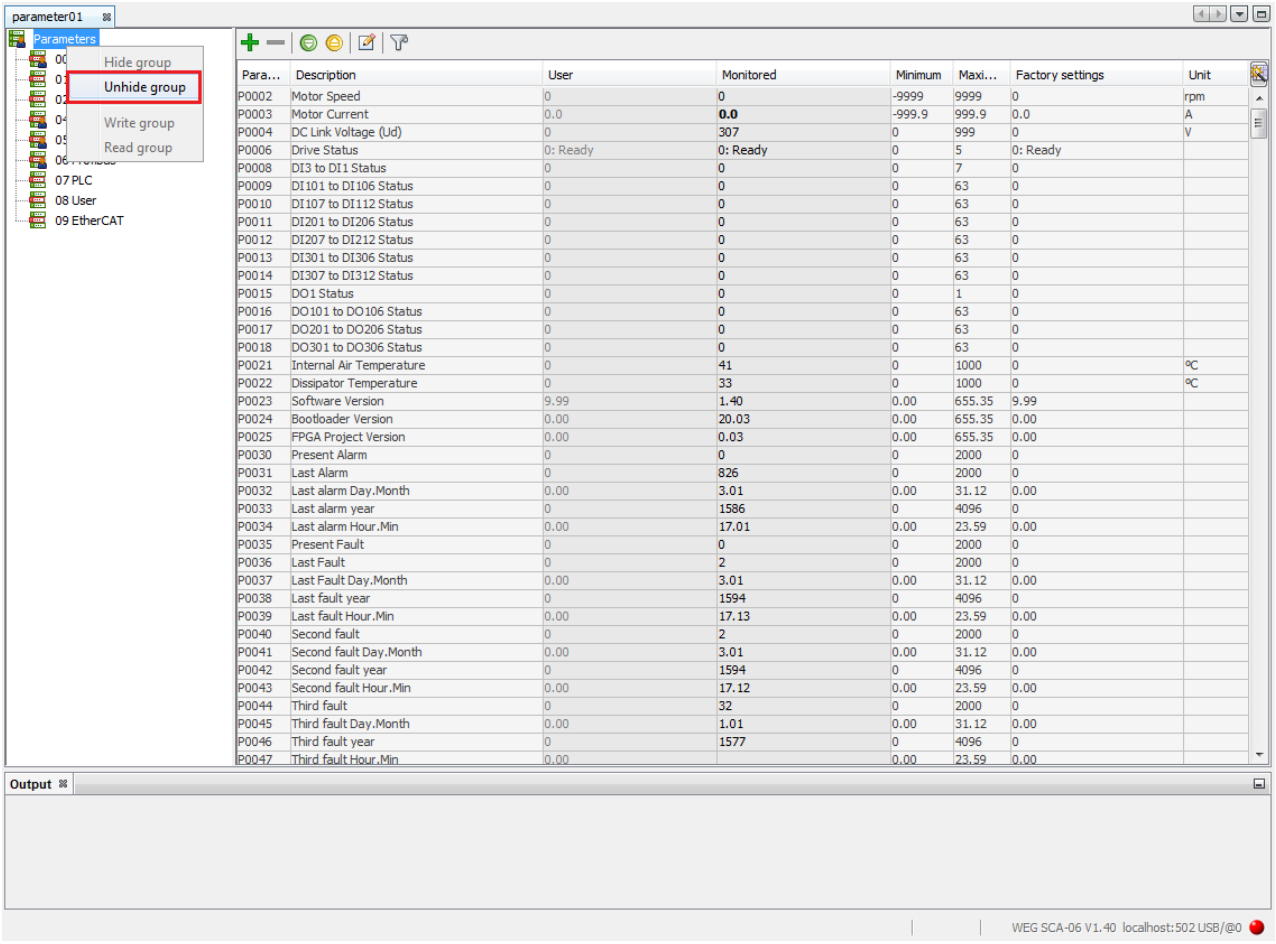


3. **Hide Group of Parameters**. In order to hide a group of parameters, just select the group on the tree and use the **Hide Group** option.

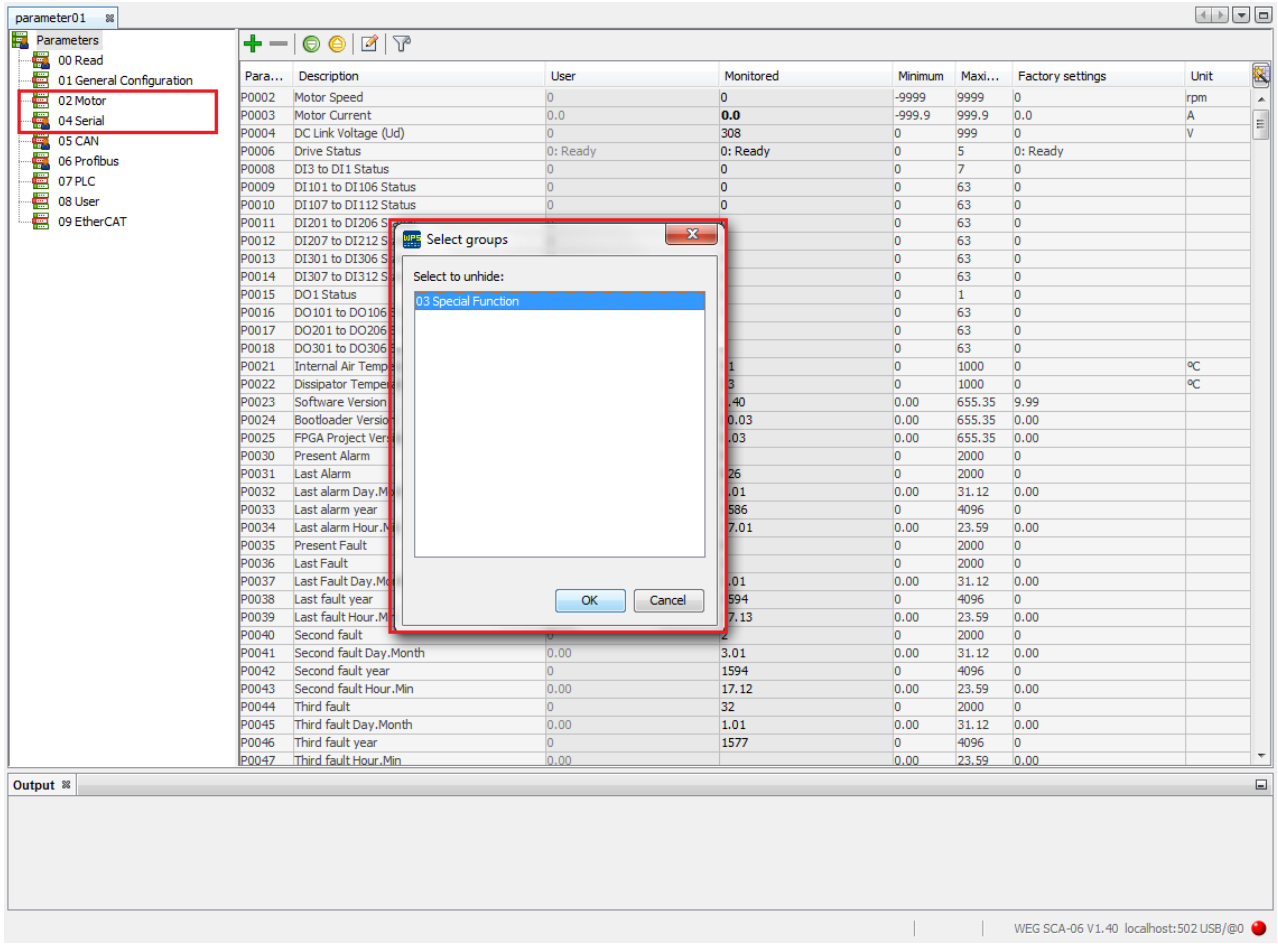

parameter01
Parameters
00 Read
01 General Configuration
02 Motor
03 Sp
04 Se
05 CA
06 Pro
07 PLC
08 Use
09 EtherCAT
Hide group
Unhide group
Write group
Read group
Param...
Description
User
Mo...
Minimum
Maximum
Factory settings
Unit
P0365 Encoder temperature 0.0 -3276.7 3276.7 0.0
P0368 Manufacture password 0 0 0 65535 0
P0369 Electrical position 0 50828 0 65535 0
P0370 Encoder temperature 0 -32767 32767 0
P0371 Read Encoder filter 3000 3000 3000 37500 3000
P0372 Encoder filter 3000 3000 3000 37500 3000
P0375 Encoder Offset 0 0 -32767 32767 0
P0490 Load Absolute position 0 0 0 1 0
P0492 User reference: fractions of revolution 0 0 -16383 16383 0
P0493 User reference: number of revolutions 0 0 -32768 32767 0
P0500 Count Mode: Standart Counter 0: Disabled 0: Di... 0 4 0: Disabled
P0502 Load counter 0 0 0 1 0
P0503 Counter Value - Low 0 0 0 65535 0
P0504 Counter Value - High 0 0 0 65535 0
P0506 Counter number of pulses/turns 1024 1024 1 65535 1024
P0507 Counter speed filter 500 500 1 4000 500 Hz
P0510 Count Mode: Counter 1 0: Disabled 0: Di... 0 3 0: Disabled
P0511 Null Pulse 1 options 0: Disabled 0: Di... 0 5 0: Disabled
P0512 Load Counter 1 value 0 0 0 1 0
P0513 Counter 1 value - Low 0 0 0 65535 0
P0514 Counter 1 value - High 0 0 0 65535 0
P0516 Counter 1 number of pulses/turns 1024 1024 1 65535 1024
P0517 Counter 1 speed filter 500 500 1 4000 500 Hz
P0519 Error Enable 0: Disable 0: Di... 0 2 0: Disable
P0520 Count Mode: Counter 2 0: Disabled 0: Di... 0 3 0: Disabled
P0521 Null Pulse 2 options 0: Disabled 0: Di... 0 5 0: Disabled
P0522 Load Counter 2 value 0 0 0 1 0
P0523 Counter 2 value - Low 0 0 0 65535 0
P0524 Counter 2 value - High 0 0 0 65535 0
P0526 Counter 2 number of pulses/turns 1024 1024 1 65535 1024
P0527 Counter 2 speed filter 500 500 1 4000 500 Hz
P0529 Error Enable 0: Disabled 0: Di... 0 2 0: Disabled
P0531 Stop function:level or edge 0: stop activation by level 0: st... 0 1 0: stop activation by level
P0550 Trigger 1 Signal Source 0: Disabled 0: Di... 0 48 0: Disabled
P0551 Trigger 1 Value 0 0 -32768 32767 0
P0552 Trigger 1 Condition 0: Higher or equal than reference value 0: Hi... 0 1 0: Higher or equal than reference value
P0553 Trigger 2 Signal Source 0: Disable 0: Di... 0 48 0: Disable
P0554 Trigger 2 Value 0 -32768 32767 0
Output
WEG SCA-06 V1.40 localhost:502 USB/@0



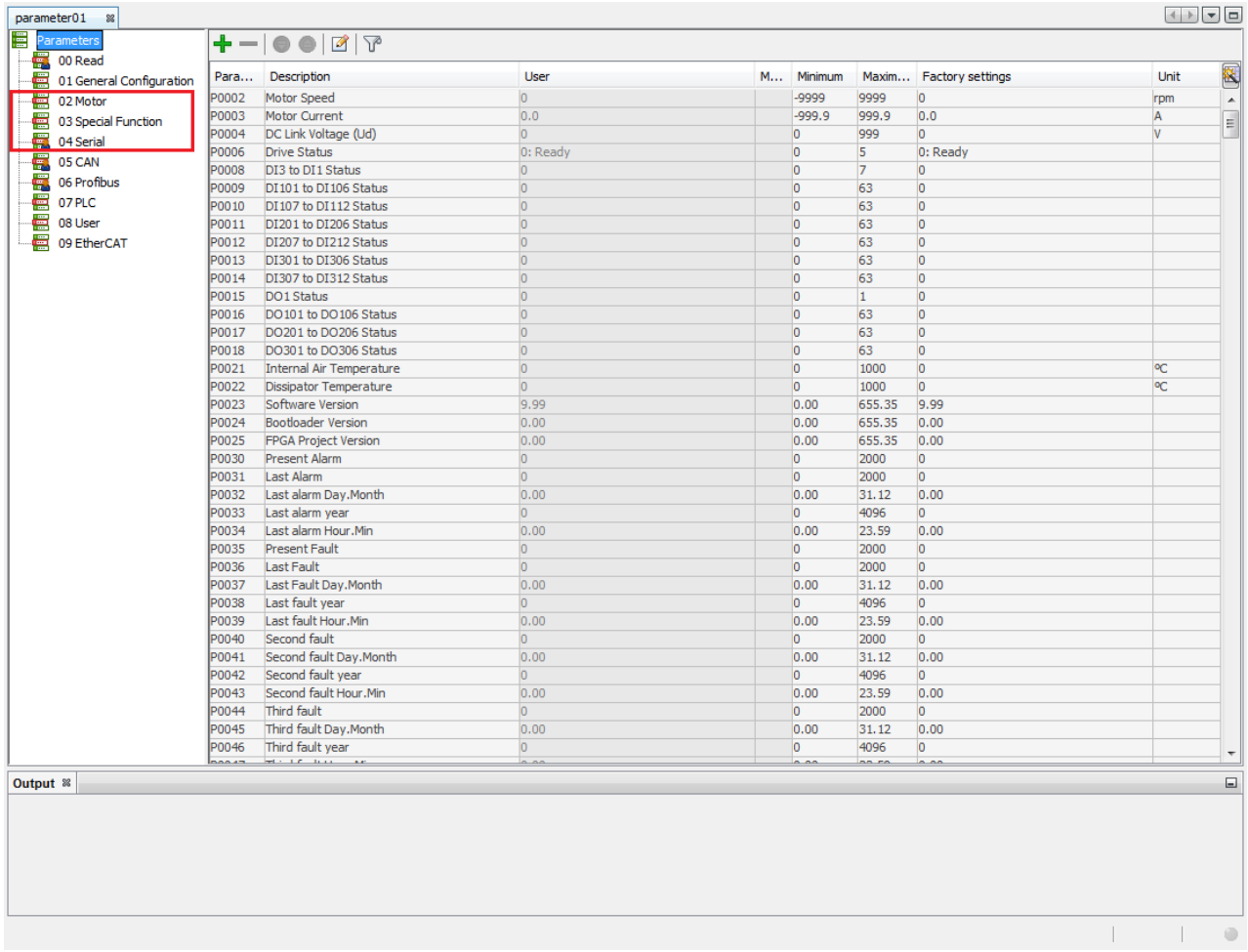
4. **Unhide Group of Parameters**. In order to show a hidden group of parameters, just select the root of the group tree and select the **Unhide Group** option. A window will open showing the groups that are hidden; then just select the group you wish to unhide.



Para...	Description	User	Monitored	Minimum	Maxi...	Factory settings	Unit
P0002	Motor Speed	0	0	-9999	9999	0	rpm
P0003	Motor Current	0.0	**0.0**	-999.9	999.9	0.0	A
P0004	DC Link Voltage (Ud)	0	307	0	999	0	V
P0006	Drive Status	0: Ready	0: Ready	0	5	0: Ready	
P0008	DI3 to DI1 Status	0	0	0	7	0	
P0009	DI101 to DI106 Status	0	0	0	63	0	
P0010	DI107 to DI112 Status	0	0	0	63	0	
P0011	DI201 to DI206 Status	0	0	0	63	0	
P0012	DI207 to DI212 Status	0	0	0	63	0	
P0013	DI301 to DI306 Status	0	0	0	63	0	
P0014	DI307 to DI312 Status	0	0	0	63	0	
P0015	DO1 Status	0	0	0	1	0	
P0016	DO101 to DO106 Status	0	0	0	63	0	
P0017	DO201 to DO206 Status	0	0	0	63	0	
P0018	DO301 to DO306 Status	0	0	0	63	0	
P0021	Internal Air Temperature	0	41	0	1000	0	°C
P0022	Dissipator Temperature	0	33	0	1000	0	°C
P0023	Software Version	9.99	1.40	0.00	655.35	9.99	
P0024	Bootloader Version	0.00	20.03	0.00	655.35	0.00	
P0025	FPGA Project Version	0.00	0.03	0.00	655.35	0.00	
P0030	Present Alarm	0	0	0	2000	0	
P0031	Last Alarm	0	826	0	2000	0	
P0032	Last alarm Day.Month	0.00	3.01	0.00	31.12	0.00	
P0033	Last alarm year	0	1586	0	4096	0	
P0034	Last alarm Hour.Min	0.00	17.01	0.00	23.59	0.00	
P0035	Present Fault	0	0	0	2000	0	
P0036	Last Fault	0	2	0	2000	0	
P0037	Last Fault Day.Month	0.00	3.01	0.00	31.12	0.00	
P0038	Last fault year	0	1594	0	4096	0	
P0039	Last fault Hour.Min	0.00	17.13	0.00	23.59	0.00	
P0040	Second fault	0	2	0	2000	0	
P0041	Second fault Day.Month	0.00	3.01	0.00	31.12	0.00	
P0042	Second fault year	0	1594	0	4096	0	
P0043	Second fault Hour.Min	0.00	17.12	0.00	23.59	0.00	
P0044	Third fault	0	32	0	2000	0	
P0045	Third fault Day.Month	0.00	1.01	0.00	31.12	0.00	
P0046	Third fault year	0	1577	0	4096	0	
P0047	Third fault Hour.Min	0.00		0.00	23.59	0.00	

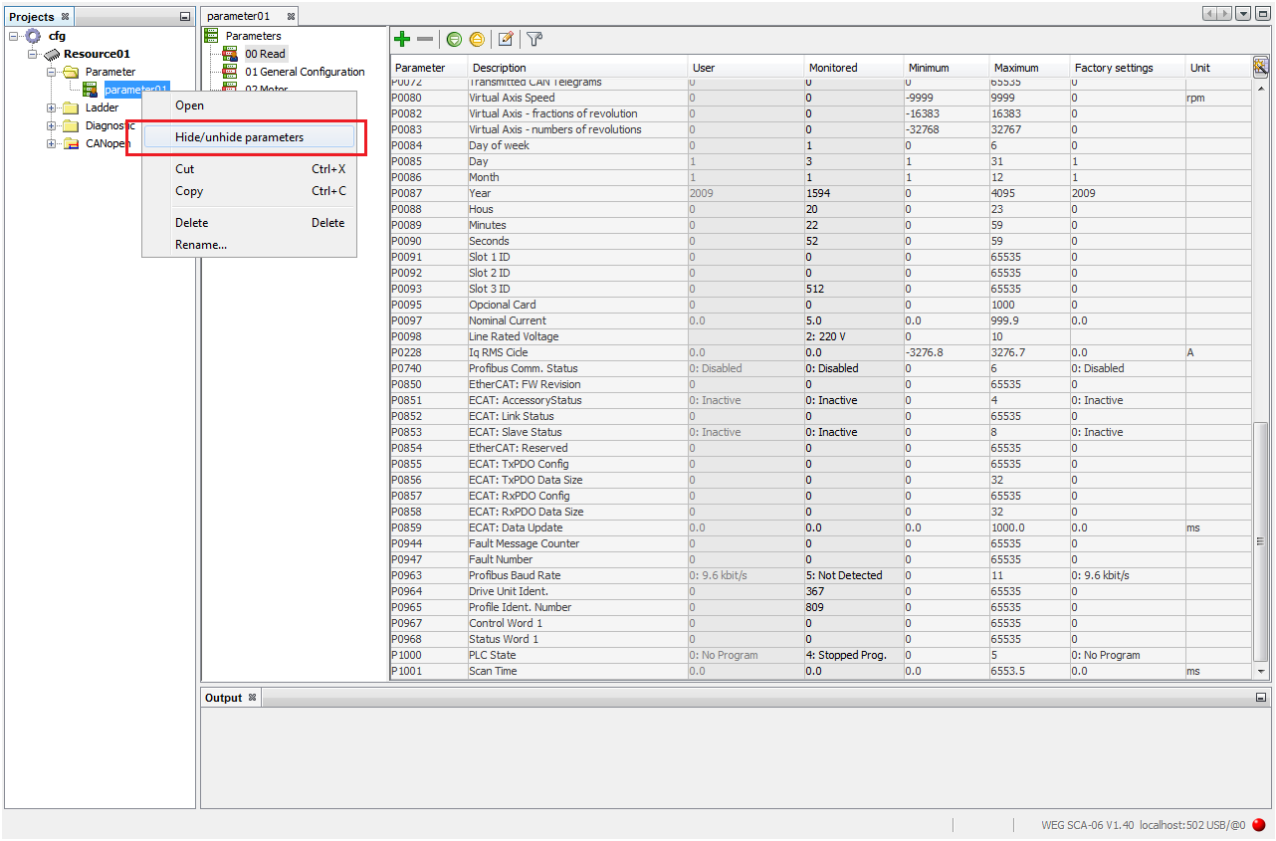


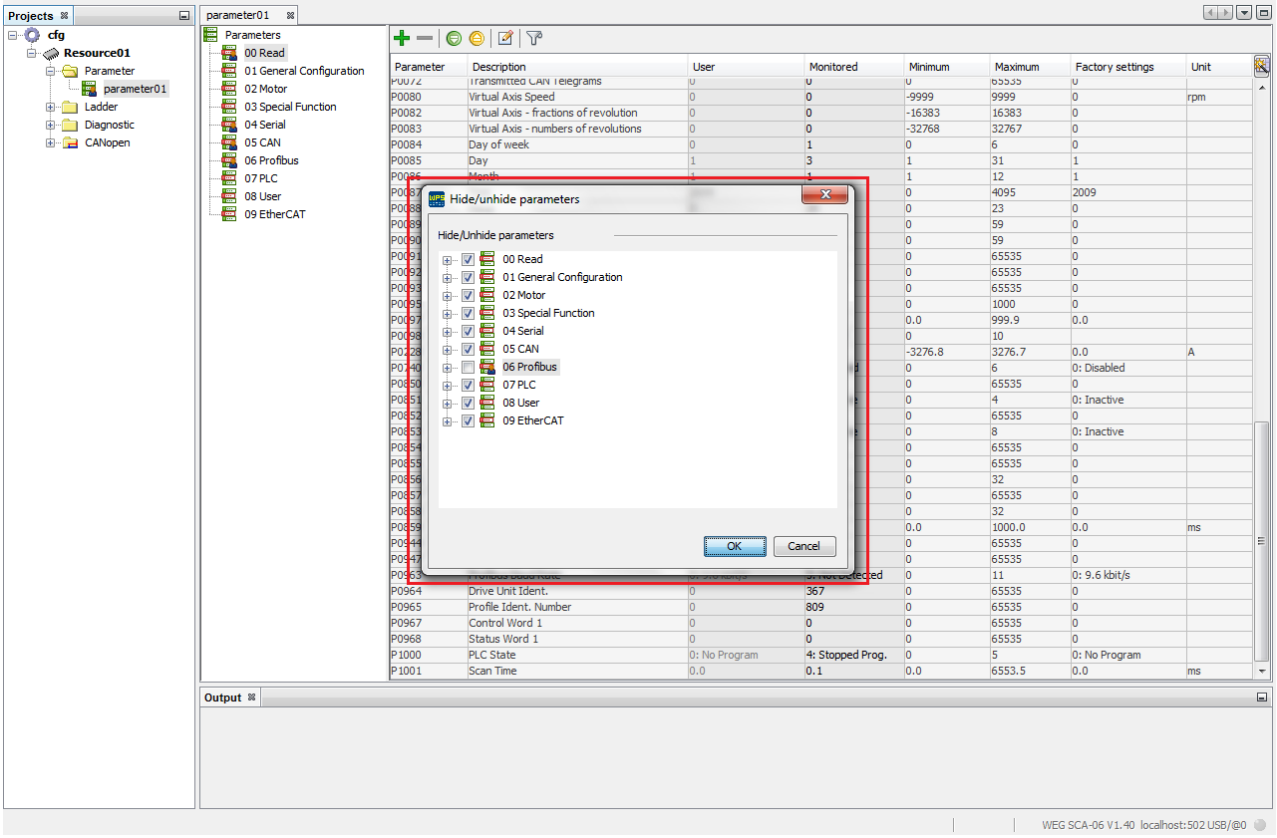
Para...	Description	User	Monitored	Minimum	Maxi...	Factory settings	Unit
P0002	Motor Speed	0	0	-9999	9999	0	rpm
P0003	Motor Current	0.0	**0.0**	-999.9	999.9	0.0	A
P0004	DC Link Voltage (Ud)	0	308	0	999	0	V
P0006	Drive Status	0: Ready	0: Ready	0	5	0: Ready	
P0008	DI3 to DI1 Status	0	0	0	7	0	
P0009	DI101 to DI106 Status	0	0	0	63	0	
P0010	DI107 to DI112 Status	0	0	0	63	0	
P0011	DI201 to DI206 S			0	63	0	
P0012	DI207 to DI212 S			0	63	0	
P0013	DI301 to DI306 S			0	63	0	
P0014	DI307 to DI312 S			0	63	0	
P0015	DO1 Status			0	1	0	
P0016	DO101 to DO106			0	63	0	
P0017	DO201 to DO206			0	63	0	
P0018	DO301 to DO306			0	63	0	
P0021	Internal Air Temp		1	0	1000	0	°C
P0022	Dissipator Temper		3	0	1000	0	°C
P0023	Software Version		.40	0.00	655.35	9.99	
P0024	Bootloader Versio		0.03	0.00	655.35	0.00	
P0025	FPGA Project Ver		.03	0.00	655.35	0.00	
P0030	Present Alarm			0	2000	0	
P0031	Last Alarm		26	0	2000	0	
P0032	Last alarm Day.M		.01	0.00	31.12	0.00	
P0033	Last alarm year		586	0	4096	0	
P0034	Last alarm Hour.M		7.01	0.00	23.59	0.00	
P0035	Present Fault			0	2000	0	
P0036	Last Fault			0	2000	0	
P0037	Last Fault Day.M		.01	0.00	31.12	0.00	
P0038	Last fault year		594	0	4096	0	
P0039	Last fault Hour.M		7.13	0.00	23.59	0.00	
P0040	Second fault		2	0	2000	0	
P0041	Second fault Day.Month	0.00	3.01	0.00	31.12	0.00	
P0042	Second fault year	0	1594	0	4096	0	
P0043	Second fault Hour.Min	0.00	17.12	0.00	23.59	0.00	
P0044	Third fault	0	32	0	2000	0	
P0045	Third fault Day.Month	0.00	1.01	0.00	31.12	0.00	
P0046	Third fault year	0	1577	0	4096	0	
P0047	Third fault Hour.Min	0.00		0.00	23.59	0.00	



Para...	Description	User	M...	Minimum	Maxim...	Factory settings	Unit
P0002	Motor Speed	0		-9999	9999	0	rpm
P0003	Motor Current	0.0		-999.9	999.9	0.0	A
P0004	DC Link Voltage (Ud)	0		0	999	0	V
P0006	Drive Status	0: Ready		0	5	0: Ready	
P0008	DI3 to DI1 Status	0		0	7	0	
P0009	DI101 to DI106 Status	0		0	63	0	
P0010	DI107 to DI112 Status	0		0	63	0	
P0011	DI201 to DI206 Status	0		0	63	0	
P0012	DI207 to DI212 Status	0		0	63	0	
P0013	DI301 to DI306 Status	0		0	63	0	
P0014	DI307 to DI312 Status	0		0	63	0	
P0015	DO1 Status	0		0	1	0	
P0016	DO101 to DO106 Status	0		0	63	0	
P0017	DO201 to DO206 Status	0		0	63	0	
P0018	DO301 to DO306 Status	0		0	63	0	
P0021	Internal Air Temperature	0		0	1000	0	ºC
P0022	Dissipator Temperature	0		0	1000	0	ºC
P0023	Software Version	9.99		0.00	655.35	9.99	
P0024	Bootloader Version	0.00		0.00	655.35	0.00	
P0025	FPGA Project Version	0.00		0.00	655.35	0.00	
P0030	Present Alarm	0		0	2000	0	
P0031	Last Alarm	0		0	2000	0	
P0032	Last alarm Day.Month	0.00		0.00	31.12	0.00	
P0033	Last alarm year	0		0	4096	0	
P0034	Last alarm Hour.Min	0.00		0.00	23.59	0.00	
P0035	Present Fault	0		0	2000	0	
P0036	Last Fault	0		0	2000	0	
P0037	Last Fault Day.Month	0.00		0.00	31.12	0.00	
P0038	Last fault year	0		0	4096	0	
P0039	Last fault Hour.Min	0.00		0.00	23.59	0.00	
P0040	Second fault	0		0	2000	0	
P0041	Second fault Day.Month	0.00		0.00	31.12	0.00	
P0042	Second fault year	0		0	4096	0	
P0043	Second fault Hour.Min	0.00		0.00	23.59	0.00	
P0044	Third fault	0		0	2000	0	
P0045	Third fault Day.Month	0.00		0.00	31.12	0.00	
P0046	Third fault year	0		0	4096	0	

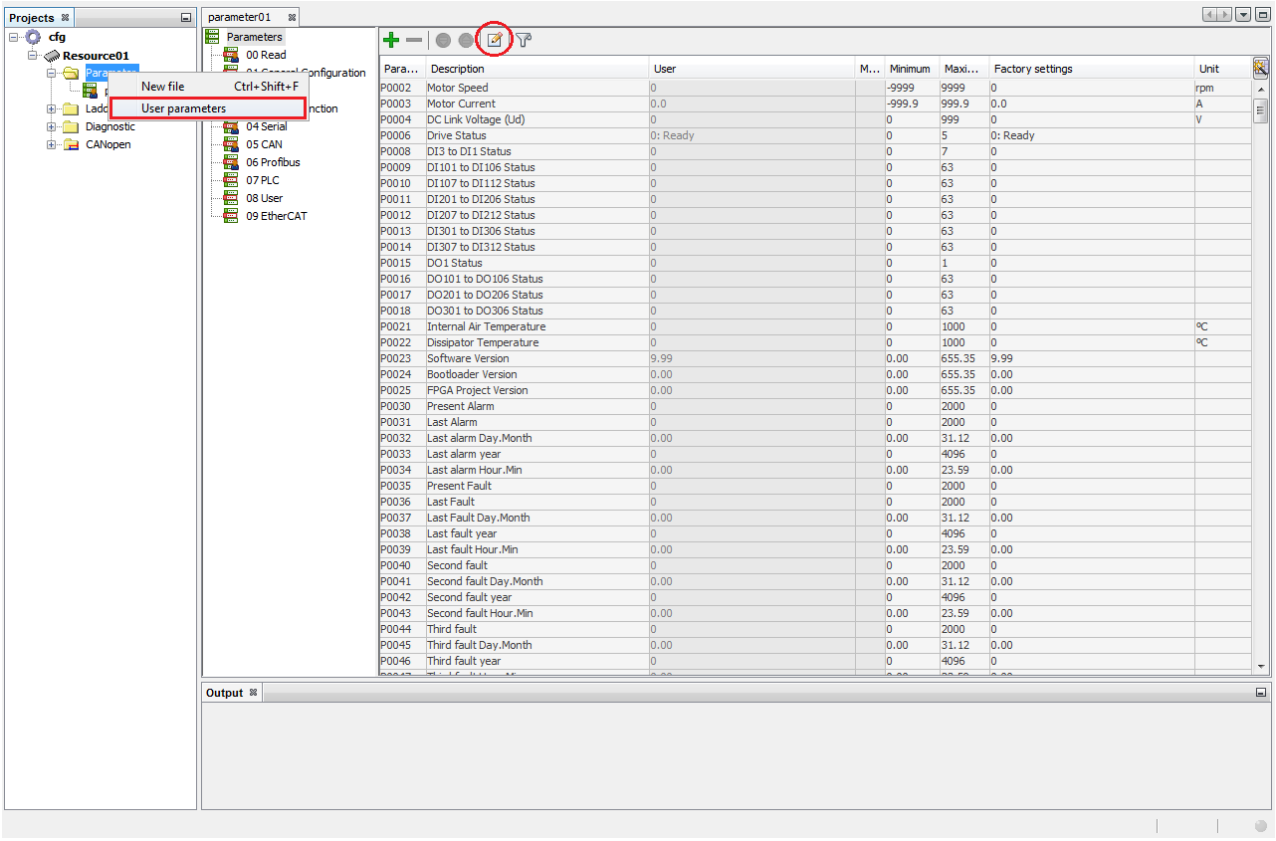
5. **Hide and Show Parameters and Groups of Parameters**. By means of this option, you have full control of the parameters and groups of parameters. It is possible to hide and unhide individual parameters, multiple parameters, individual groups and multiple groups in the same action.


Projects
cfg
Resource01
Parameter
parameter01
Ladder
Diagnostic
CANopen
Open
Hide/unhide parameters
Cut Ctrl+X
Copy Ctrl+C
Delete Delete
Rename...
parameter01
Parameters
00 Read
01 General Configuration
Parameter
Description
User
Monitored
Minimum
Maximum
Factory settings
Unit
P0080 Virtual Axis Speed
P0082 Virtual Axis - fractions of revolution
P0083 Virtual Axis - numbers of revolutions
P0084 Day of week
P0085 Day
P0086 Month
P0087 Year
P0088 Hous
P0089 Minutes
P0090 Seconds
P0091 Slot 1 ID
P0092 Slot 2 ID
P0093 Slot 3 ID
P0095 Opcional Card
P0097 Nominal Current
P0098 Line Rated Voltage
P0228 Iq RMS Cicle
P0740 Profibus Comm. Status
P0850 EtherCAT: FW Revision
P0851 ECAT: AccessoryStatus
P0852 ECAT: Link Status
P0853 ECAT: Slave Status
P0854 EtherCAT: Reserved
P0855 ECAT: TxPDO Config
P0856 ECAT: TxPDO Data Size
P0857 ECAT: RxPDO Config
P0858 ECAT: RxPDO Data Size
P0859 ECAT: Data Update
P0944 Fault Message Counter
P0947 Fault Number
P0963 Profibus Baud Rate
P0964 Drive Unit Ident.
P0965 Profile Ident. Number
P0967 Control Word 1
P0968 Status Word 1
P1000 PLC State
P1001 Scan Time
Output
WEG SCA-06 V1.40 localhost:502 USB/@0


Projects
cfg
Resource01
Parameter
parameter01
Ladder
Diagnostic
CANopen
parameter01
Parameters
00 Read
01 General Configuration
02 Motor
03 Special Function
04 Serial
05 CAN
06 Profibus
07 PLC
08 User
09 EtherCAT
Hide/unhide parameters
Hide/Unhide parameters
OK
Cancel
Output
WEG SCA-06 V1.40 localhost:502 USB/@0

11.7.2.5 User Parameters

In order to open the configuration screen of the user parameters, just click on the **User Parameters** option on the Parameter node of the project tree or click on the icon indicated on the tool bar of the parameter file.



Para...	Description	User	M...	Minimum	Maxi...	Factory settings	Unit
P0002	Motor Speed	0		-9999	9999	0	rpm
P0003	Motor Current	0.0		-999.9	999.9	0.0	A
P0004	DC Link Voltage (Ud)	0		0	999	0	V
P0006	Drive Status	0: Ready		0	5	0: Ready	
P0008	DI3 to DI1 Status	0		0	7	0	
P0009	DI101 to DI106 Status	0		0	63	0	
P0010	DI107 to DI112 Status	0		0	63	0	
P0011	DI201 to DI206 Status	0		0	63	0	
P0012	DI207 to DI212 Status	0		0	63	0	
P0013	DI301 to DI306 Status	0		0	63	0	
P0014	DI307 to DI312 Status	0		0	63	0	
P0015	DO1 Status	0		0	1	0	
P0016	DO101 to DO106 Status	0		0	63	0	
P0017	DO201 to DO206 Status	0		0	63	0	
P0018	DO301 to DO306 Status	0		0	63	0	
P0021	Internal Air Temperature	0		0	1000	0	°C
P0022	Dissipator Temperature	0		0	1000	0	°C
P0023	Software Version	9.99		0.00	655.35	9.99	
P0024	Bootloader Version	0.00		0.00	655.35	0.00	
P0025	FPGA Project Version	0.00		0.00	655.35	0.00	
P0030	Present Alarm	0		0	2000	0	
P0031	Last Alarm	0		0	2000	0	
P0032	Last alarm Day.Month	0.00		0.00	31.12	0.00	
P0033	Last alarm year	0		0	4096	0	
P0034	Last alarm Hour.Min	0.00		0.00	23.59	0.00	
P0035	Present Fault	0		0	2000	0	
P0036	Last Fault	0		0	2000	0	
P0037	Last Fault Day.Month	0.00		0.00	31.12	0.00	
P0038	Last fault year	0		0	4096	0	
P0039	Last fault Hour.Min	0.00		0.00	23.59	0.00	
P0040	Second fault	0		0	2000	0	
P0041	Second fault Day.Month	0.00		0.00	31.12	0.00	
P0042	Second fault year	0		0	4096	0	
P0043	Second fault Hour.Min	0.00		0.00	23.59	0.00	
P0044	Third fault	0		0	2000	0	
P0045	Third fault Day.Month	0.00		0.00	31.12	0.00	
P0046	Third fault year	0		0	4096	0	

Configuration Table.

On the user parameter configuration table, it is possible to define several attributes to the parameters, such as description, minimum and maximum values, unit, digits, data type, etc.

NOTE!
These settings will be automatically displayed in the parameter table. However, to be sent to the device, you need to *download* the resource.

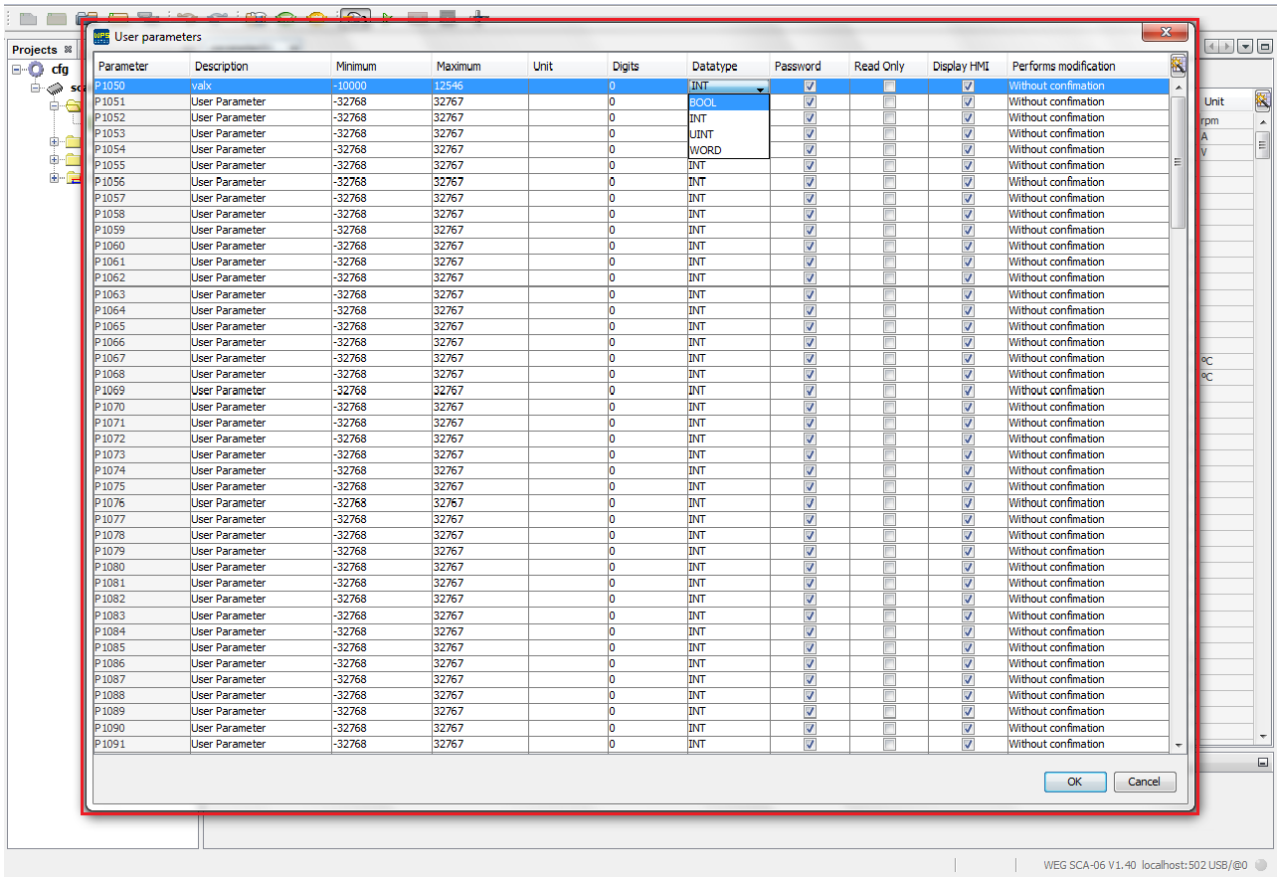


Table fields:

- Parameter: User parameter identification.
- Description: Description of the user parameter in the parameter table. On devices that have text-based HMIs, the description is sent to the machine and displayed on the HMI.
- Minimum: Minimum input value for parameter.
- Maximum: Maximum input value for parameter.
- Unit: Unit displayed on the device's HMI.
- Default: Value loaded when restore factory default is selected.
- Retentive: Retain value after rebooting devices.
- Hexadecimal: Displays the value in hexadecimal.
- Digits: Number of decimal digits for displaying value.
- Datatype: Parameter datatype used by the ladder application.
- Password: Enables password request by changing parameter value.

- Read only: It does not allow the writing of values in the parameter by the communication network or the HMI. Writing is done only by the ladder application.

- Display HMI: Displays the parameter in the HMI.

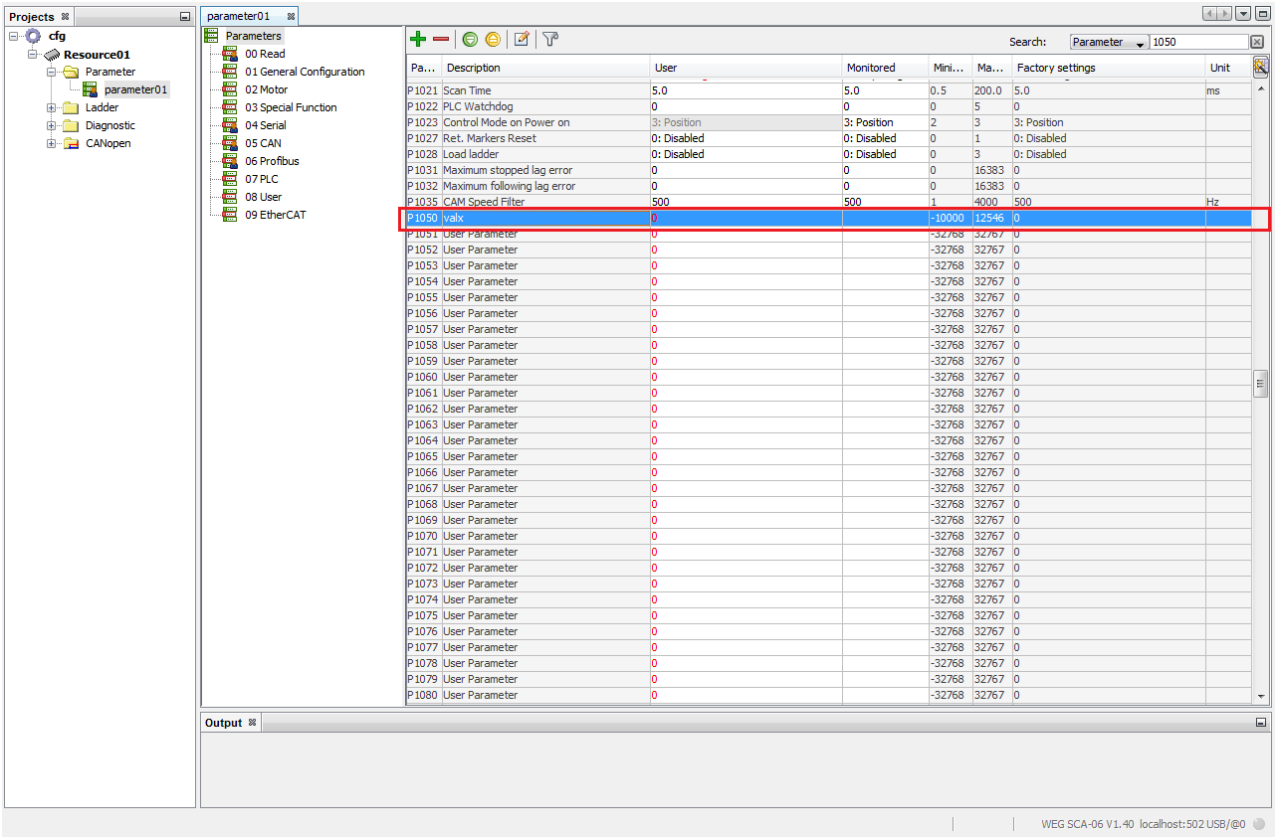
- Performs modification: Confirmation options when changing the parameter:
o No confirmation: Does not prompt for confirmation when changing parameter.
o With confirmation and engine stopped: Request confirmation and allow change only with engine stopped.
o With confirmation: Prompt for confirmation when changing parameter.

- Stopped motor: Perform change only with motor stopped.

- Help: On devices that have text-based HMI, you can edit a help text for the parameter.

View the user parameter

In the parameter table, the user parameters will be shown as they are configured on the configuration screen.



Pa...	Description	User	Monitored	Mini...	Ma...	Factory settings	Unit
P1021	Scan Time	5.0	5.0	0.5	200.0	5.0	ms
P1022	PLC Watchdog	0	0	0	5	0	
P1023	Control Mode on Power on	3: Position	3: Position	2	3	3: Position	
P1027	Ret. Markers Reset	0: Disabled	0: Disabled	0	1	0: Disabled	
P1028	Load ladder	0: Disabled	0: Disabled	0	3	0: Disabled	
P1031	Maximum stopped lag error	0	0	0	16383	0	
P1032	Maximum following lag error	0	0	0	16383	0	
P1035	CAM Speed Filter	500	500	1	4000	500	Hz
P1050	valx	0		-10000	12546	0	
P1051	User Parameter	0		-32768	32767	0	
P1052	User Parameter	0		-32768	32767	0	
P1053	User Parameter	0		-32768	32767	0	
P1054	User Parameter	0		-32768	32767	0	
P1055	User Parameter	0		-32768	32767	0	
P1056	User Parameter	0		-32768	32767	0	
P1057	User Parameter	0		-32768	32767	0	
P1058	User Parameter	0		-32768	32767	0	
P1059	User Parameter	0		-32768	32767	0	
P1060	User Parameter	0		-32768	32767	0	
P1061	User Parameter	0		-32768	32767	0	
P1062	User Parameter	0		-32768	32767	0	
P1063	User Parameter	0		-32768	32767	0	
P1064	User Parameter	0		-32768	32767	0	
P1065	User Parameter	0		-32768	32767	0	
P1066	User Parameter	0		-32768	32767	0	
P1067	User Parameter	0		-32768	32767	0	
P1068	User Parameter	0		-32768	32767	0	
P1069	User Parameter	0		-32768	32767	0	
P1070	User Parameter	0		-32768	32767	0	
P1071	User Parameter	0		-32768	32767	0	
P1072	User Parameter	0		-32768	32767	0	
P1073	User Parameter	0		-32768	32767	0	
P1074	User Parameter	0		-32768	32767	0	
P1075	User Parameter	0		-32768	32767	0	
P1076	User Parameter	0		-32768	32767	0	
P1077	User Parameter	0		-32768	32767	0	
P1078	User Parameter	0		-32768	32767	0	
P1079	User Parameter	0		-32768	32767	0	
P1080	User Parameter	0		-32768	32767	0	

11.8 PLC300

11.8.1 Description

PLC300 is a PLC with integrated HMI, developed to meet the need of interface with the user on panels and machines and at the same time a complete expandable PLC, fast and with several communication ports, enabling the product to be master of CANopen networks (CAN network) and/or Modbus RTU (RS-485 network)

as well as Modbus TCP (Ethernet network).

Description of the Models

PLC300 is available in six different models. With our without HMI; standard or plus, and a version with HMI, though without membrane, in which the user can personalize the PLC appearance following specifications present in the product CD.

The identification of PLC300 is done by a two or three letter suffix. The letters have the following meaning:
- **H**: HMI
- **B**: Blind (without HMI)
- **P**: Plus
- **S**: Standard
- **C**: Custom (without membrane)

Specifications	PLC300HP	PLC300BP	PLC300HS	PLC300BS	PLC300HPC	PLC300HSC
Code WEG	12358193	12358194	12358192	12358195	12358196	12358197
IHM	✓	✗	✓	✗	✓	✓
Membrane	✓	✗	✓	✗	✗	✗
Ethernet	✓	✓	✗	✗	✓	✗
Encoder	✓	✓	✗	✗	✓	✗
Expansion	✓	✓	✗	✗	✓	✗
SD Card	✓	✓	✓	✓	✓	✓
CAN	✓	✓	✓	✓	✓	✓
RS 485	✓	✓	✓	✓	✓	✓
RS 232	✓	✓	✓	✓	✓	✓
USB	✓	✓	✓	✓	✓	✓
PWM Output	✓	✓	✓	✓	✓	✓
Digital I/O	✓	✓	✓	✓	✓	✓
Analog I/O	✓	✓	✓	✓	✓	✓

For further details on the product, refer to PLC300 equipment's user manual.

11.8.2 New Features and Corrections

PLC300 V4.11

New Functions:

- ARRAY_COPY block;
- SCALE block;
- Conversion block MUX2;
- Conversion block DEMUX2;
- Conversion block BYTES_TO_DWORD;
- Conversion block DWORD_TO_BYTES;
- Conversion block BYTES_TO_WORD;
- Conversion block WORD_TO_BYTES;

- Conversion block WORDS_TO_DWORD;
- Conversion block DWORD_TO_WORD;
- CALL block;
- System markers for accessing the function key LEDs F1...F6;
- PID2 block;
- IMMEDIATE_OUTPUT block also for the expansion modules;
- SWAP2 block.

Corrections of functional deviation:

- Status LED: In some undesired situation the Status LED was turning red, overwriting the green LED, signaling PLC without application or stopped;
- Numeric input fix: When using decimal places, it was truncating the value rather than rounding it at CAST time, and it could change the value entered by the user.

PLC300 V3.41

New Functions:

- ARRAY_COPY block;
- SCALE block;
- Conversion block MUX2;
- Conversion block DEMUX2;
- Conversion block BYTES_TO_DWORD;
- Conversion block DWORD_TO_BYTES;
- Conversion block BYTES_TO_WORD;
- Conversion block WORD_TO_BYTES;
- Conversion block WORDS_TO_DWORD;
- Conversion block DWORD_TO_WORD;
- CALL block;
- System markers for accessing the function key LEDs F1...F6;
- PID2 block;
- IMMEDIATE_OUTPUT block also for the expansion modules;
- SWAP2 block.

Corrections of functional deviation:

- Status LED: In some undesired situation the Status LED was turning red, overwriting the green LED, signaling PLC without application or stopped;
- Numeric input fix: When using decimal places, it was truncating the value rather than rounding it at CAST time, and it could change the value entered by the user.

PLC300 V3.30

New Functions:

- Follow CANopen;
- "Change of State" log event with variable list;
- Ethernet function blocks with 128 bytes of data.

Corrections of functional deviation:

- Inconsistent data on the component "Input text".

PLC300 V3.00

New Functions:

- Internal memory increased to 1MB.

Corrections of functional deviation:

- After the firmware download the watchdog alarm was activated.

> **NOTE!**
> It's strongly recommended to always upgrade the PLC300 firmware to the latest available version.
> - Hardware version 1 (H1): v1.76 or higher
> - Hardware version 2 (H2): v2.42 or higher
> - Hardware version 2 w/ 1MB (H2-1MB): v3.08 or higher

PLC300 V2.40

Corrections of functional deviation:

- Locking of Modbus RTU blocks when utilized in USERFB. When disabling USERFB with a Modbus RTU block activated (which reserves the modbus master resource) other Modbus RTU blocks couldn´t be executed anymore;
- General backup in version 2.3x was presenting problems during the backup of files in SD Card; and
- The Watchdog was being activated in some devices during the initialization causing it to stop responding.

> **NOTE!**
> It's strongly recommended to always upgrade the PLC300 firmware to the latest available version.
> - Hardware version 1 (H1): v1.76 or higher
> - Hardware version 2 (H2): v2.42 or higher
> - Hardware version 2 w/ 1MB (H2-1MB): v3.08 or higher

PLC300 V2.30

New Functions:

- Hot Download: Implementation that allows the realization of hot download on the resource, in other words, its possible to change the ladder program, screens, alarms and logs with the actual program running and after the download conclusion the new program is executed automatically;
- Watchdog: Implementation of a configurable Watchdog with minimal time of 300ms. The outputs state can be configure in case of Watchdog, as also one exit for exclusive use;

- Markers for monitoring the maximum and minimum time of scan cycle;
- Markers to disable the keys HOME, SETUP and ALARM.

Modification of existing functions:

- Encoder default value changed from 12V to 5V.

PLC300 V2.10

New Functions:

- Creation of blocks that support Strings:
  - STR_COMPARE
  - STR_COPY
  - STR_COPY_LAST
  - STR_DELETE
  - STR_FIND
  - STR_FIND_LAST
  - STR_INSERT
  - STR_LENGTH
  - STR_REPLACE
  - DWORD_TO_STRING
  - REAL_TO_STRING
  - STRING_TO_DWORD
  - STRING_TO_REAL
- Creation of block READENC4, reads encoder, calculating the positioning and velocity of the same, allows use of filter;
- External and counting tasks using DI1 to DI8;
- Creation of block TRUNC, realizes variable truncation;
- Creation of block ROUND, realizes variable rounding;
- Increment and decrement of fields "Numeric Input" using direction keys (during field editing);
- Log variables of type STRING;
- Block STORE with variables of type STRING;
- Block SEL with variables of type STRING;
- Block ISTORE with variables of type STRING;
- Block ILOAD with variables of type STRING;
- Use of variables of type STRING in "Text Input" fields;
- Creation of structures and recipes with variables of type STRING;

Modification of existing functions

- Base address modified from 3000 to 0.

PLC300 V2.00

New Functions:

- Creation of block P_RMAP that generates train pulses with frequency in form of "ramp";

- Inputs DI9 and DI10 as encoder inputs;
- DI10 can be used as rapid count, and DI9 determines the counting direction;
- Change in the analog input 10bits to 12bits;
- Automatic recovery (ASR) application, setup, firmware ... through FLASH memory, requiring no more SD Card for this feature.

PLC300 V1.70

New Functions:

- Creation of a "Text Input" component type;
- Creation of one new memory area for preserving the variable values during download

NOTE!
It's strongly recommended to always upgrade the PLC300 firmware to the latest available version.
- Hardware version 1 (H1): v1.76 or higher
- Hardware version 2 (H2): v2.42 or higher
- Hardware version 2 w/ 1MB (H2-1MB): v3.08 or higher

PLC300 V1.60

Modification of existing functions

- Creation of new instructions for program size and scan cycle reduction

---

PLC300 V1.50

New Functions:

- Status System Markers (%S_):
  - STS_ASR_OCC
  - STS_INPUT
- Command System Markers (%C_):
  - RS232_TIMEOUT_MS
  - RS232_RX_END_CHARACTER
  - RS232_TX_INITIAL_ADDRESS
  - RS232_RX_INITIAL_ADDRESS
  - RS232_TX_BUFFER_LENGTH
  - RS232_MAX_RX_BUFFER_LENGTH
  - RS232_ENABLE_END_CHARACTER
  - RS232_START_TX
  - RS485_TIMEOUT_MS
  - RS485_RX_END_CHARACTER
  - RS485_TX_INITIAL_ADDRESS
  - RS485_RX_INITIAL_ADDRESS
  - RS485_TX_BUFFER_LENGTH

  - RS485_MAX_RX_BUFFER_LENGTH
  - RS485_ENABLE_END_CHARACTER
  - RS485_START_TX
- Automatic Software Recovery (ASR) function

Modification of existing functions

- Modbus RTU block modification to support reading and writing 16 registers
- Modified timer blocks (TON, TOF and TP) with adjustable time base (milliseconds centisegundos, minutes and seconds)
- Increased capacity of the amount of user screens

PLC300 V1.40

New Functions:

- Status System Markers (%S_):
  - KEY_NUMERIC
  - KEY_HOME
  - KEY_ESC
  - KEY_DEL
  - KEY_ALARM
  - KEY_SETUP
  - KEY_SHIFT
  - KEY_UP
  - KEY_DOWN
  - KEY_LEFT
  - KEY_RIGHT
  - KEY_ENTER
  - KEY_F1 ... KEY_F12
- Program upload
- Force I/O
- English language on the PLC300
- Presentation of variables on alarm screens
- Leading zeros in the "Numeric Input" and "Numeric Output" fields on the user's screens
- Download option:
  - Initialize output and volatile variables
  - Stop/Start the execution of the program automatically
- Password-protected commands for recording and loading setup and firmware files on the SD card.

PLC300 V1.30

New Functions:

- Status System Markers (%S_):
  - BOOTLOADER
  - INTERVAL_TASK9_WATCHDOG ... INTERVAL_TASK16_WATCHDOG
  - SINGLE_TASK9_WATCHDOG ... SINGLE_TASK16_WATCHDOG

  - COUNT_TASK9_WATCHDOG ... COUNT_TASK16_WATCHDOG
  - STS_SD_INVALID
- Command System Markers (%C_):
  - INTERVAL_TASK9_DISABLE ... INTERVAL_TASK16_DISABLE
  - SINGLE_TASK9_DISABLE ... SINGLE_TASK16_DISABLE
  - COUNT_TASK9_DISABLE ... COUNT_TASK9_DISABLE
- Modbus TCP Blocks
  - MBTCP_ReadBinary
  - MBTCP_WriteBinary
  - MBTCP_ReadRegister
  - MBTCP_WriteRegister
  - MBTCP_ServerStatus
  - MBTCP_ClientControlStatus
- New Data Transfer Blocks (Recipes)
  - ReadRecipe
  - WriteRecipe
- Log of
  - Alarms
  - Events
- Backup in the SD Card of
  - Firmware
  - Resource
  - Setup
- Ethernet
  - WPS gateway connection with the PLC300

PLC300 V1.20

New Functions:

- Tasks
  - INTERVAL
  - SINGLE
  - EXTERN EVENT (DI9, DI10 and Z pulse)
  - COUNT (DI9, DI10, A, B, Z pulses and quadrature AB)
  - SYSTEM (start and stop)
- Status System Markers (%S_)
  - TICK_100US
  - INTERVAL_TASK1_WATCHDOG...INTERVAL_TASK8_WATCHDOG
  - SINGLE_TASK1_WATCHDOG...SINGLE_TASK8_WATCHDOG
  - EXT_EVENT_TASK1_WATCHDOG...EXT_EVENT_TASK3_WATCHDOG
  - COUNT_TASK1_WATCHDOG...COUNT_TASK8_WATCHDOG
  - MAIN_TASK_WATCHDOG
  - START_TASK_WATCHDOG
  - STOP_TASK_WATCHDOG
- Command System Markers (%C_)
  - INTERVAL_TASK1_DISABLE...INTERVAL_TASK8_DISABLE
  - SINGLE_TASK1_DISABLE...SINGLE_TASK8_DISABLE
  - EXT_EVENT_TASK1_WATCHDOG...EXT_EVENT_TASK3_WATCHDOG

  - COUNT_TASK1_DISABLE...COUNT_TASK8_DISABLE
- Hardware Blocks
  - IMMEDIATEINPUT
  - IMMEDIATEOUTPUT
  - READENC3
- Coil Block
  - IMMEDIATECOIL

Modification of existing functions

- Options to initialize or not the retentive variables and alarm history in the download.
- PWM Block - frequency value of 0 Hz allowed
- ReadEnc and ReadEnc2 Blocks - Value Data Type can be DINT when the counted pulses are Quadrature_AB, allowing negative values according to the rotation direction of the encoder.

Corrections of functional deviation

- MB_WriteBinary Block - in previous versions, the block always wrote the value 1.
- Variable addresses modified for access via Modbus.

PLC300 Versions previous to V1.20

We recommend to update the firmware.

PLC300 V1.10

New Functions:

- ASCII RS232 Protocol

PLC300 V1.00

Initial version.

11.8.3 I/O's

Hardware information can be found in the Manual of the PLC300 at the website www.weg.net.

Digital Inputs

Address	Bit	Modbus	Tag	Description
%IB0	0	16000	DI1	Digital input 1
%IB0	1	16001	DI2	Digital input 2
%IB0	2	16002	DI3	Digital input 3

%IB0	3	16003	DI4	Digital input 4
%IB0	4	16004	DI5	Digital input 5
%IB0	5	16005	DI6	Digital input 6
%IB0	6	16006	DI7	Digital input 7
%IB0	7	16007	DI8	Digital input 8
%IB1	0	16008	DI9	Digital input 9
%IB1	1	16009	DI10	Digital input 10
%IB2	0	16016	DI101	Digital input 1 - Slot 1
%IB2	1	16017	DI102	Digital input 2 - Slot 1
%IB2	2	16018	DI103	Digital input 3 - Slot 1
%IB2	3	16019	DI104	Digital input 4 - Slot 1
%IB2	4	16020	DI105	Digital input 5 - Slot 1
%IB2	5	16021	DI106	Digital input 6 - Slot 1
%IB2	6	16022	DI107	Digital input 7 - Slot 1
%IB2	7	16023	DI108	Digital input 8 - Slot 1
%IB3	0	16024	DI109	Digital input 9 - Slot 1
%IB3	1	16025	DI110	Digital input 10 - Slot 1
%IB3	2	16026	DI111	Digital input 11 - Slot 1
%IB3	3	16027	DI112	Digital input 12 - Slot 1
%IB3	4	16028	DI113	Digital input 13 - Slot 1
%IB3	5	16029	DI114	Digital input 14 - Slot 1
%IB3	6	16030	DI115	Digital input 15 - Slot 1
%IB3	7	16031	DI116	Digital input 16 - Slot 1
%IB4	0	16032	DI201	Digital input 1 - Slot 2
%IB4	1	16033	DI202	Digital input 2 - Slot 2
%IB4	2	16034	DI203	Digital input 3 - Slot 2
%IB4	3	16035	DI204	Digital input 4 - Slot 2
%IB4	4	16036	DI205	Digital input 5 - Slot 2
%IB4	5	16037	DI206	Digital input 6 - Slot 2
%IB4	6	16038	DI207	Digital input 7 - Slot 2
%IB4	7	16039	DI208	Digital input 8 - Slot 2
%IB5	0	16040	DI209	Digital input 9 - Slot 2
%IB5	1	16041	DI210	Digital input 10 - Slot 2
%IB5	2	16042	DI211	Digital input 11 - Slot 2
%IB5	3	16043	DI212	Digital input 12 - Slot 2
%IB5	4	16044	DI213	Digital input 13 - Slot 2
%IB5	5	16045	DI214	Digital input 14 - Slot 2
%IB5	6	16046	DI215	Digital input 15 - Slot 2
%IB5	7	16047	DI216	Digital input 16 - Slot 2

Analog Inputs

Address	Bit	Modbus	Tag	Description
%IW6	--	5003	AI1	Analog input 1
%IW8	--	5004	AI101	Analog input 1 - Slot 1
%IW10	--	5005	AI102	Analog input 2 - Slot 1
%IW12	--	5006	AI103	Analog input 3 - Slot 1
%IW14	--	5007	AI104	Analog input 4 - Slot 1
%IW16	--	5008	AI105	Analog input 5 - Slot 1
%IW18	--	5009	AI201	Analog input 1 - Slot 2
%IW20	--	5010	AI202	Analog input 2 - Slot 2
%IW22	--	5011	AI203	Analog input 3 - Slot 2
%IW24	--	5012	AI204	Analog input 4 - Slot 2
%IW26	--	5013	AI205	Analog input 5 - Slot 2

Digital Outputs

Address	Bit	Modbus	Tag	Description
%QB0	0	16000	DO1	Digital output 1
%QB0	1	16001	DO2	Digital output 2
%QB0	2	16002	DO3	Digital output 3
%QB0	3	16003	DO4	Digital output 4
%QB0	4	16004	DO5	Digital output 5
%QB0	5	16005	DO6	Digital output 6
%QB0	6	16006	DO7	Digital output 7
%QB0	7	16007	DO8	Digital output 8
%QB1	0	16008	DO9	Digital output 9
%QB2	0	16016	DO101	Digital output 1 - Slot 1
%QB2	1	16017	DO102	Digital output 2 - Slot 1
%QB2	2	16018	DO103	Digital output 3 - Slot 1
%QB2	3	16019	DO104	Digital output 4 - Slot 1
%QB2	4	16020	DO105	Digital output 5 - Slot 1
%QB2	5	16021	DO106	Digital output 6 - Slot 1
%QB2	6	16022	DO107	Digital output 7 - Slot 1
%QB2	7	16023	DO108	Digital output 8 - Slot 1
%QB3	0	16024	DO109	Digital output 9 - Slot 1
%QB3	1	16025	DO110	Digital output 10 - Slot 1
%QB3	2	16026	DO111	Digital output 11 - Slot 1
%QB3	3	16027	DO112	Digital output 12 - Slot 1
%QB3	4	16028	DO113	Digital output 13 - Slot 1
%QB3	5	16029	DO114	Digital output 14 - Slot 1
%QB3	6	16030	DO115	Digital output 15 - Slot 1
%QB3	7	16031	DO116	Digital output 16 - Slot 1
%QB4	0	16032	DO201	Digital output 1 - Slot 2
%QB4	1	16033	DO202	Digital output 2 - Slot 2
%QB4	2	16034	DO203	Digital output 3 - Slot 2
%QB4	3	16035	DO204	Digital output 4 - Slot 2
%QB4	4	16036	DO205	Digital output 5 - Slot 2
%QB4	5	16037	DO206	Digital output 6 - Slot 2
%QB4	6	16038	DO207	Digital output 7 - Slot 2
%QB4	7	16039	DO208	Digital output 8 - Slot 2
%QB5	0	16040	DO209	Digital output 9 - Slot 2
%QB5	1	16041	DO210	Digital output 10 - Slot 2
%QB5	2	16042	DO211	Digital output 11 - Slot 2
%QB5	3	16043	DO212	Digital output 12 - Slot 2
%QB5	4	16044	DOI13	Digital output 13 - Slot 2
%QB5	5	16045	DO214	Digital output 14 - Slot 2
%QB5	6	16046	DO215	Digital output 15 - Slot 2
%QB5	7	16047	DO216	Digital output 16 - Slot 2

Analog Outputs

Address	Bit	Modbus	Tag	Description
%QW6	--	5003	AO1	Analog output 1
%QW8	--	5004	AO101	Analog output 1 - Slot 1
%QW10	--	5005	AO102	Analog output 2 - Slot 1
%QW12	--	5006	AO201	Analog output 1 - Slot 2
%QW14	--	5007	AO202	Analog output 2 - Slot 2

11.8.4 System Markers

The following variables contained in the **GLOBAL_SYSTEM** group of the variables table, have the fixed tag.

Some markers of the BYTE type (%SB or %CB) are allocated in the same modbus address for registers. In this case:
- (L): indicates that it is the least significant byte of the WORD;
- (H): indicates that it is the most significant byte of the WORD;

NOTE!
The base address was modified from 3000 to 0 in firmware version 2.10.

Reading System Markers (Status)

Address	Bit	Modbus	Tag	Description
Ladder				
%SB0004	0	32	FALSE	Always in 0
%SB0004	1	33	TRUE	Always in 1
%SB0004	2	34	FREQ_2HZ	Oscillator 2Hz
%SB0004	3	35	PULSE_1SCAN	Pulse during the first scan cycle
%SB0004	4	36	ENC_DIR	Encoder speed direction (0-Clockwise, 1-Anticlockwise)
PLC300 Status				
%SW0000	--	3000	FIRMWARE	Firmware version
%SW0002	--	3001	SCAN_CICLE	Scan cycle time
%SB0006	0	48	STS_BAT	Low battery
%SB0006	1	49	STS_DOS	Digital output fault
%SB0006	2	50	STS_ENC	Encoder fault
%SB0006	3	51	STS_AI1	Analog input 1 break detected
%SB0006	4	52	STS_SD_INVALID	SD card missing or invalid
%SB0006	4	53	STS_DOS_SLOT1	Digital output slot 1 error
%SB0006	5	54	STS_DOS_SLOT2	Digital output slot 2 error
%SB0006	6	55	STS_ASR_OCC	Automatic Software Recovery (ASR) occurred
%SD0008	--	3004	ENC_FREQ	Encoder frequency

%SD0012	--	3006	TICK_100US	Marker increased every 100us
%SW0016	--	3008	BOOTLOADER	Module Version Upgrade Firmware
%SB0540	0	3270	WATCHDOG_OCC	Watchdog occured
%SW0542	--	3271	WATCHDOG_DATE	Watchdog date
%SD0556	--	3278	WATCHDOG_CODE	Watchdog code
HMI Keys and Screen				
%SB0020	--	3010	SCREEN	Actual screen
%SB0022	--	3011	KEY_NUMERIC	ASCII code of numeric key
%SB0024	0	192	KEY_HOME	HOME key pressed
%SB0024	1	193	KEY_ESC	ESC key pressed
%SB0024	2	194	KEY_DEL	DEL key pressed
%SB0024	3	195	KEY_ALARM	ALARM key pressed
%SB0024	4	196	KEY_SETUP	SETUP key pressed
%SB0024	5	197	KEY_SHIFT	SHIFT key pressed
%SB0024	6	198	KEY_UP	UP key pressed
%SB0024	7	199	KEY_DOWN	DOWN key pressed
%SB0025	0	200	KEY_LEFT	LEFT key pressed
%SB0025	1	201	KEY_RIGHT	RIGHT key pressed
%SB0025	2	202	KEY_ENTER	ENTER key pressed
%SB0025	3	203	STS_INPUT	Editing element 'Numeric Input' on screen
%SB0026	0	208	KEY_F1	F1 key pressed
%SB0026	1	209	KEY_F2	F2 key pressed
%SB0026	2	210	KEY_F3	F3 key pressed
%SB0026	3	2011	KEY_F4	F4 key pressed
%SB0026	4	212	KEY_F5	F5 key pressed
%SB0026	5	213	KEY_F6	F6 key pressed
%SB0026	6	214	KEY_F7	F7 key pressed
%SB0026	7	215	KEY_F8	F8 key pressed
%SB0027	0	216	KEY_F9	F9 key pressed
%SB0027	1	217	KEY_F10	F10 key pressed
%SB0027	2	218	KEY_F11	F11 key pressed
%SB0027	3	219	KEY_F12	F12 key pressed
Real Time Clock (RTC)				
%SW0030	--	3015	HOUR	Real time clock hour
%SW0032	--	3016	MINUTE	Real time clock minute
%SW0034	--	3017	SECOND	Real time clock second
%SW0036	--	3018	DAY	Real time clock day
%SW0038	--	3019	MONTH	Real time clock month
%SW0040	--	3020	YEAR	Real time clock year
%SW0042	--	3021	WEEKDAY	Weekday (0-Sunday, 1-Monday ... 6-Saturday)
Task Watchdog				
%SB0050	0	400	INTERVAL_TASK1_WATCHDOG	Time task 1 watchdog

%SB0050	1	401	INTERVAL_TASK2_WATCHDOG	Time task 2 watchdog
%SB0050	2	402	INTERVAL_TASK3_WATCHDOG	Time task 3 watchdog
%SB0050	3	403	INTERVAL_TASK4_WATCHDOG	Time task 4 watchdog
%SB0050	4	404	INTERVAL_TASK5_WATCHDOG	Time task 5 watchdog
%SB0050	5	405	INTERVAL_TASK6_WATCHDOG	Time task 6 watchdog
%SB0050	6	406	INTERVAL_TASK7_WATCHDOG	Time task 7 watchdog
%SB0050	7	407	INTERVAL_TASK8_WATCHDOG	Time task 8 watchdog
%SB0051	0	408	INTERVAL_TASK9_WATCHDOG	Time task 9 watchdog
%SB0051	1	409	INTERVAL_TASK10_WATCHDOG	Time task 10 watchdog
%SB0051	2	410	INTERVAL_TASK11_WATCHDOG	Time task 11 watchdog
%SB0051	3	411	INTERVAL_TASK12_WATCHDOG	Time task 12 watchdog
%SB0051	4	412	INTERVAL_TASK13_WATCHDOG	Time task 13 watchdog
%SB0051	5	413	INTERVAL_TASK14_WATCHDOG	Time task 14 watchdog
%SB0051	6	414	INTERVAL_TASK15_WATCHDOG	Time task 15 watchdog
%SB0051	7	415	INTERVAL_TASK16_WATCHDOG	Time task 16 watchdog
%SB0052	0	416	SINGLE_TASK1_WATCHDOG	Event task 1 watchdog
%SB0052	1	417	SINGLE_TASK2_WATCHDOG	Event task 2 watchdog
%SB0052	2	418	SINGLE_TASK3_WATCHDOG	Event task 3 watchdog
%SB0052	3	419	SINGLE_TASK4_WATCHDOG	Event task 4 watchdog
%SB0052	4	420	SINGLE_TASK5_WATCHDOG	Event task 5 watchdog
%SB0052	5	421	SINGLE_TASK6_WATCHDOG	Event task 6 watchdog
%SB0052	6	422	SINGLE_TASK7_WATCHDOG	Event task 7 watchdog
%SB0052	7	423	SINGLE_TASK8_WATCHDOG	Event task 8 watchdog
%SB0053	0	424	SINGLE_TASK9_WATCHDOG	Event task 9 watchdog
%SB0053	1	425	SINGLE_TASK10_WATCHDOG	Event task 10 watchdog
%SB0053	2	426	SINGLE_TASK11_WATCHDOG	Event task 11 watchdog
%SB0053	3	427	SINGLE_TASK12_WATCHDOG	Event task 12 watchdog
%SB0053	4	428	SINGLE_TASK13_WATCHDOG	Event task 13 watchdog
%SB0053	5	429	SINGLE_TASK14_WATCHDOG	Event task 14 watchdog
%SB0053	6	430	SINGLE_TASK15_WATCHDOG	Event task 15 watchdog
%SB0053	7	431	SINGLE_TASK16_WATCHDOG	Event task 16 watchdog
%SB0054	0	3027	EXT_EVENT_TASK1_WATCHDOG	External event 1 task watchdog
%SB0054	1	3027	EXT_EVENT_TASK2_WATCHDOG	External event 2 task watchdog
%SB0054	2	3027	EXT_EVENT_TASK3_WATCHDOG	External event 3 task watchdog
%SB0056	0	3028	COUNT_TASK1_WATCHDOG	Count task 1 watchdog
%SB0056	1	3028	COUNT_TASK2_WATCHDOG	Count task 2 watchdog
%SB0056	2	3028	COUNT_TASK3_WATCHDOG	Count task 3 watchdog
%SB0056	3	3028	COUNT_TASK4_WATCHDOG	Count task 4 watchdog
%SB0056	4	3028	COUNT_TASK5_WATCHDOG	Count task 5 watchdog
%SB0056	5	3028	COUNT_TASK6_WATCHDOG	Count task 6 watchdog
%SB0056	6	3028	COUNT_TASK7_WATCHDOG	Count task 7 watchdog
%SB0056	7	3028	COUNT_TASK8_WATCHDOG	Count task 8 watchdog

%SB0057	0	3028	COUNT_TASK9_WATCHDOG	Count task 9 watchdog
%SB0057	1	3028	COUNT_TASK10_WATCHDOG	Count task 10 watchdog
%SB0057	2	3028	COUNT_TASK11_WATCHDOG	Count task 11 watchdog
%SB0057	3	3028	COUNT_TASK12_WATCHDOG	Count task 12 watchdog
%SB0057	4	3028	COUNT_TASK13_WATCHDOG	Count task 13 watchdog
%SB0057	5	3028	COUNT_TASK14_WATCHDOG	Count task 14 watchdog
%SB0057	6	3028	COUNT_TASK15_WATCHDOG	Count task 15 watchdog
%SB0057	7	3028	COUNT_TASK16_WATCHDOG	Count task 16 watchdog
%SB0058	0	3029	MAIN_TASK_WATCHDOG	Main task watchdog
%SB0058	1	3029	START_TASK_WATCHDOG	System start task watchdog
%SB0058	2	3029	STOP_TASK_WATCHDOG	System stop task watchdog
RS232				
%SW0068	--	3034	RS232_TX_TELEGRAM_COUNTER	Sent telegrams counter
%SW0070	--	3035	RS232_RX_TELEGRAM_COUNTER	Received telegrams counter
%SW0072	--	3036	RS232_RX_BYTE_COUNTER	Counter of received bytes for each telegram
%SB0078	0	624	RS232_TIMEOUT	Response timeout
%SB0078	1	625	RS232_RX_TELEGRAM_FINISHED	Telegram received
%SB0078	2	626	RS232_TX_TELEGRAM_FINISHED	Telegram sent
%SB0114	0	912	RS232_RX_FINISHED	ASCII reception finished
RS485				
%SW0084	--	3042	RS485_TX_TELEGRAM_COUNTER	Sent telegrams counter
%SW0086	--	3043	RS485_RX_TELEGRAM_COUNTER	Received telegrams counter
%SW0088	--	3044	RS485_RX_BYTE_COUNTER	Counter of received bytes for each telegram
%SB0094	0	752	RS485_TIMEOUT	Response timeout
%SB0094	1	753	RS485_RX_TELEGRAM_FINISHED	Telegram received
%SB0094	2	754	RS485_TX_TELEGRAM_FINISHED	Telegram transmitted
Modbus RTU				
%SB0100	--	3050	MBUS_INTERFACE_DISABLED	Interface Modbus RTU Master disabled.
%SW0102	--	3051	MBUS_REQUEST_COUNT	Counter requests made by the Master Modbus RTU
%SW0104	--	3052	MBUS_RESPONSE_COUNT	Counter of successful responses received by the Modbus master
%SW0106	--	3053	MBUS_NO_ANSWER_COUNT	Counter of requests without answer received by the master (timeout)
%SW0108	--	3054	MBUS_RESP_ERROR_COUNT	Counter error responses received by the Modbus RTU Master
%SB0110	--	3055 (L)	MBUS_LAST_ERROR_ADDR	Last error detected: Modbus RTU slave address
%SB0111	--	3055 (H)	MBUS_LAST_ERROR_TYPE	Last detected error: error type
%SB0112	--	3056	MBUS_LAST_ERROR_CODE	Last detected error: error code
%SW0120	--	3060	MBUS_SLAVE_REQUEST_COUNT	Counter telegrams received successfully by PLC300 as Modbus RTU slave
%SW0122	--	3061	MBUS_SLAVE_RESPONSE_COUNT	Response Counter transmitted successfully by PLC300 as Modbus RTU slave
CAN				
%SB0150	--	3075	CAN_STATUS	CAN interface status

%SB0151	0	1208	CAN_BUS_POWER	CAN bus power supply
%SW0152	--	3076	CAN_RX_COUNTER	Received CAN telegram counter
%SW0154	--	3077	CAN_TX_COUNTER	Transmitted CAN telegram counter
%SW0156	--	3078	CAN_BUS_OFF_COUNTER	Detected buss off error counter
%SW0158	--	3079	CAN_OVERRUN_COUNTER	Lost (overrun) CAN telegram counter
CANopen				
%SB0180	--	3090 (L)	CO_STATUS	CANopen communication status
%SB0181	--	3090 (H)	CO_NODE_STATE	CANopen node status (pre-operational, operational, stopped)
%SB0200	0	1600	CO_STS_MASTER_CONTACTED	The CANopen master contacted all the slaves
%SB0200	1	1601	CO_STS_MASTER_CONFIG_OK	The CANopen master downloaded the configurations of the slaves
%SB0200	2	1602	CO_STS_MASTER_ERROR_CTRL_OK	Error control protocol (node guarding/heartbeat) initiated with the slaves
%SB0200	3	1603	CO_STS_MASTER_INIT_FINISHED	The CANopen master initialized all the slaves
%SB0200	4	1604	CO_STS_MASTER_INIT_ERROR	A slave presented an initialization error
%SB0200	5	1605	CO_STS_MASTER_ERROR_CTRL	The CANopen master detected a fault in a slave through the error detection protocol
%SB0200	6	1606	CO_STS_MASTER_EMCY	A slave reported EMCY
%SB0201	0	1608	CO_STS_MASTER_NMT_TOGGLE	NMT command toggle bit feedback
%SB0206	5	1613	CO_STS_MASTER_BUS_OFF	The CANopen master is in bus off
%SB0201	6	1614	CO_STS_MASTER_POWER_OFF	The CANopen master has no power supply at the CAN interface
%SB0201	7	1615	CO_STS_MASTER_COMM_DISABLED	Disabled CANopen master communication
%SB0202	0	1616	CO_STS_SLAVE1_CONTACTED	The CANopen master successfully contacted the slave in the indicated address
%SB0202	1	1617	CO_STS_SLAVE1_CONFIG_OK	The CANopen master successfully configured the slave
%SB0202	2	1618	CO_STS_SLAVE1_ERROR_CTRL_OK	Error control protocol (node guarding/heartbeat) initiated with the slave
%SB0202	3	1619	CO_STS_SLAVE1_INIT_FINISHED	Concluded slave initialization
%SB0202	4	1620	CO_STS_SLAVE1_INIT_ERROR	Initialization error in the indicated address slave
%SB0202	5	1621	CO_STS_SLAVE1_ERROR_CTRL_FAIL	Fault detected in some slave from the CANopen master error detection protocol
%SB0202	6	1632	CO_STS_SLAVE1_EMCY	The slave in the indicated address reported EMCY error
%SB0204	0	1633	CO_STS_SLAVE2_CONTACTED	The CANopen master successfully contacted the slave in the indicated address
%SB0204	1	1634	CO_STS_SLAVE2_CONFIG_OK	The CANopen master successfully configured the slave
%SB0204	2	1635	CO_STS_SLAVE2_ERROR_CTRL_OK	Error control protocol (node guarding/heartbeat) initiated with the slave
%SB0204	3	1636	CO_STS_SLAVE2_INIT_FINISHED	Concluded slave initialization
%SB0204	4	1637	CO_STS_SLAVE2_INIT_ERROR	Initialization error in the indicated address slave
%SB0204	5	1638	CO_STS_SLAVE2_ERROR_CTRL_FAIL	Fault detected in some slave from the CANopen master error detection protocol
%SB0204	6	1639	CO_STS_SLAVE2_EMCY	The slave in the indicated address reported EMCY error
...		...	...	...
%SB0454	0	3632	CO_STS_SLAVE127_CONTACTED	The CANopen master successfully contacted the slave

				in the indicated address
%SB0454	1	3633	CO_STS_SLAVE127_CONFIG_OK	The CANopen master successfully configured the slave
%SB0454	2	3634	CO_STS_SLAVE127_ERROR_CTRL_OK	Error control protocol (node guarding/heartbeat) initiated with the slave
%SB0454	3	3635	CO_STS_SLAVE127_INIT_FINISHED	Concluded slave initialization
%SB0454	4	3636	CO_STS_SLAVE127_INIT_ERROR	Initialization error in the indicated address slave
%SB0454	5	3637	CO_STS_SLAVE127_ERROR_CTRL_FAIL	Fault detected in some slave from the CANopen master error detection protocol
%SB0454	6	3638	CO_STS_SLAVE127_EMCY	The slave in the indicated address reported EMCY error
%SW0460	--	3230	CO_SDO_ERROR_NODE_ID	SDO error: address of the slave with the last detected SDO error
%SW0462	--	3231	CO_SDO_ERROR_OBJECT_INDEX	SDO error: object index
%SW0464	--	3232	CO_SDO_ERROR_OBJECT_SUBINDEX	SDO error: object sub-index
%SW0466	--	3233	CO_SDO_ERROR_FUNCTION	SDO error: function (writing/reading)
%SW0468	--	3234	CO_SDO_ERROR_VALUE	SDO error: value
%SW0472	--	3236	CO_SDO_ERROR_CODE	SDO error: error code
%SW0480	--	3240	CO_EMCY_SLAVE_ID	Last reported EMCY: slave address
%SW0482	--	3241	CO_EMCY_DATA	Last reported EMCY: object data
Ethernet				
%SB0492	--	3246	ETH_MAC[6]	Physical Address
%SB0498	--	3249	ETH_STS_SPD_DUP	Ethernet Communication Mode
%SD0500	--	3250	ETH_STS_IP	IP address
%SD0504	--	3252	ETH_STS_MASK	Subnet Mask
%SD0508	--	3254	ETH_STS_GW	Default gateway
Modbus TCP				
%SW0512	--	3256	MBTCP_SERVER_REQUEST_COUNT	Counter of successful telegrams received by the PLC300 as Modbus TCP server
%SW0514	--	3257	MBTCP_SERVER_RESPONSE_COUNT	Counter of successful telegrams received by the PLC300 as Modbus TCP server
%SW0516	--	3258	MBTCP_SERVER_CNXNS	Number of Modbus TCP server connections active
%SB0520	--	3260	MBTCP_CLIENT_DISABLED	Disabled Modbus TCP client
%SW0522	--	3261	MBTCP_REQUEST_COUNT	Counter of the requests done by the Modbus TCP client
%SW0524	--	3262	MBTCP_RESPONSE_COUNT	Counter of successful responses received by the Modbus TCP client
%SW0526	--	3263	MBTCP_NO_ANSWER_COUNT	Counter of requests without answer received by the client (timeout)
%SW0528	--	3264	MBTCP_RESP_ERROR_COUNT	Counter of error responses received by the Modbus TCP client
%SW0530	--	3265	MBTCP_LAST_ERROR_TCP_PORT	Last detected error: Modbus TCP server TCP port
%SD0532	--	3266	MBTCP_LAST_ERROR_IP	Last detected error: Modbus TCP server IP address
%SB0536	--	3268 (L)	MBTCP_LAST_ERROR_UNITID	Last detected error: Modbus TCP server Unit ID
%SB0537	--	3268 (H)	MBTCP_LAST_ERROR_TYPE	Last detected error: error type
%SB0538	--	3269	MBTCP_LAST_ERROR_CODE	Last detected error: error code

Writing / Reading System Markers (Command)

Address	Bit	Modbus	Tag	Description
PLC300				
%CB0000	0	0	ERASE_RET	Erase retain markers
%CB0000	1	2	CLEAR_ENC_ALARM	Clear encoder fault alarm
%CB0000	2	3	BUZZER_ACTIVE	Buzzer active
Real Time Clock (RTC)				
%CW0030	--	3015	WR_HOUR	Write real time clock hour
%CW0032	--	3016	WR_MINUTE	Write real time clock minute
%CW0034	--	3017	WR_SECOND	Write real time clock seconds
%CW0036	--	3018	WR_DAY	Write real time clock day
%CW0038	--	3019	WR_MONTH	Write real time clock month
%CW0040	--	3020	WR_YEAR	Write real time clock year
CAN				
%CB0052	--	3026	CAN_ADDRESS	PLC300 address for CAN interface
%CB0055	--	3027	CAN_BAUDRATE	CAN interface baud rate
RS232				
%CB0061	--	3030	RS232_MODE	RS232 interface operation mode: 0=Modbus RTU 2=ASCII
%CB0062	--	3031 (L)	RS232_BYTE_FORMAT	RS232 interface parity and stop bits
%CB0063	--	3031 (H)	RS232_BAUDRATE	RS232 interface baud rate
%CB0114	0	912	RS232_RX_CLEAR	Start new ASCII reception
%CB0124	--	3062	RS232_ASCII_STRING	ASCII reception buffer
%CB0124	--	3062	RS232_ASCII_BYTEBUFFER	ASCII reception buffer
RS485				
%CB0068	--	3034 (L)	RS485_ADDRESS	PLC300 address for RS485 interface
%CB0069	--	3034 (H)	RS485_MODE	RS485 interface operation mode (master or slave)
%CB0070	--	3035 (L)	RS485_BYTE_FORMAT	RS485 interface parity and stop bits
%CB0071	--	3035 (H)	RS485_BAUDRATE	RS485 interface baud rate
Modbus RTU				
%CW0100	--	3050	MBUS_DISABLE_COMM	Disable Interface Modbus RTU Master
CANopen				
%CB0120	--	3060	CO_NMT_COMMAND	NMT command transmission by the CANopen master: command code
%CB0121	0	968	CO_NMT_TOGGLE	NMT command transmission by the CANopen master: toggle bit
%CB0121	7	975	CO_DISABLE	Disables the CANopen communication
%CB0122	--	3061	CO_NMT_SLAVE_ADDR	NMT command transmission by the CANopen master: slave address
Time, event and count task				
%CB0400	0	3200	INTERVAL_TASK1_DISABLE	Disable time task 1
%CB0400	1	3201	INTERVAL_TASK2_DISABLE	Disable time task 2
%CB0400	2	3202	INTERVAL_TASK3_DISABLE	Disable time task 3
%CB0400	3	3203	INTERVAL_TASK4_DISABLE	Disable time task 4
%CB0400	4	3204	INTERVAL_TASK5_DISABLE	Disable time task 5
%CB0400	5	3205	INTERVAL_TASK6_DISABLE	Disable time task 6
%CB0400	6	3206	INTERVAL_TASK7_DISABLE	Disable time task 7
%CB0400	7	3207	INTERVAL_TASK8_DISABLE	Disable time task 8
%CB0401	0	3208	INTERVAL_TASK9_DISABLE	Disable time task 9
%CB0401	1	3209	INTERVAL_TASK10_DISABLE	Disable time task 10
%CB0401	2	3210	INTERVAL_TASK11_DISABLE	Disable time task 11
%CB0401	3	3211	INTERVAL_TASK12_DISABLE	Disable time task 12
%CB0401	4	3212	INTERVAL_TASK13_DISABLE	Disable time task 13

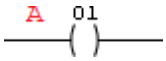
11.8.5 Ladder

11.8.5.1 Coil

11.8.5.1.1 DIRECTCOIL

Logical block used to assign direct values of the output variables.

Ladder Representation



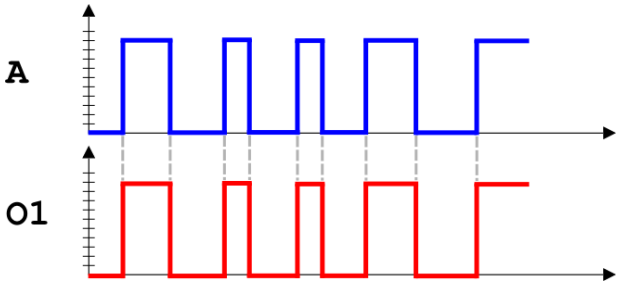
Block Structure

Variable Type	Name	Data Type	Description
VAR_OUTPUT	O1	BOOL	Block log output

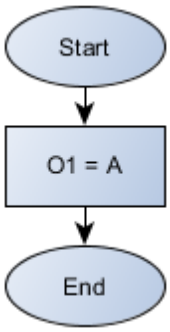
Operation

The block transfers the value of A for the memory address corresponding to O1.

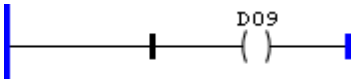
Diagram



Block Flowchart



Example

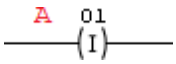


The above example keeps the digital output DO9 permanently connected, because the value of A in this case is the value of the left bus which is always considered high logic level (TRUE).

11.8.5.1.2 IMMEDIATECOIL

Logical block used for assigning values to standard digital outputs instantly.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_OUTPUT	O1	BOOL	Block log output

Operation

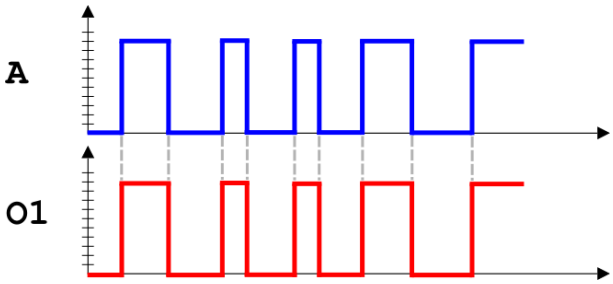
The block transfers the value of A for the digital output corresponding to O1.
Unlike the direct coil, this block does not wait until the end of the scan cycle so that the output value is updated; this is done at the same time the block is activated.

NOTE!
This block only works with standard digital outputs of the product.

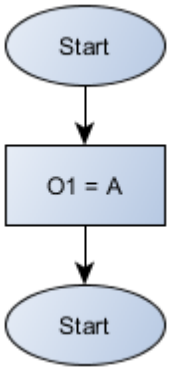
Compatibility

Device	Version
PLC300	1.20 or higher
SCA06	2.00 or higher

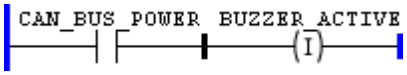
Diagram



Block Flowchart



Example

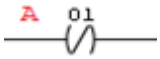


The above example immediately activates the internal buzzer when it detects that the power of the CANopen bus was stopped and remains on until the power is restored.

11.8.5.1.3 INVERTEDCOIL

Logical block used for assigning values denied to output variables.

Ladder Representation



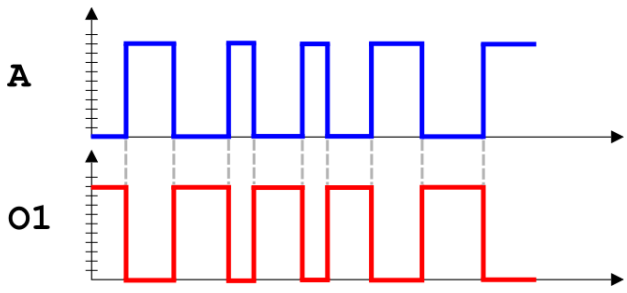
Block Structure

Variable Type	Name	Data Type	Description
VAR_OUTPUT	O1	BOOL	Block log output

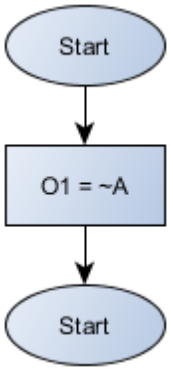
Operation

The block transfers the denied value of A for the memory address corresponding to O1.

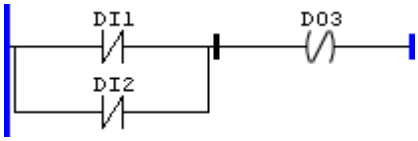
Diagram



Block Flowchart



Example

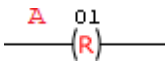


The above example disables the digital output DO3 when some of the digital inputs DI1 and DI2 are with FALSE value. When both inputs are with a TRUE value, DO3 activates.

11.8.5.1.4 RESETCOIL

Logical block used for indefinite disabling of output variables.

Ladder Representation



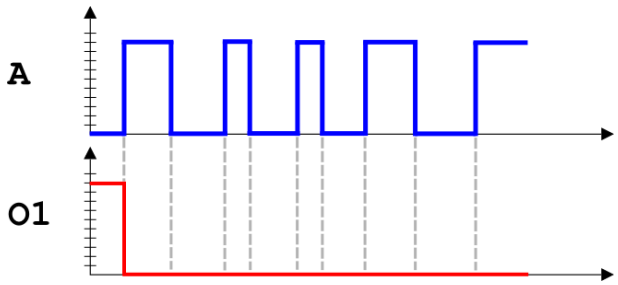
Block Structure

Variable Type	Name	Data Type	Description
VAR_OUTPUT	O1	BOOL	Block log output

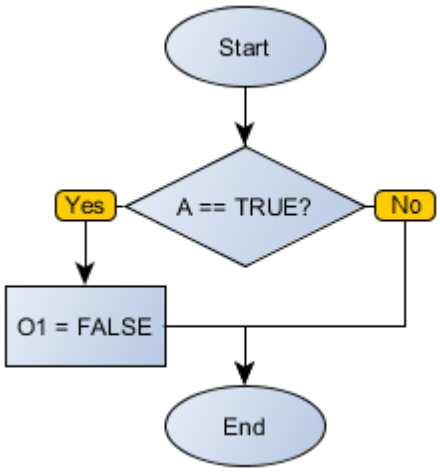
Operation

When identifying a TRUE value in A, this block transfers a FALSE value to the memory address corresponding to O1.
When identifying a FALSE value in A, this block performs no operation.

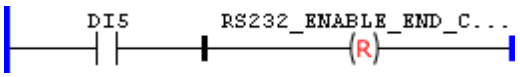
Diagram



Block Flowchart



Example

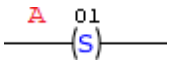


The example above activates permanently the system control marker that enables end-of-message character in RS232 communication to identify a TRUE level at the digital input DI5.

11.8.5.1.5 SETCOIL

Logical block used for indefinite enabling of output variables.

Ladder Representation



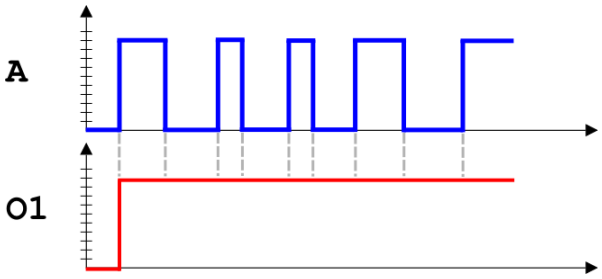
Block Structure

Variable Type	Name	Data Type	Description
VAR_OUTPUT	O1	BOOL	Block log output

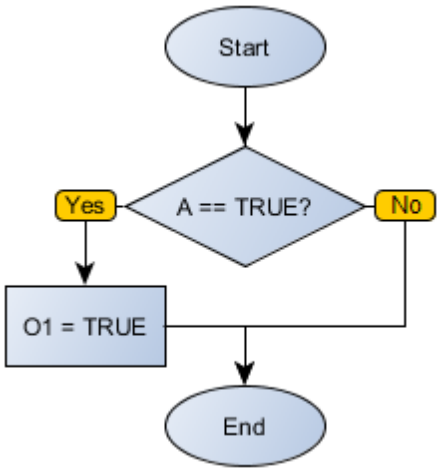
Operation

When identifying a TRUE value in A, this block transfers the value of A for the memory address corresponding to O1.
When identifying a FALSE value in A, this block performs no operation.

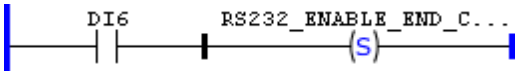
Diagram



Block Flowchart



Example

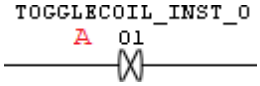


The example above activates permanently the system control marker that enables end-of-message character in RS232 communication to identify a TRUE level at the digital input DI6.

11.8.5.1.6 TOGGLECOIL

Logical block used for output variables alternance.

Ladder Representation



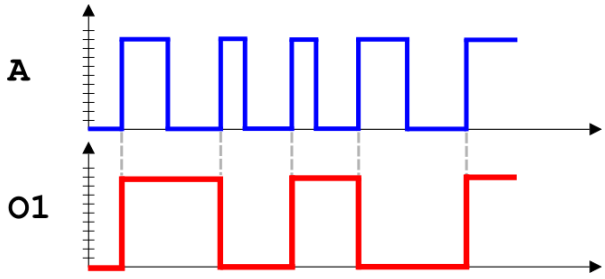
Block Structure

Variable Type	Name	Data Type	Description
VAR_OUTPUT	O1	BOOL	Block log output
VAR	TOGGLECOIL_INST_0	TOGGLECOIL	Instance of access to block structure

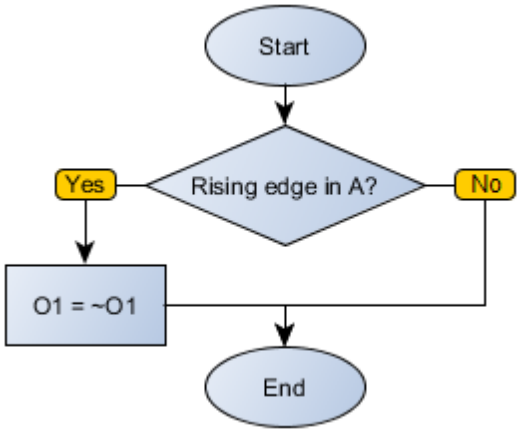
Operation

When identifying a transition from FALSE to TRUE (leading edge) on A, the block reverses the status of O1.

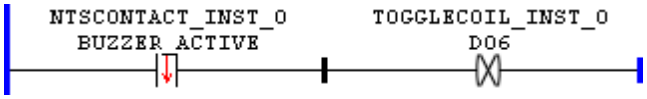
Diagram



Block Flowchart



Example



The above example inverts the state of the digital output DO6 to each disabling the internal buzzer.

11.8.5.2 Communication Network

11.8.5.2.1 CANopen

11.8.5.2.1.1 CANopen Overview

Operation in the CANopen Network - Master Mode

Besides the operation as a slave, the PLC300 programmable controller also allows operation as a master for the CANopen network. PLC300 characteristics and functions as a master for the CANopen network will be described as follows.

Enabling Function CANopen Master

By default, PLC300 programmable controller is programmed to operate as a slave for the CANopen network. Programming the equipment as a master for the network must be done by using the WSCAN software that also allows the configuration of the whole CANopen network. A detailed description of the WSCAN software windows and functions must be obtained in the menu "Help" in the software itself.

After the master configuration has been created, it is necessary to download the configurations by using one of the product's programming interfaces - refer to the user manual for further information. Once programmed as a master for the network, in case it is necessary to delete said configurations, the function to delete the user program - available in the Setup menu - also deletes the CANopen master configurations.

NOTE!
CANopen network is a flexible network that allows several different ways of configuring and operating. Nonetheless, such a flexibility requires the user to have a good knowledge of the communication functions and objects used to configure the network, as well as knowledge of the WSCAN programming software.

CANopen Master Characteristics

PLC300 programmable controller allows the control of a group of up to 63 slaves, using the following communication tasks and resources:

- Network management task (NMT)
- 63 transmission PDOs
- 63 reception PDOs
- 63 Heartbeat Consumers
- Heartbeat Producer
- SDO Client
- SYNC producer/consumer
- 512 bytes of input network markers
- 512 bytes of output network markers

Physical characteristics - installation, connector, cable, etc. - are the same for PLC300 operating both as a master and a slave. Address configurations and communication rate are also necessary for the operation as a master, but these configurations are programmed by the WSCAN software according to the properties defined for the master in the software itself.

NOTE!
Input network markers are used to map data in RPDOs, while output network markers are used to map data in TPDOs. They can be accessed in Byte (%IB or %QB), Word (%IW or %QW), or Double Word (%ID or %QD). Nonetheless, their function is not pre-defined and depends on the Ladder application developed for the PLC300 controller.

Master Operation

Once programmed to operate as a master, the PLC300 programmable controller will perform the following steps in order to perform the sequential initialization for each one of the slaves.

1. By sending the communication reset command to the whole network, so that the slaves initialize with values known for the communication objects.
2. Equipment identification in the network, through the reading via SDO of the 1000h/00h object - Object Identification.
3. Writing via SDO of all objects programmed for the slave, which usually include the configuration and mapping of TPDOs and RPDOs, node guarding, heartbeat, besides the manufacturer's specific objects, in case they are programmed.
4. Error control task initialized - node guarding or heartbeat - in case they are programmed.
5. Sending of the slave to the operating mode.

If one of these steps fails, communication error with the slave will be indicated. Depending on the configurations, the initialization of slaves will be aborted, and the master will initialize the following slave, returning to the slave presenting error, after trying to initialize all other slaves in the network.

Similarly, if during the operation of a slave, an error in the error control task is identified, depending on the configurations made for the master, the slave will be automatically reset and the initialization procedure will be performed over.

NOTE!
The communication state and the state of each slave can be observed in input system markers.

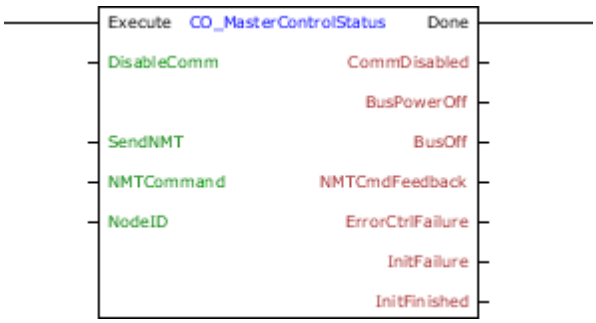
Blocks for the CANopen Master

Besides the communication and configuration objects made in the WSCAN software, blocks for the monitoring and sending of commands, which can be used during the creation of the application in Ladder for the PLC300 programmable controller, are also available. It is not necessary to use said blocks during the equipment operation, but its use provides more flexibility and facilitates the diagnosis of communication problems during the PLC300 programmable controller operation.

11.8.5.2.1.2 CO_MasterControlStatus

Block that allows monitoring various statuses of the CANopen network master.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	Execute	BOOL	Block enabling
	DisableComm	BOOL	Disables CANopen communication
	SendNMT	BOOL	Triggers management command sending
	NMTCommand	BYTE	Management command code to be sent
	NodeID	BYTE USINT	Slave address for sending the NMT
VAR_OUTPUT	Done	BOOL	Output enabling
	CommDisabled	BOOL	Disabled communication flag
	BusPowerOff	BOOL	Flag of a power failure of CAN interface
	BusOff	BOOL	Flag indicating BusOff error at the CAN interface
	NMTCmdFeedback	BOOL	Flag of NMT command sent by the master
	ErrorCtrlFailure	BOOL	Flag indicating error of nodeguarding or heartbeat in a slave network
	InitFailure	BOOL	Flag indicating error in the initialization of the slavee network
	InitFinished	BOOL	Flag indicating the initialization of the slaves was completed

Operation

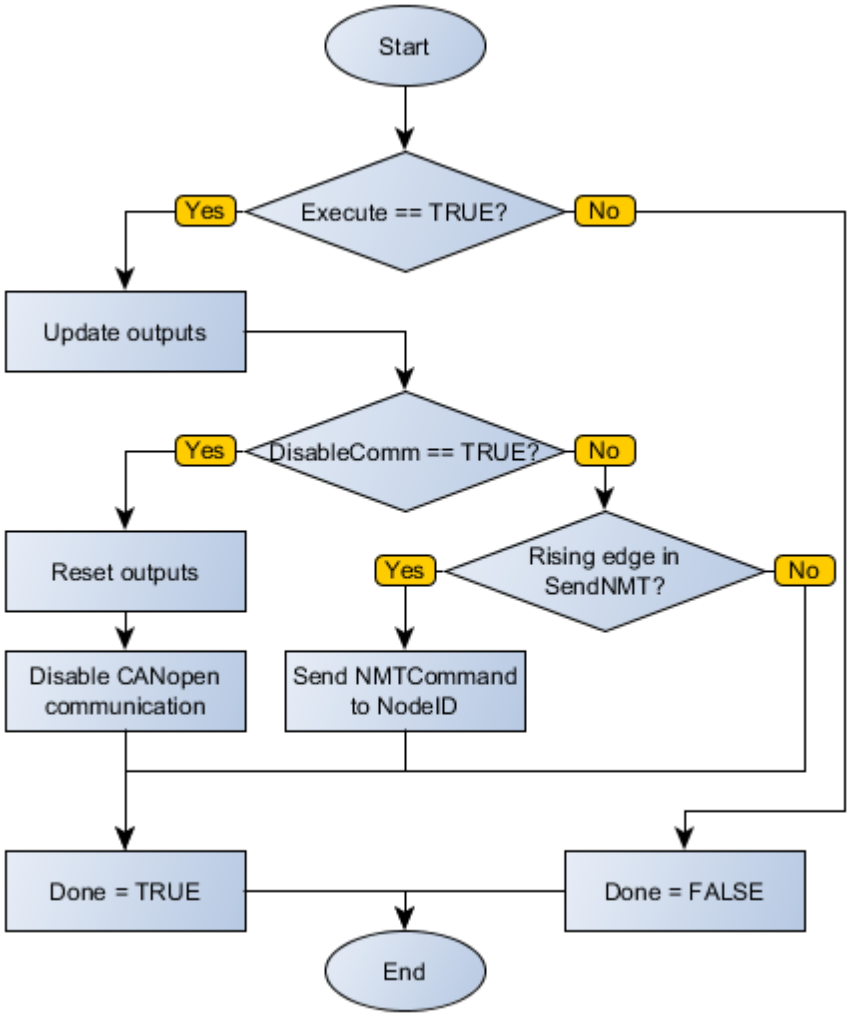
This block remains active while Execute is at TRUE level, updating its outputs according to the monitoring of the master and input requests. When Execute receives FALSE level, the inputs are ignored and the outputs are zeroed. The Done output receives TRUE level when Execute has TRUE level and block finished its execution.

A TRUE level DisableComm disables the CANopen communication and resets the status counters and markers of the master.

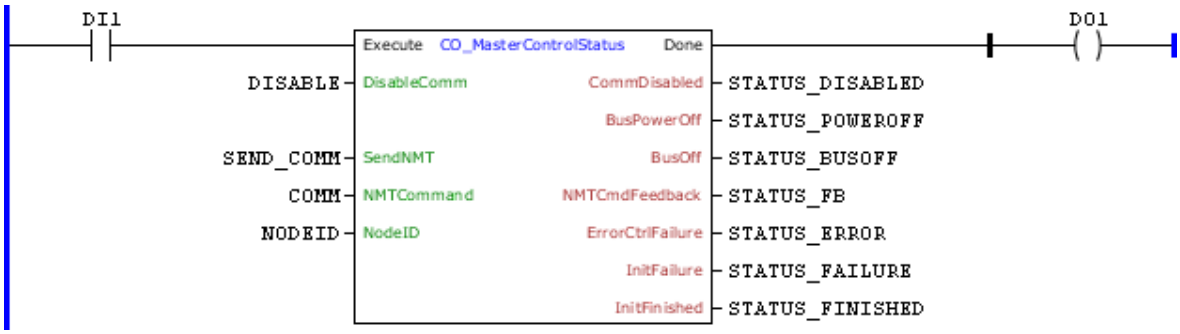
A leading edge on SendNMT sends a command management (NMT) indicated at NMTCommand to the slave of NodeID address.

Outputs receive TRUE level when the status of the CANopen master matches description (disabled communication, power failure, etc.).

Block Flowchart



Example

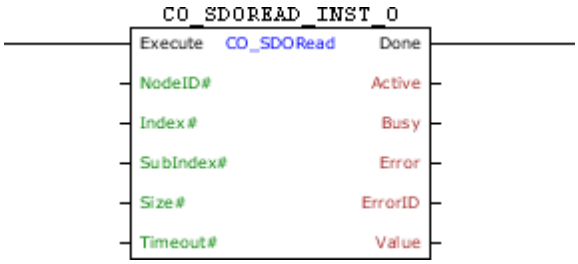


The example above requests status data of the CANopen network master, and allows disabling communication through DISABLE and to send commands NMT to NODEID through SEND_COMM. The block ends successfully, Done output is activated.

11.8.5.2.1.3 CO_SDORead

Block that performs a reading of data via SDO from a remote slave in CANopen network.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	Execute	BOOL	Block enabling
	NodeID#	BYTE	Slave address
	Index#	WORD	Index of the object to be accessed in slave
	SubIndex#	BYTE	Sub-index of the object
	Size#	BYTE	Size of data accessed, in bytes
	Timeout#	WORD	Maximum w aiting time for arrival of data, from the beginning of the request [ms]
VAR_OUTPUT	Done	BOOL	Output enabling
	Active	BOOL	Aw aiting response flag
	Busy	BOOL	Flag of the SDO client is busy w ith another request
	Error	BOOL	Error in the execution flag
	ErrorID	BYTE USINT	Identifier of the occurred error
	Value	BYTE USINT	Variable that stores the received data
VAR	CO_SDOREAD_INST_0	CO_SDOREAD	Instance of access to block structure

Operation

When this block detects a leading edge on Execute it checks whether the SDO client in the specified NodeID # address is free to send data (Busy variable at FALSE level). If so, it sends the reading request to the object of Size# size located in Index# and SubIndex# and sets the Active output, resetting it when receiving the response from the slave. The received data is stored in the Value variable. If the slave is not free, the block waits Busy go to FALSE level to resubmit the request.

NOTE!
If Execute goes to FALSE level and Busy is still at TRUE level, the request is canceled.

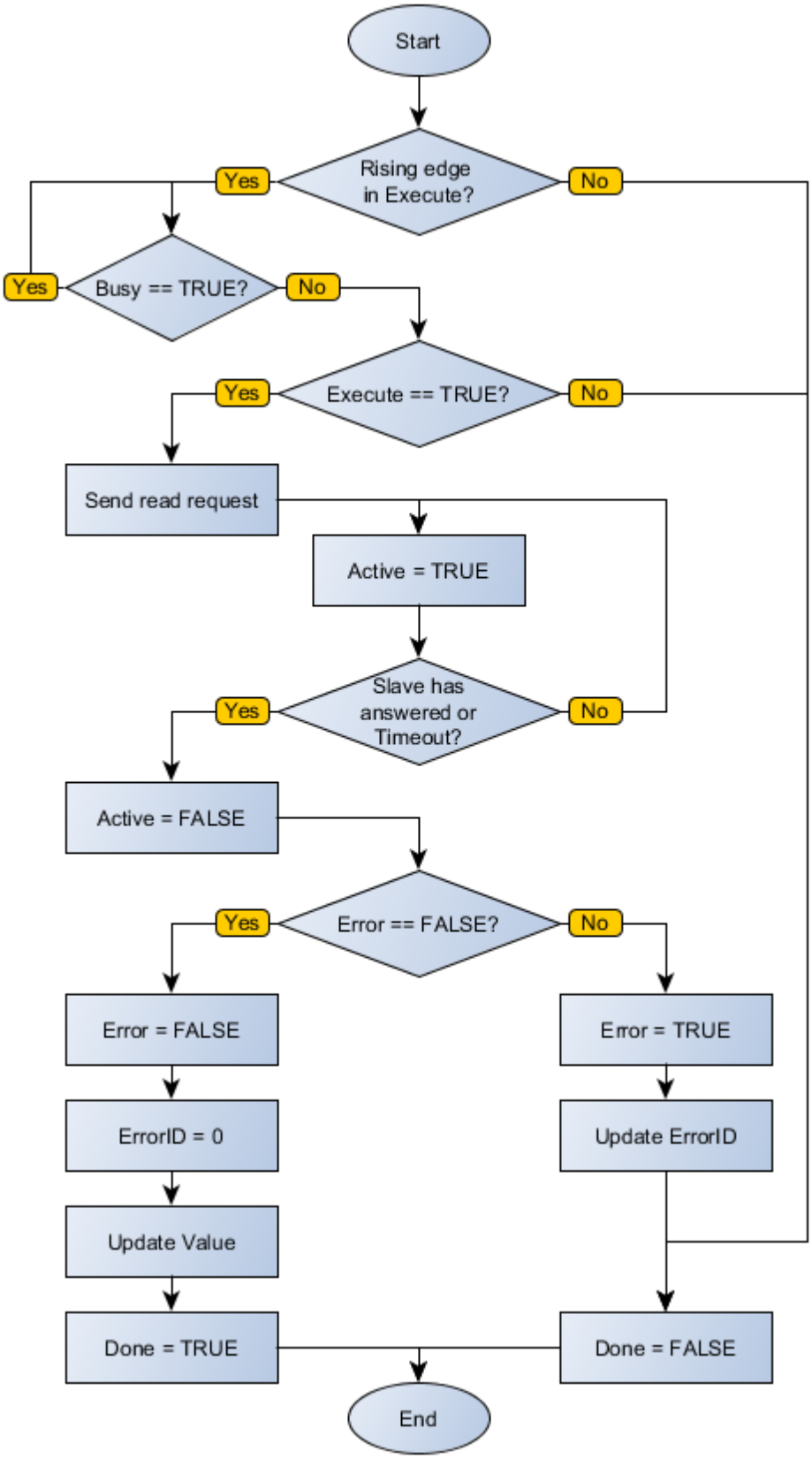
NOTE!
Value is an array of size equal to Size#. It is important to check this compatibility not to generate errors in the block.

When Execute has FALSE value, Done remains FALSE. The Done output is only activated when the block finishes executing successfully, remaining at TRUE level until Execute receives FALSE.

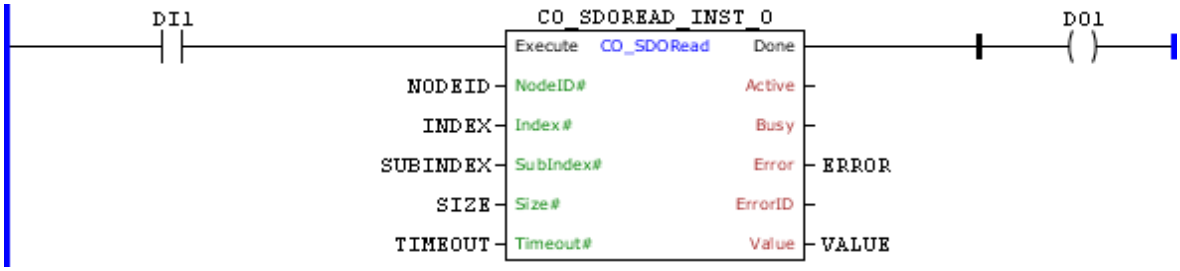
If there is any error in the execution, the Error output is enabled and ErrorID displays an error code according to the table below.

Code	Description
0	Executed successfully
1	Card cannot execute the function
2	Timeout in slave response
3	Slave returned error

Block Flowchart


Start
Rising edge in Execute?
Yes
No
Busy == TRUE?
Yes
No
Execute == TRUE?
Yes
No
Send read request
Active = TRUE
Slave has answered or Timeout?
Yes
No
Active = FALSE
Error == FALSE?
Yes
No
Error = FALSE
ErrorID = 0
Update Value
Done = TRUE
Error = TRUE
Update ErrorID
Done = FALSE
End

Example

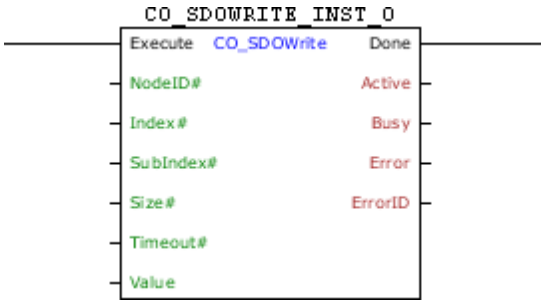


The example above requests reading of the data size SIZE, located in INDEX - SUBINDEX, of the NODEID device. This data is forwarded to VALUE. The block ends successfully, Done output is activated.

11.8.5.2.1.4 CO_SDOWrite

Block that performs a writing of data via SDO from a remote slave in CANopen network.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	Execute	BOOL	Block enabling
	NodeID#	BYTE	Slave address
	Index#	WORD	Index of the object to be accessed in slave
	SubIndex#	BYTE	Sub-index of the object
	Size#	BYTE	Size of data accessed, in bytes
	Timeout#	WORD	Maximum waiting time for arrival of data, from the beginning of the request [ms]
	Value	BYTE USINT	Variable that has the data to be written
VAR_OUTPUT	Done	BOOL	Output enabling
	Active	BOOL	Awaiting response flag
	Busy	BOOL	Flag of the SDO client is busy with another request
	Error	BOOL	Error in the execution flag
	ErrorID	BYTE USINT	Identifier of the occurred error
VAR	CO_SDOWRITE_INST_0	CO_SDOWRITE	Instance of access to block structure

Operation

When this block detects a leading edge on Execute it checks whether the SDO client in the specified NodeID # address is free to send data (Busy variable at FALSE level). If so, it sends the writing request to the object of Size# size located in Index# and SubIndex# and sets the Active output, resetting it when receiving the response from the slave. If the slave is not free, the block waits Busy go to FALSE level to resubmit the request.

NOTE!
If Execute goes to FALSE level and Busy is still at TRUE level, the request is canceled.

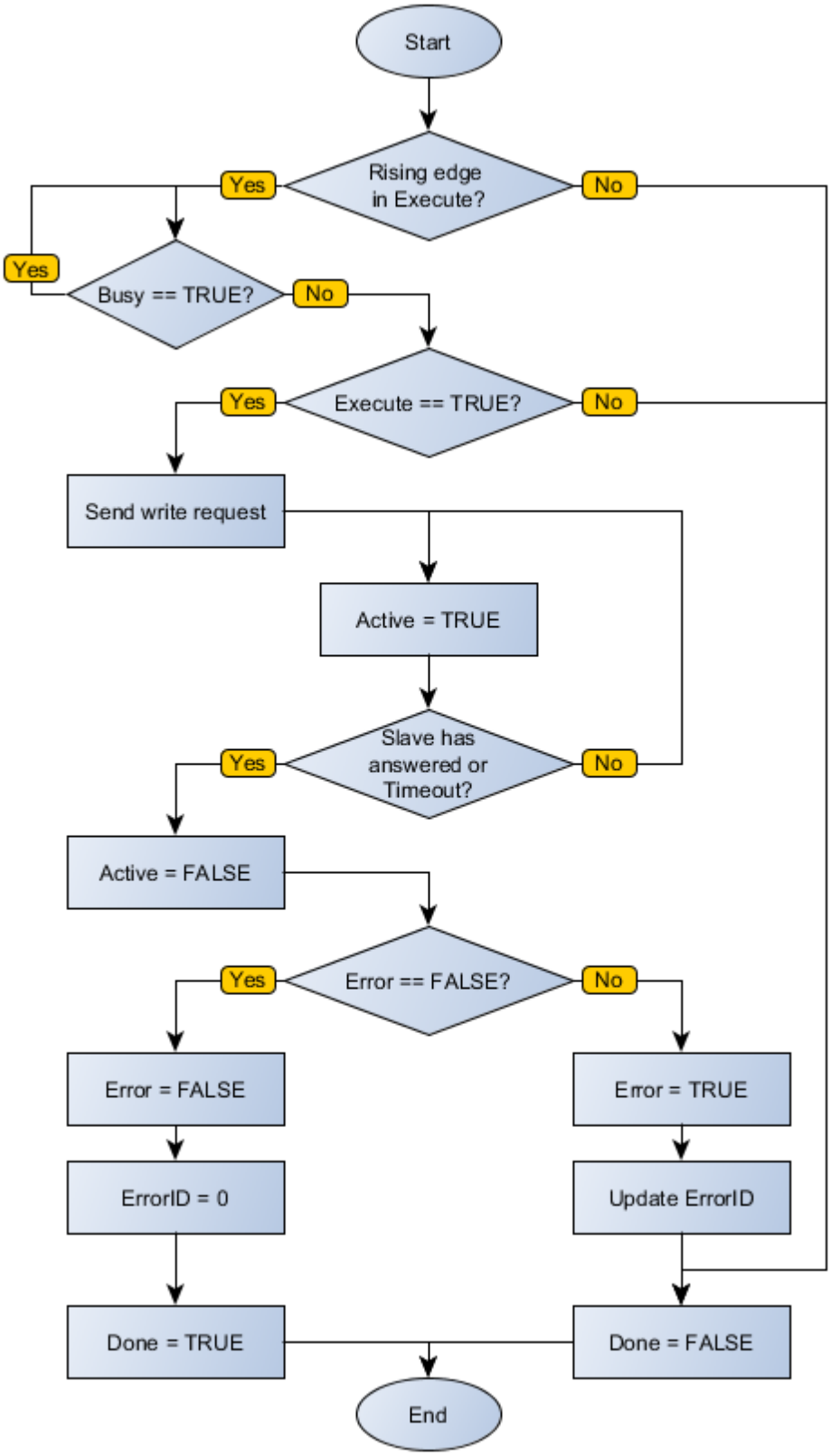
NOTE!
Value is an array of size equal to Size#. It is important to check this compatibility not to generate errors in the block.

When Execute has FALSE value, Done remains FALSE. The Done output is only activated when the block finishes executing successfully, remaining at TRUE level until Execute receives FALSE.

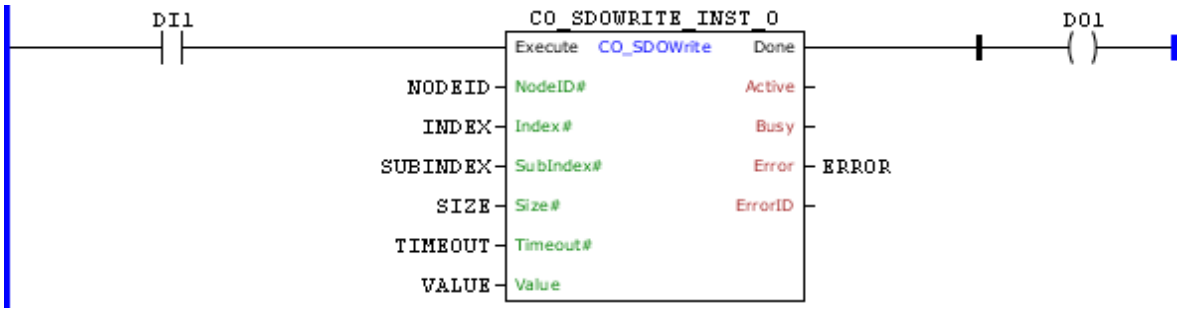
If there is any error in the execution, the Error output is enabled and ErrorID displays an error code according to the table below.

Code	Description
0	Executed successfully
1	Card cannot execute the function
2	Timeout in slave response
3	Slave returned error

Block Flowchart


Start
Rising edge in Execute?
Yes
No
Yes
Busy == TRUE?
No
Yes
Execute == TRUE?
No
Send write request
Active = TRUE
Slave has answered or Timeout?
Yes
No
Active = FALSE
Error == FALSE?
Yes
No
Error = FALSE
Error = TRUE
ErrorID = 0
Update ErrorID
Done = TRUE
Done = FALSE
End

Example

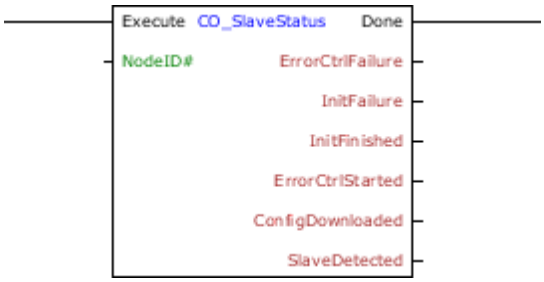


The example above requests writing of the data size VALUE, located in INDEX - SUBINDEX, of the NODEID device. The block ends successfully, Done output is activated.

11.8.5.2.1.5 CO_SlaveStatus

Block that allows monitoring various statuses of the CANopen network master.

Ladder Representation



Block Structure

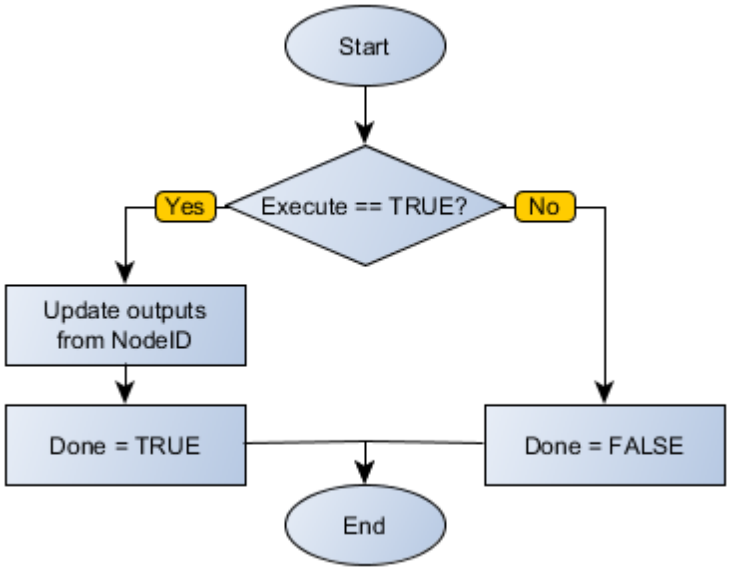
Variable Type	Name	Data Type	Description
VAR_INPUT	Execute	BOOL	Block enabling
	NodeID	BYTE USINT	Slave address for monitoring
VAR_OUTPUT	Done	BOOL	Output enabling
	ErrorCtrlFailure	BOOL	Flag indicating error of nodeguarding or heartbeat in a slave network
	InitFailure	BOOL	Flag indicating error in the initialization of the slavee network
	InitFinished	BOOL	Flag indicating the initialization of the slaves was completed
	ErrorCtrlStarted	BOOL	Flag of start of the error control service
	ConfigDownloaded	BOOL	Flag of success in the download of SDO settings in the slave
	SlaveDetected	BOOL	Flag of success in reading slave ID via SDO

Operation

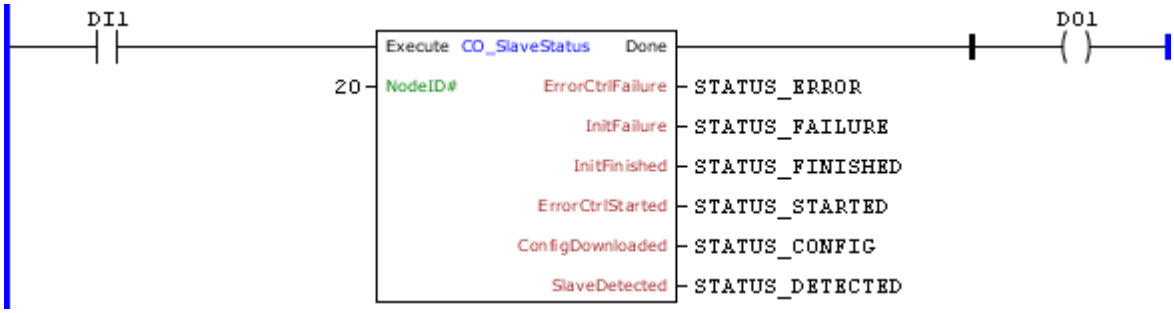
This block remains active while Execute is at TRUE level, updating its outputs according to the monitoring of the slave. When Execute receives FALSE level, the inputs are ignored and the outputs are zeroed. The Done output receives TRUE level when Execute has TRUE level and block finished its execution.

Outputs receive TRUE level when the status of the CANopen slave indicated by address NodeID matches description (boot error, download success, etc.).

Block Flowchart



Example



The example above requests data from the slave 20 of the CANopen network. The block ends successfully, Done output is activated.

11.8.5.2.2 Modbus RTU

11.8.5.2.2.1 Modbus RTU Overview

Operation in the Modbus RTU Network - Master Mode

Besides the operation as a slave, the PLC300 programmable controller also allows operation as a master for the Modbus RTU network. For this operation, it is necessary to observe the following points:

- Only interface RS485 allows operation as a network master.
- It is necessary to program, in product configurations, the operation mode as "Master", besides the communication rate, parity, and stop bits, which must be the same for the whole equipment in the network.
- The Modbus RTU network master does not have an address, so the address configured in the PLC300 is not used.
- Sending and receiving telegrams via RS485 interface using the Modbus RTU is programmed by using blocks in Ladder programming language. It is necessary to know the available blocks and the Ladder programming software in order to be able to program the network master.
- The following functions are available for the sending of requisitions by the Modbus master:
  - Function 01: Read Coils
  - Function 02: Read Discrete Inputs
  - Function 03: Read Holding Registers
  - Function 04: Read Input Registers
  - Function 05: Write Single Coil
  - Function 06: Write Single Register
  - Function 15: Write Multiple Coils
  - Function 16: Write Multiple Registers

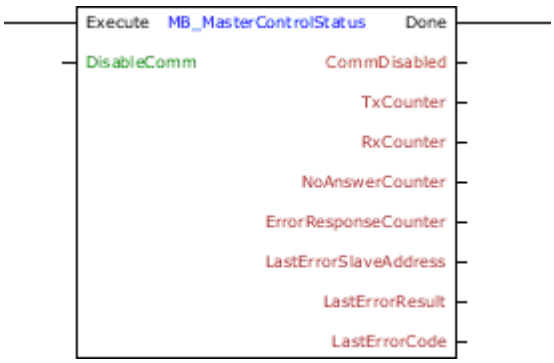
Blocks to program the master

In order to control and monitor the Modbus RTU communication using the PLC300 programmable controller, the following blocks were developed, and they must be used when programming in Ladder.

11.8.5.2.2.2 MB_MasterControlStatus

Block that allows monitoring various statuses of the Modbus RTU network master.

Ladder Representation



Block Structure

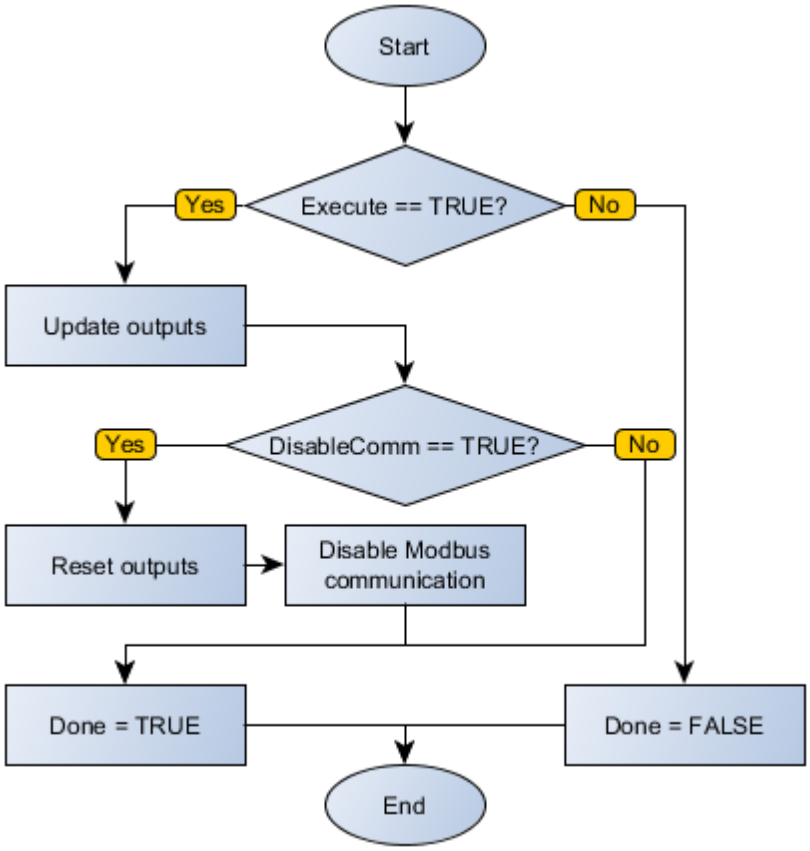
Variable Type	Name	Data Type	Description
VAR_INPUT	Execute	BOOL	Block enabling
	DisableComm	BOOL	Disables Modbus RTU communication
VAR_OUTPUT	Done	BOOL	Output enabling
	CommDisabled	BOOL	Disabled communication flag
	TxCounter	WORD UINT	Counter of requests sent
	RxCounter	WORD UINT	Counter of telegrams received
	NoAnswerCounter	WORD UINT	Counter of requests not answered
	ErrorResponseCounter	WORD UINT	Counter of responses received with error information
	LastErrorSlaveAddress	BYTE USINT	Slave address in which the last communication error was detected
	LastErrorResult	BYTE USINT	Operation result of the last communication error received (0 = No error) (4 – Response Timeout) (5 = Slave returned error)
	LastErrorCode	BYTE USINT	Code of the last communication error received

Operation

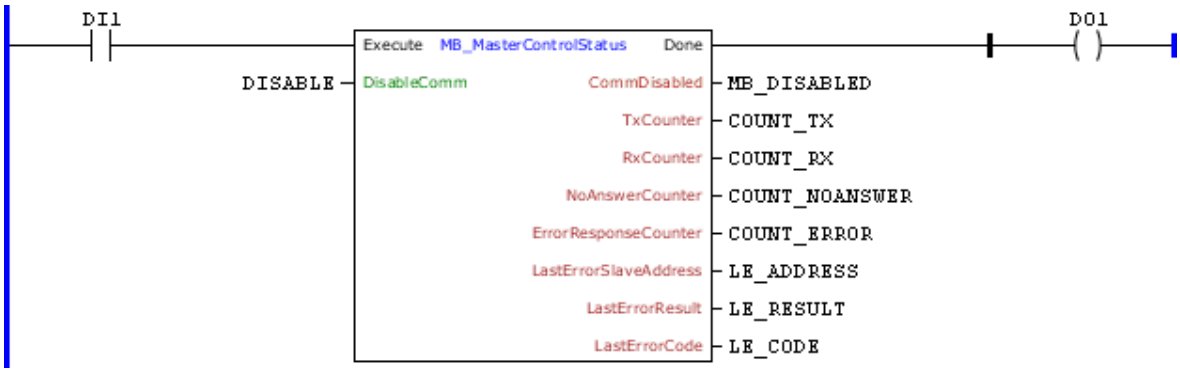
This block remains active while Execute is at TRUE level, updating its outputs according to the monitoring of the master and input requests. When Execute receives FALSE level, the inputs are ignored and the outputs are zeroed. The Done output receives TRUE level when Execute has TRUE level and block finished its execution.

A TRUE level DisableComm disables the Modbus RTU communication and resets the status counters and markers of the master. These markers and counters are displayed in the output block each having some data corresponding to its description. Their values are also cleared at shutdown of the master.

Block Flowchart



Example

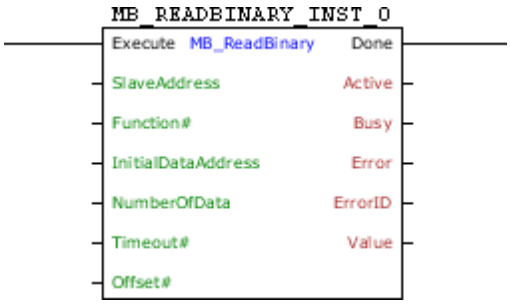


The example above requests status data of the Modbus RTU network master, and allows disabling communication through DISABLE. The block ends successfully, Done output is activated.

11.8.5.2.2.3 MB_ReadBinary

Block that performs a reading of up to 128 binary data (via Read Coils or Read Discrete Inputs) of a slave on the Modbus RTU network.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	Execute	BOOL	Block enabling
	SlaveAddress	BYTE	Slave address
	Function#	BYTE	Reading function code
	InitialDataAddress	WORD	Initial bit address of the data to be read
	NumberOfData	BYTE	Number of bits to be read (1 to 128)
	Timeout#	WORD	Maximum w aiting time for the slave response [ms]
	Offset#	BOOL	Offset Indication in InitialDataAddress, i.e., need to subtract 1 from this number
VAR_OUTPUT	Done	BOOL	Output enabling
	Active	BOOL	Aw aiting response flag
	Busy	BOOL	Flag indicating the RS485 interface is busy w ith another request
	Error	BOOL	Error in the execution flag
	ErrorID	BYTE	Identifier of the occurred error
	Value	BOOL	Variable that stores the received data
VAR	MB_READBINARY_INST_0	MB_READBINARY	Instance of access to block structure

Operation

When this block detects a leading edge on Execute, it checks whether the Modbus slave RTU in specified address in SlaveAddress is free to send data (Busy variable at FALSE level). If so, it sends the reading request of a number of bits indicated by NumberOfData in InitialDataAddress address using chosen function in Function# and sets the Active output, resetting it when receiving the response from the slave. The received data is stored in the Value variable. If the slave is not free, the block waits Busy go to FALSE level to resubmit the request.

NOTE!
If Execute goes to FALSE level and Busy is still at TRUE level, the request is canceled.

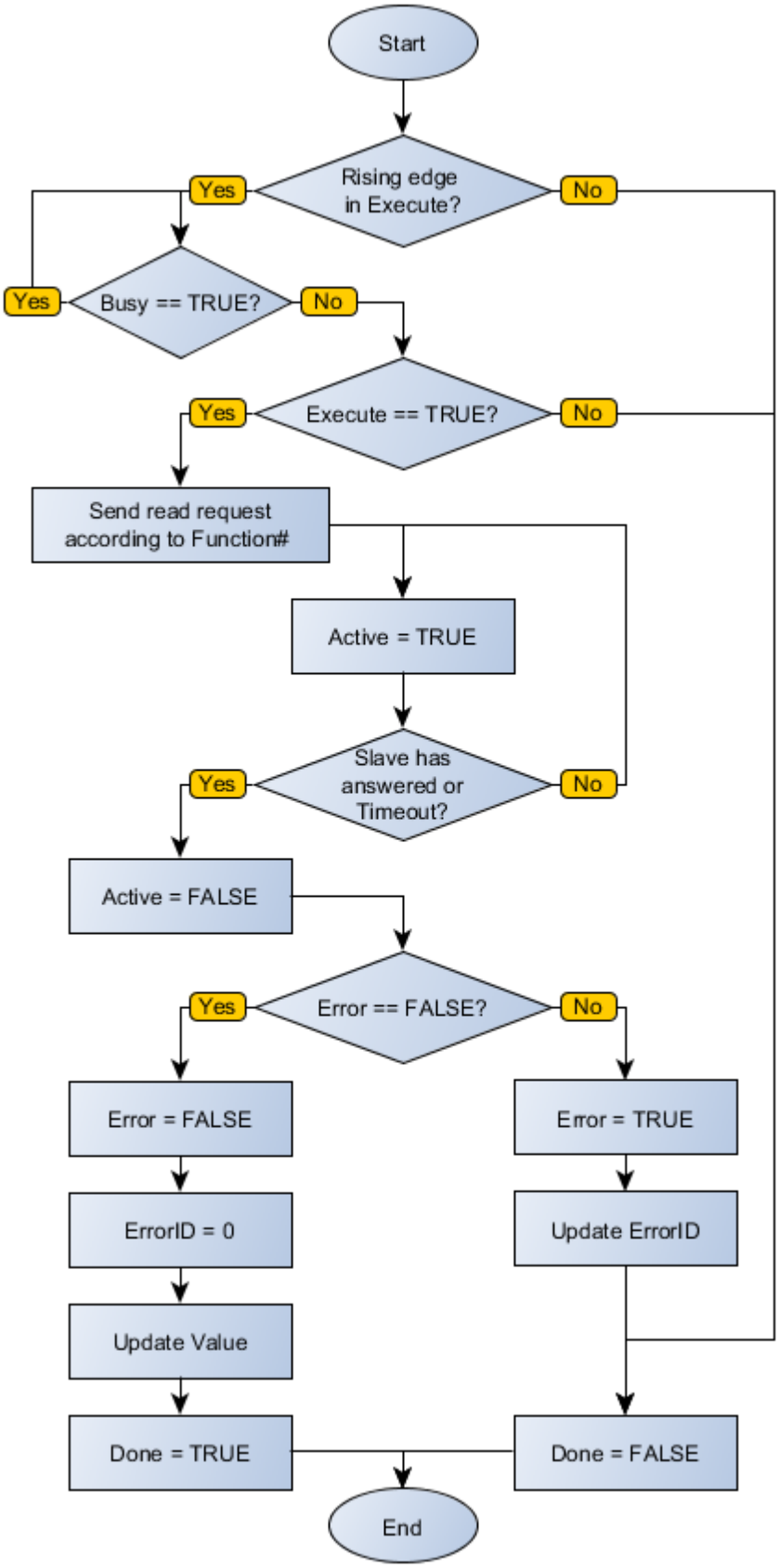
NOTE!
Value is an array of size equal to NumberOfData. It is important to check this compatibility not to generate errors in the block.

When Execute has FALSE value, Done remains FALSE. The Done output is only activated when the block finishes executing successfully, remaining at TRUE level until Execute receives FALSE.

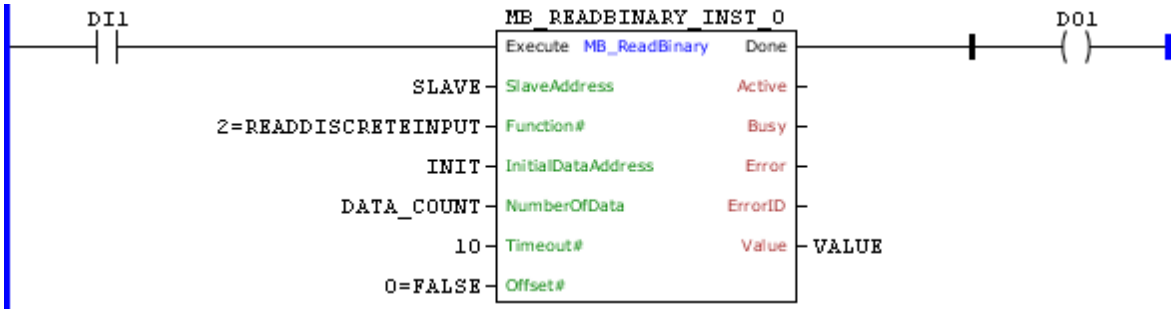
If there is any error in the execution, the Error output is enabled and ErrorID displays an error code according to the table below.

Code	Description
0	Executed successfully
1	Invalid input data
2	Master not enabled
4	Timeout in slave response
5	Slave returned error

Block Flowchart


Start
Rising edge in Execute?
Yes
No
Busy == TRUE?
Yes
No
Execute == TRUE?
Yes
No
Send read request according to Function#
Active = TRUE
Slave has answered or Timeout?
Yes
No
Active = FALSE
Error == FALSE?
Yes
No
Error = FALSE
ErrorID = 0
Update Value
Done = TRUE
Error = TRUE
Update ErrorID
Done = FALSE
End

Example

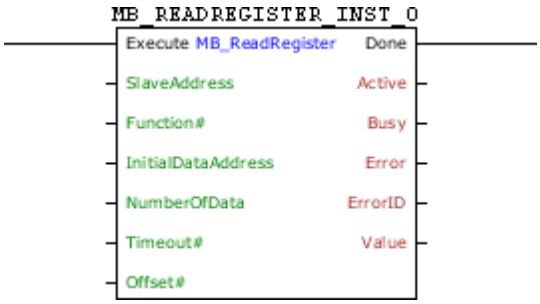


The above example requests reading of a number of binary data described by DATA_COUNT positioned in the INIT Modbus RTU slave of SLAVE address through the Read Discrete Input function. These data are forwarded to VALUE. The block ends successfully, Done output is activated.

11.8.5.2.2.4 MB_ReadRegister

Block that performs a reading of up to 64 16-bit registers (via Read Holding Registers or Read Input Registers) of a slave on the Modbus RTU network.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	Execute	BOOL	Block enabling
	SlaveAddress	BYTE	Slave address
	Function#	BYTE	Reading function code
	InitialDataAddress	WORD	Initial register address to be read
	NumberOfData	BYTE	Number of registers to be read (1 to 64)
	Timeout#	WORD	Maximum waiting time for the slave response [ms]
	Offset#	BOOL	Offset Indication in InitialDataAddress, i.e., need to subtract 1 from this number
VAR_OUTPUT	Done	BOOL	Output enabling
	Active	BOOL	Awaiting response flag
	Busy	BOOL	Flag indicating the RS485 interface is busy with another request
	Error	BOOL	Error in the execution flag
	ErrorID	BYTE	Identifier of the occurred error
	Value	BYTE SINT USINT WORD UINT INT DWORD UDINT DINT REAL	Variable that stores the received data
VAR	MB_READREGISTER_INST_0	MB_READREGISTER	Instance of access to block structure

Operation

When this block detects a leading edge on Execute, it checks whether the Modbus RTU slave in specified address in SlaveAddress is free to send data (Busy variable at FALSE level). If so, it sends the reading request of a number of registers indicated by NumberOfData in InitialDataAddress address using chosen function in Function# and sets the Active output, resetting them when receiving the response from the slave. The received data is stored in the Value variable. If the slave is not free, the block waits Busy go to FALSE level to resubmit the request.

NOTE!
If Execute goes to FALSE level and Busy is still at TRUE level, the request is canceled.

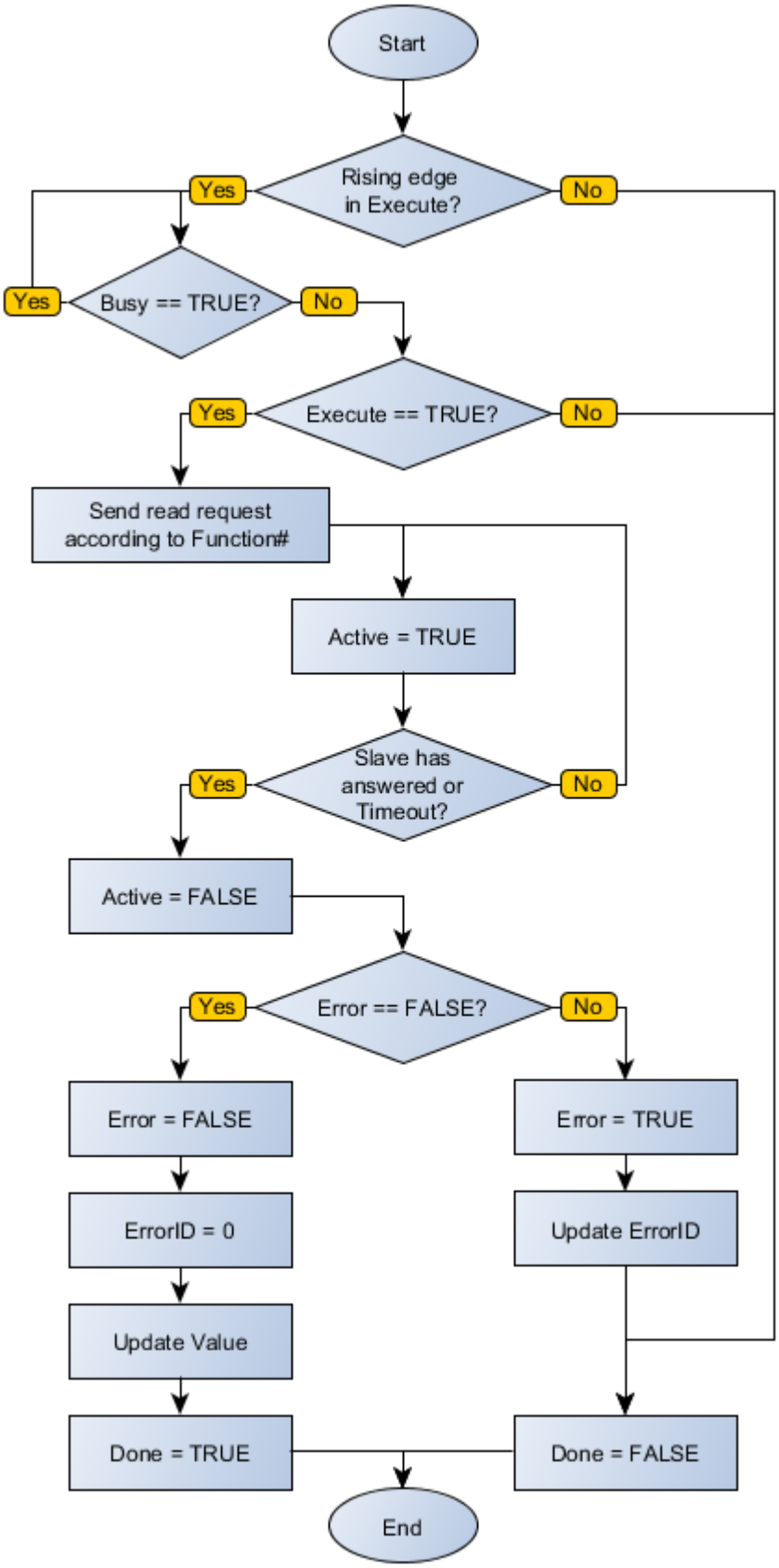
NOTE!
Value is an array of number of bits NumberOfData multiplied by 16. That is, if NumberOfData is 16, Value can be an array of 32 BYTE positions, 16 WORD positions or 8 DWORD positions. It is important to check this compatibility not to generate errors in the block.

When Execute has FALSE value, Done remains FALSE. The Done output is only activated when the block finishes executing successfully, remaining at TRUE level until Execute receives FALSE.

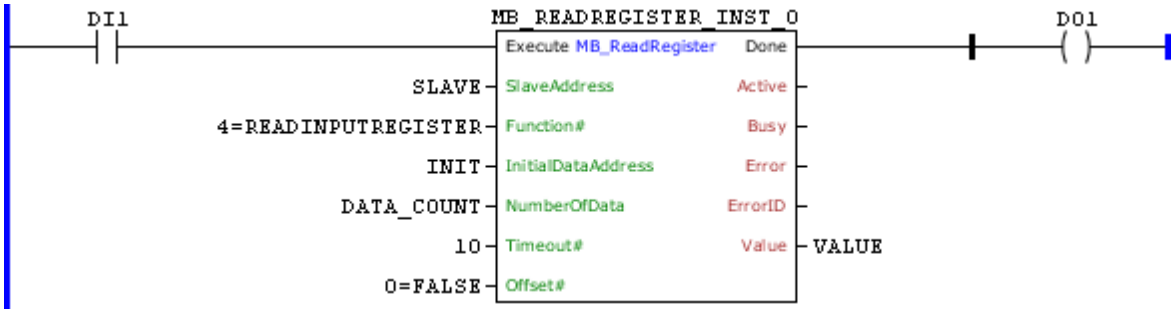
If there is any error in the execution, the Error output is enabled and ErrorID displays an error code according to the table below.

Code	Description
0	Executed successfully
1	Invalid input data
2	Master not enabled
4	Timeout in slave response
5	Slave returned error

Block Flowchart


Start
Rising edge in Execute?
Yes
No
Busy == TRUE?
Yes
No
Execute == TRUE?
Yes
No
Send read request according to Function#
Active = TRUE
Slave has answered or Timeout?
Yes
No
Active = FALSE
Error == FALSE?
Yes
No
Error = FALSE
ErrorID = 0
Update Value
Done = TRUE
Error = TRUE
Update ErrorID
Done = FALSE
End

Example



The above example requests reading of a number of binary data described by DATA_COUNT positioned in the INIT in the Modbus RTU slave of SLAVE address through the Read Input Register function. These data are forwarded to VALUE. The block ends successfully, Done output is activated.

11.8.5.2.2.5 MB_SlaveStatus

Block that allows monitoring the status of 4 slaves of the Modbus RTU network.

Ladder Representation

Block Structure

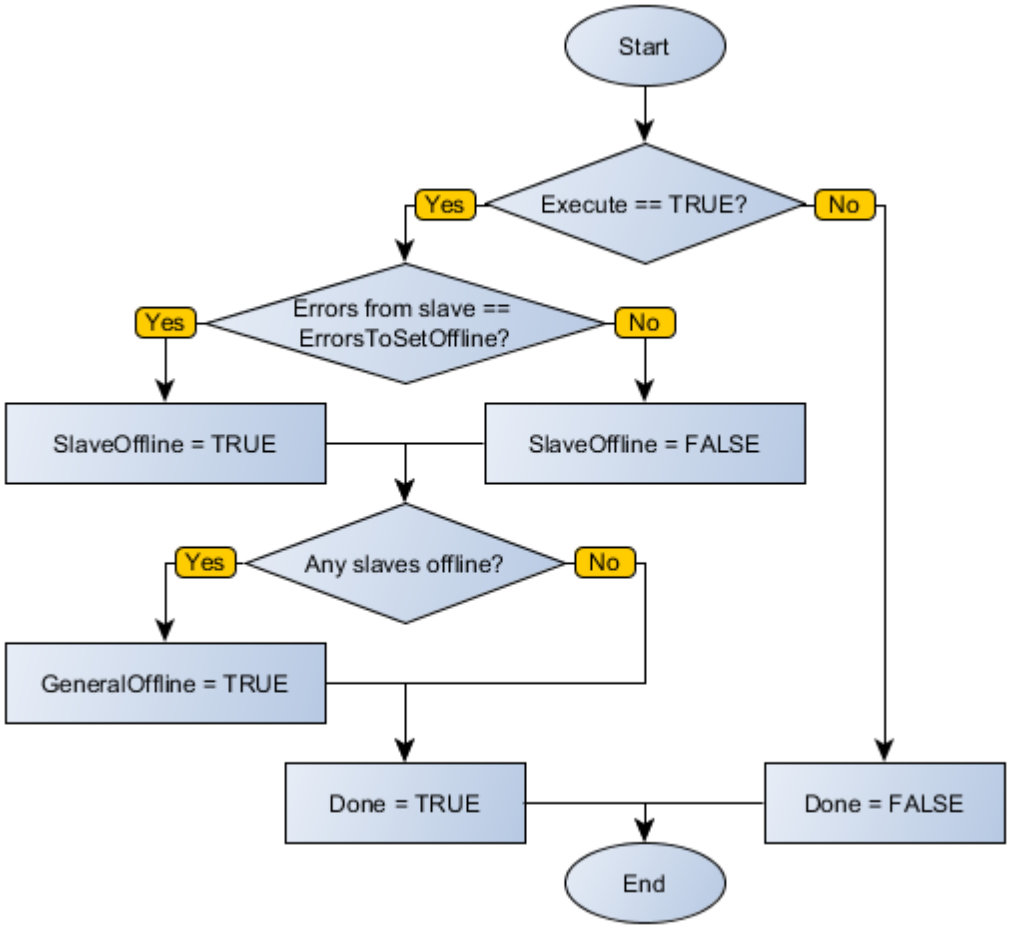
Variable Type	Name	Data Type	Description
VAR_INPUT	Execute	BOOL	Block enabling
	ErrorsToSetOffline#	BYTE	Amount of errors that the master must identify until it considers communication w ith an offline slave
	AddressSlave1#	BYTE	Slave address 1 to be monitored
	AddressSlave2#	BYTE	Slave address 2 to be monitored
	AddressSlave3#	BYTE	Slave address 3 to be monitored
	AddressSlave4#	BYTE	Slave address 4 to be monitored
VAR_OUTPUT	Done	BOOL	Output enabling
	GeneralOffline	BOOL	Flag indicating any one of the monitored communication is offline
	Slave1Offline	BOOL	Flag of offline status slave 1
	Slave2Offline	BOOL	Flag of offline status slave 2
	Slave3Offline	BOOL	Flag of offline status slave 3
	Slave4Offline	BOOL	Flag of offline status slave 4

Operation

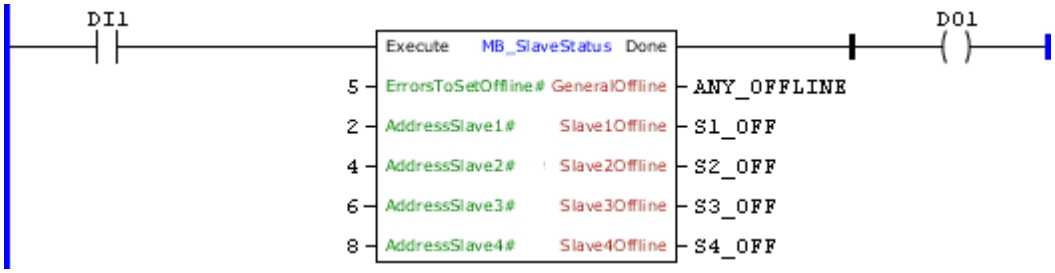
This block remains active while Execute is at TRUE level, updating its outputs according to the number of errors recorded for each slave. When Execute receives FALSE level, the inputs are ignored and the outputs are zeroed. The Done output receives TRUE level when Execute has TRUE level and block finished its execution.

The ErrorsToSetOffline # input allows registering the number of errors identified in a slave that will feature an offline communication. AddressSlave inputs allow inserting four slave addresses to be monitored. When this monitored slave reports the programmed number of errors, its corresponding SlaveOffline output is set to TRUE level. If any of SlaveOffline outputs is at TRUE level, GeneralOffline also receives TRUE level.

Block Flowchart



Example

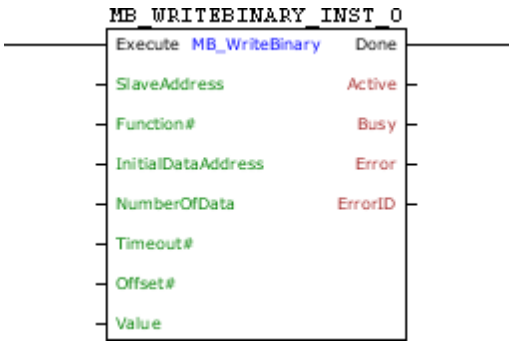


The above example checks the number of error responses sent by the slaves 2, 4, 6 and 8 of the Modbus RTU. If any of them is greater than 5, its SX_OFF status is led to TRUE level. The block ends successfully, Done output is activated.

11.8.5.2.2.6 MB_WriteBinary

Block that performs a writing of up to 128 binary data (via Write Single Coil or Write Multiple Coils) in a slave on the Modbus RTU network.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	Execute	BOOL	Block enabling
	SlaveAddress	BYTE	Slave address
	Function#	BYTE	Writing function code
	InitialDataAddress	WORD	Initial bit address where the data will be written
	NumberOfData	BYTE	Number of bits to be written (1 to 128)
	Timeout#	WORD	Maximum waiting time for the slave response [ms]
	Offset#	BOOL	Offset Indication in InitialDataAddress, i.e., need to subtract 1 from this number
	Value	BOOL	Variable that stores the data to be written
VAR_OUTPUT	Done	BOOL	Output enabling
	Active	BOOL	Awaiting response flag
	Busy	BOOL	Flag indicating the RS485 interface is busy with another request
	Error	BOOL	Error in the execution flag
	ErrorID	BYTE	Identifier of the occurred error
VAR	MB_WRITEBINARY_INST_0	MB_WRITEBINARY	Instance of access to block structure

Operation

When this block detects a leading edge on Execute, it checks whether the Modbus RTU slave in specified address in SlaveAddress is free to send data (Busy variable at FALSE level). If so, it sends the writing request of a number of bits indicated by NumberOfData in InitialDataAddress address using chosen function in Function# and sets the Active output, resetting it when receiving the response from the slave. If the slave is not free, the block waits Busy go to FALSE level to resubmit the request.

NOTE!
If Execute goes to FALSE level and Busy is still at TRUE level, the request is canceled.

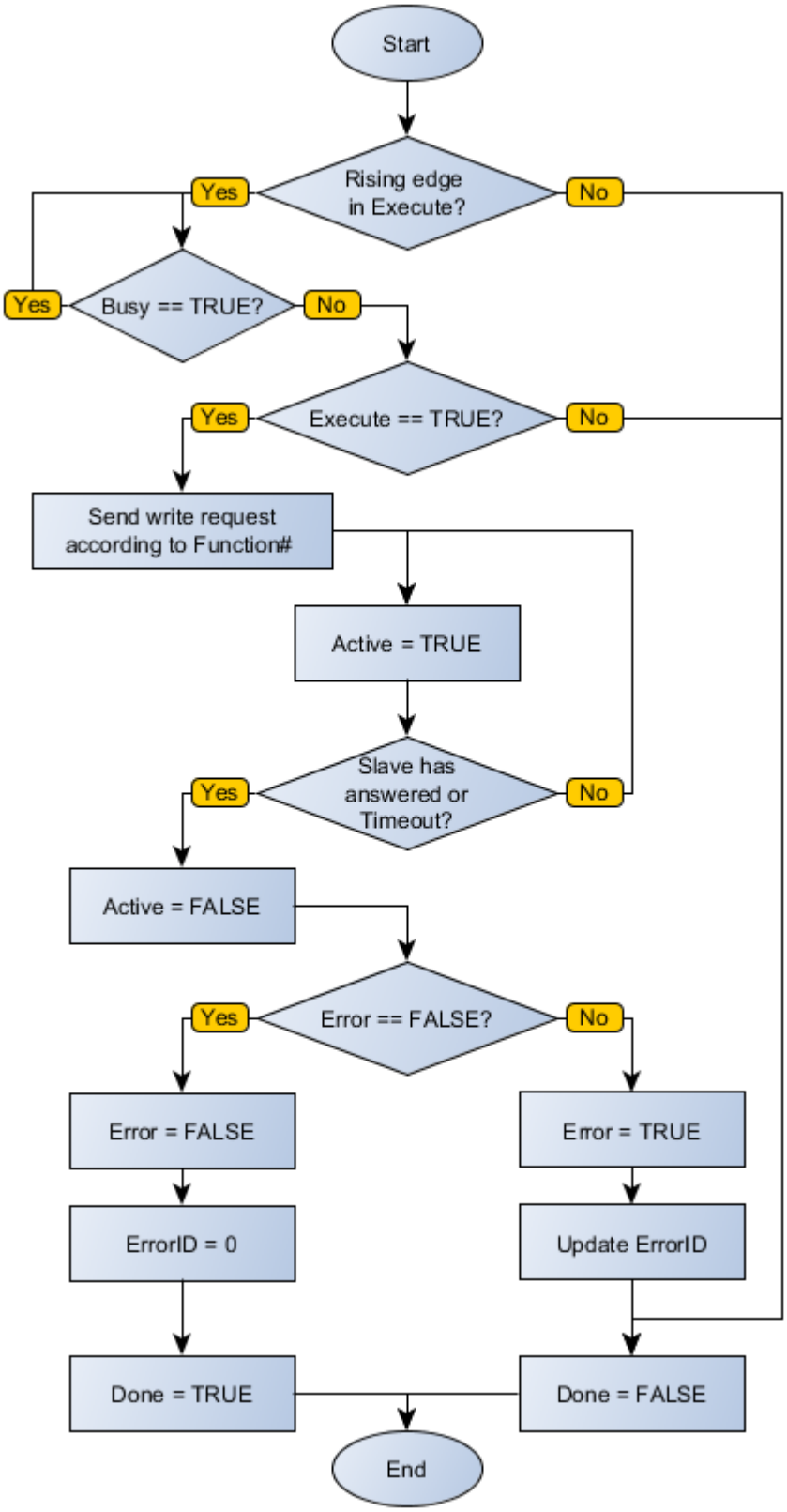
NOTE!
Value is an array of size equal to NumberOfData. It is important to check this compatibility not to generate errors in the block.

When Execute has FALSE value, Done remains FALSE. The Done output is only activated when the block finishes executing successfully, remaining at TRUE level until Execute receives FALSE.

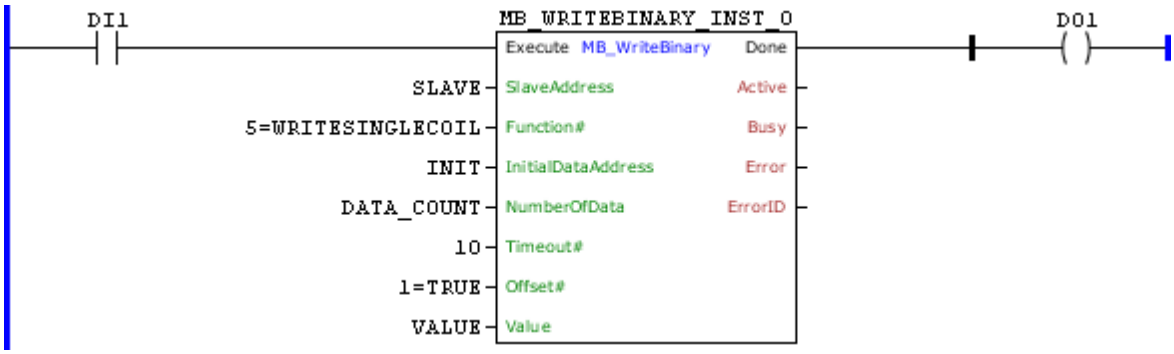
If there is any error in the execution, the Error output is enabled and ErrorID displays an error code according to the table below.

Code	Description
0	Executed successfully
1	Invalid input data
2	Master not enabled
4	Timeout in slave response
5	Slave returned error

Block Flowchart


Start
Rising edge in Execute?
Yes
No
Busy == TRUE?
Yes
No
Execute == TRUE?
Yes
No
Send write request according to Function#
Active = TRUE
Slave has answered or Timeout?
Yes
No
Active = FALSE
Error == FALSE?
Yes
No
Error = FALSE
Error = TRUE
ErrorID = 0
Update ErrorID
Done = TRUE
Done = FALSE
End

Example

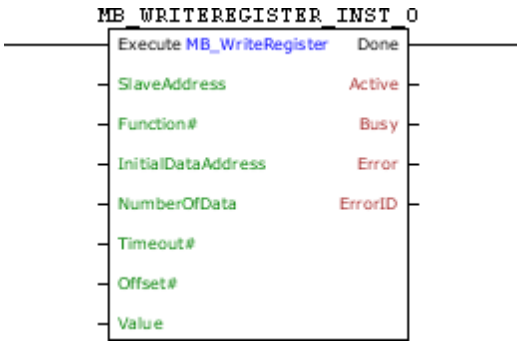


The example above requests written data contained in VALUE, with size described by DATA_COUNT, at addresses positioned from INIT on Modbus RTU slave at address SLAVE using the function Write Single Coil. The block ends successfully, Done output is activated.

11.8.5.2.2.7 MB_WriteRegister

Block that performs a reading of up to sixteen 16-bit registers (via Write Single Register or Write Multiple Registers) of a slave on the Modbus RTU network.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	Execute	BOOL	Block enabling
	SlaveAddress	BYTE	Slave address
	Function#	BYTE	Writing function code
	InitialDataAddress	WORD	Initial register address to be written
	NumberOfData	BYTE	Number of registers to be written (1 to 16)
	Timeout#	WORD	Maximum waiting time for the slave response [ms]
	Offset#	BOOL	Offset Indication in InitialDataAddress, i.e., need to subtract 1 from this number
	Value	BYTE SINT USINT WORD UINT INT DWORD UDINT DINT REAL	Variable that stores the data to be written
VAR_OUTPUT	Done	BOOL	Output enabling
	Active	BOOL	Awaiting response flag
	Busy	BOOL	Flag indicating the RS485 interface is busy with another request
	Error	BOOL	Error in the execution flag
	ErrorID	BYTE	Identifier of the occurred error
VAR	MB_WRITEREGISTER_INST_0	MB_WRITEREGISTER	Instance of access to block structure

Operation

When this block detects a leading edge on Execute, it checks whether the Modbus RTU slave in specified address in SlaveAddress is free to send data (Busy variable at FALSE level). If so, it sends the writing request of Value values in a number of registers indicated by NumberOfData in InitialDataAddress address using chosen function in Function# and sets the Active output, resetting it when receiving the response from the slave. If the slave is not free, the block waits Busy go to FALSE level to resubmit the request.

NOTE!
If Execute goes to FALSE level and Busy is still at TRUE level, the request is canceled.

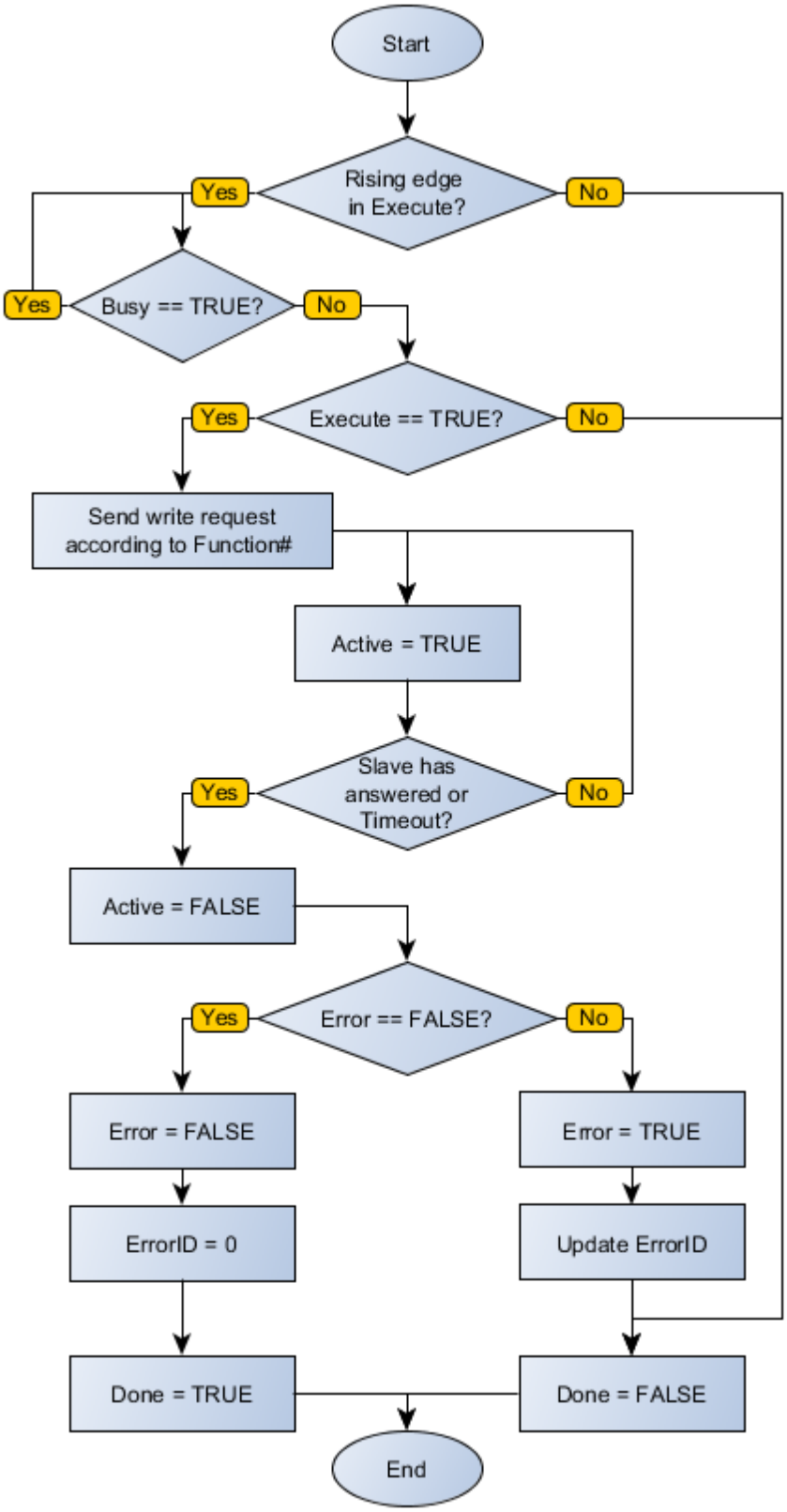
NOTE!
Value is an array of number of bits NumberOfData multiplied by 16. That is, if NumberOfData is 16, Value can be an array of 32 BYTE positions, 16 WORD positions or 8 DWORD positions. It is important to check this compatibility not to generate errors in the block.

When Execute has FALSE value, Done remains FALSE. The Done output is only activated when the block finishes executing successfully, remaining at TRUE level until Execute receives FALSE.

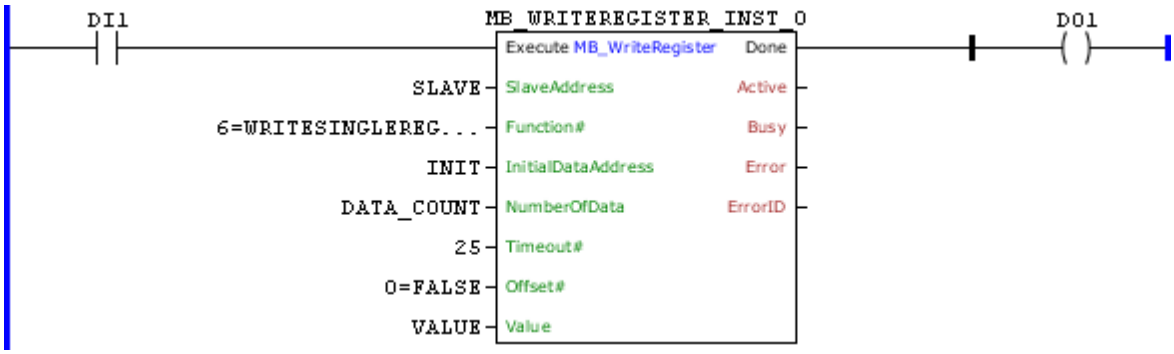
If there is any error in the execution, the Error output is enabled and ErrorID displays an error code according to the table below.

Code	Description
0	Executed successfully
1	Invalid input data
2	Master not enabled
4	Timeout in slave response
5	Slave returned error

Block Flowchart


Start
Rising edge in Execute?
Yes
No
Busy == TRUE?
Yes
No
Execute == TRUE?
Yes
No
Send write request according to Function#
Active = TRUE
Slave has answered or Timeout?
Yes
No
Active = FALSE
Error == FALSE?
Yes
No
Error = FALSE
Error = TRUE
ErrorID = 0
Update ErrorID
Done = TRUE
Done = FALSE
End

Example



The example above requests written data contained in VALUE, with size described by DATA_COUNT, at addresses positioned from INIT on Modbus RTU slave at address SLAVE using the function Write Single Register. The block ends successfully, Done output is activated.

11.8.5.2.3 Modbus TCP

11.8.5.2.3.1 Modbus TCP Overview

Operation in Modbus TCP Network – Client Mode

Besides the operation as server, the programmable controller PLC300 also allows the operation as client of the Modbus TCP network. For this operation, it is necessary to observe the following points:

- The sending and receiving of telegrams via Ethernet interface using the Modbus TCP protocol is programmed by using blocks in Ladder programming language. It is necessary to know the available blocks and the Ladder programming software in order to program the network client.
- The following functions are available to send requests by the Modbus TCP client:
  - Function 01: Read Coils
  - Function 02: Read Discrete Inputs
  - Function 03: Read Holding Registers
  - Function 04: Read Input Registers
  - Function 05: Write Single Coil
  - Function 06: Write Single Register
  - Function 15: Write Multiple Coil
  - Function 16: Write Multiple Registers

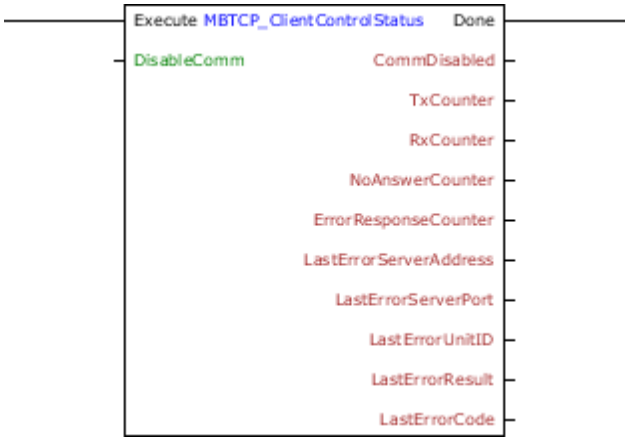
Blocks for Programming of the Client

In order to control and monitor Modbus TCP communication using the programmable controller PLC300, the following blocks were developed, which must be used during Ladder programming.

11.8.5.2.3.2 MBTCP_ClientControlStatus

Block that allows monitoring various statuses of the Modbus TCP network client.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	Execute	BOOL	Block enabling
	DisableComm	BOOL	Disables Modbus TCP communication
VAR_OUTPUT	Done	BOOL	Output enabling
	CommDisabled	BOOL	Disabled communication flag
	TxCounter	WORD UINT	Counter of requests sent
	RxCounter	WORD UINT	Counter of telegrams received
	NoAnswerCounter	WORD UINT	Counter of requests not answered
	ErrorResponseCounter	WORD UINT	Counter of responses received with error information
	LastErrorServerAddress	DWORD	Server address in which the last communication error was detected
	LastErrorServerPort	WORD UINT	Server port in which the last communication error was detected
	LastErrorUnitID	BYTE USINT	Server UnitID in which the last communication error was detected
	LastErrorResult	BYTE USINT	Operation result of the last communication error received (0 - No error) (4 - Server response timeout) (5 - Server returned error) (6 - Connection to server has failed) (7 - TCP/IP Connection ended prematurely)
	LastErrorCode	BYTE USINT	Code of the last communication error received

Operation

This block remains active while Execute is at TRUE level, updating its outputs according to the monitoring of the master and input requests. When Execute receives FALSE level, the inputs are ignored and the outputs are zeroed. The Done output receives TRUE level when Execute has TRUE level and block finished its execution.

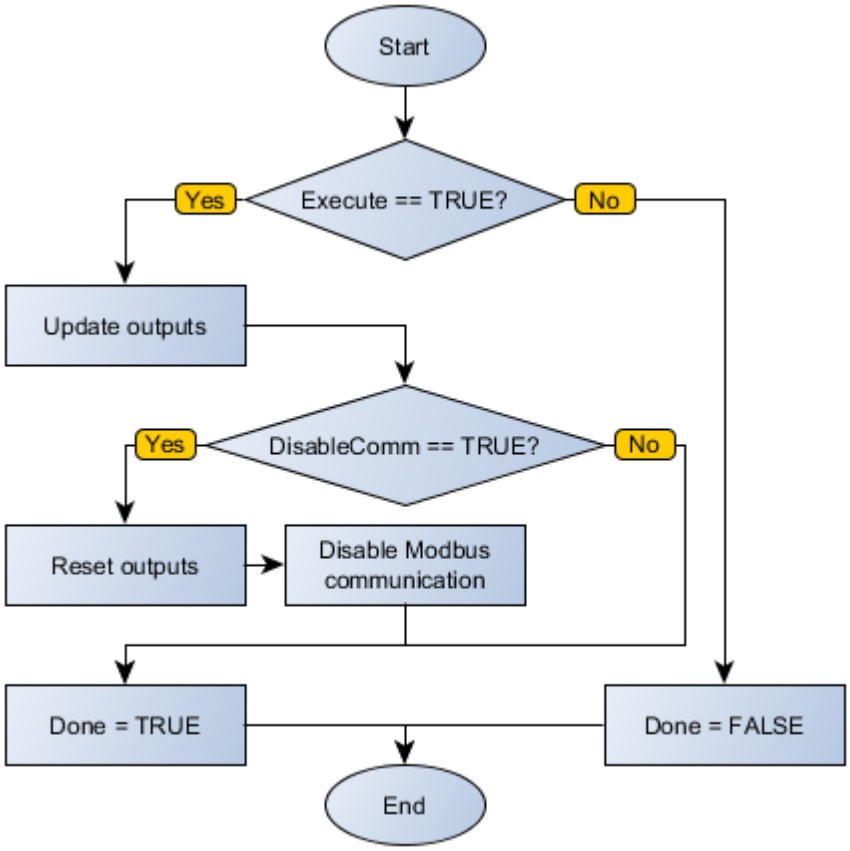
A TRUE level DisableComm disables the Modbus TCP communication and resets the status counters

and markers of the client. These markers and counters are displayed in the output block each having some data corresponding to its description. Their values are also cleared at shutdown of the client.

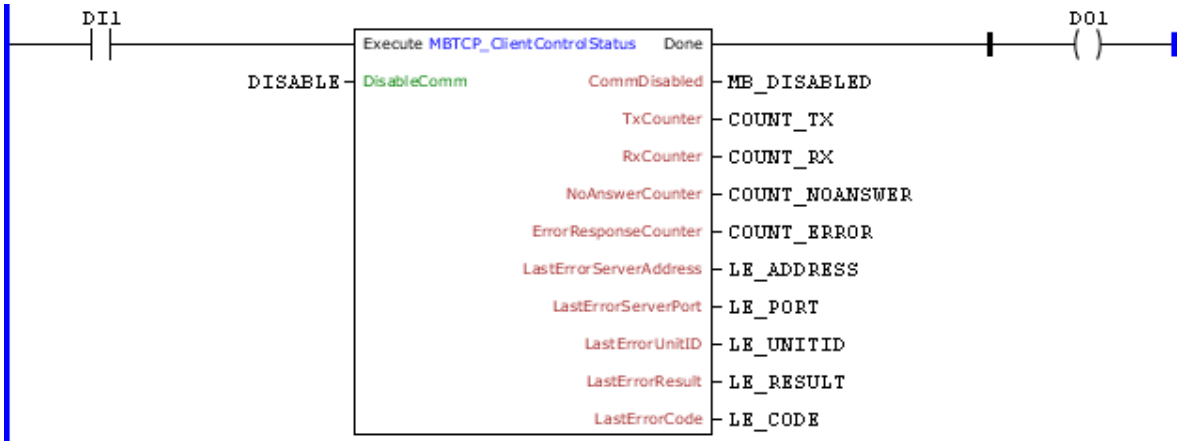
Compatibility

Device	Version
PLC300	1.30 or higher

Block Flowchart



Example

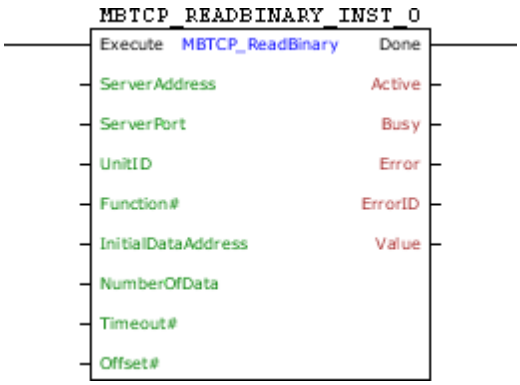


The example above requests status data of the Modbus RTU network client, and allows disabling communication through DISABLE. The block ends successfully, Done output is activated.

11.8.5.2.3.3 MBTCP_ReadBinary

Block that performs a reading of up to 128 binary data (via Read Coils or Read Discrete Inputs) of a server on the Modbus TCP network.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	Execute	BOOL	Block enabling
	ServerAddress	DWORD	Server IP address (Ex: 192.168.0.1)
	ServerPort	WORD	Modbus TCP Port of the server (Standard: 502)
	UnitID	BYTE	UnitID do servidor (Standard: 255)
	Function#	BYTE	Reading function code
	InitialDataAddress	WORD	Initial bit address of the data to be read
	NumberOfData	BYTE	Number of bits to be read (1 to 128)
	Timeout#	WORD	Maximum waiting time for the server response [ms]
	Offset#	BOOL	Offset Indication in InitialDataAddress, i.e., need to subtract 1 from this number
VAR_OUTPUT	Done	BOOL	Output enabling
	Active	BOOL	Awaiting response flag
	Busy	BOOL	Flag indicating the connection is busy with another request
	Error	BOOL	Error in the execution flag
	ErrorID	BYTE	Identifier of the occurred error
	Value	BOOL	Variable that stores the received data
VAR	MBTCP_READBINARY_INST_0	MBTCP_READBINARY	Instance of access to block structure

Operation

When this block detects a leading edge on Execute, it checks whether the Modbus TCP server in specified address in ServerAddress is free to send data (Busy variable at FALSE level). If so, it sends the reading request of a number of bits indicated by NumberOfData in InitialDataAddress address using chosen function in Function# and sets the Active output, resetting it when receiving the response from the server. The received data is stored in the Value variable. If the server is not free, the block waits Busy go to FALSE level to resubmit the request.

NOTE!
If Execute goes to FALSE level and Busy is still at TRUE level, the request is canceled.

NOTE!
Value is an array of size equal to NumberOfData. It is important to check this compatibility not to generate errors in the block.

When Execute has FALSE value, Done remains FALSE. The Done output is only activated when the block finishes executing successfully, remaining at TRUE level until Execute receives FALSE.

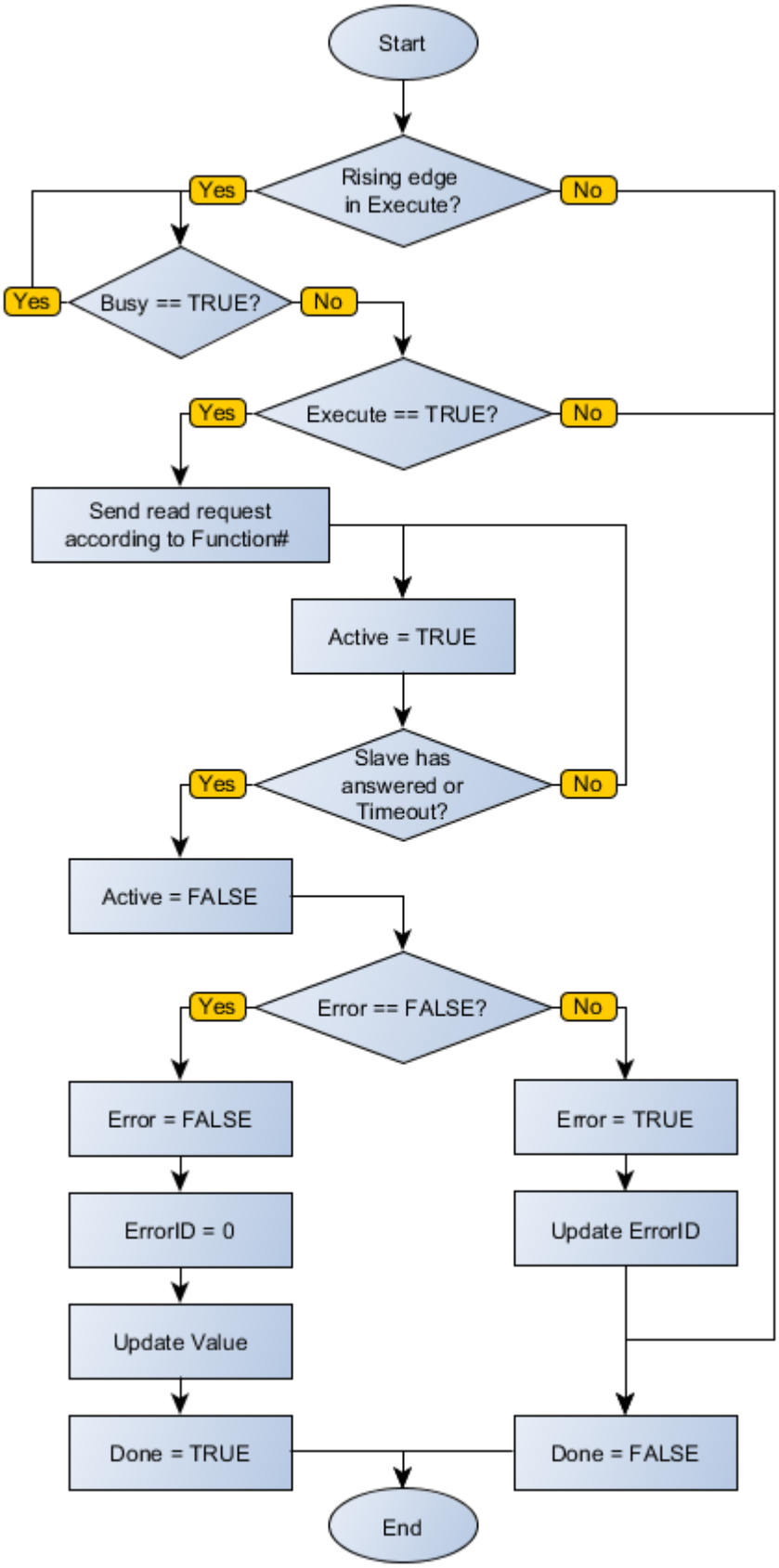
If there is any error in the execution, the Error output is enabled and ErrorID displays an error code according to the table below.

Code	Description
0	Executed successfully
1	Invalid input data
2	Client not enabled
4	Timeout in server response
5	Server returned error
6	Failed to connect to server
7	TCP/IP connection terminated prematurely

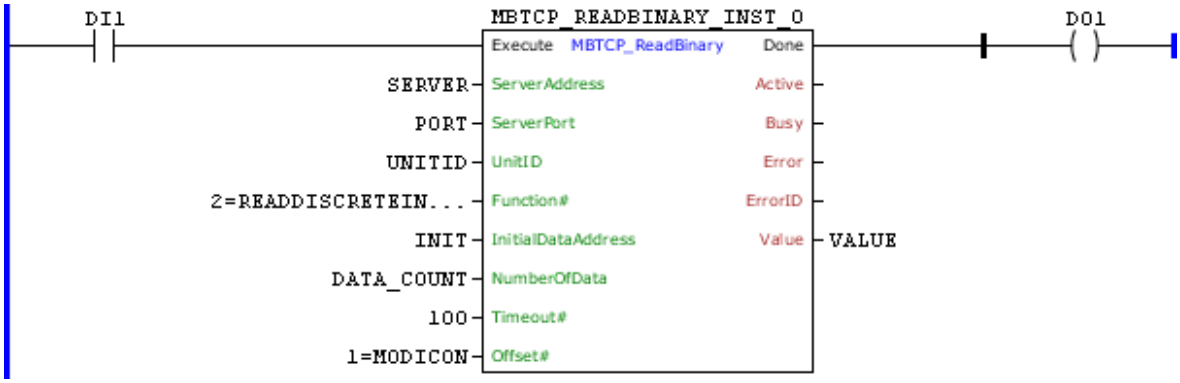
Compatibility

Device	Version
PLC300	1.30 or higher

Block Flowchart


Start
Rising edge in Execute?
Yes
No
Busy == TRUE?
Yes
No
Execute == TRUE?
Yes
No
Send read request according to Function#
Active = TRUE
Slave has answered or Timeout?
Yes
No
Active = FALSE
Error == FALSE?
Yes
No
Error = FALSE
ErrorID = 0
Update Value
Done = TRUE
Error = TRUE
Update ErrorID
Done = FALSE
End

Example

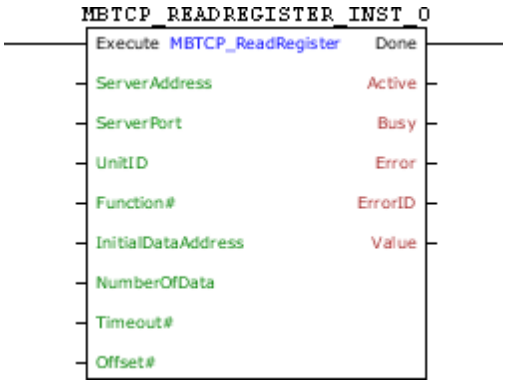


The above example requests reading of a number of binary data described by DATA_COUNT positioned in the INIT – 1 in Modbus TCP server of SERVER:PORT address through the Read Discrete Input function. These data are forwarded to VALUE. The block ends successfully, Done output is activated.

11.8.5.2.3.4 MBTCP_ReadRegister

Block that performs a reading of up to 64 16-bit registers (via Read Holding Registers or Read Input Registers) of a server on the Modbus TCP network.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	Execute	BOOL	Block enabling
	ServerAddress	DWORD	Server IP address (Ex: 192.168.0.1)
	ServerPort	WORD	Modbus TCP Port of the server (Standard: 502)
	UnitID	BYTE	UnitID do servidor (Standard: 255)
	Function#	BYTE	Reading function code
	InitialDataAddress	WORD	Initial register address to be read
	NumberOfData	BYTE	Number of registers to be read (1 to 64)
	Timeout#	WORD	Maximum waiting time for the server response [ms]
	Offset#	BOOL	Offset Indication in InitialDataAddress, i.e., need to subtract 1 from this number
VAR_OUTPUT	Done	BOOL	Output enabling
	Active	BOOL	Awaiting response flag
	Busy	BOOL	Flag indicating the connection is busy with another request
	Error	BOOL	Error in the execution flag
	ErrorID	BYTE	Identifier of the occurred error
	Value	BYTE SINT USINT WORD UINT INT DWORD UDINT DINT REAL	Variable that stores the received data
VAR	MBTCP_READREGISTER_INST_0	MBTCP_READREGISTER	Instance of access to block structure

Operation

When this block detects a leading edge on Execute, it checks whether the Modbus TCP server in specified address in ServerAddress is free to send data (Busy variable at FALSE level). If so, it sends the reading request of a number of registers indicated by NumberOfData in InitialDataAddress address using chosen function in Function# and sets the Active output, resetting it when receiving the response from the server. The received data is stored in the Value variable. If the server is not free, the block waits Busy go to FALSE level to resubmit the request.

NOTE!
If Execute goes to FALSE level and Busy is still at TRUE level, the request is canceled.

NOTE!
Value is an array of number of bits NumberOfData multiplied by 16. That is, if NumberOfData is 16, Value can be an array of 32 BYTE positions, 16 WORD positions or 8 DWORD positions. It is important to check this compatibility not to generate errors in the block.

When Execute has FALSE value, Done remains FALSE. The Done output is only activated when the block finishes executing successfully, remaining at TRUE level until Execute receives FALSE.

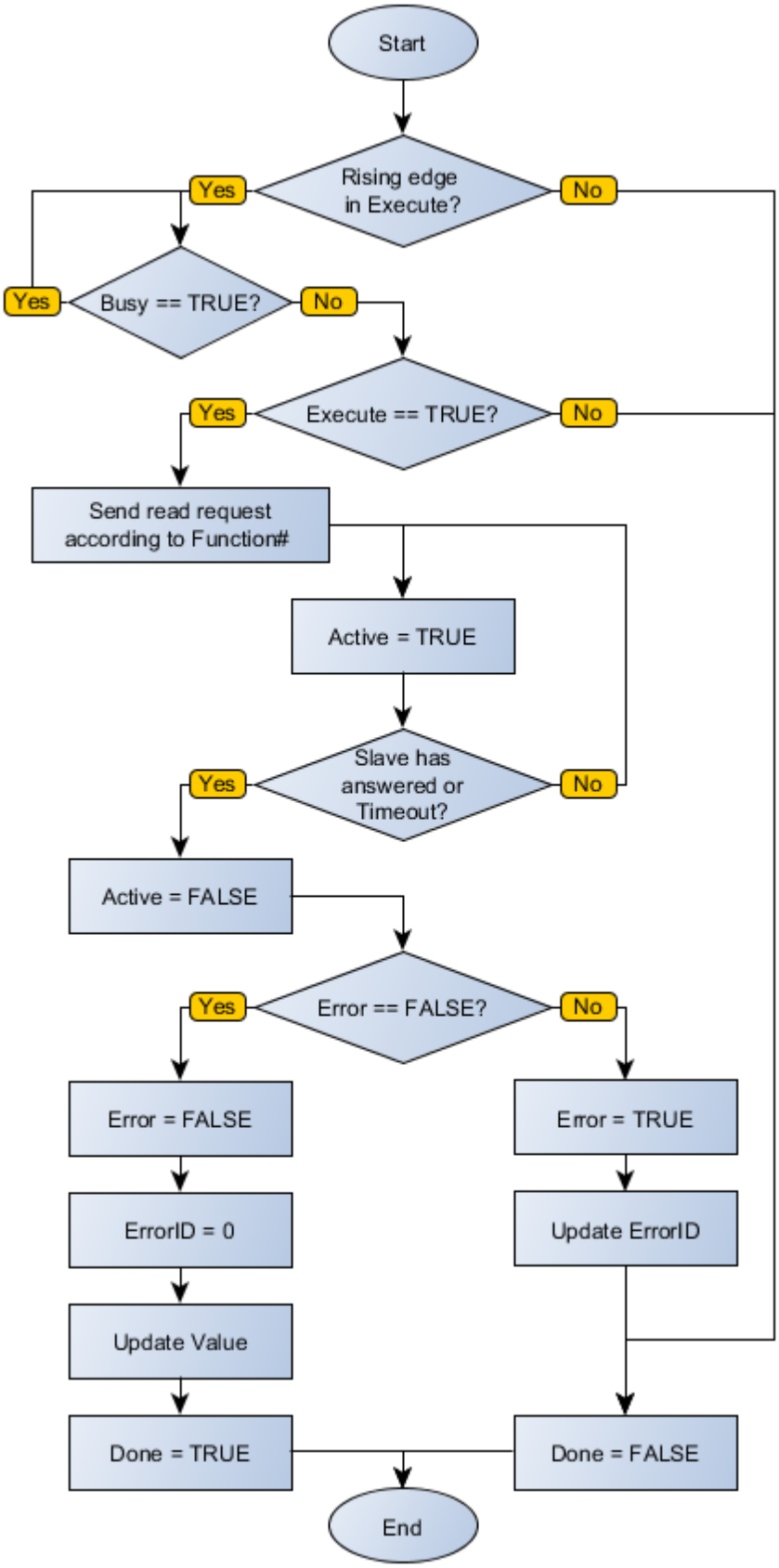
If there is any error in the execution, the Error output is enabled and ErrorID displays an error code according to the table below.

Code	Description
0	Executed successfully
1	Invalid input data
2	Client not enabled
4	Timeout in server response
5	Server returned error
6	Failed to connect to server
7	TCP/IP connection terminated prematurely

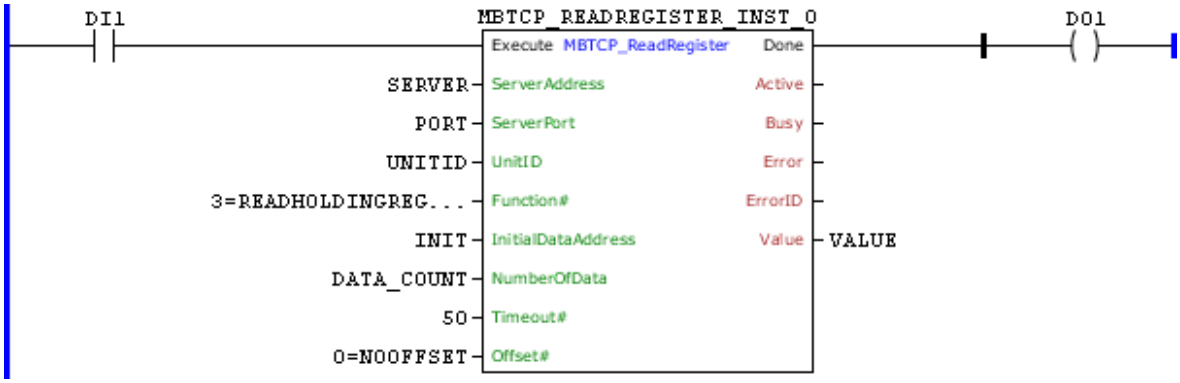
Compatibility

Device	Version
PLC300	1.30 or higher

Block Flowchart


Start
Rising edge in Execute?
Yes
No
Busy == TRUE?
Yes
No
Execute == TRUE?
Yes
No
Send read request according to Function#
Active = TRUE
Slave has answered or Timeout?
Yes
No
Active = FALSE
Error == FALSE?
Yes
No
Error = FALSE
ErrorID = 0
Update Value
Done = TRUE
Error = TRUE
Update ErrorID
Done = FALSE
End

Example

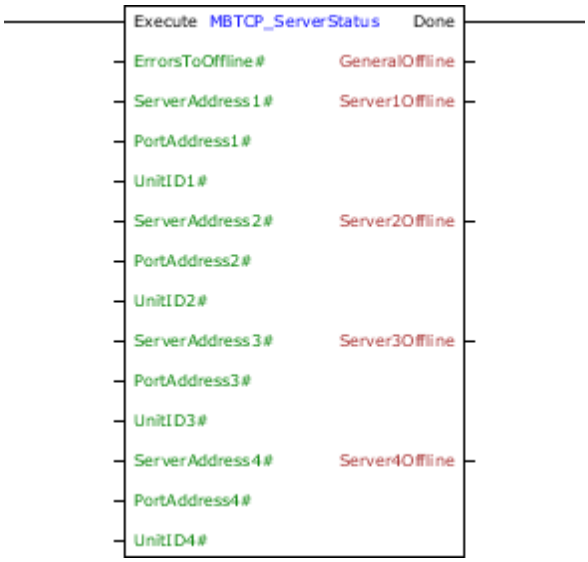


The above example requests reading of a number of register data described by DATA_COUNT positioned in the INIT1 in Modbus TCP server of SERVER:PORT address through the Read Holding Register function. These data are forwarded to VALUE. The block ends successfully, Done output is activated.

11.8.5.2.3.5 MBTCP_ServerStatus

Block that allows monitoring the status of 4 servers of the Modbus TCP network.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	Execute	BOOL	Block enabling
	ErrorsToSetOffline#	BYTE	Amount of errors that the must be identified until it considers communication w ith an offline server
	ServerAddress1#	DWORD	Server 1 address to be monitored (Ex: 192.168.0.1)
	PortAddress1#	WORD	Server 1 port to be monitored (Standard: 502)
	UnitID1#	BYTE	Server 1 UnitID to be monitored (Standard: 255)
	ServerAddress2#	DWORD	Server 2 address to be monitored (Ex: 192.168.0.1)
	PortAddress2#	WORD	Server 2 port to be monitored (Standard: 502)
	UnitID2#	BYTE	Server 2 UnitID to be monitored (Standard: 255)
	ServerAddress3#	DWORD	Server 3 address to be monitored (Ex: 192.168.0.1)
	PortAddress3#	WORD	Server 3 port to be monitored (Standard: 502)
	UnitID3#	BYTE	Server 3 UnitID to be monitored (Standard: 255)
	ServerAddress4#	DWORD	Server 4 address to be monitored (Ex: 192.168.0.1)
	PortAddress4#	WORD	Server 4 port to be monitored (Standard: 502)
	UnitID4#	BYTE	Server 4 UnitID to be monitored (Standard: 255)
VAR_OUTPUT	Done	BOOL	Output enabling
	GeneralOffline	BOOL	Flag indicating any one of the monitored communication is offline
	Server1Offline	BOOL	Flag of offline status for server 1
	Server2Offline	BOOL	Flag of offline status for server 2
	Server3Offline	BOOL	Flag of offline status for server 3
	Server4Offline	BOOL	Flag of offline status for server 4

Operation

This block remains active while Execute is at TRUE level, updating its outputs according to the monitoring of the number of errors recorded for each server. When Execute receives FALSE level, the inputs are ignored and the outputs are zeroed. The Done output receives TRUE level when Execute has TRUE level and block finished its execution.

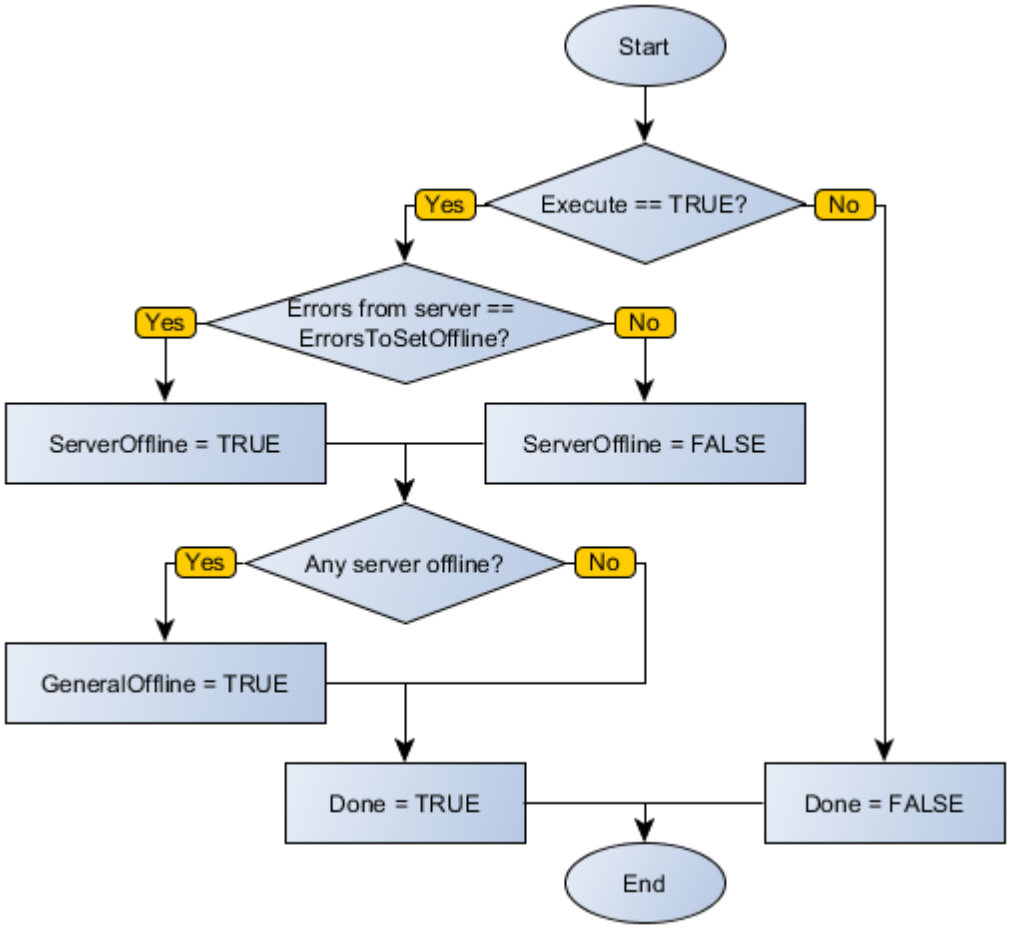
The ErrorsToSetOffline # input allows registering the number of errors identified in a server that will feature an offline communication. AddressServer inputs allow inserting four server addresses to be monitored. If you want to leave a channel ignored, enter the value 0 in the server address. When this monitored server reports the programmed number of errors, its corresponding SlaveOffline output is set to TRUE level. If any of SlaveOffline outputs is at TRUE level, GeneralOffline also receives TRUE

level.

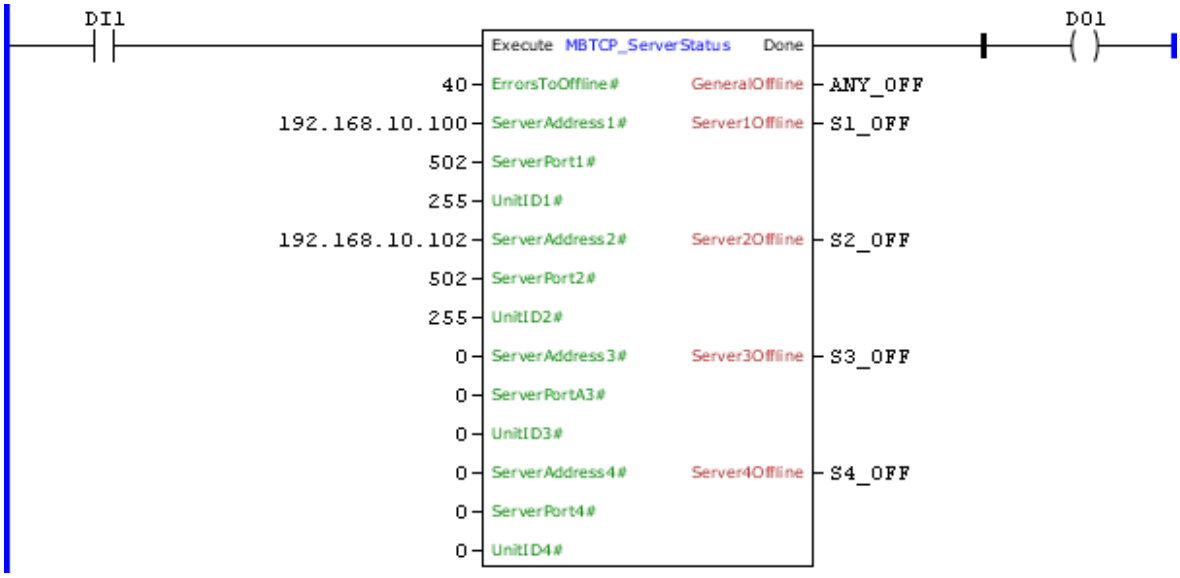
Compatibility

Device	Version
PLC300	1.30 or higher

Block Flowchart



Example

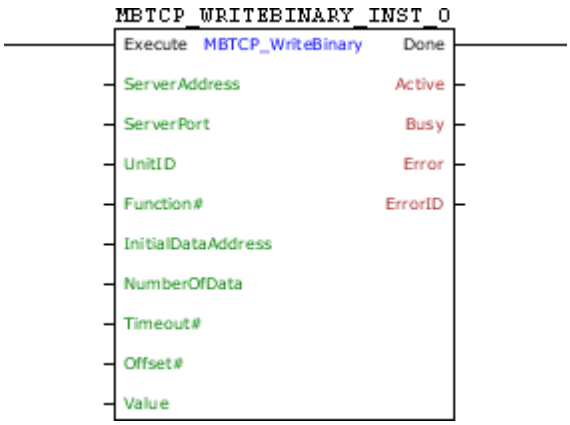


The above example checks the number of error responses sent by the servers 192.168.10.100:502 and 192.168.10.102:502 of the Modbus TCP network. If any of them is greater than 40, its SX_OFF status is led to TRUE level. The block ends successfully, Done output is activated.

11.8.5.2.3.6 MBTCP_WriteBinary

Block that performs a writing of up to 128 binary data (via Write Single Coil or Write Multiple Coils) in a server on the Modbus TCP network.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	Execute	BOOL	Block enabling
	ServerAddress	DWORD	Server IP address (Ex: 192.168.0.1)
	ServerPort	WORD	Modbus TCP Port of the server (Standard: 502)
	UnitID	BYTE	UnitID do servidor (Standard: 255)
	Function#	BYTE	Writing function code
	InitialDataAddress	WORD	Initial bit address where the data will be written
	NumberOfData	BYTE	Number of bits to be written (1 to 128)
	Timeout#	WORD	Maximum waiting time for the server response [ms]
	Offset#	BOOL	Offset Indication in InitialDataAddress, i.e., need to subtract 1 from this number
	Value	BOOL	Variable that stores the data to be written
VAR_OUTPUT	Done	BOOL	Output enabling
	Active	BOOL	Awaiting response flag
	Busy	BOOL	Flag indicating the connection is busy with another request
	Error	BOOL	Error in the execution flag
	ErrorID	BYTE	Identifier of the occurred error
VAR	MBTCP_WRITEBINARY_INST_0	MBTCP_WRITEBINARY	Instance of access to block structure

Operation

When this block detects a leading edge on Execute, it checks whether the Modbus TCP server in specified address in ServerAddress is free to send data (Busy variable at FALSE level). If so, it sends the writing request of the Value values in a number of bits indicated by NumberOfData in InitialDataAddress address using chosen function in Function# and sets the Active output, resetting it when receiving the response from the server. If the server is not free, the block waits Busy go to FALSE level to resubmit the request.

NOTE!
If Execute goes to FALSE level and Busy is still at TRUE level, the request is canceled.

NOTE!
Value is an array of size equal to NumberOfData. It is important to check this compatibility not to generate errors in the block.

When Execute has FALSE value, Done remains FALSE. The Done output is only activated when the block finishes executing successfully, remaining at TRUE level until Execute receives FALSE.

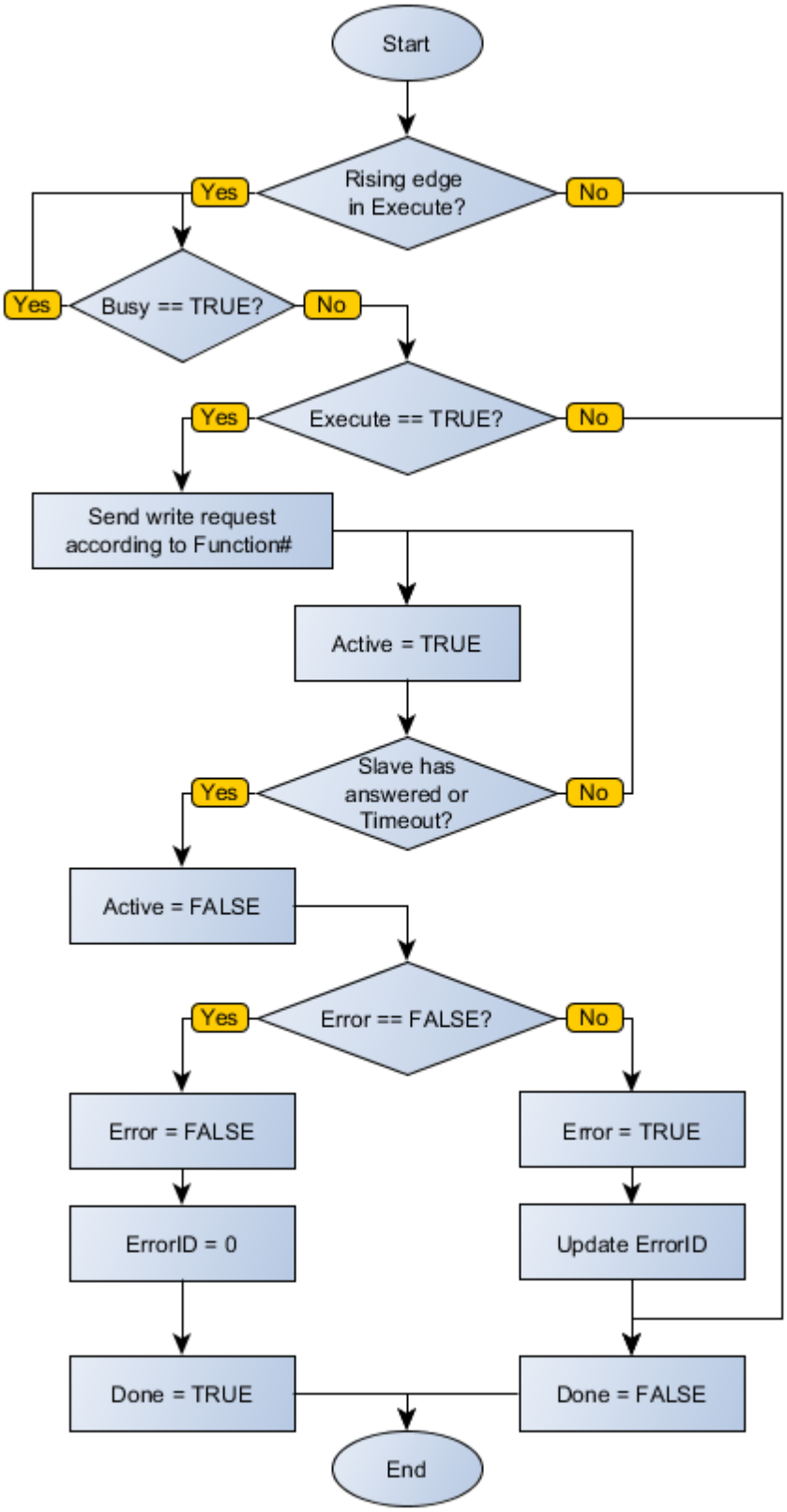
If there is any error in the execution, the Error output is enabled and ErrorID displays an error code according to the table below.

Code	Description
0	Executed successfully
1	Invalid input data
2	Client not enabled
4	Timeout in server response
5	Server returned error
6	Failed to connect to server
7	TCP/IP connection terminated prematurely

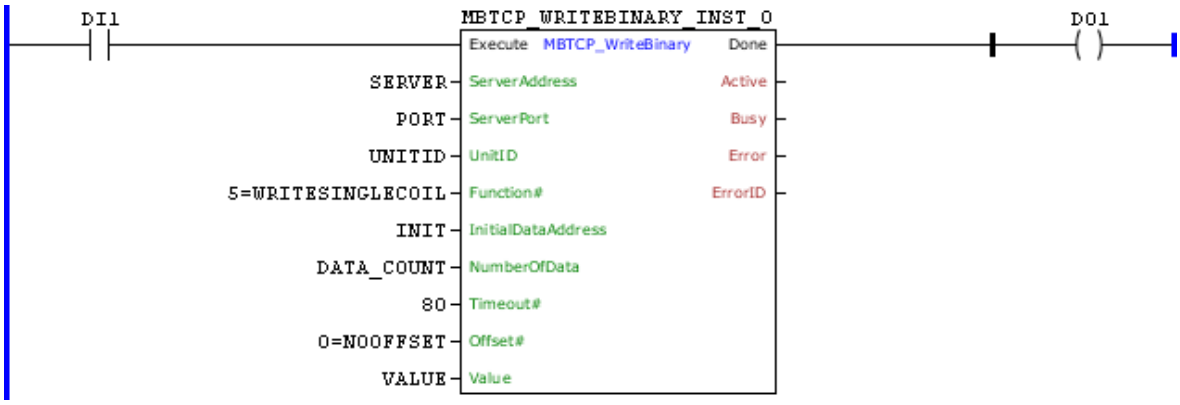
Compatibility

Device	Version
PLC300	1.30 or higher

Block Flowchart


Start
Rising edge in Execute?
Yes
No
Busy == TRUE?
Yes
No
Execute == TRUE?
Yes
No
Send write request according to Function#
Active = TRUE
Slave has answered or Timeout?
Yes
No
Active = FALSE
Error == FALSE?
Yes
No
Error = FALSE
Error = TRUE
ErrorID = 0
Update ErrorID
Done = TRUE
Done = FALSE
End

Example

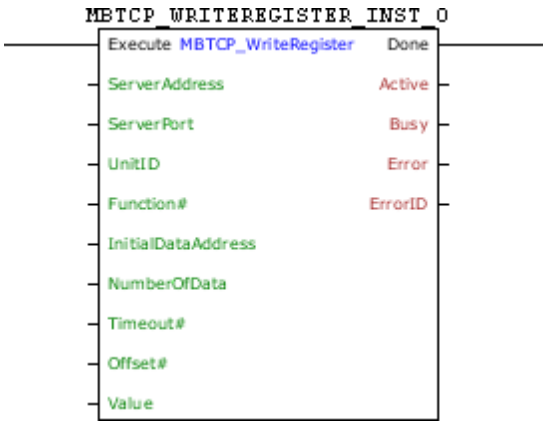


The example above requests written data contained in VALUE, with size described by DATA_COUNT, at addresses positioned from INIT on Modbus TCP server at SERVER:PORT address using the function Write Single Coil. The block ends successfully, Done output is activated.

11.8.5.2.3.7 MBTCP_WriteRegister

Block that performs a writing of up to sixteen 16-bit registers (via Write Single Register or Write Multiple Registers) of a server on the Modbus TCP network.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	Execute	BOOL	Block enabling
	ServerAddress	DWORD	Server IP address (Ex: 192.168.0.1)
	ServerPort	WORD	Modbus TCP Port of the server (Standard: 502)
	UnitID	BYTE	UnitID do servidor (Standard: 255)
	Function#	BYTE	Writing function code
	InitialDataAddress	WORD	Initial register address to be written
	NumberOfData	BYTE	Number of registers to be written (1 to 16)
	Timeout#	WORD	Maximum waiting time for the server response [ms]
	Offset#	BOOL	Offset Indication in InitialDataAddress, i.e., need to subtract 1 from this number
	Value	BYTE SINT USINT WORD UINT INT DWORD UDINT DINT REAL	Variable that stores the data to be written
VAR_OUTPUT	Done	BOOL	Output enabling
	Active	BOOL	Awaiting response flag
	Busy	BOOL	Flag indicating the connection is busy with another request
	Error	BOOL	Error in the execution flag
	ErrorID	BYTE	Identifier of the occurred error
VAR	MB_TCPWRITEREGISTER_INST_0	MB_TCPWRITEREGISTER	Instance of access to block structure

Operation

When this block detects a leading edge on Execute, it checks whether the Modbus TCP server in specified address in ServerAddress is free to send data (Busy variable at FALSE level). If so, it sends the writing request of Value values in a number of registers indicated by NumberOfData in InitialDataAddress address using chosen function in Function# and sets the Active output, resetting it when receiving the response from the server. If the server is not free, the block waits Busy go to FALSE level to resubmit the request.

NOTE!
If Execute goes to FALSE level and Busy is still at TRUE level, the request is canceled.

NOTE!
Value is an array of number of bits NumberOfData multiplied by 16. That is, if NumberOfData is 16, Value can be an array of 32 BYTE positions, 16 WORD positions or 8 DWORD positions. It is important to check this compatibility not to generate errors in the block.

When Execute has FALSE value, Done remains FALSE. The Done output is only activated when the block finishes executing successfully, remaining at TRUE level until Execute receives FALSE.

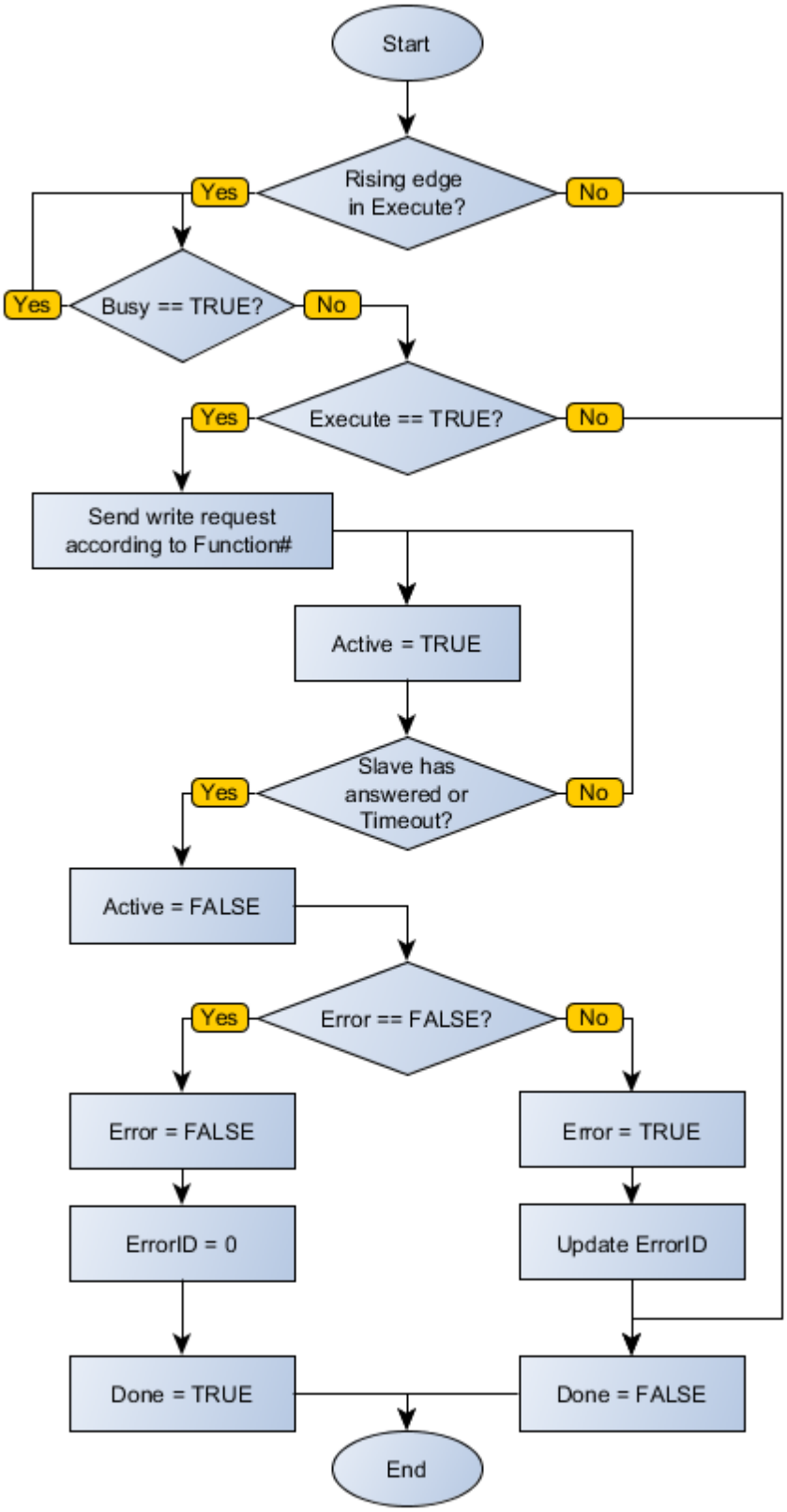
If there is any error in the execution, the Error output is enabled and ErrorID displays an error code according to the table below.

Code	Description
0	Executed successfully
1	Invalid input data
2	Client not enabled
4	Timeout in server response
5	Server returned error
6	Failed to connect to server
7	TCP/IP connection terminated prematurely

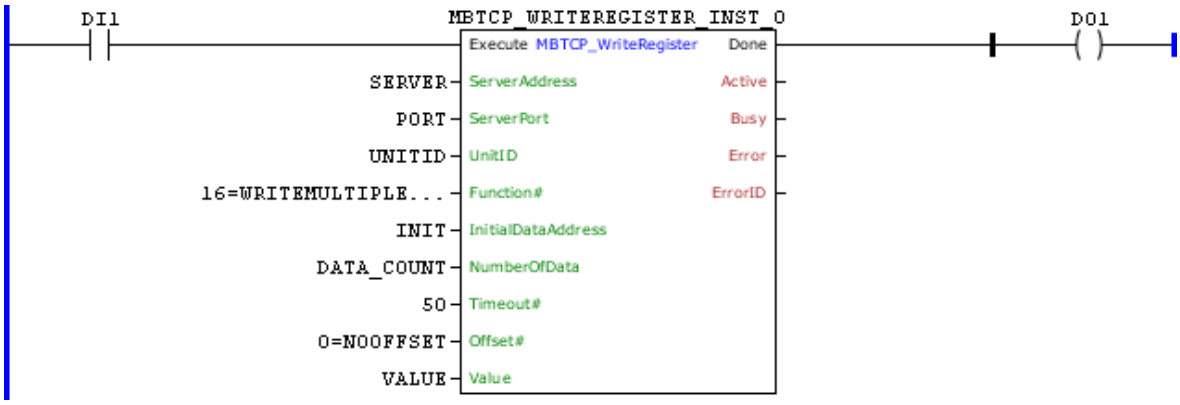
Compatibility

Device	Version
PLC300	1.30 or higher

Block Flowchart


Start
Rising edge in Execute?
Yes
No
Busy == TRUE?
Yes
No
Execute == TRUE?
Yes
No
Send write request according to Function#
Active = TRUE
Slave has answered or Timeout?
Yes
No
Active = FALSE
Error == FALSE?
Yes
No
Error = FALSE
Error = TRUE
ErrorID = 0
Update ErrorID
Done = TRUE
Done = FALSE
End

Example



The example above requests written data contained in VALUE, with size described by DATA_COUNT, at addresses positioned from INIT on Modbus TCP server at SERVER:PORT address using the function Write Multiple Registers. The block ends successfully, Done output is activated.

11.8.5.3 Compare

11.8.5.3.1 COMP_EQ

Block that compares the values of Value1 and Value2, enabling the output Q if both are equal.

Ladder Representation

Block Structure

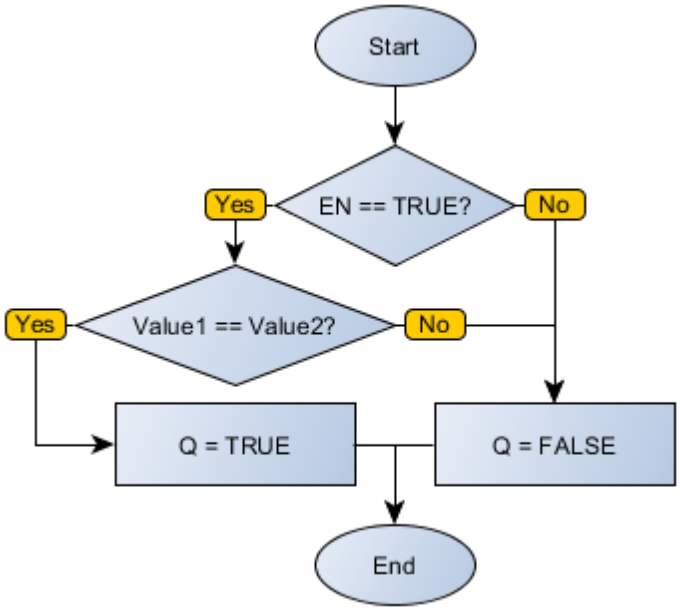
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	First value of comparison
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Second value of comparison
VAR_OUTPUT	Q	BOOL	Indicator of equality

Operation

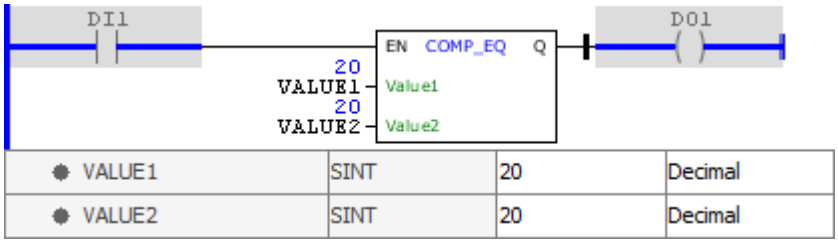
When this block has a TRUE value in EN, it sends to the output Q the TRUE value if Value1 and Value2 are the same. Otherwise, Q receives FALSE.

When EN has FALSE value, Q remains in FALSE.

Block Flowchart

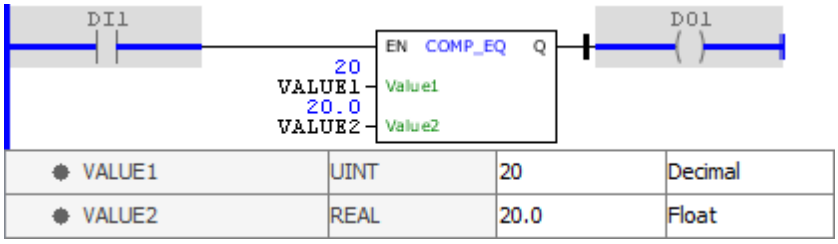


Example



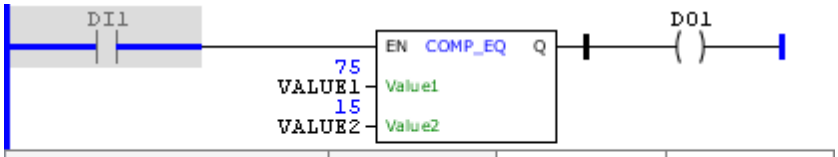
VALUE1	SINT	20	Decimal
VALUE2	SINT	20	Decimal

The example above checks equality between VALUE1 and VALUE2. Since both variables have the same value, the Q output is activated.



VALUE1	UINT	20	Decimal
VALUE2	REAL	20.0	Float

The example above checks equality between VALUE1 and VALUE2. Since both variables have the same value, the Q output is activated. Notice that the types of the input variables can be different without causing execution problems.



VALUE1	UINT	75	Decimal
VALUE2	UINT	15	Decimal

The example above checks equality between VALUE1 and VALUE2. Since both variables have different values, the Q output is disabled.

11.8.5.3.2 COMP_GE

Block that compares the values of Value1 and Value2, enabling the output Q if Value1 is higher than or equal to Value2.

Ladder Representation

Block Structure

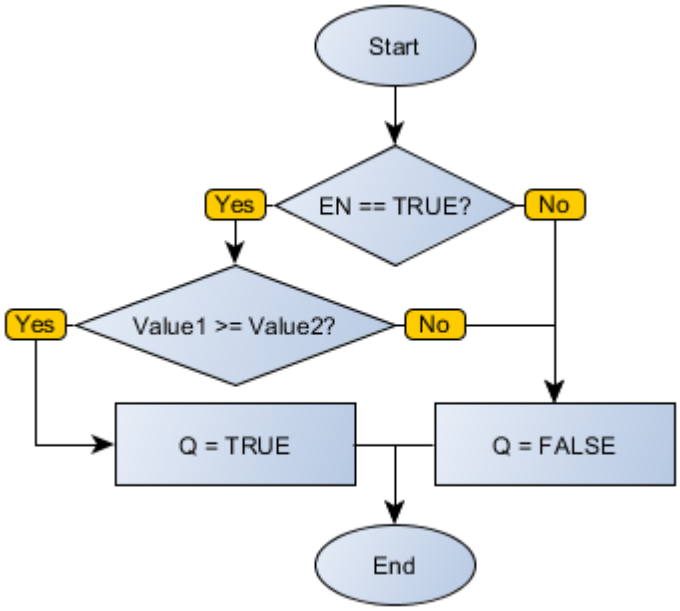
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	First value of comparison
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Second value of comparison
VAR_OUTPUT	Q	BOOL	Indicator of equality or majority of Value1

Operation

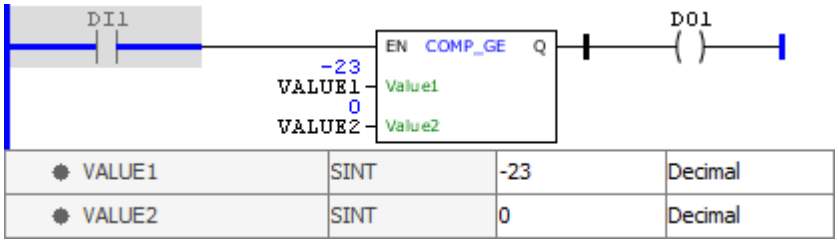
When this block has a TRUE value in EN it sends the Q output to the TRUE value if Value1 is higher than or equal to Value2. Otherwise, Q receives FALSE.

When EN has FALSE value, Q remains in FALSE.

Block Flowchart

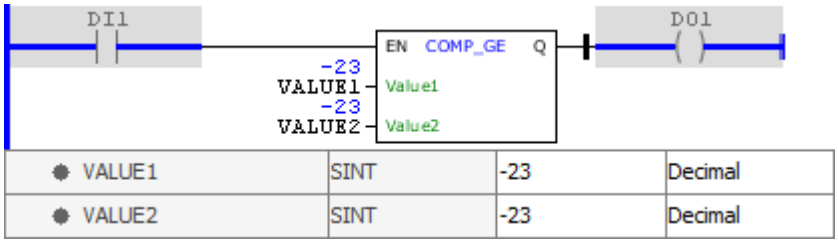


Example



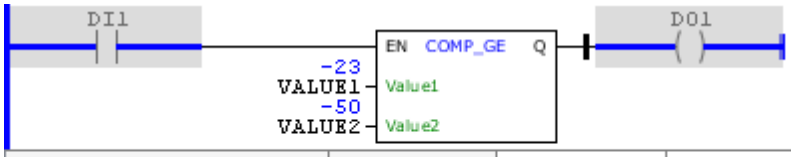
VALUE1	SINT	-23	Decimal
VALUE2	SINT	0	Decimal

The example above checks equality or majority of VALUE1 in relation to VALUE2. Since VALUE1 has lower value than VALUE2, the Q output is disabled.



VALUE1	SINT	-23	Decimal
VALUE2	SINT	-23	Decimal

The example above checks equality or majority of VALUE1 in relation to VALUE2. Since both variables have the same value, the Q output is activated.



VALUE1	SINT	-23	Decimal
VALUE2	SINT	-50	Decimal

The example above checks equality or majority of VALUE1 in relation to VALUE2. Since VALUE1 has higher value than VALUE2, the Q output is activated.

11.8.5.3.3 COMP_GT

Block that compares the values of Value1 and Value2, enabling the output Q if Value1 is higher than Value2.

Ladder Representation

Block Structure

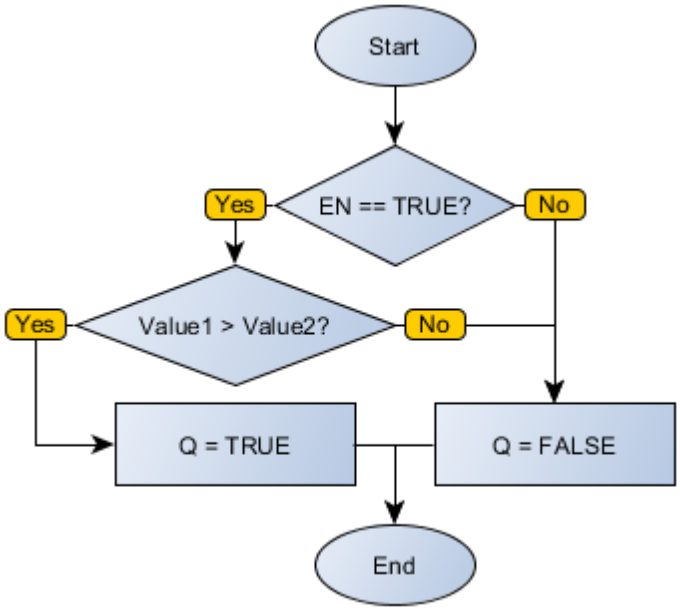
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	First value of comparison
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Second value of comparison
VAR_OUTPUT	Q	BOOL	Indicator of majority of Value1

Operation

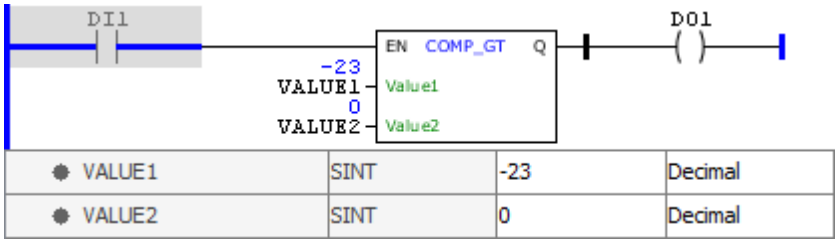
When this block has a TRUE value in EN, it sends to the Q output the TRUE value if Value1 is higher than Value2. Otherwise, Q receives FALSE.

When EN has FALSE value, Q remains in FALSE.

Block Flowchart

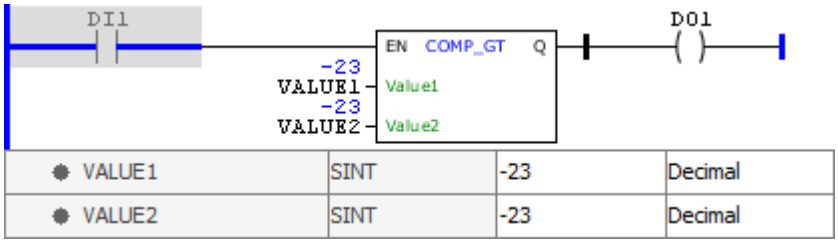


Example



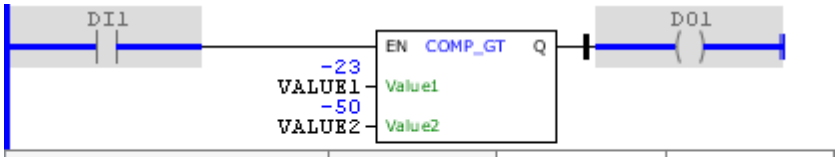
VALUE1	SINT	-23	Decimal
VALUE2	SINT	0	Decimal

The example above checks the majority of VALUE1 in relation to VALUE2. Since VALUE1 has lower value than VALUE2, the Q output is disabled.



VALUE1	SINT	-23	Decimal
VALUE2	SINT	-23	Decimal

The example above checks the majority of VALUE1 in relation to VALUE2. Since both variables have the same value, the Q output is disabled.



VALUE1	SINT	-23	Decimal
VALUE2	SINT	-50	Decimal

The example above checks the majority of VALUE1 in relation to VALUE2. Since VALUE1 has higher value than VALUE2, the Q output is activated.

11.8.5.3.4 COMP_LE

Block that compares the values of Value1 and Value2, enabling the output Q if Value1 is lower than or equal to Value2.

Ladder Representation

Block Structure

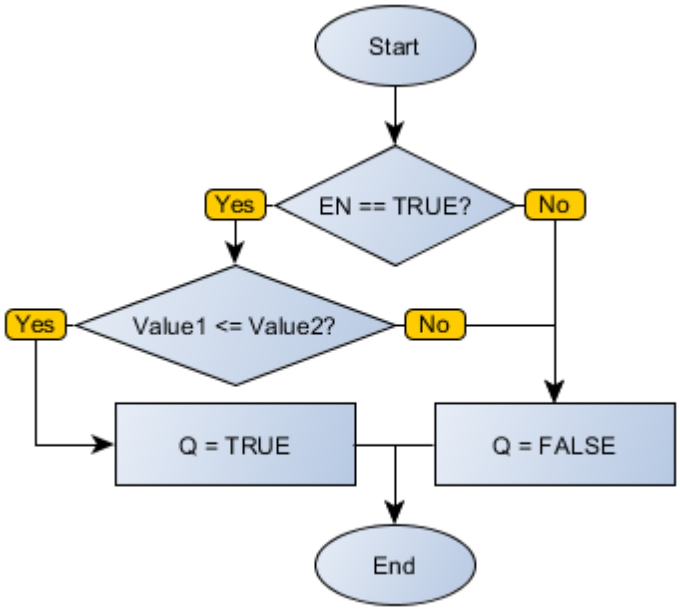
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	First value of comparison
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Second value of comparison
VAR_OUTPUT	Q	BOOL	Indicator of equality or minority of Value1

Operation

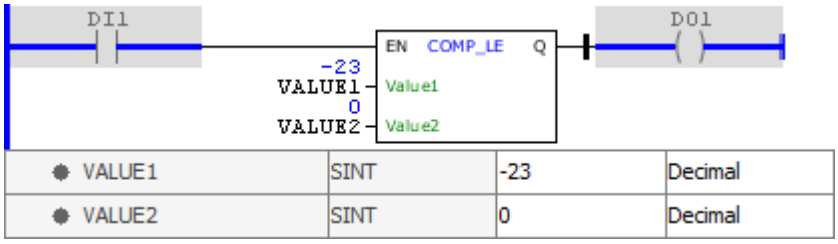
When this block has a TRUE value in EN, it sends to the Q output the TRUE value if Value1 is lower than or equal to Value2. Otherwise, Q receives FALSE.

When EN has FALSE value, Q remains in FALSE.

Block Flowchart

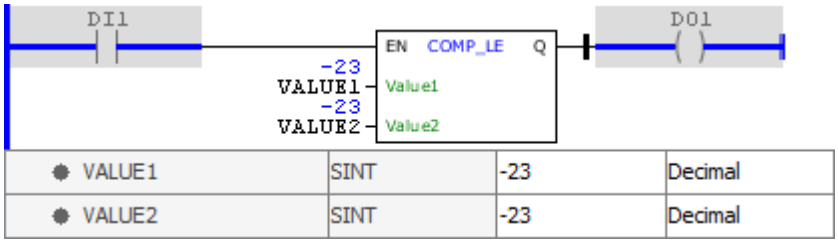


Example



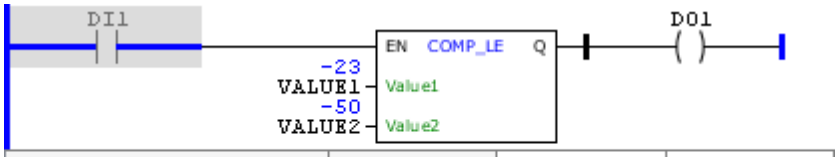
VALUE1	SINT	-23	Decimal
VALUE2	SINT	0	Decimal

The example above checks equality or minority of VALUE1 in relation to VALUE2. Since VALUE1 has lower value than VALUE2, the Q output is activated.



VALUE1	SINT	-23	Decimal
VALUE2	SINT	-23	Decimal

The example above checks equality or minority of VALUE1 in relation to VALUE2. Since both variables have the same value, the Q output is activated.



VALUE1	SINT	-23	Decimal
VALUE2	SINT	-50	Decimal

The example above checks equality or minority of VALUE1 in relation to VALUE2. Since VALUE1 has higher value than VALUE2, the Q output is disabled.

11.8.5.3.5 COMP_LT

Block that compares the values of Value1 and Value2, enabling the output Q if Value1 is lower than Value2.

Ladder Representation

Block Structure

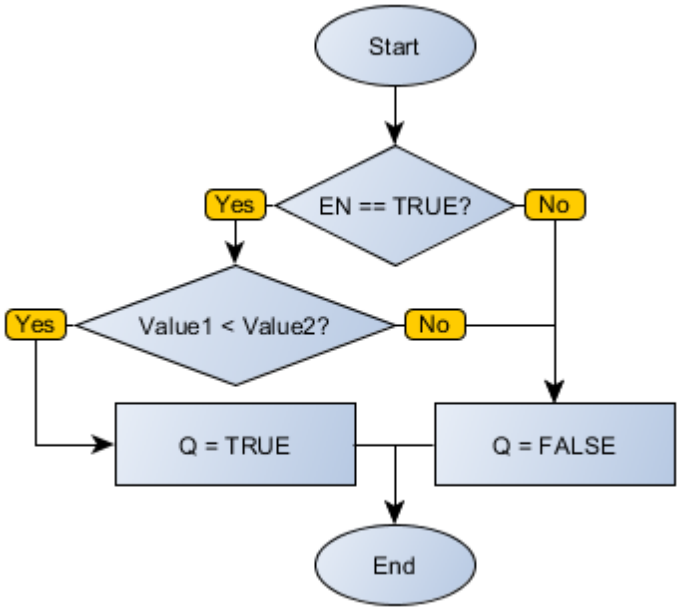
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	First value of comparison
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Second value of comparison
VAR_OUTPUT	Q	BOOL	Indicator of minority of Value1

Operation

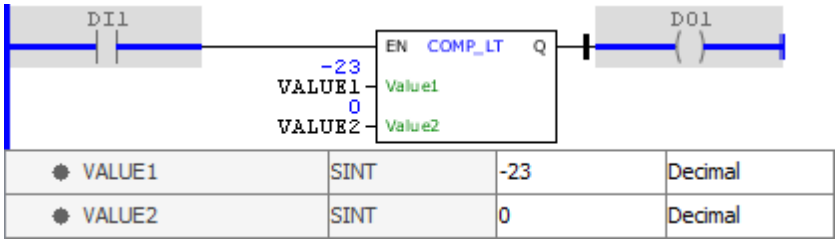
When this block has a TRUE value in EN, it sends to the Q output the TRUE value if Value1 is lower than or equal to Value2. Otherwise, Q receives FALSE.

When EN has FALSE value, Q remains in FALSE.

Block Flowchart

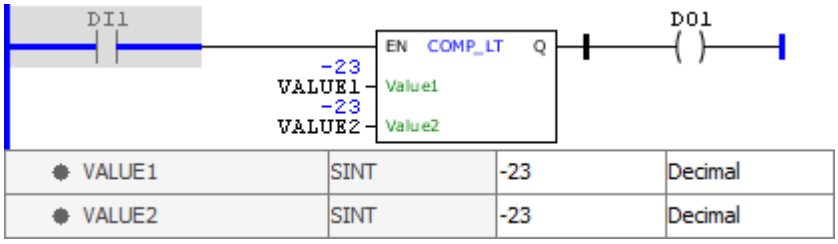


Example



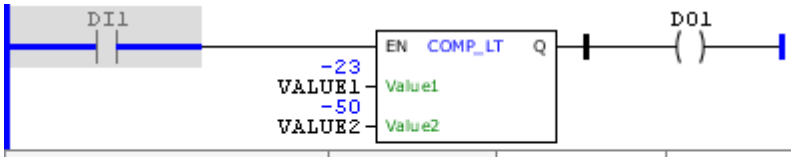
VALUE1	SINT	-23	Decimal
VALUE2	SINT	0	Decimal

The example above checks minority of VALUE1 in relation to VALUE2. Since VALUE1 has lower value than VALUE2, the Q output is activated.



VALUE1	SINT	-23	Decimal
VALUE2	SINT	-23	Decimal

The example above checks the minority of VALUE1 in relation to VALUE2. Since both variables have the same value, the Q output is disabled.



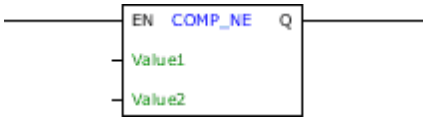
VALUE1	SINT	-23	Decimal
VALUE2	SINT	-50	Decimal

The example above checks the minority of VALUE1 in relation to VALUE2. Since VALUE1 has higher value than VALUE2, the Q output is disabled.

11.8.5.3.6 COMP_NE

Block that compares the values of Value1 and Value2, enabling the Q output if Value1 is different from Value2.

Ladder Representation



Block Structure

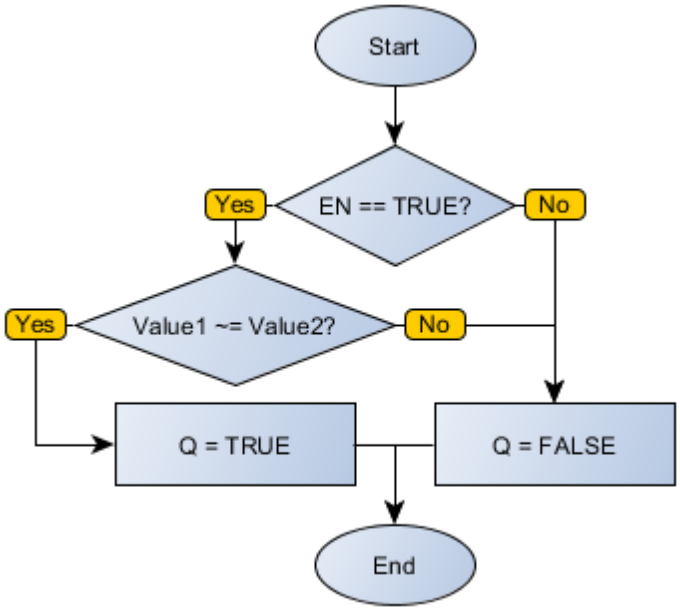
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	First value of comparison
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Second value of comparison
VAR_OUTPUT	Q	BOOL	Indicator of inequality

Operation

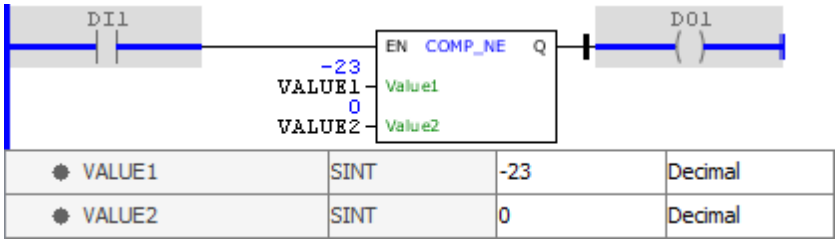
When this block has a TRUE value in EN, it sends to the Q output the TRUE value if Value1 is different from Value2. Otherwise, Q receives FALSE.

When EN has FALSE value, Q remains in FALSE.

Block Flowchart

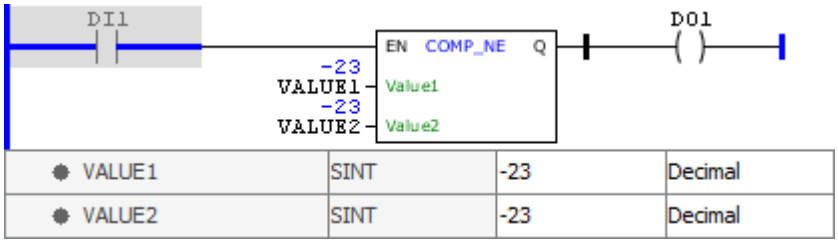


Example



VALUE1	SINT	-23	Decimal
VALUE2	SINT	0	Decimal

The example above checks inequality between VALUE1 and VALUE2. Since both variables have different values, the Q output is activated.



VALUE1	SINT	-23	Decimal
VALUE2	SINT	-23	Decimal

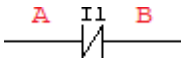
The example above checks equality between VALUE1 and VALUE2. Since both variables have the same value, the Q output is disabled.

11.8.5.4 Contact

11.8.5.4.1 NCCONTACT

Normally closed contact.

Ladder Representation



Block Structure

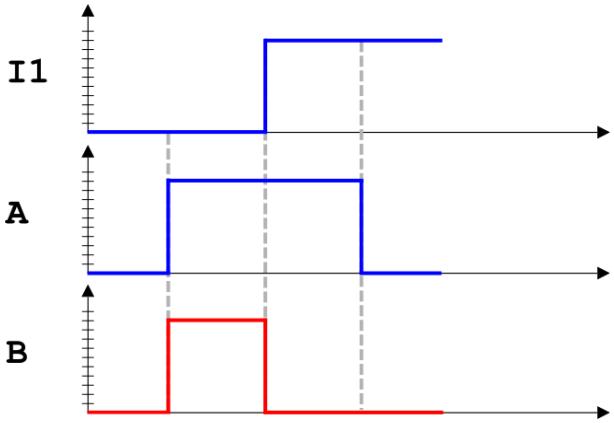
Variable Type	Name	Data Type	Description
VAR_OUTPUT	I1	BOOL	Block control input

Operation

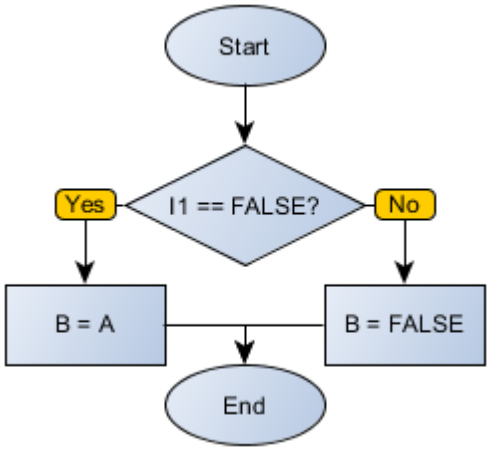
When variable I1 is with TRUE value, B receives FALSE.
When variable I1 is with FALSE value, B receives the value of A.

NOTE!
Watch out for series and parallel associations of contacts. Refer to section Contact Logic for further information.

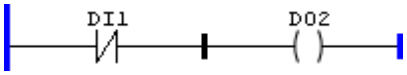
Diagram



Block Flowchart



Example

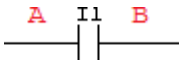


The above example performs the transfer of the opposite value of digital input DI1 to the digital output DO2.

11.8.5.4.2 NOCONTACT

Normally open contact.

Ladder Representation



Block Structure

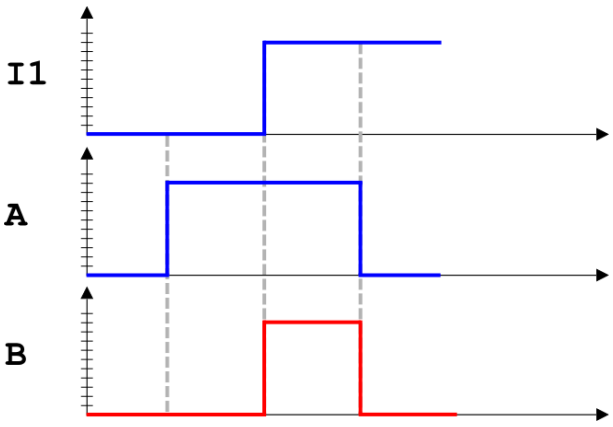
Variable Type	Name	Data Type	Description
VAR_OUTPUT	I1	BOOL	Block control input

Operation

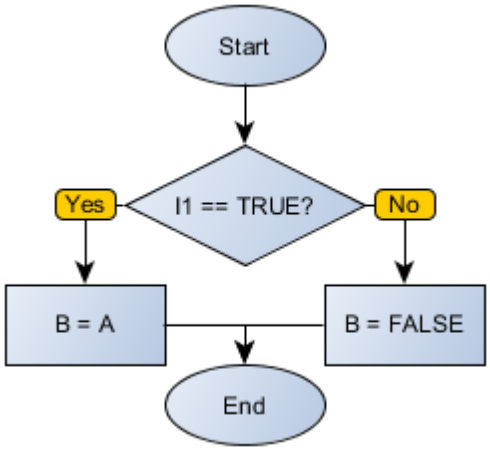
When variable I1 is with FALSE value, B receives FALSE.
When variable I1 is with TRUE value, B receives the value of A.

NOTE!
Watch out for series and parallel associations of contacts. Refer to section Contact Logic for further information.

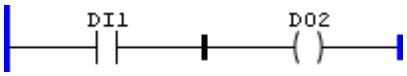
Diagram



Block Flowchart



Example



The above example performs the transfer of the value of digital input DI1 to the digital output DO2.

11.8.5.4.3 NTSCONTACT

Falling edge transition contact.

Ladder Representation

Block Structure

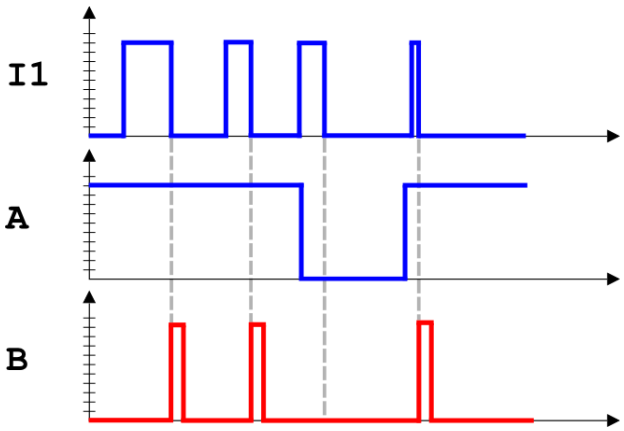
Variable Type	Name	Data Type	Description
VAR_INPUT	I1	BOOL	Block control input
VAR	NTSCONTACT_INST_0	NTSCONTACT	Instance of access to block structure

Operation

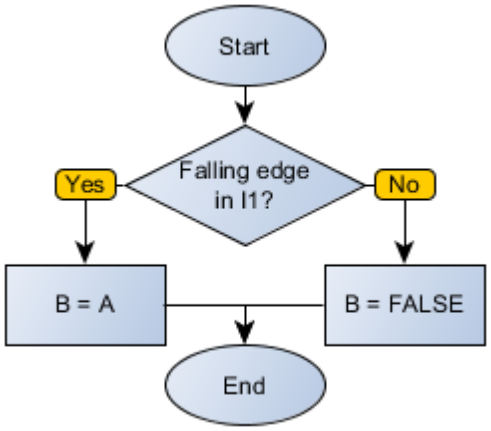
At the instant the variable I1 transitions from TRUE to FALSE (falling edge or negative edge transition), B receives the value of A for a scan cycle.
At all other times, B receives the FALSE value.

NOTE!
Watch out for series and parallel associations of contacts. Refer to section Contact Logic for further information.

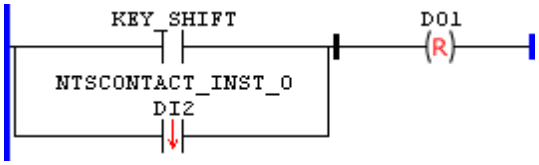
Diagram



Block Flowchart



Example

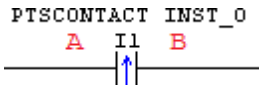


The above example resets the digital output DO1 if the SHIFT key is pressed or a positive pulse on the digital input DI2 is given.

11.8.5.4.4 PTSCONTACT

Leading edge transition contact.

Ladder Representation



Block Structure

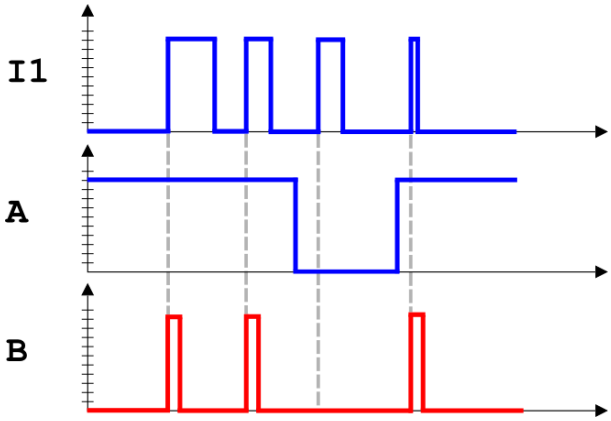
Variable Type	Name	Data Type	Description
VAR_INPUT	I1	BOOL	Block control input
VAR	PTSCONTACT_INST_0	PTSCONTACT	Instance of access to block structure

Operation

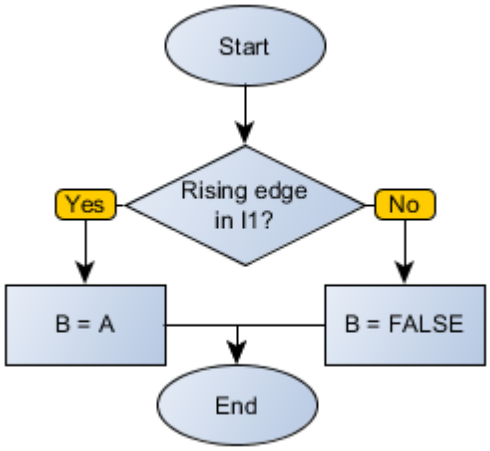
At the instant the variable I1 transitions from FALSE to TRUE (leading edge or positive edge transition), B receives the value of A for a scan cycle.
At all other times, B receives the FALSE value.

NOTE!
Watch out for series and parallel associations of contacts. Refer to section Contact Logic for further information.

Diagram



Block Flowchart



Example

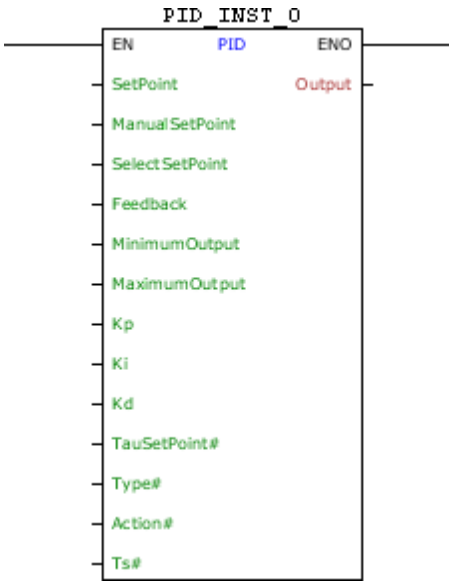
The above example resets the digital output DO1 if the SHIFT key is pressed and a positive pulse on the digital input DI2 is given.

11.8.5.5 Control

11.8.5.5.1 PID

Block that performs the function of a discrete PID controller. From the input variables, it calculates the corresponding controller output.

Ladder Representation



Block Structure

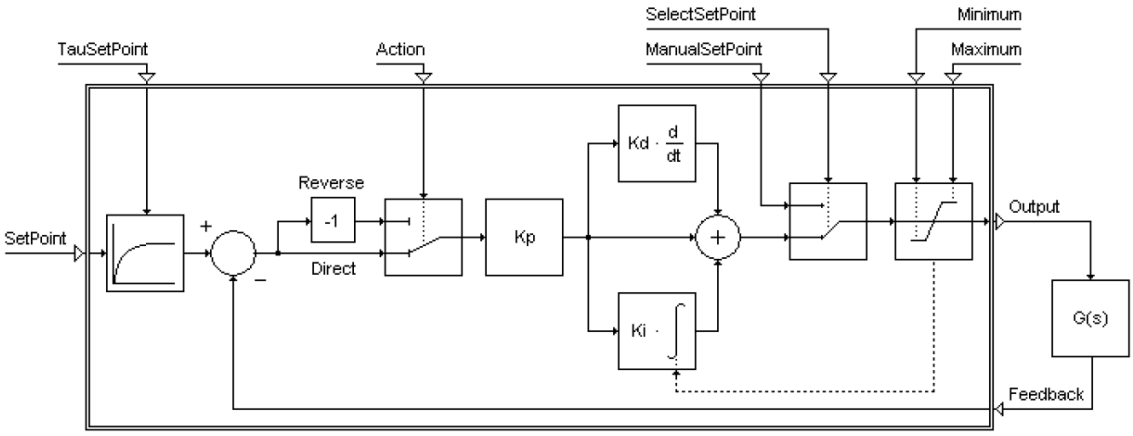
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	SetPoint	REAL	Automatic reference (pre-control)
	ManualSetPoint	REAL	Forced reference (post control)
	SelectSetPoint	BOOL	Selects w hich reference to use
	Feedback	REAL	Feedback loop variable
	MinimumOutput	REAL	Minimum value of the controller output
	MaximumOutput	REAL	Maximum value of the controller output
	Kp	REAL	Proportional gain
	Ki	REAL	Integral gain
	Kd	REAL	Derivative gain
	TauSetPoint#	REAL	Time constant of the automatic reference in put filter
	Type#	BYTE	Controller type
	Action#	BYTE	Control action
	Ts#	UINT	Sampling time [ms]
VAR_OUTPUT	ENO	BOOL	Output enabling
	Output	REAL	Controller output
VAR	PID_INST_0	PID	Instance of access to block structure

Operation

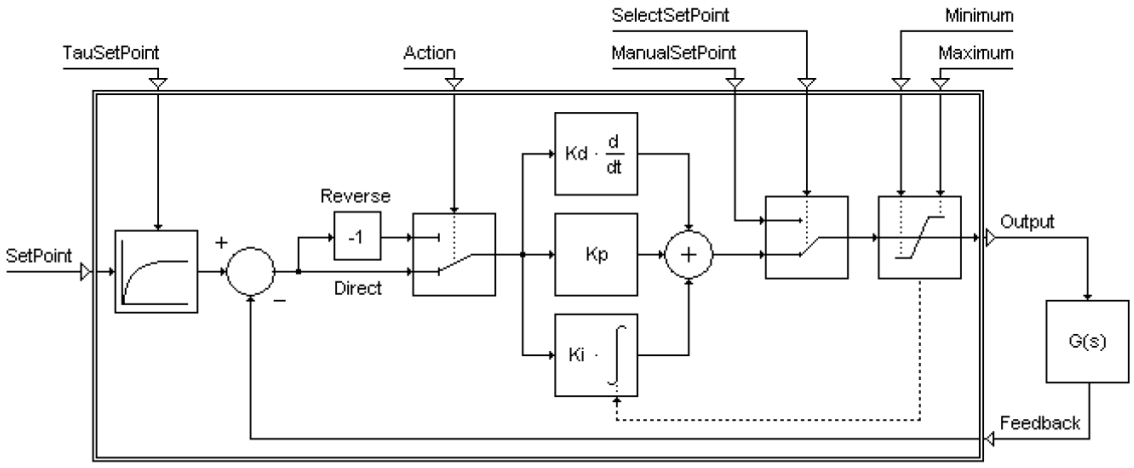
On the positive transition edge in EN, Output receives zero value, and the block executes its functionality as EN is at TRUE level.

When enabled, this block performs a routine PID control with the Kp, Ki and Kd parameters chosen. The PID topology used may be the Academic or Parallel, depending on what is chosen in Type#.

Academic Form:



Parallel Form:



The levels of the output signal of the controller are saturated at value MinimumOutput and MaximumOutput. The SelectSetPoint input level FALSE causes the SetPoint reference be adopted, allowing the controller maintains control over the process. When SelectSetPoint goes to TRUE level, the controller has no more domain, and ManualSetPoint becomes to be considered the output signal of the controller.

Action# will define the feedback operation. If Action# is DIRECT, the operation will be SetPoint – Feedback. If Action# is REVERSE, the operation will be Feedback – SetPoint.

Feedback receives the process variable considered as the plant output. Ts# receives the sampling time for the controller and # TauSetPoint receives the time constant for the input filter of the automatic reference.

When EN has FALSE value, Output remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

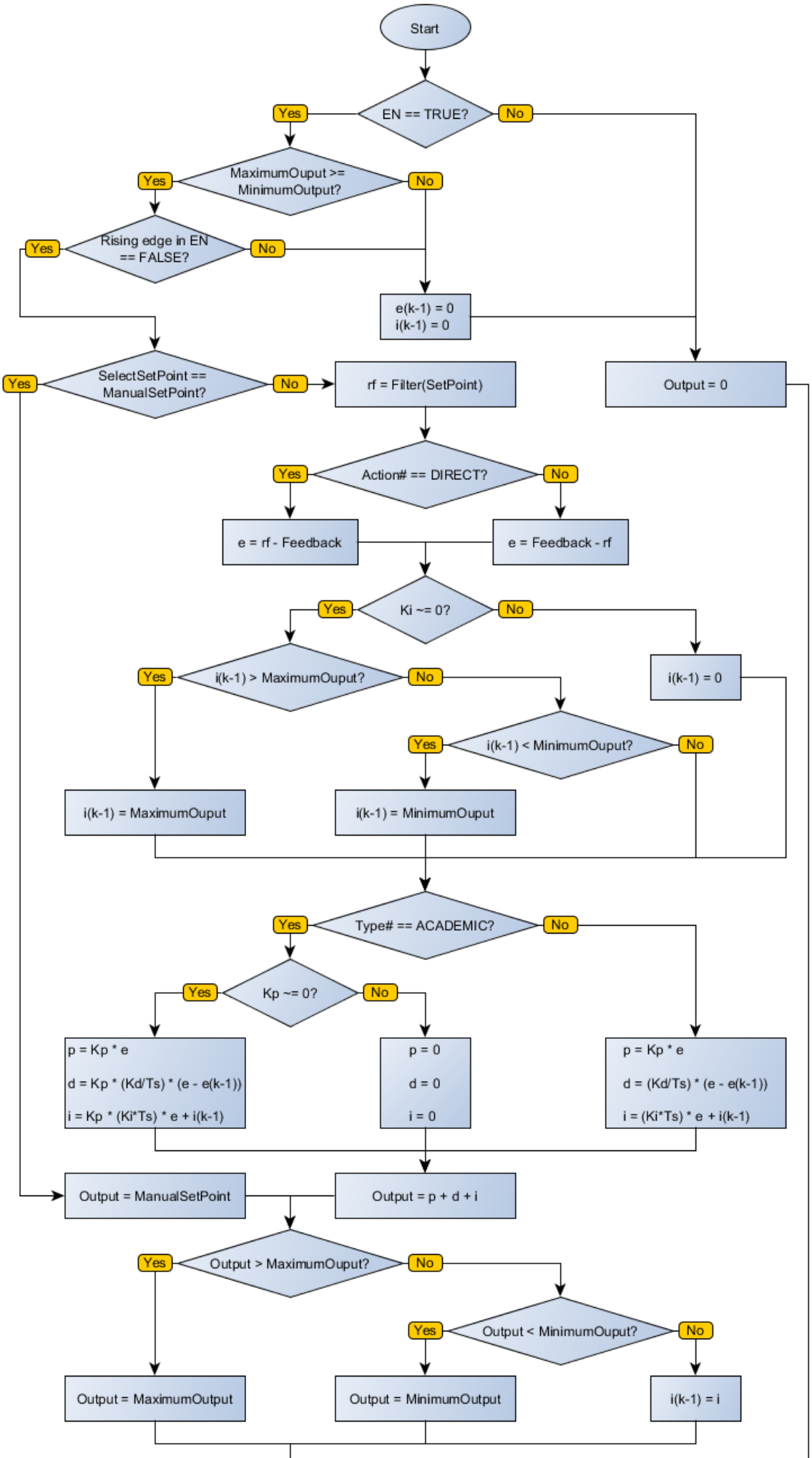
NOTE!
Effects of the alteration of gains on the process

- If Kp decreases, the process becomes slower; generally more stable or less oscillating; it has less overshoot.
- If Kp increases, the process responds faster; it may become more unstable or more oscillating; it has more overshoot.
- If Ki decreases, the process becomes slower, lagging to reach the "SetPoint"; it becomes more stable or less oscillating; it has less overshoot.
- If Ki increases, the process becomes faster, quickly reaching the "SetPoint"; it becomes more unstable or more oscillating; it has more overshoot.
- If Kd decreases, the process becomes slower; it has less overshoot.
- If Kd increases, it has more overshoot.

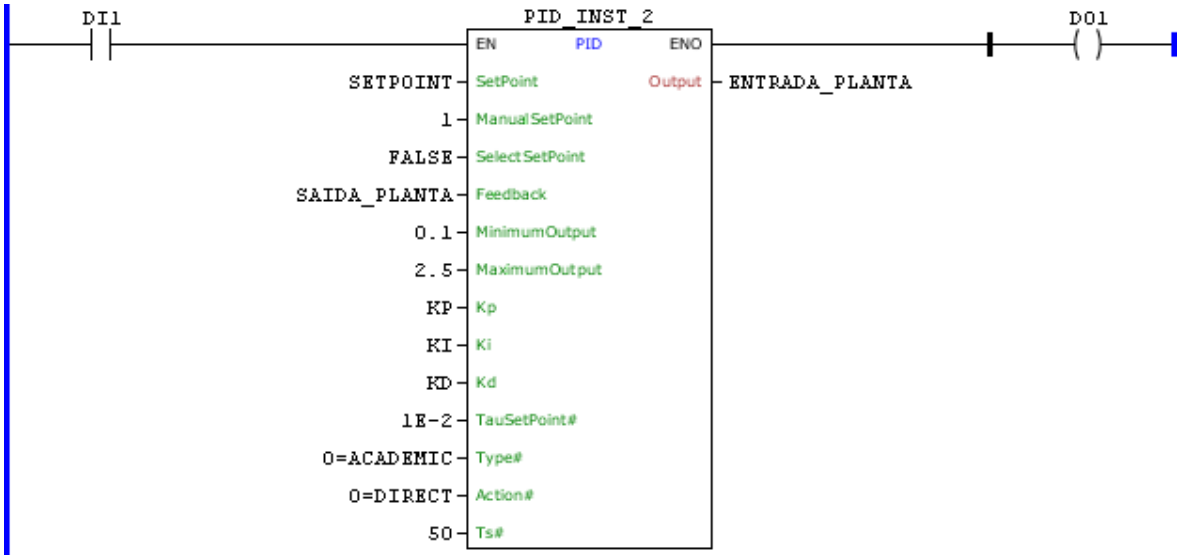
NOTE!
How to improve the performance of the process through the adjustment of gains (valid for the Academic PID)

- If the performance of the process is almost good, but the overshoot is a bit high, try to: (1) decrease Kp 20%, (2) decrease Ki 20% and/or (3) decrease Kd 50%.
- If the performance of the process is almost good, but it does not have overshoot and lags to reach the "SetPoint", try to: (1) increase Kp 20%, (2) increase Ki 20% and/or (3) increase Kd 50%.
- If the performance of the process is good, but the process output is varying too much, try to: (1) increase Kd 50%, (2) decrease Kp 20%.
- If the performance of the process is bad, i.e. after start up, the transitory lasts several periods of oscillation that reduce very slowly or never reduce at all, try to: (1) decrease Kp 50%.
- If the performance of the process is bad, i.e. after start up it slowly moves towards the "SetPoint" without overshoot, but is still very far and the process output is less than the rated value, try to: (1) increase Kp 50%, (2) increase Ki 50%, (3) increase Kd 70%.

Block Flowchart


Start
EN == TRUE?
Yes
No
MaximumOuput >= MinimumOutput?
Yes
No
Rising edge in EN == FALSE?
Yes
No
e(k-1) = 0
i(k-1) = 0
SelectSetPoint == ManualSetPoint?
Yes
No
rf = Filter(SetPoint)
Output = 0
Action# == DIRECT?
Yes
No
e = rf - Feedback
e = Feedback - rf
Ki ~= 0?
Yes
No
i(k-1) > MaximumOuput?
Yes
No
i(k-1) = 0
i(k-1) < MinimumOuput?
Yes
No
i(k-1) = MaximumOuput
i(k-1) = MinimumOuput
Type# == ACADEMIC?
Yes
No
Kp ~= 0?
Yes
No
p = Kp * e
d = Kp * (Kd/Ts) * (e - e(k-1))
i = Kp * (Ki*Ts) * e + i(k-1)
p = 0
d = 0
i = 0
p = Kp * e
d = (Kd/Ts) * (e - e(k-1))
i = (Ki*Ts) * e + i(k-1)
Output = ManualSetPoint
Output = p + d + i
Output > MaximumOuput?
Yes
No
Output < MinimumOuput?
Yes
No
Output = MaximumOutput
Output = MinimumOutput
i(k-1) = i

Example

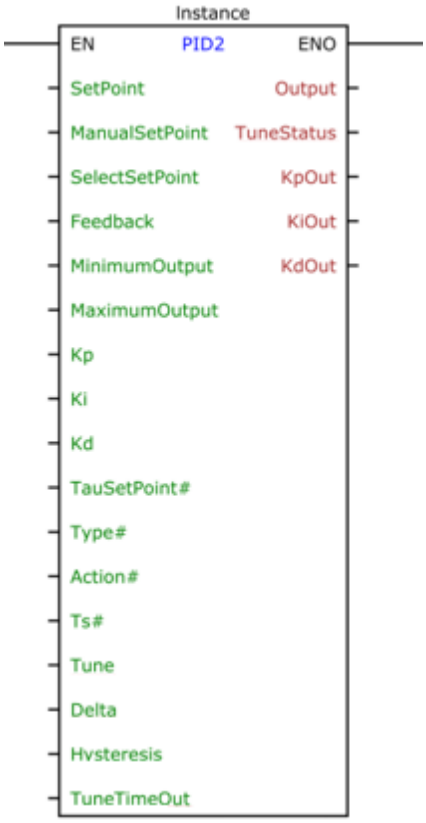


The above example creates a loop of a digital PID form with sampling time 50 ms, using the constants KP, KI and KD for control. Automatic reference SETPOINT, filtered by a first order filter with time constant of 0:01 is used. The error signal is calculated as the difference between the filtered reference and variable SAIDA_PLANTA. The controller output is saturated between the values 0.1 and 2.5 and sent to the variable ENTRADA_PLANTA.

11.8.5.5.2 PID2

Block that performs auto tuning of a discrete PID controller. From the input variables, it calculates the corresponding controller output PID or PI throw relay method. This block also implements the obtained controller or another that the user wishes.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enable
	SetPoint	REAL	Automatic reference (pre-control)
	ManualSetPoint	REAL	Forced reference (post-control)
	SelectSetPoint	BOOL	Select which reference to use
	Feedback	REAL	Mesh Feedback Variable
	MinimumOutput	REAL	Minimum value of controller output
	MaximumOutput	REAL	Maximum value of controller output
	Kp	REAL	Proportional gain
	Ki	REAL	Integral gain
	Kd	REAL	Derivative Gain
	TauSetPoint#	REAL	Auto reference input filter time constant
	Type#	BYTE	Controller Type
	Action#	BYTE	Control action
	Ts#	UINT	Sampling period [ms]
	Tune	BYTE	It starts the tuning process according to Table 2
	Delta	REAL	Parameter of the method of the relays that represents the variation of the manipulated variable with respect to the value that

			reached the reference.
	Hysteresis	REAL	Determines the hysteresis in the switching of the Relay
	TuneTimeOut	DWORD	Maximum wait time until tuning is complete [ms]
VAR_OUTPUT	ENO	BOOL	Output enable
	Output	REAL	Controller Output
	TuneStatus	BYTE	Automatic tuning status as per Table 3
	KpOut	REAL	Proportional gain obtained in tuning
	KiOut	REAL	Full gain achieved in tuning
	KdOut	REAL	Derivative gain obtained in tuning
VAR	PID2_INST_0	PID2	Block Structure Access Instance

Table1

Operation

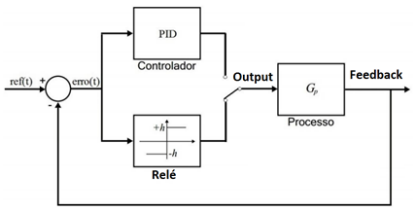
The operation of the PID2 block is divided into two parts: Control and Automatic Tuning.
The Tune variable defines which mode of operation. Whenever Tune = 0, the system operates in control mode, with values of the user defined gains, in the same way as the PID block.
The control part is exactly the same as the PID block, see the PID Help.

When Tune receives a value other than 0, the automatic tuning process starts. Some rules need to be obeyed so that tuning can occur. They will be displayed in the sequence.

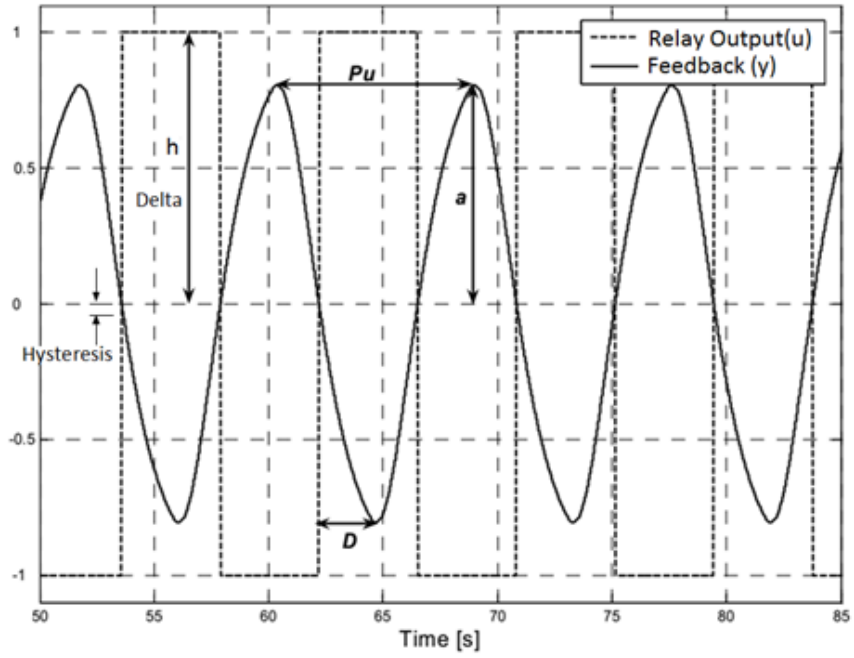
The Auto-Tuning

For the Automatic Tuning, the Relay Method is used, which is based on obtaining the critical gain K_u and critical period P_u, which can be used to obtain PID controller gains through several tuning rules: Ziegler-Nichols, Ciancone-Marlin, Tyreus- Luyben, ITAE, among others.

During the tuning process, the control is switched off and a "relay" determines the output of the controller (Output), as shown below.



At start, the relay (Output) goes to the value $+h$ and after a period D the output y (Feedback) starts to increase. When the output becomes higher than the value of the setpoint + hysteresis, the relay switches to $-h$, and so on, as shown below.



Since the MV value of the manipulated variable (Output) stabilized at the point of operation, we have:

$$h = Delta + MV$$

In the case of image above, $MV = 0$.

From the obtained subtraction curve, the necessary parameters are obtained to obtain the controller.

Criteria for tuning to be performed

In order for automatic tuning to be performed properly, two criteria must be obeyed:

1) place the system at the operating point to be calibrated (the difference must be less than 20%);

2) variable $Output$, which is the manipulated variable, may not be showing oscillation greater than 20%;

OBS: you can use the Manual mode of the block to reach the operating point without large oscillations (via the variable $ManualSetPoint$).

Step by step to perform tuning (portuguese only)

1) $Tune$ must start with the value zero;

2) Set a value for $Delta$. An initial value of 10% of the manipulated variable value is recommended ($Output$)

which reached the system setpoint;

Ex: Assuming that to stabilize at the point of operation $Output$ = 12.3. In this case, use $Delta = 1.23$;

3) Set an initial value for $Hysteresis$. This value should be slightly larger than the noise present in the system. An initial value of 2% of the Setpoint value can be used if no noise is known;

Ex: for $Setpoit$ = 60.0, use $Hysteresis = 1.2$.

4) Set a value in milliseconds (ms) to $TuneTimeOut$. This value depends on the system being applied to the block. Slower systems will require a longer time. It should be sufficient for at least 10 cycles of the relay to be complete, although with 5 cycles the system generally already stabilizes;

5) Place the system in the operating point. If a pre-control has already been done, it can be applied. Another option is to use the manual mode and vary input $ManualSetPoint$ until the system stabilizes at the point of operation, that is, $Feedback$ as close as possible to $SetPoint$;

6) Enable tuning by choosing the type of control (PI or PID and the rule used) through the variable $Tune$, as shown in Table 2;

7) The output $TuneStatus$ will inform you the status of the automatic tuning process, as shown in Table 9. At the end of the process, the output $TuneStatus$ value will be received if the calibration is performed successfully and the outputs $KpOut$, $KiOut$ and $KdOut$ will be updated with the values obtained in the tuning process.

Tune	Controller
0	Disabled
1	Automatic PID Controller
2	Automatic PI Controller
3	Tyreus-Luyben PID Controller
4	Tyreus-Luyben PI Controller
5	PID ITAE Controller
6	PI ITAE Controller
7	Ciancone-Marlin PID Controller
8	Ciancone-Marlin PI Controller
9	Ziegler-Nichols PID Controller
10	Ziegler-Nichols PI Controller

Table 2

Tune Status	Meaning
0	Disabled

1	High level relay
2	Low level relay
4	Stabilized system
5	Stable and Relay at high level
6	Stable and Relay at low level
8	Tuning completed
16	Reserved
32	Timeout
64	Method nonexistent
128	Busy: another block in tune

Table 3

The Tyreus-Luyben methods are recommended for systems with the dominant time constant in relation to the transport delay. Ciancone-Marlin methods are recommended for systems with a long transport delay. The minimum ITAE method is for intermediary situations.

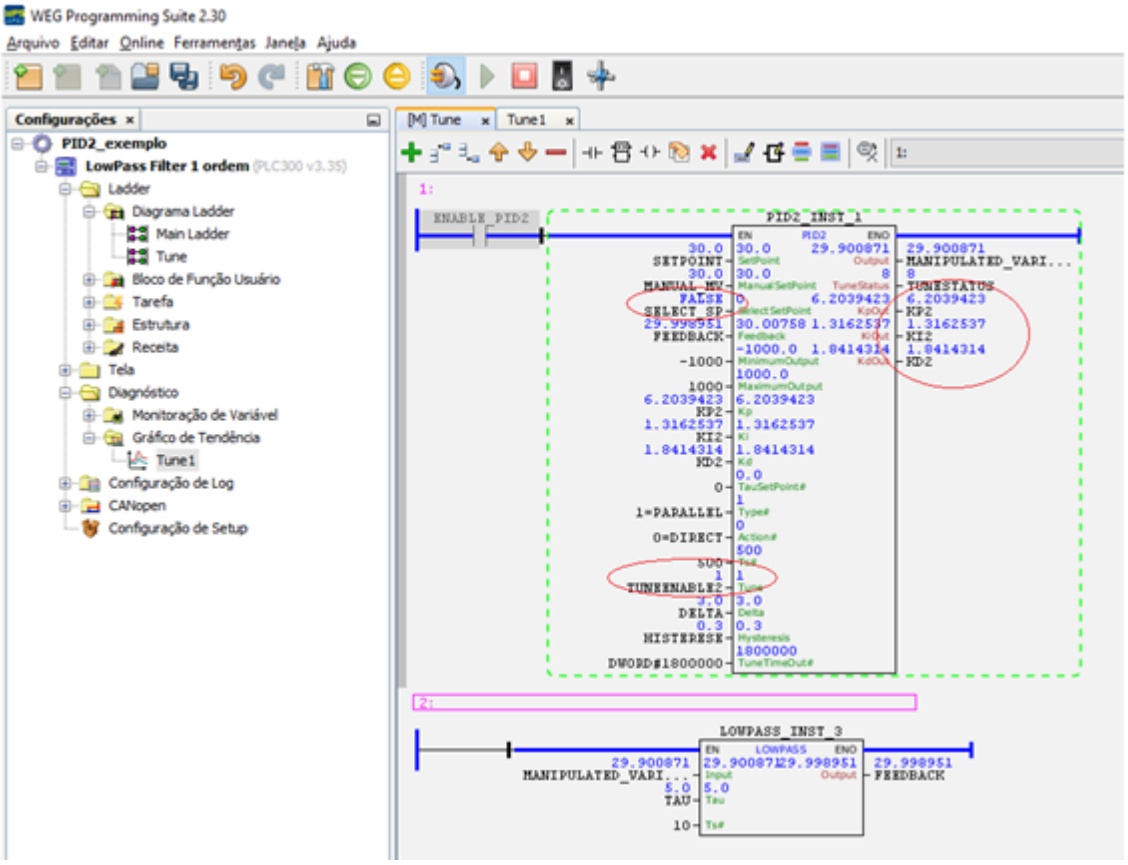
It is recommended for initial tuning or for users without some practice with the method of the relays, the use of automatic 1 or 2 for $Tune$, leaving the system to choose the most appropriate way.

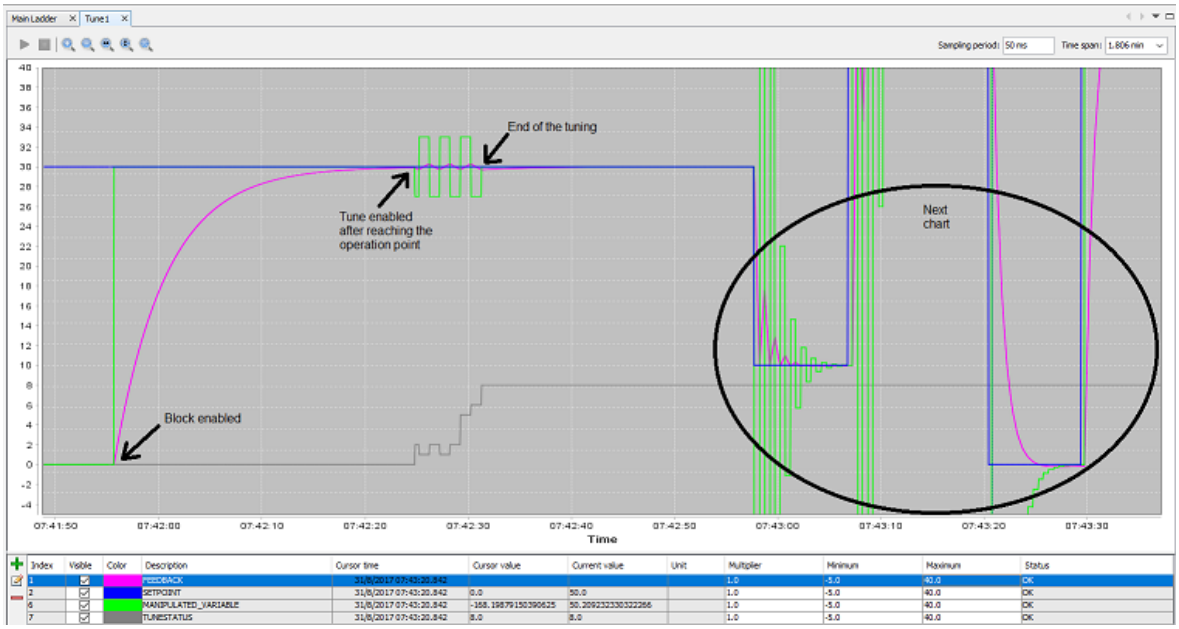
After tuning, the method can be changed causing a new controller to be calculated without re-tuning, since the required data is stored internally. This allows other methods to be easily experienced.

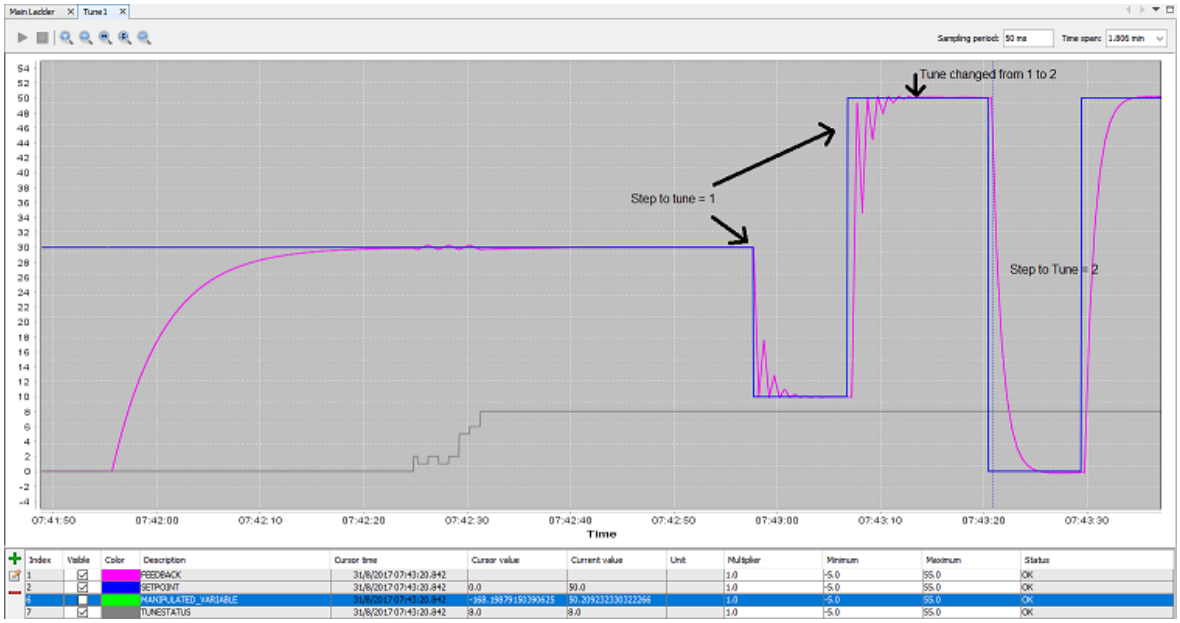
For a new tuning to be made, $Tune$ need to receive the value zero.

Block Flowchart
N/A

Example


WEG Programming Suite 2.30
Arquivo Editar Online Ferramentas Janela Ajuda
Configurações
PID2_exemplo
LowPass Filter 1 ordem (PLC300 v3.35)
Ladder
Diagrama Ladder
Main Ladder
Tune
Bloco de Função Usuário
Tarefa
Estrutura
Receita
Tela
Diagnóstico
Monitoração de Variável
Gráfico de Tendência
Tune1
Configuração de Log
CANopen
Configuração de Setup
[M] Tune
Tune1
ENABLE_PID2
PID2_INST_1
SETPOINT
MANUAL_MV
SELECT_SP
FEEDBACK
KP2
KI2
KD2
TUNEENABLE2
DELTA
HISTERESE
LOWPASS_INST_3
MANIPULATED_VARI...
TAU
FEEDBACK


End of the tuning
Tune enabled
after reaching the
operation point
Block enabled
Next
chart
Time



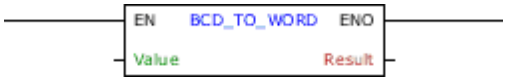
11.8.5.6 Conversion

11.8.5.6.1 BCD

11.8.5.6.1.1 BCD_TO_WORD

Block that performs the conversion of a BCD code into a WORD value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	WORD UINT INT	Value in BCD
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	WORD UINT INT	Value in WORD

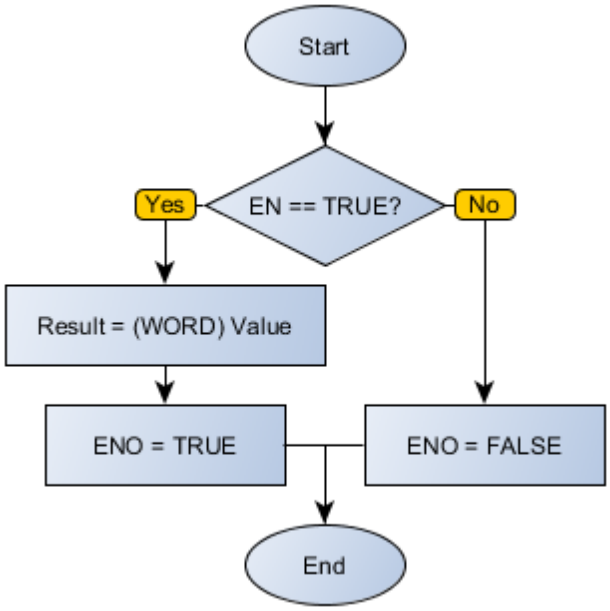
Operation

When this block has a TRUE value in EN, it interprets the Value value as BCD and converts it to WORD, storing in Result.

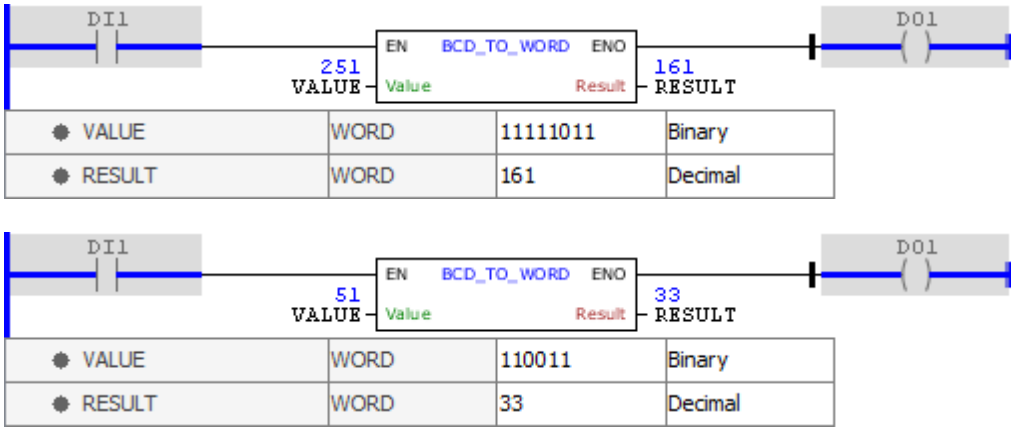
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example

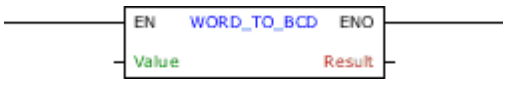


The above example converts the VALUE variable, in BCD, into a WORD value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.8.5.6.1.2 WORD_TO_BCD

Block that performs the conversion of a WORD value into a BCD code.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	WORD UINT INT	Value in WORD
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	WORD UINT INT	Value in BCD

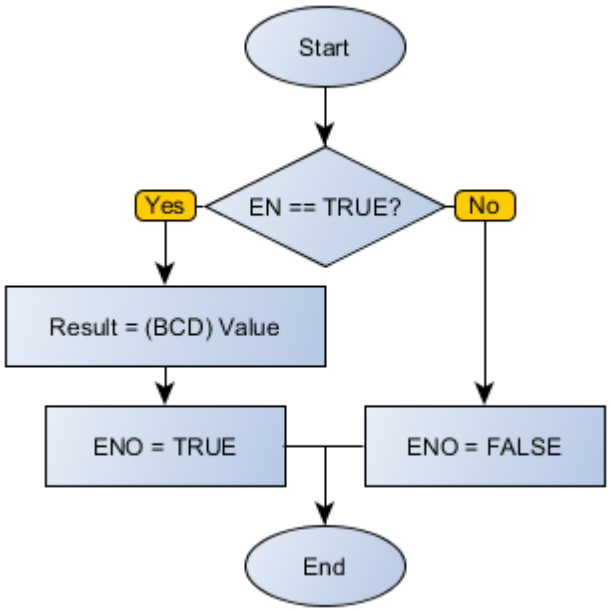
Operation

When this block has a TRUE value in EN, it interprets the Value value as WORD and converts it into BCD, storing in Result.

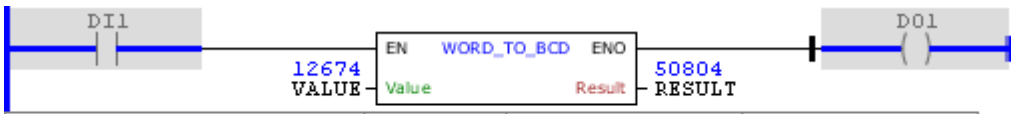
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

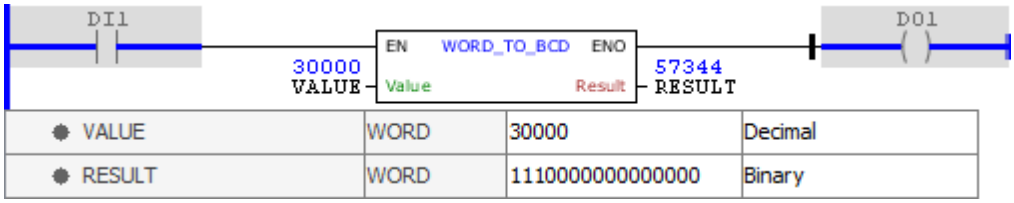
Block Flowchart



Example



VALUE	WORD	12674	Decimal
RESULT	WORD	1100011001110100	Binary



VALUE	WORD	30000	Decimal
RESULT	WORD	1110000000000000	Binary

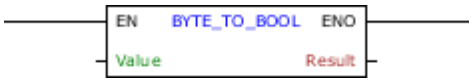
The examples above perform the conversion of VALUE variable, in WORD, into a BCD value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.8.5.6.2 BOOL

11.8.5.6.2.1 BYTE_TO_BOOL

Block that performs the conversion of a BYTE value into a BOOL value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT	Value in BYTE
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BOOL	Value in BOOL

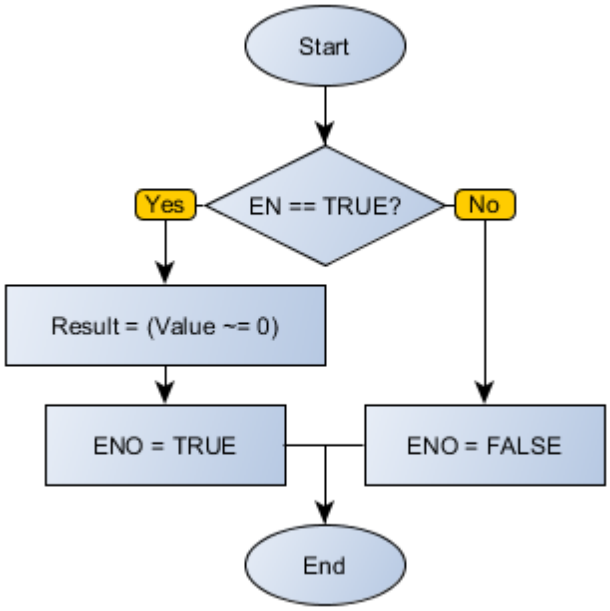
Operation

When this block has a TRUE value in EN, it interprets the Value value as BYTE and converts it into BOOL, storing in Result.

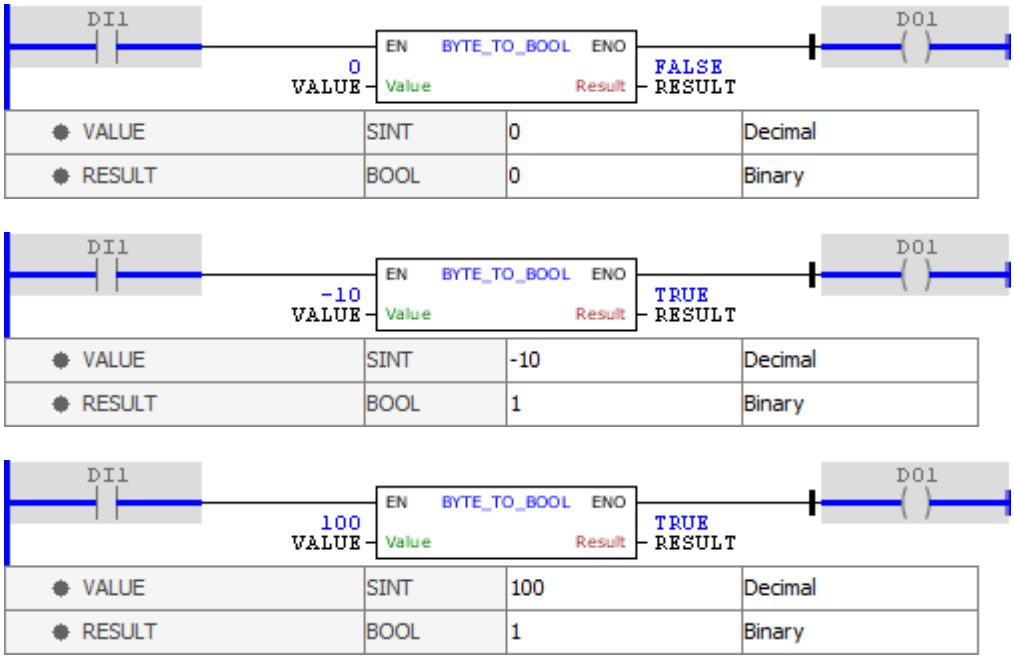
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example

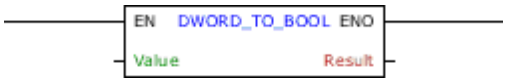


The examples above perform the conversion of VALUE variable, in BYTE, into a BOOL value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.8.5.6.2.2 DWORD_TO_BOOL

Block that performs the conversion of a DWORD value into a BOOL value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	DWORD UDINT DINT	Value in DWORD
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BOOL	Value in BOOL

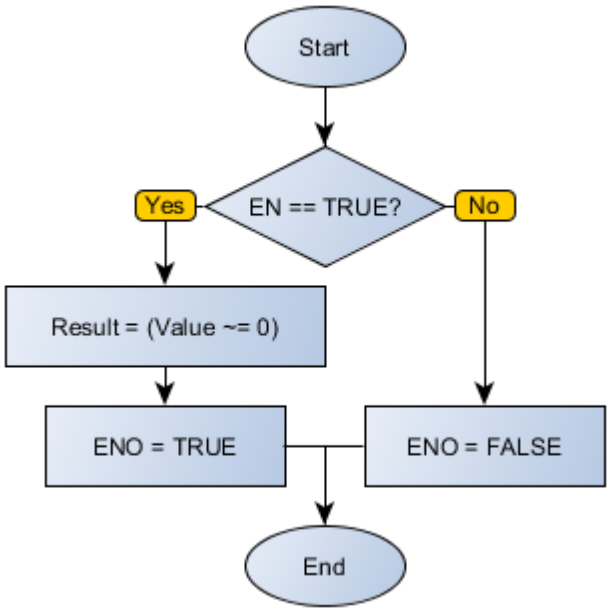
Operation

When this block has a TRUE value in EN, it interprets the Value value as DWORD and converts it into BOOL, storing in Result.

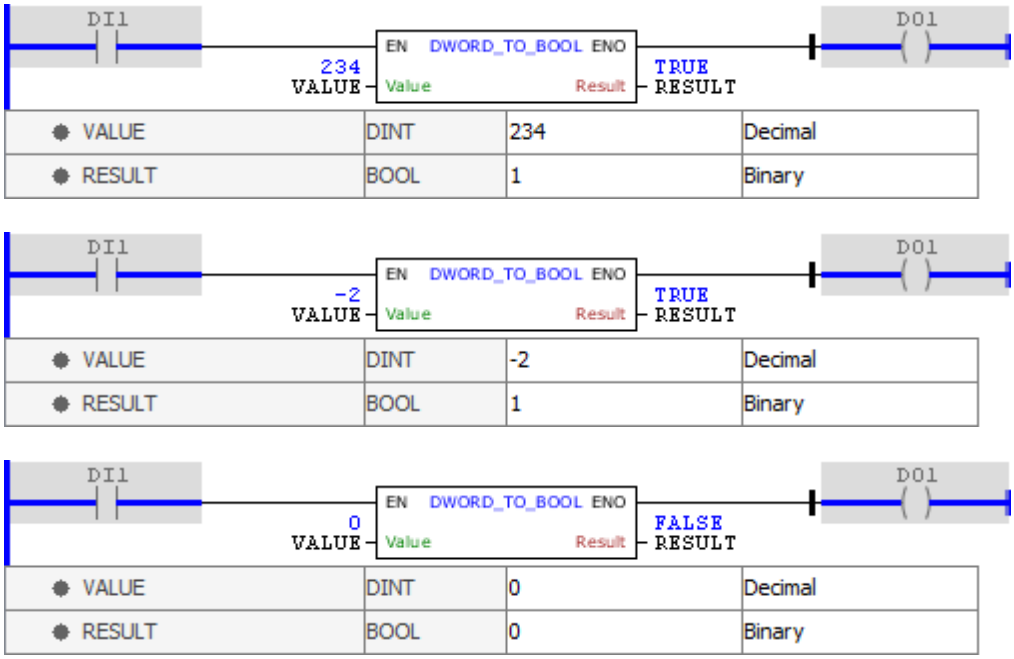
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



VALUE	DINT	234	Decimal
RESULT	BOOL	1	Binary

VALUE	DINT	-2	Decimal
RESULT	BOOL	1	Binary

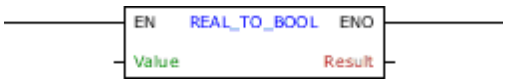
VALUE	DINT	0	Decimal
RESULT	BOOL	0	Binary

The examples above perform the conversion of VALUE variable, in DWORD, into a BOOL value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.8.5.6.2.3 REAL_TO_BOOL

Block that performs the conversion of a REAL value into a BOOL value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Value in REAL
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BOOL	Value in BOOL

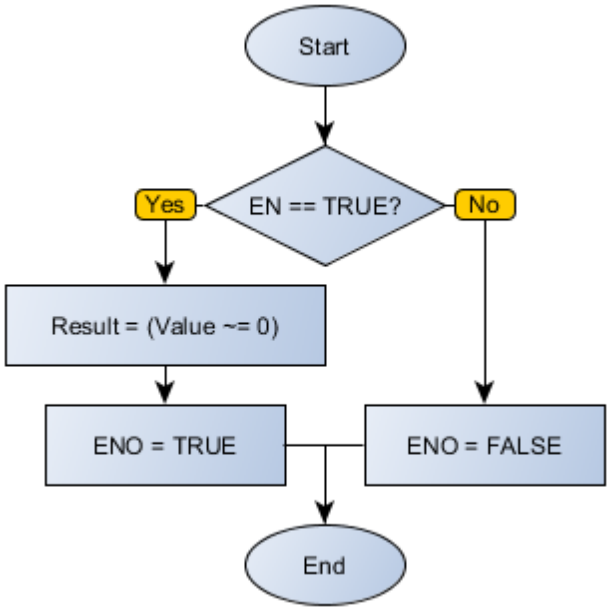
Operation

When this block has a TRUE value in EN, it interprets the Value value as REAL and converts it into BOOL, storing in Result.

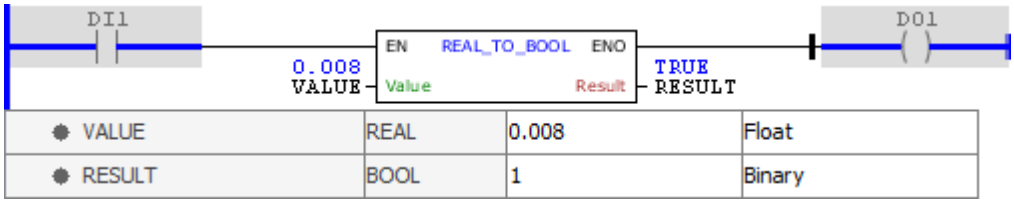
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

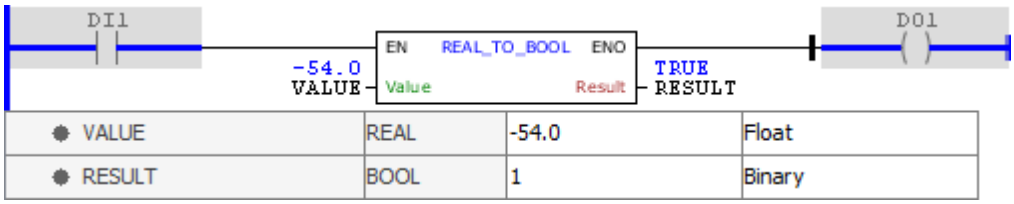
Block Flowchart



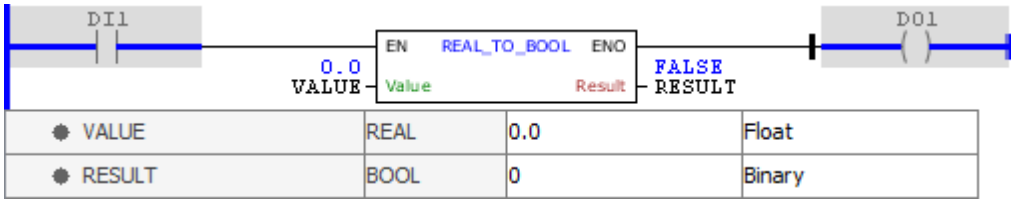
Example



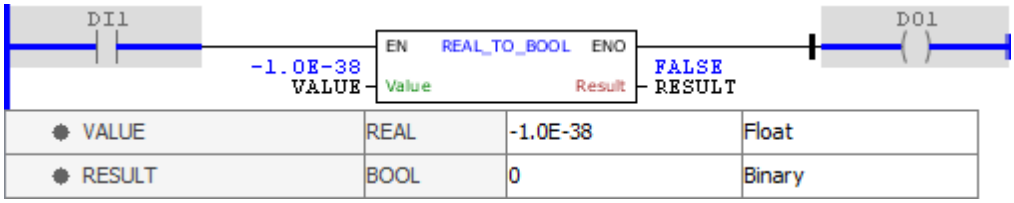
VALUE	REAL	0.008	Float
RESULT	BOOL	1	Binary



VALUE	REAL	-54.0	Float
RESULT	BOOL	1	Binary



VALUE	REAL	0.0	Float
RESULT	BOOL	0	Binary



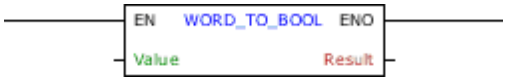
VALUE	REAL	-1.0E-38	Float
RESULT	BOOL	0	Binary

The examples above perform the conversion of VALUE variable, in REAL, into a BOOL value storing the final result in RESULT. The block ends with success and ENO output is activated. Notice in the last example that the values very close to the machine epsilon may result in an interpretation of the FALSE value.

11.8.5.6.2.4 WORD_TO_BOOL

Block that performs the conversion of a WORD value into a BOOL value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	WORD UINT INT	Value in WORD
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BOOL	Value in BOOL

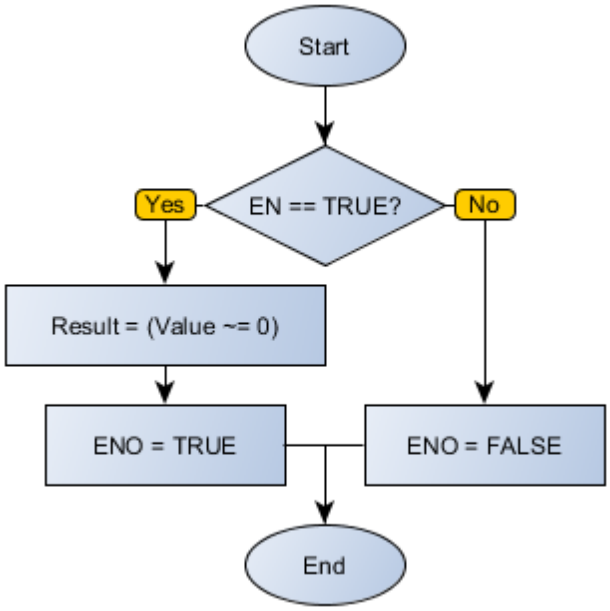
Operation

When this block has a TRUE value in EN, it interprets the Value value as WORD and converts it into BOOL, storing in Result.

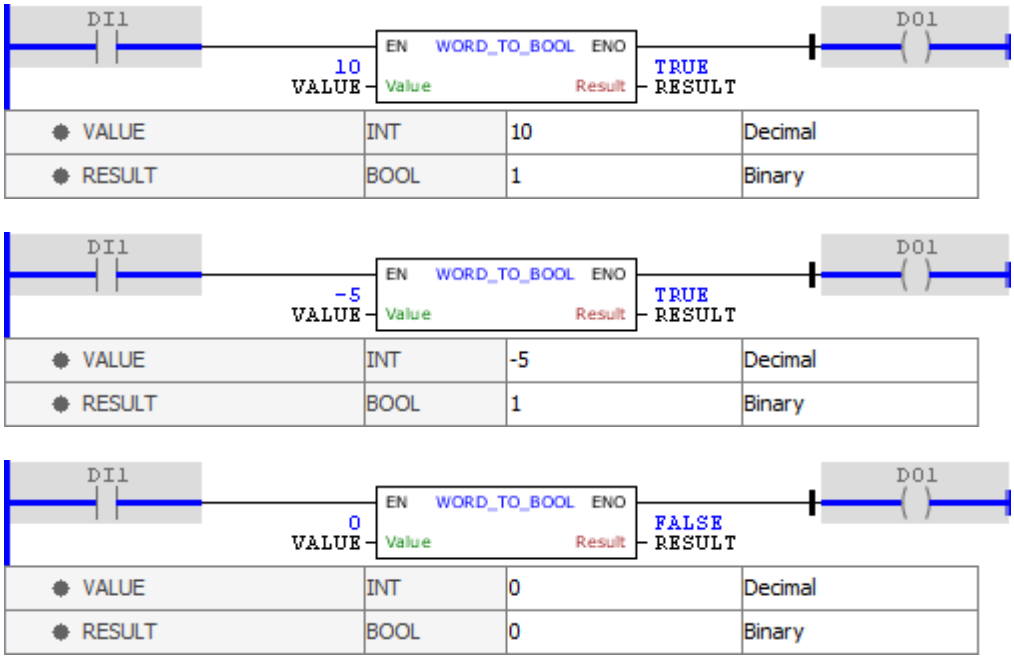
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



VALUE	INT	10	Decimal
RESULT	BOOL	1	Binary

VALUE	INT	-5	Decimal
RESULT	BOOL	1	Binary

VALUE	INT	0	Decimal
RESULT	BOOL	0	Binary

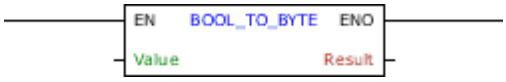
The examples above perform the conversion of VALUE variable, in WORD, into a BOOL value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.8.5.6.3 BYTE

11.8.5.6.3.1 BOOL_TO_BYTE

Block that performs the conversion of a BOOL value into a BYTE value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BOOL	Value in BOOL
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT	Value in BYTE

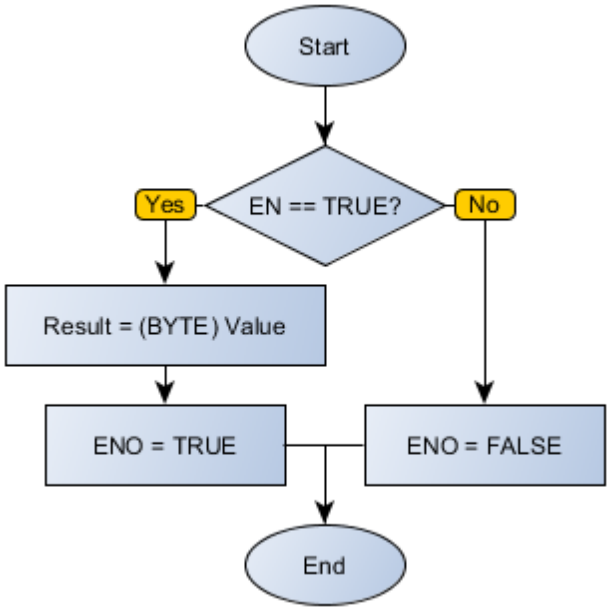
Operation

When this block has a TRUE value in EN, it interprets the Value value as BOOL and converts it into BYTE, storing in Result.

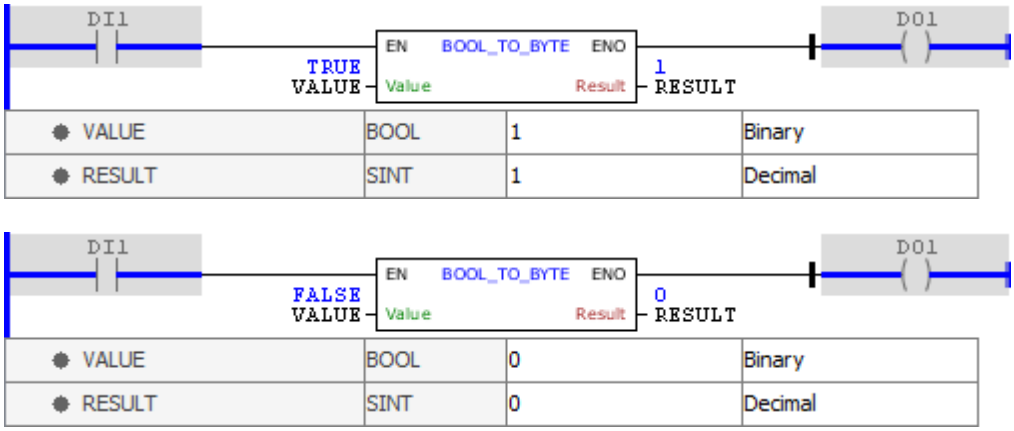
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



VALUE	BOOL	1	Binary
RESULT	SINT	1	Decimal

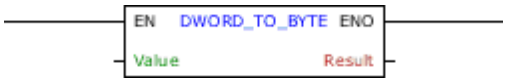
VALUE	BOOL	0	Binary
RESULT	SINT	0	Decimal

The examples above perform the conversion of variable VALUE, in BOOL, into a BYTE value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.8.5.6.3.2 DWORD_TO_BYTE

Block that performs the conversion of a DWORD value into a BYTE value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	DWORD UDINT DINT	Value in DWORD
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT	Value in BYTE

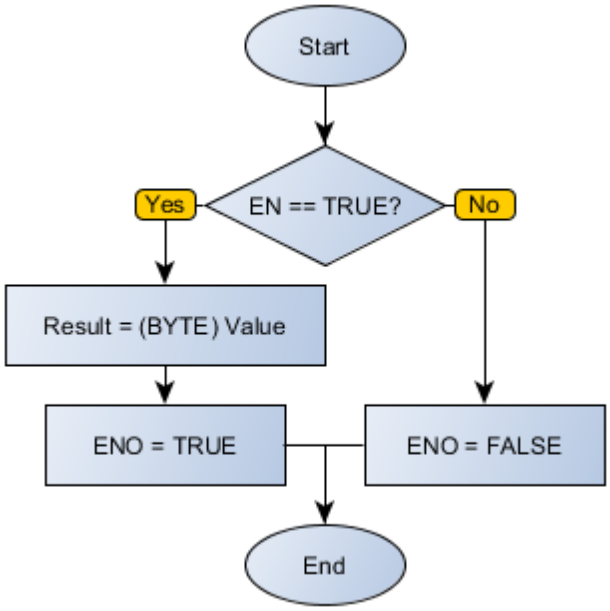
Operation

When this block has a TRUE value in EN, it interprets the Value value as DWORD and converts it into BYTE, storing in Result.

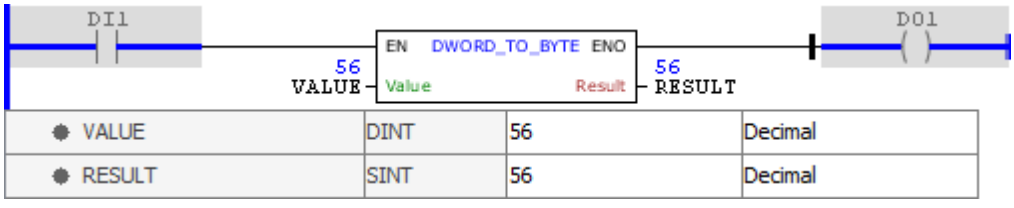
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

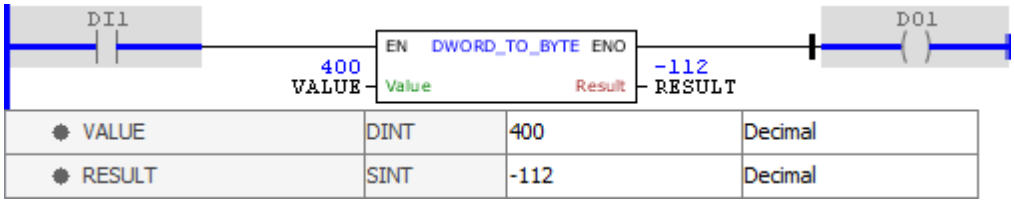
Block Flowchart



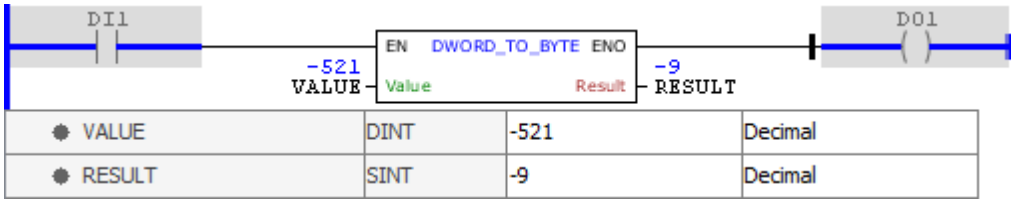
Example



VALUE	DINT	56	Decimal
RESULT	SINT	56	Decimal



VALUE	DINT	400	Decimal
RESULT	SINT	-112	Decimal



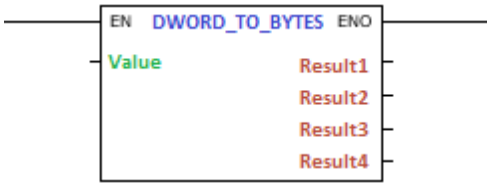
VALUE	DINT	-521	Decimal
RESULT	SINT	-9	Decimal

The examples above perform the conversion of variable VALUE, in DWORD, into a BYTE value storing the final result in RESULT. The block ends with success and ENO output is activated. Notice that only the eight least significant bits are taken into account.

11.8.5.6.3.3 DWORD_TO_BYTES

Block that performs the conversion of a 32 bits (DWORD) value into four 8 bits (4 BYTES) value.

Ladder Representation



Block Structure

Variable Type	Name	Data type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	DWORD UDINT DINT	Value in DWORD
VAR_OUTPUT	ENO	BOOL	End of operation
	Result1	BYTE USINT SINT	Value in BYTE (LSB)
	Result2	BYTE USINT SINT	Value in BYTE
	Result3	BYTE USINT SINT	Value in BYTE
	Result4	BYTE USINT SINT	Value in BYTE (MSB)

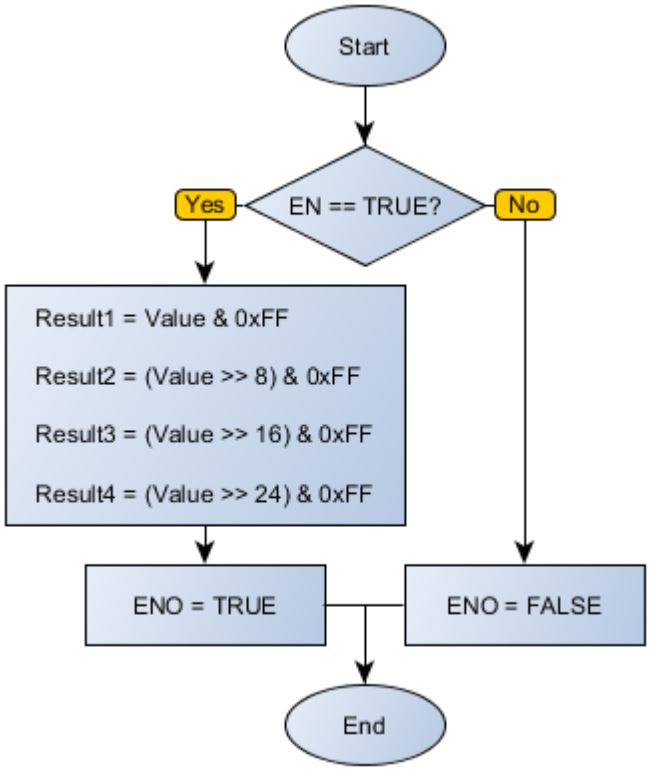
Operation

When this block has a TRUE value in EN, it interprets the Value value as DWORD and converts it into four BYTE values (from Result1 to Result4, where Result 1 is LSB), storing in Result.

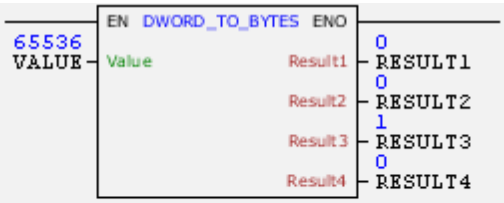
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

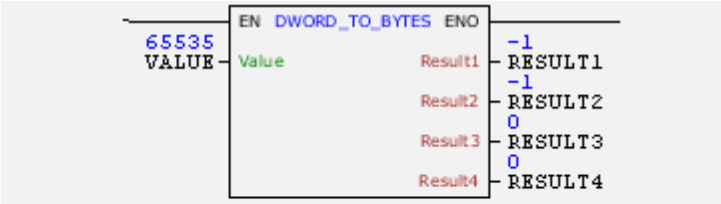
Block Flowchart



Example



RESULT1	SINT	0	0	Decimal
RESULT2	SINT	0	0	Decimal
RESULT3	SINT	0	1	Decimal
RESULT4	SINT	0	0	Decimal
VALUE	DINT	0	65536	Decimal



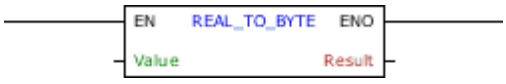
RESULT1	SINT	0	-1	Decimal
RESULT2	SINT	0	-1	Decimal
RESULT3	SINT	0	0	Decimal
RESULT4	SINT	0	0	Decimal
VALUE	DINT	0	65535	Decimal

The examples above perform the conversion of variable VALUE, in DWORD, into four BYTE value storing the final result in RESULT1, RESULT2, RESULT3 and RESULT4. The block ends with success and ENO output is activated.

11.8.5.6.3.4 REAL_TO_BYTE

Block that performs the conversion of a REAL value into a BYTE value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Value in REAL
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT	Value in BYTE

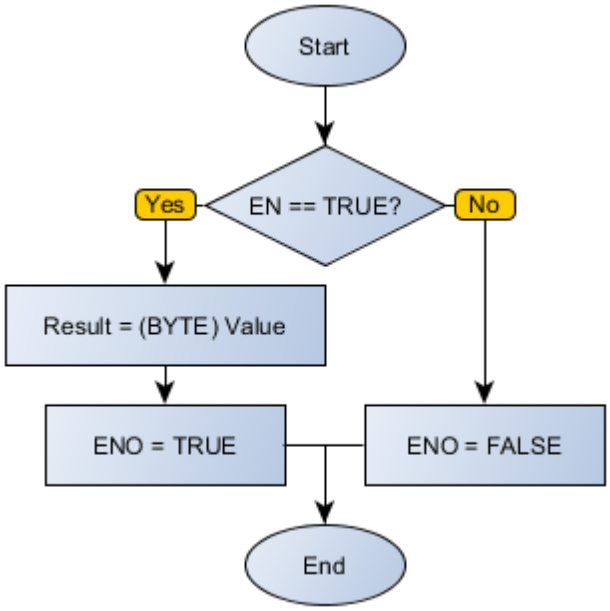
Operation

When this block has a TRUE value in EN, it interprets the Value value as REAL and converts it into BYTE, storing in Result.

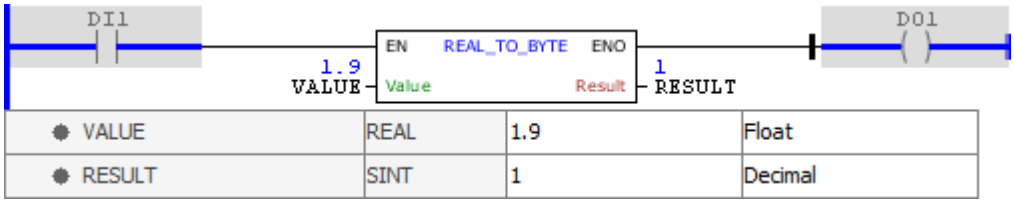
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

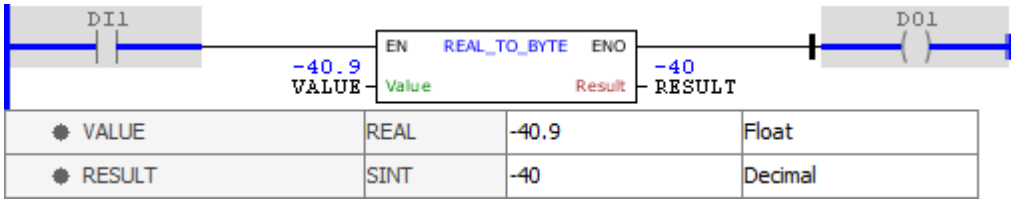
Block Flowchart



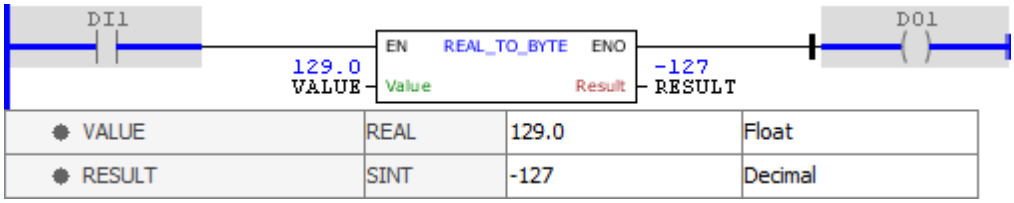
Example



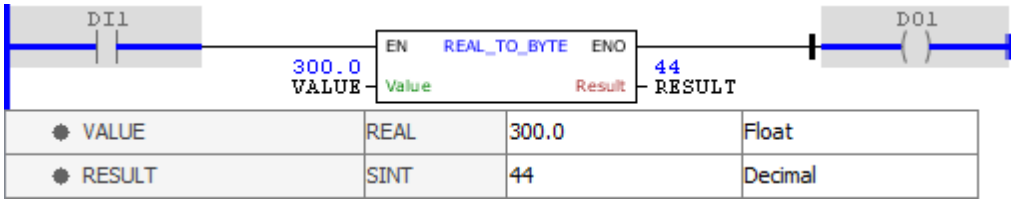
VALUE	REAL	1.9	Float
RESULT	SINT	1	Decimal



VALUE	REAL	-40.9	Float
RESULT	SINT	-40	Decimal



VALUE	REAL	129.0	Float
RESULT	SINT	-127	Decimal



VALUE	REAL	300.0	Float
RESULT	SINT	44	Decimal

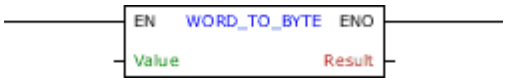
The examples above perform the conversion of variable VALUE, in REAL, into a BYTE value storing

the final result in RESULT. The block ends with success and ENO output is activated. Notice that the results are truncated in decimal and only the eight least significant bits are taken into account.

11.8.5.6.3.5 WORD_TO_BYTE

Block that performs the conversion of a WORD value into a BYTE value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	WORD UINT INT	Value in WORD
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT	Value in BYTE

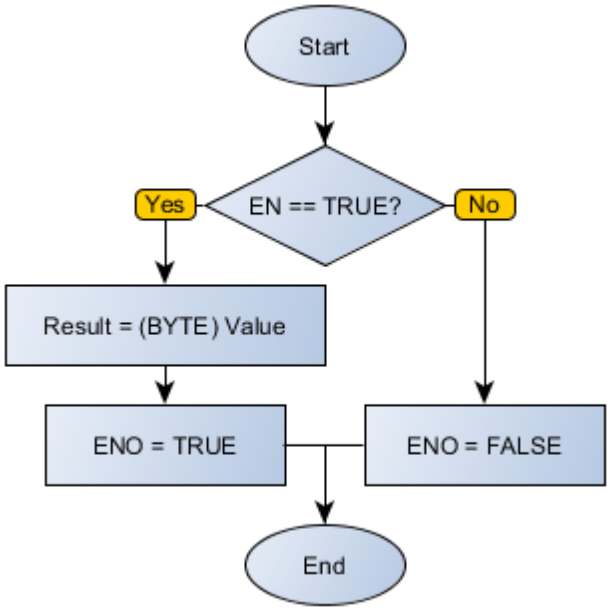
Operation

When this block has a TRUE value in EN, it interprets the Value value as WORD and converts it into BYTE, storing in Result.

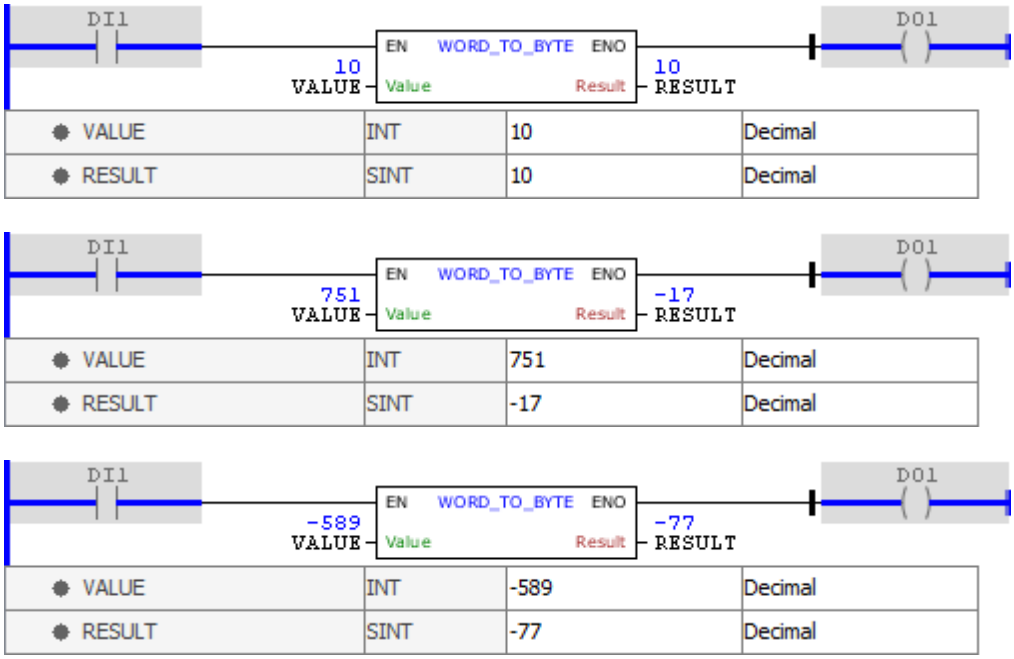
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



The examples above perform the conversion of variable VALUE, in WORD, into a BYTE value storing the final result in RESULT. The block ends with success and ENO output is activated. Notice that only the eight least significant bits are taken into account.

11.8.5.6.3.6 WORD_TO_BYTES

Block that performs the conversion of a 16 bits (WORD) value in two 8 bits (2 BYTES) value.

Ladder Representation

Block Structure

Variable Type	Name	Data type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	WORD UINT INT	Value in WORD
VAR_OUTPUT	ENO	BOOL	End of operation
	Result1	BYTE USINT SINT	Value in BYTE (LSB)
	Result2	BYTE USINT SINT	Value in BYTE (MSB)

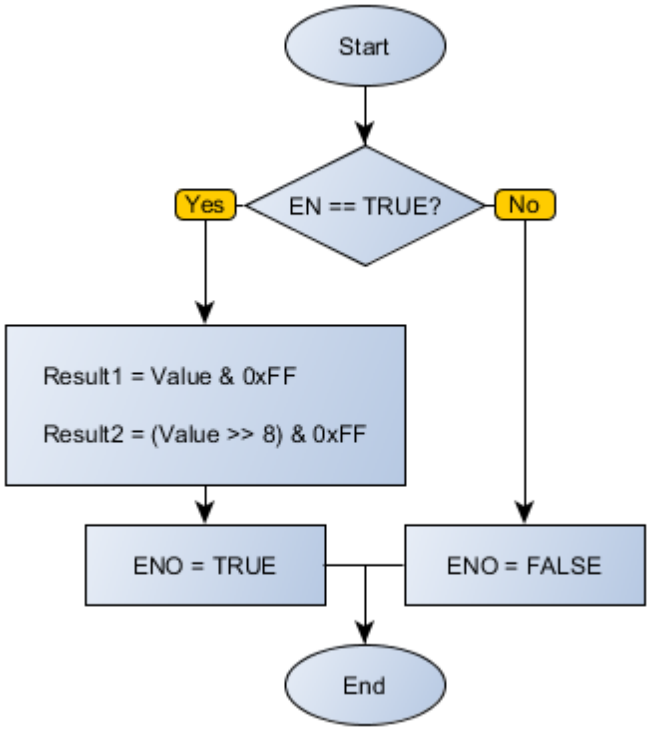
Operation

When this block has a TRUE value in EN, it interprets the Value value as WORD and converts it in two BYTE variables, storing in Result.

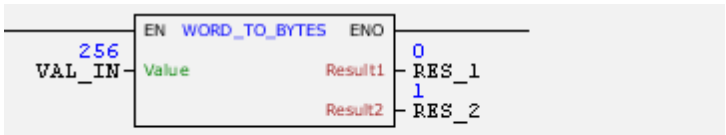
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

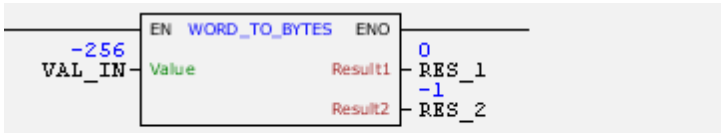
Block Flowchart



Example



RES_1	SINT	0	0	Decimal
RES_2	SINT	0	1	Decimal
VAL_IN	INT	0	256	Decimal



RES_1	SINT	0	0	Decimal
RES_2	SINT	0	-1	Decimal
VAL_IN	INT	0	-256	Decimal

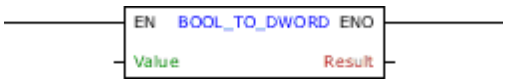
The examples above perform the conversion of variable VALUE "VAL_IN", in WORD, in two BYTE values storing the final result in RESULT1 and RESULT2. The block ends with success and ENO output is activated.

11.8.5.6.4 DWORD

11.8.5.6.4.1 BOOL_TO_DWORD

Block that performs the conversion of a BOOL value into a DWORD value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BOOL	Value in BOOL
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	DWORD UDINT DINT	Value in DWORD

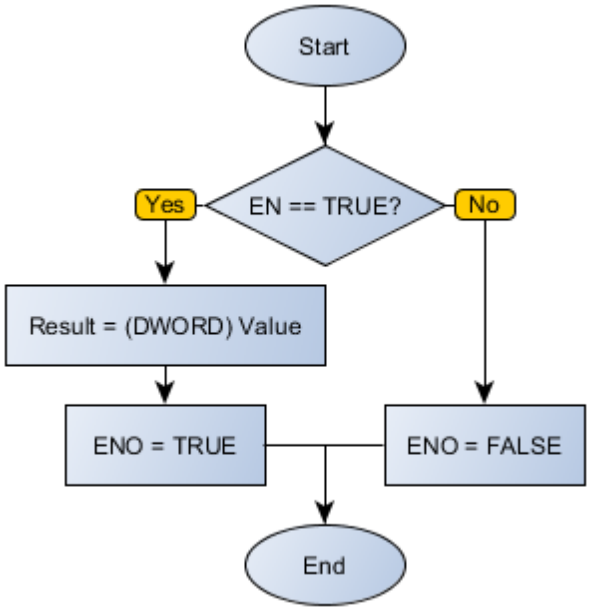
Operation

When this block has a TRUE value in EN, it interprets the Value value as BOOL and converts it into DWORD, storing in Result.

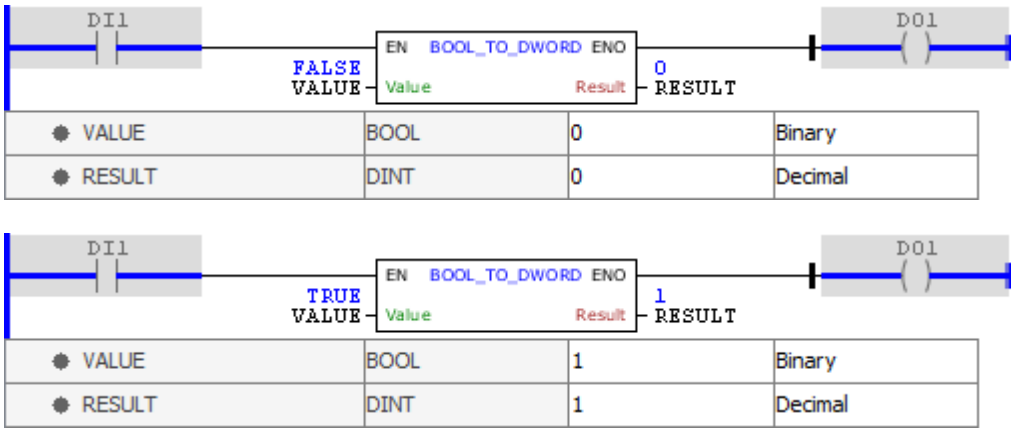
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



VALUE	BOOL	0	Binary
RESULT	DINT	0	Decimal

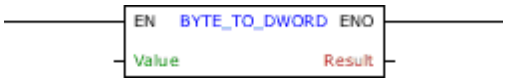
VALUE	BOOL	1	Binary
RESULT	DINT	1	Decimal

The examples above perform the conversion of VALUE variable, in BOOL, into a DWORD value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.8.5.6.4.2 BYTE_TO_DWORD

Block that performs the conversion of a BYTE value into a DWORD value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT	Value in BYTE
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	DWORD UDINT DINT	Value in DWORD

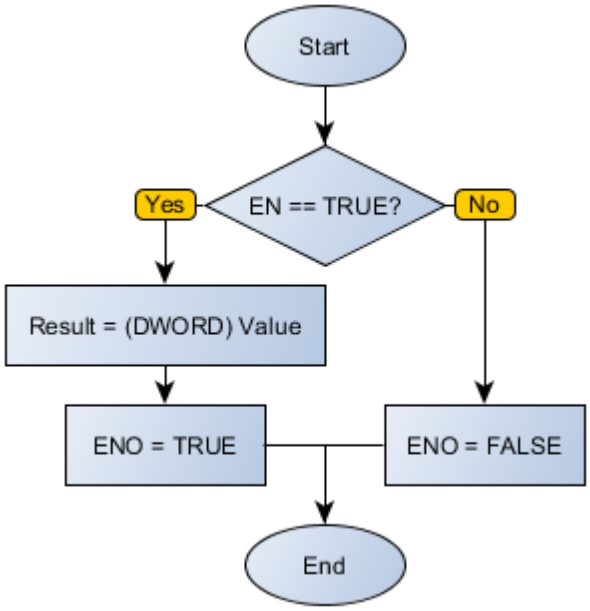
Operation

When this block has a TRUE value in EN, it interprets the Value value as BYTE and converts it into DWORD, storing in Result.

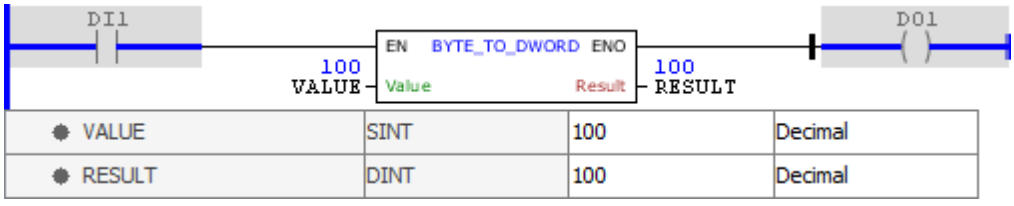
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

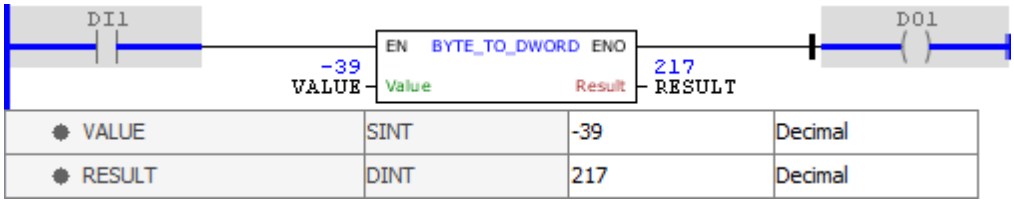
Block Flowchart



Example



VALUE	SINT	100	Decimal
RESULT	DINT	100	Decimal



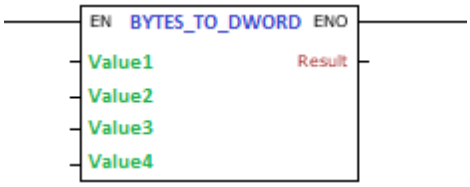
VALUE	SINT	-39	Decimal
RESULT	DINT	217	Decimal

The examples above perform the conversion of variable VALUE, in BYTE, into a DWORD value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.8.5.6.4.3 BYTES_TO_DWORD

Block that performs the conversion of four 8 bits (BYTE) values into a 32 bits (DWORD) value.

Ladder Representation



Block Structure

Variable Type	Name	Data type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT	Value in BYTE (1st byte - LSB)
	Value2	BYTE USINT SINT	Value in BYTE
	Value3	BYTE USINT SINT	Value in BYTE
	Value4	BYTE USINT SINT	Value in BYTE (4th byte - MSB)
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	DWORD UDINT DINT	Value in DWORD

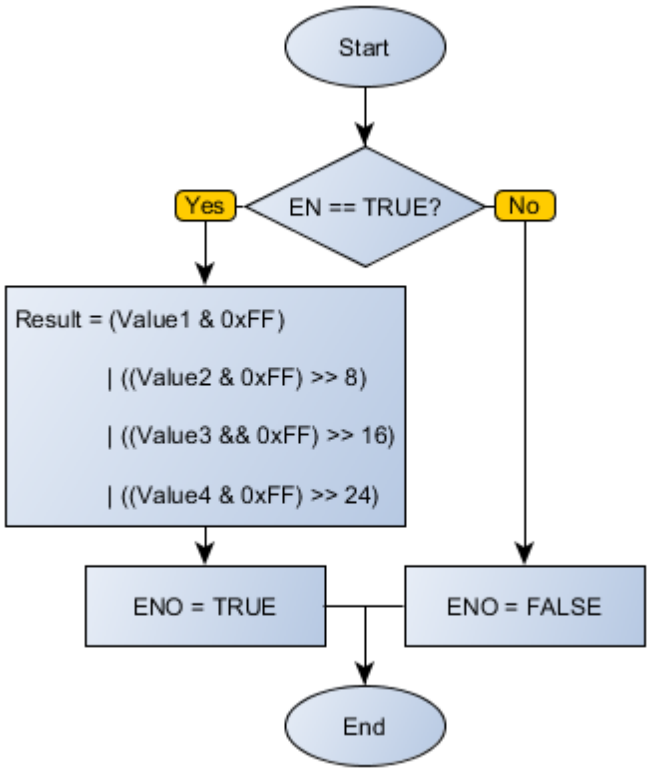
Operation

When this block has a TRUE value in EN, it interprets the Value1, Value2, Value3 and Value4 values as BYTE and converts it into a DWORD variable, storing in Result.

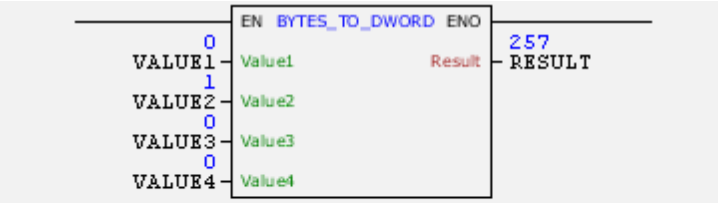
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



VALUE1	SINT	0	0	Decimal
VALUE2	SINT	0	1	Decimal
VALUE3	SINT	0	0	Decimal
VALUE4	SINT	0	0	Decimal
RESULT	DINT	0	257	Decimal

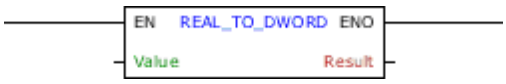
VALUE1	SINT	0	-1	Decimal
VALUE2	SINT	0	0	Decimal
VALUE3	SINT	0	0	Decimal
VALUE4	SINT	0	0	Decimal
RESULT	DINT	0	255	Decimal

The examples above perform the conversion of four variables VALUE1..4, in BYTE, into a DWORD value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.8.5.6.4.4 REAL_TO_DWORD

Block that performs the conversion of a REAL value into a DWORD value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Value in REAL
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	DWORD UDINT DINT	Value in DWORD

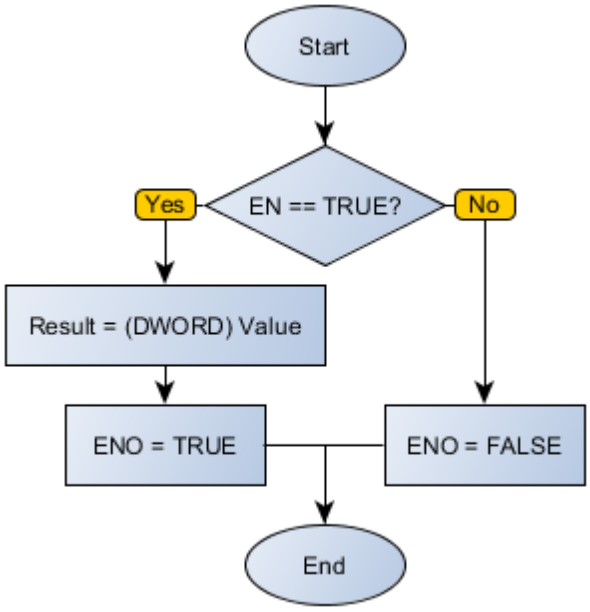
Operation

When this block has a TRUE value in EN, it interprets the Value value as REAL and converts it into DWORD, storing in Result.

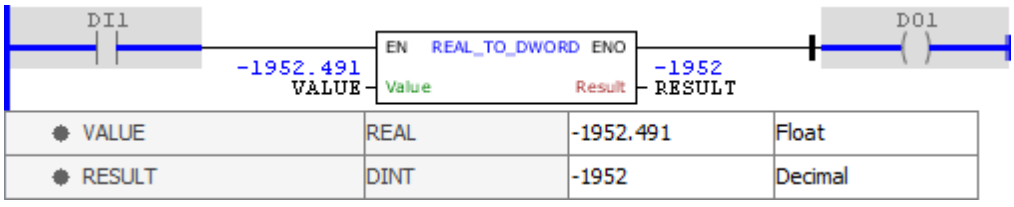
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

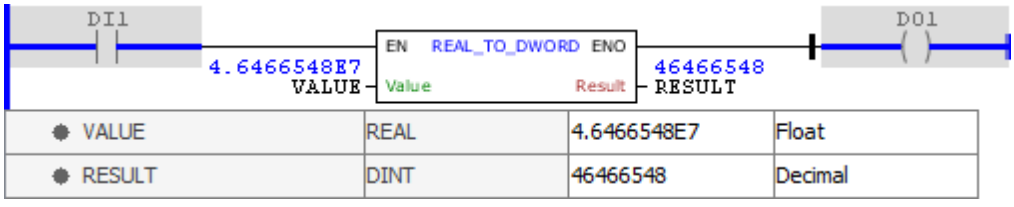
Block Flowchart



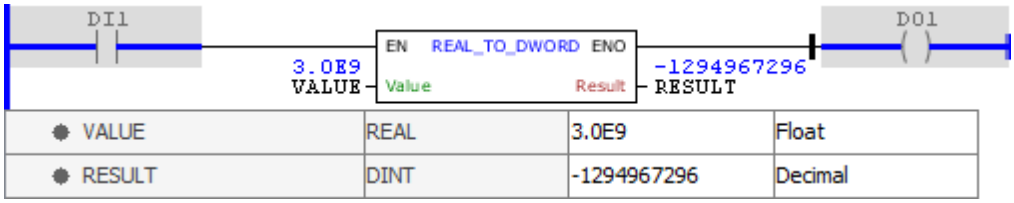
Example



VALUE	REAL	-1952.491	Float
RESULT	DINT	-1952	Decimal



VALUE	REAL	4.6466548E7	Float
RESULT	DINT	46466548	Decimal



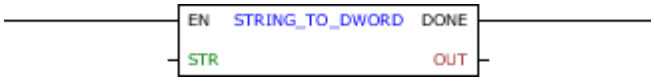
VALUE	REAL	3.0E9	Float
RESULT	DINT	-1294967296	Decimal

The examples above perform the conversion of variable VALUE, in REAL, into a DWORD value storing the final result in RESULT. The block ends with success and ENO output is activated. Note that the results are truncated in decimal and only the thirty-two least significant bits are taken into account.

11.8.5.6.4.5 STRING_TO_DWORD

Block that performs the conversion of a STRING value into a DWORD value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	STR	STRING	Value in STRING
VAR_OUTPUT	DONE	BOOL	End of operation
	OUT	DWORD UDINT DINT	Value in DWORD

Operation

When this block has a TRUE value in EN, it reads the value of STR as STRING character by character, performing the conversion to REAL and storing in OUT. If the first character is not mathematically valid, the OUT output receives zero. If there are valid characters, these characters will be converted to the end of STRING, or until it finds an invalid character.

NOTE!
If the number represented in the string is higher than the maximum supported by a DWORD, the Result value saturates in this maximum value.

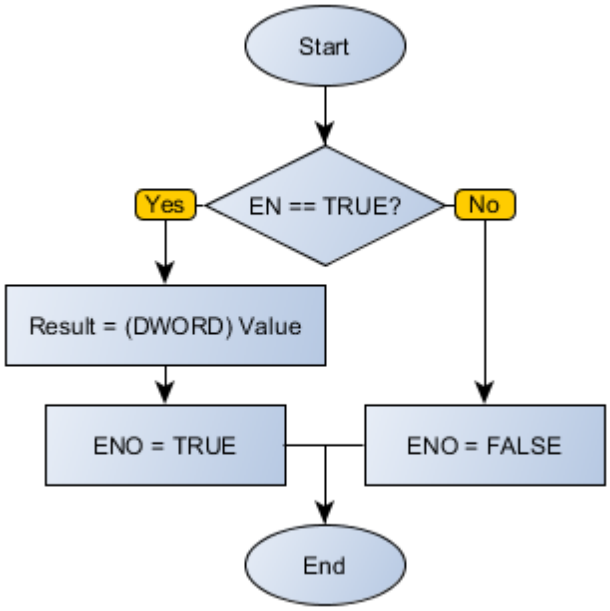
When EN has FALSE value, OUT remains unchanged and DONE remains FALSE.

The DONE value forwards to the next Ladder block the EN value after the operation is completed.

Compatibility

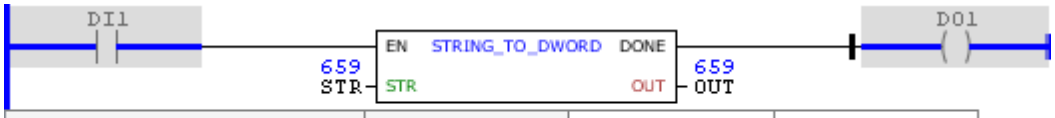
Device	Version
PLC300	2.10 or higher

Block Flowchart



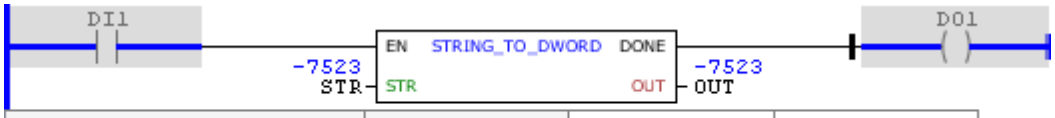
Example

The following examples show various conversions of STRING into values of DWORD type. All conversions enable the DONE output to the end of the operation.



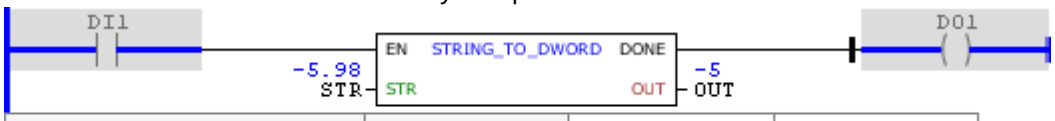
STR	STRING	659	String
OUT	DINT	659	Decimal

The conversion above was successfully completed.



STR	STRING	-7523	String
OUT	DINT	-7523	Decimal

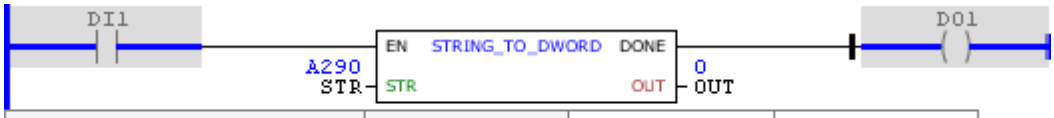
The conversion above was successfully completed. The underscore is a valid mathematical character.



STR	STRING	-5.98	String
OUT	DINT	-5	Decimal

The conversion above was successfully completed. The decimal point is not mathematically valid for

decimals and ends the conversion, truncating the result to what had already been converted.



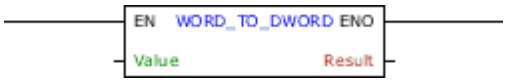
STR	STRING	A290	String
OUT	DINT	0	Decimal

The conversion above was not successfully completed. The first character was not identified as mathematically valid, and the output was zeroed.

11.8.5.6.4.6 WORD_TO_DWORD

Block that performs the conversion of a WORD value into a DWORD value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	WORD UINT INT	Value in WORD
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	DWORD UDINT DINT	Value in DWORD

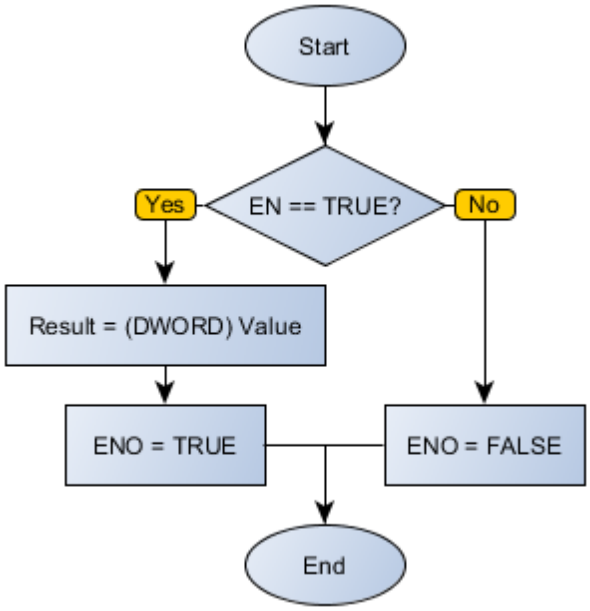
Operation

When this block has a TRUE value in EN, it interprets the Value value as WORD and converts it into DWORD, storing in Result.

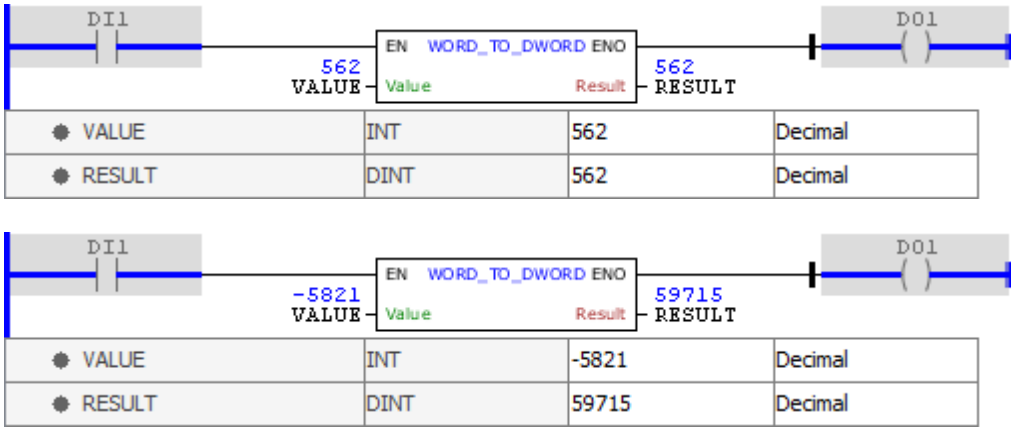
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example

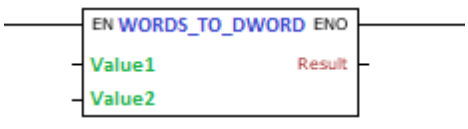


The examples above convert the VALUE variable, in WORD, into a DWORD value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.8.5.6.4.7 WORDS_TO_DWORD

Block that performs the conversion of two 16 bits (WORD) values into a 32 bits (DWORD) value.

Ladder Representation



Block Structure

Variable Type	Name	Data type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	WORD UINT INT	1st WORD (Less Significant Word)
	Value2	WORD UINT INT	2nd WORD (More Significant Word)
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	DWORD UDINT DINT	Value in DWORD

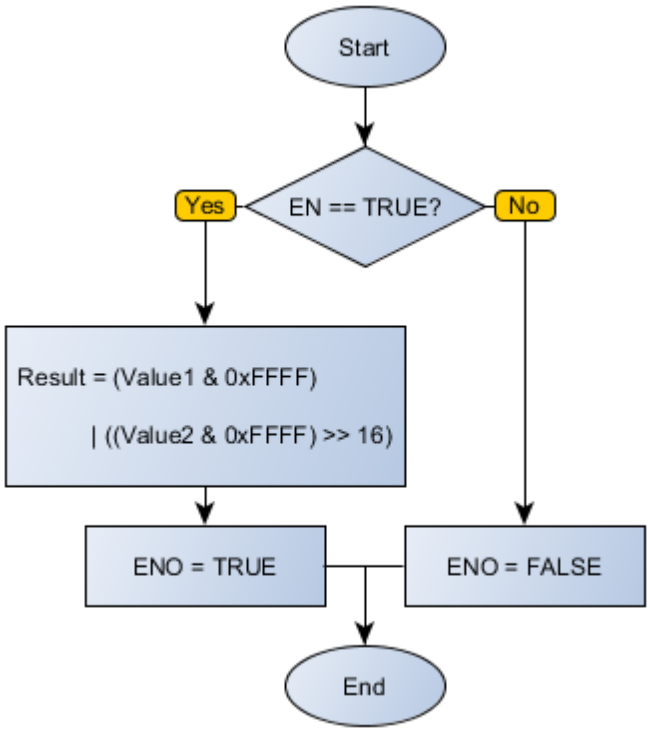
Operation

When this block has a TRUE value in EN, it interprets the Value1 and Value2 values as WORD and converts it into a DWORD variable, storing in Result.

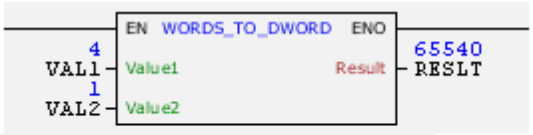
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

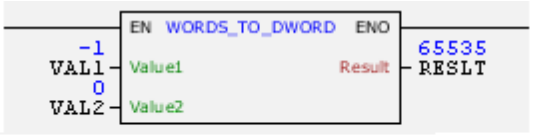
Block Flowchart



Example



RESLT	DINT	0	65540	Decimal
VAL1	INT	0	4	Decimal
VAL2	INT	0	1	Decimal



RESLT	DINT	0	65535	Decimal
VAL1	INT	0	-1	Decimal
VAL2	INT	0	0	Decimal

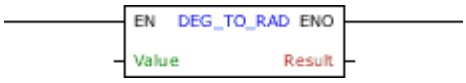
The examples above perform the conversion of two variable VALUE1 and VALUE2, in WORD, into a DWORD value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.8.5.6.5 Rad-Deg

11.8.5.6.5.1 DEG_TO_RAD

Block that performs the conversion of a value in degrees into a value in radians.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Value in degrees
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	REAL	Value in radians

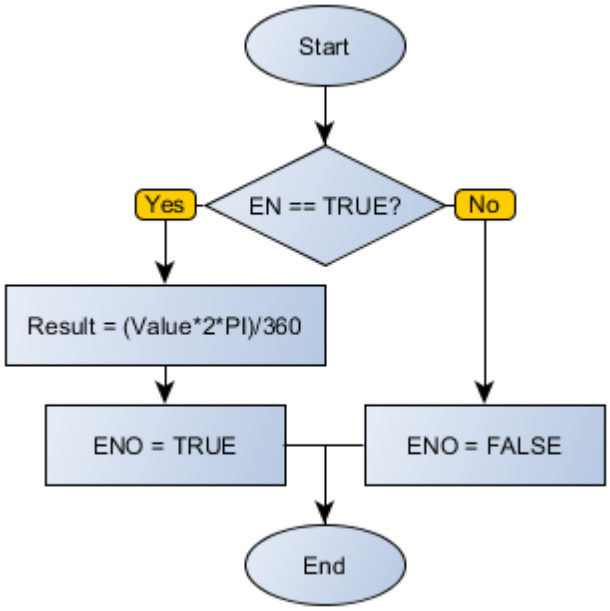
Operation

When this block has a TRUE value in EN, it interprets the Value value as in degrees and converts it into radians, storing in Result.

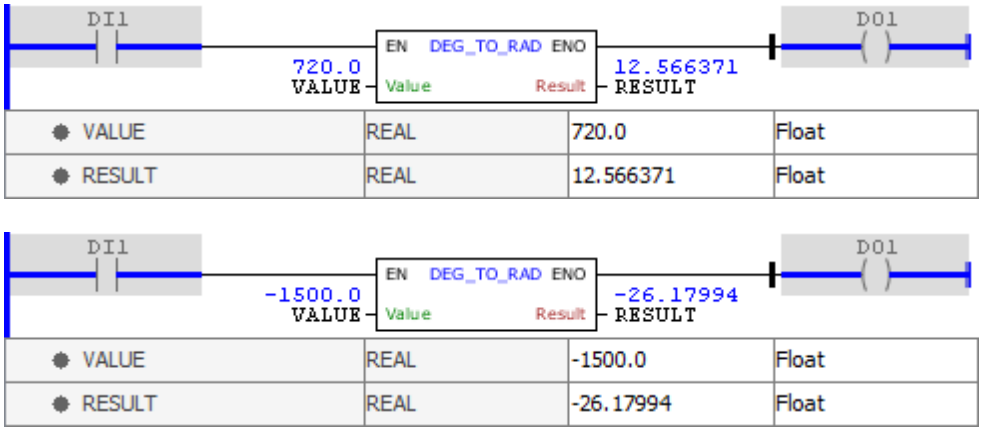
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



VALUE	REAL	720.0	Float
RESULT	REAL	12.566371	Float

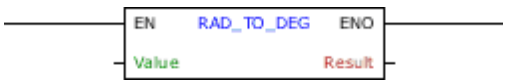
VALUE	REAL	-1500.0	Float
RESULT	REAL	-26.17994	Float

The examples above perform the conversion of variable VALUE, in degrees, into a corresponding value in radians storing the final result in RESULT. The block ends with success and ENO output is activated.

11.8.5.6.5.2 RAD_TO_DEG

Block that performs the conversion of a value in radians into a value in degrees.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Value in radianos
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	REAL	Value in graus

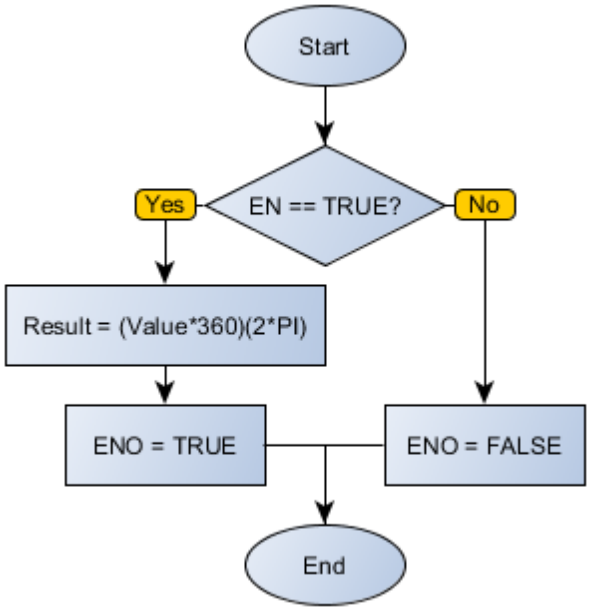
Operation

When this block has a TRUE value in EN, it interprets the Value value as in radians and converts it into degrees, storing in Result.

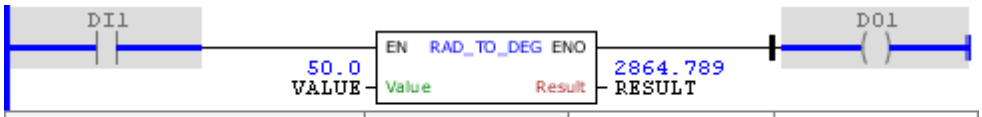
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

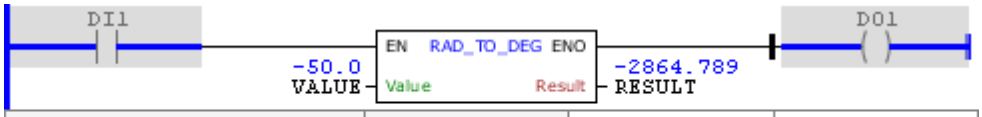
Block Flowchart



Example



VALUE	REAL	50.0	Float
RESULT	REAL	2864.789	Float



VALUE	REAL	-50.0	Float
RESULT	REAL	-2864.789	Float

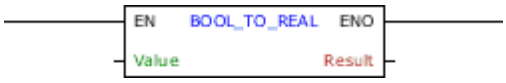
The examples above perform the conversion of variable VALUE, in radians, into a corresponding value in degrees storing the final result in RESULT. The block ends with success and ENO output is activated.

11.8.5.6.6 REAL

11.8.5.6.6.1 BOOL_TO_REAL

Block that performs the conversion of a BOOL value into a REAL value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BOOL	Value in BOOL
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	REAL	Value in REAL

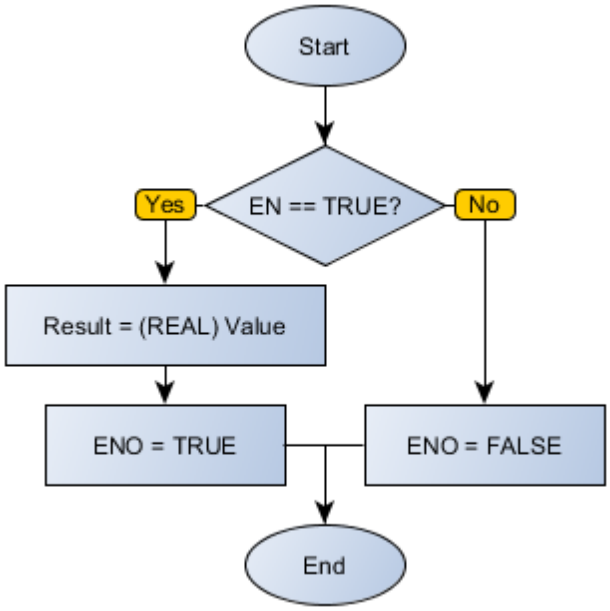
Operation

When this block has a TRUE value in EN, it interprets the Value value as BOOL and converts it into REAL, storing in Result.

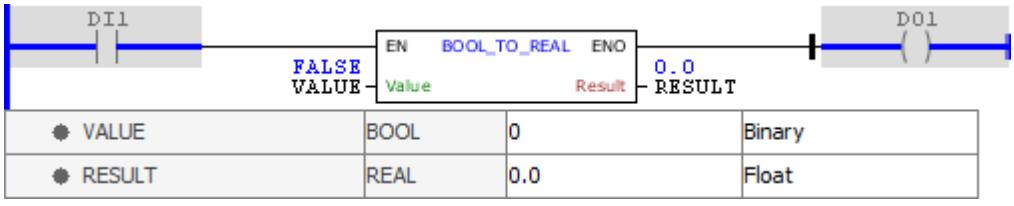
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

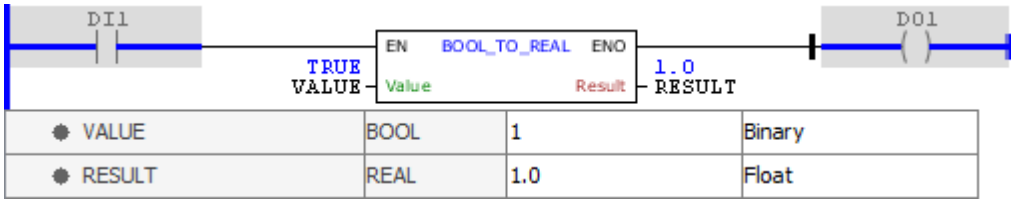
Block Flowchart



Example



VALUE	BOOL	0	Binary
RESULT	REAL	0.0	Float



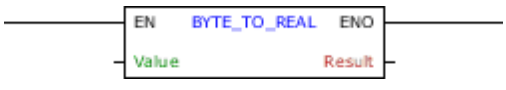
VALUE	BOOL	1	Binary
RESULT	REAL	1.0	Float

The examples above perform the conversion of variable VALUE, in BOOL, into a REAL value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.8.5.6.6.2 BYTE_TO_REAL

Block that performs the conversion of a BYTE value into a REAL value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT	Value in BYTE
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	REAL	Value in REAL

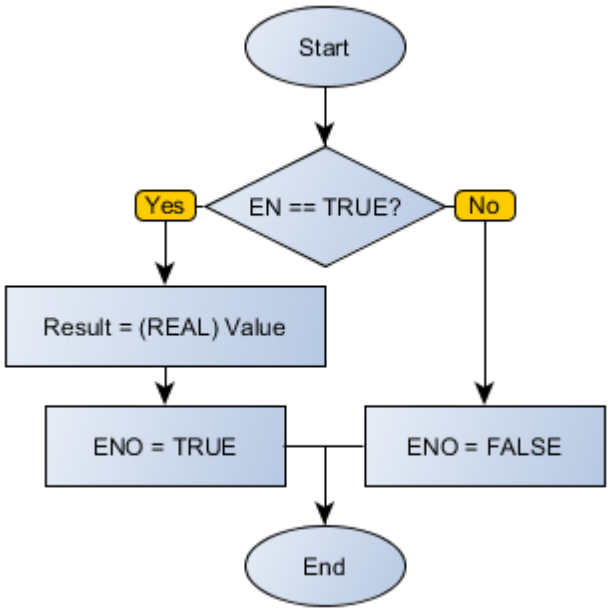
Operation

When this block has a TRUE value in EN, it interprets the Value value as BYTE and converts it into REAL, storing in Result.

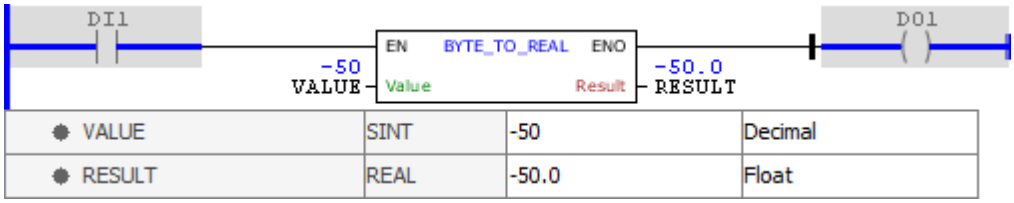
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

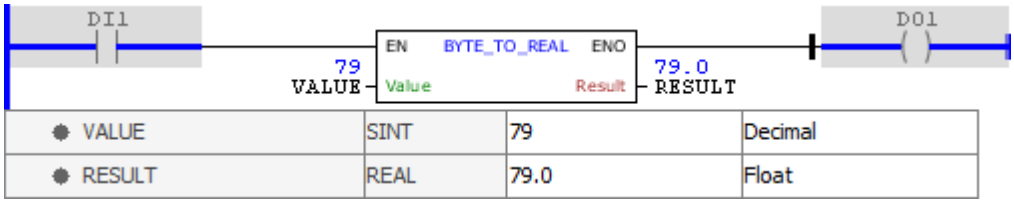
Block Flowchart



Example



VALUE	SINT	-50	Decimal
RESULT	REAL	-50.0	Float



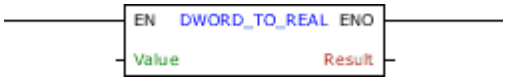
VALUE	SINT	79	Decimal
RESULT	REAL	79.0	Float

The examples above perform the conversion of variable VALUE, in BYTE, into a REAL value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.8.5.6.6.3 DWORD_TO_REAL

Block that performs the conversion of a DWORD value into a REAL value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	DWORD UDINT DINT	Value in DWORD
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	REAL	Value in REAL

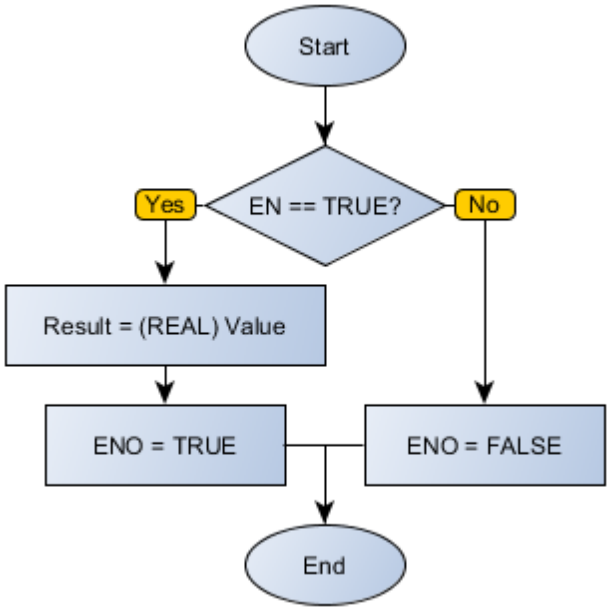
Operation

When this block has a TRUE value in EN, it interprets the Value value as DWORD and converts it into REAL, storing in Result.

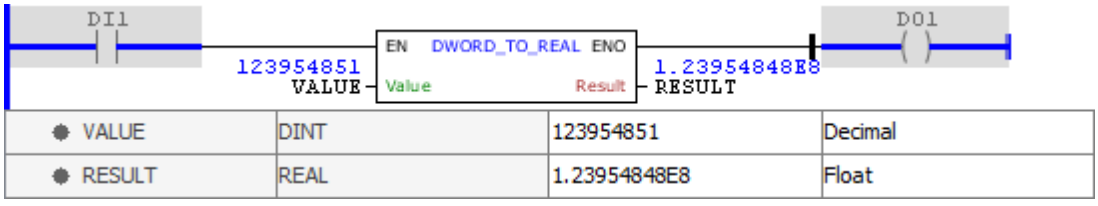
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

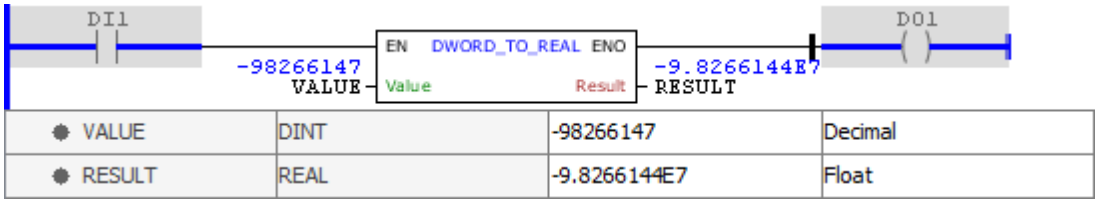
Block Flowchart



Example



VALUE	DINT	123954851	Decimal
RESULT	REAL	1.23954848E8	Float



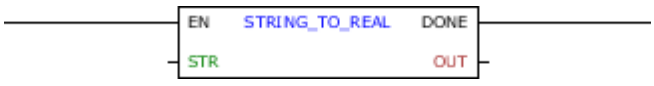
VALUE	DINT	-98266147	Decimal
RESULT	REAL	-9.8266144E7	Float

The examples above perform the conversion of variable VALUE, in DWORD, into a REAL value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.8.5.6.6.4 STRING_TO_REAL

Block that performs the conversion of a STRING value into a REAL value.

Ladder Representation



Block Structure

Tipo	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	STR	STRING	Value in STRING
VAR_OUTPUT	DONE	BOOL	End of operation
	OUT	REAL	Value in REAL

Operation

When this block has a TRUE value in EN, it reads the value of STR as STRING character by character, performing the conversion to REAL and storing in DONE. If the first character is not mathematically valid, the OUT output receives zero. If there are valid characters, these characters will be converted to the end of STRING, or until it finds an invalid character.

NOTE!
The block interprets the dot (.) as a decimal tab, and not the comma (,).

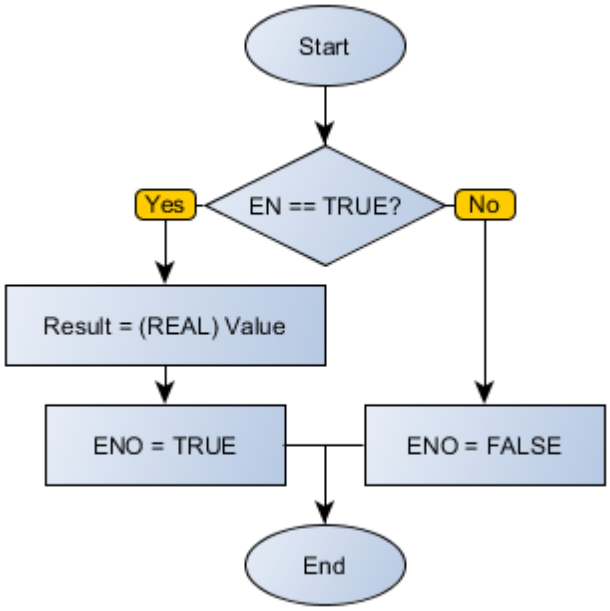
When EN has FALSE value, OUT remains unchanged and DONE remains FALSE.

The DONE value forwards to the next Ladder block the EN value after the operation is completed.

Compatibility

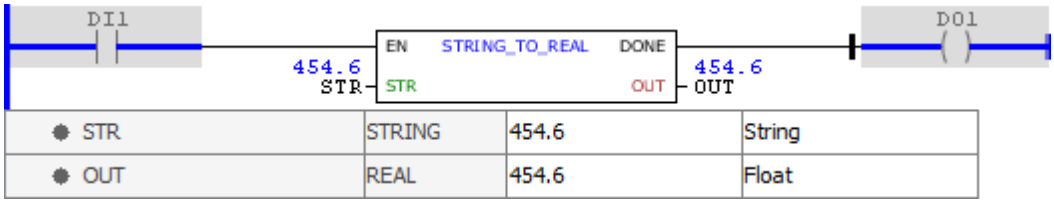
Device	Version
PLC300	2.10 or higher

Block Flowchart



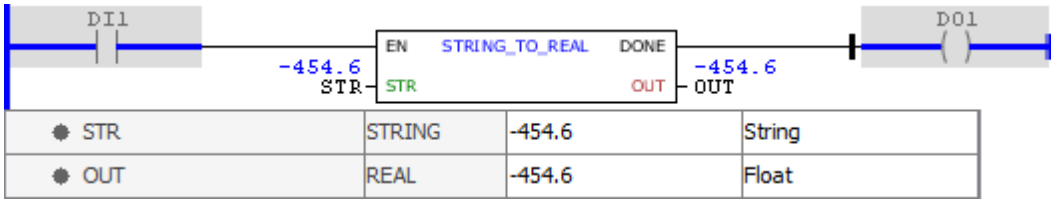
Example

The following examples show various conversions of STRING into values of REAL type. All conversions enable the DONE output to the end of the operation.



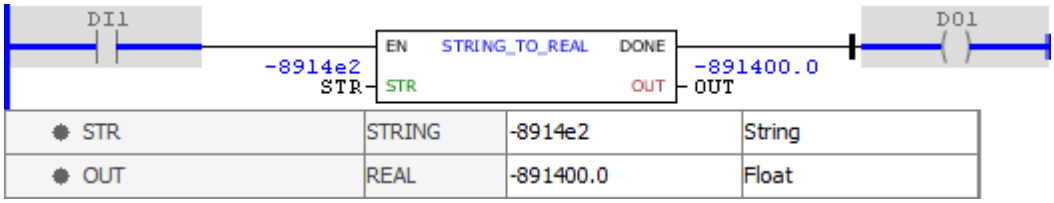
STR	STRING	454.6	String
OUT	REAL	454.6	Float

The conversion above was successfully completed. The decimal point is a valid mathematical character.



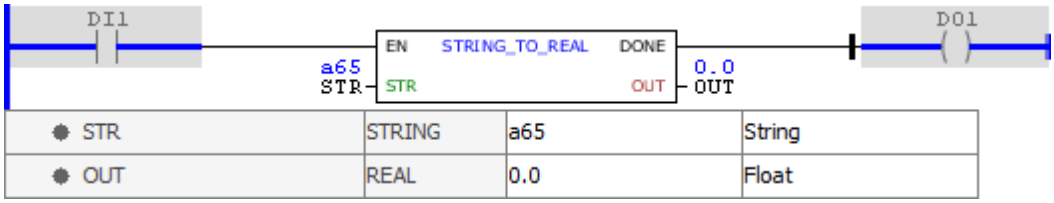
STR	STRING	-454.6	String
OUT	REAL	-454.6	Float

The conversion above was successfully completed. The dash and decimal point are valid mathematical characters.



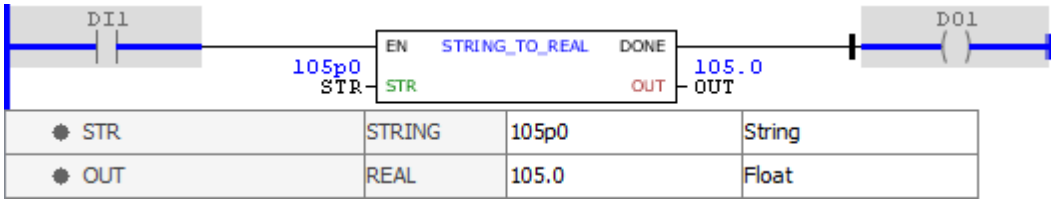
STR	STRING	-8914e2	String
OUT	REAL	-891400.0	Float

The conversion above was successfully completed. The dash and the power indicator of 10 "and" are valid mathematical characters.



STR	STRING	a65	String
OUT	REAL	0.0	Float

The conversion above was not successfully completed. The first character was not identified as mathematically valid, and the output was zeroed.



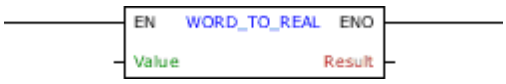
STR	STRING	105p0	String
OUT	REAL	105.0	Float

The conversion above was successfully completed. The "p" character is not mathematically valid and ends the conversion, truncating the result to what had already been converted.

11.8.5.6.6.5 WORD_TO_REAL

Block that performs the conversion of a WORD value into a REAL value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	WORD UINT INT	Value in WORD
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	REAL	Value in REAL

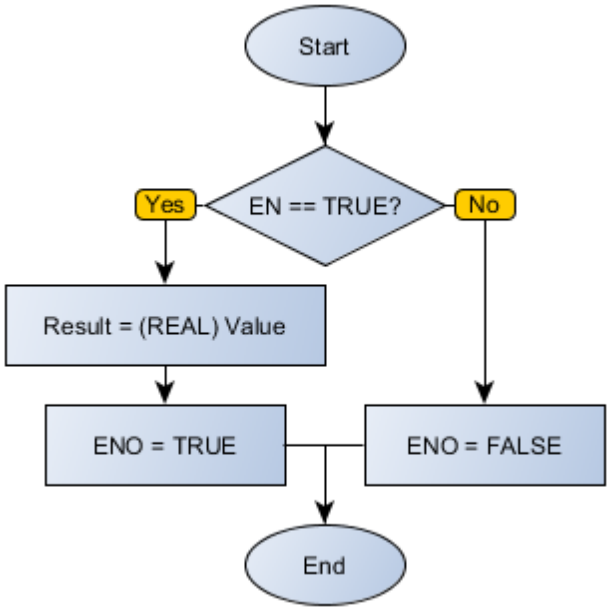
Operation

When this block has a TRUE value in EN, it interprets the Value value as WORD and converts it into REAL, storing in Result.

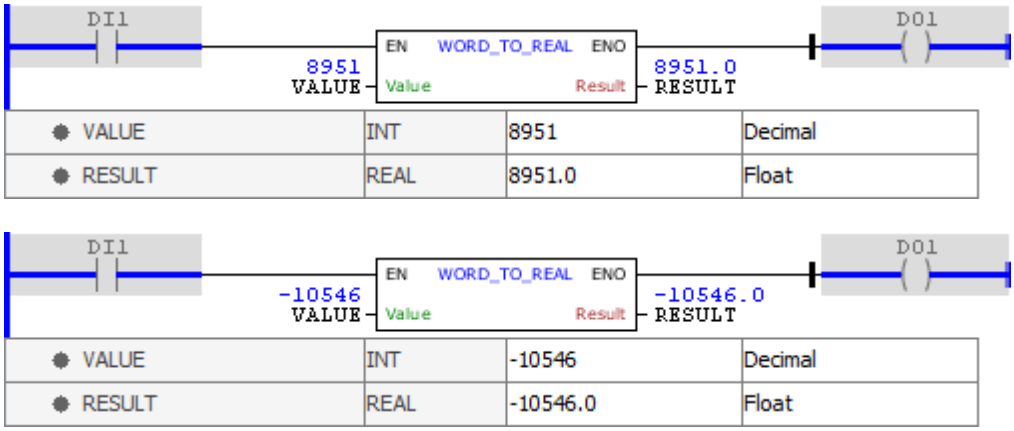
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



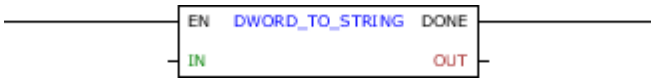
The examples above perform the conversion of variable VALUE, in WORD, into a REAL value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.8.5.6.7 STRING

11.8.5.6.7.1 DWORD_TO_STRING

Block that performs the conversion of a DWORD value into a STRING value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	IN	DWORD UDINT DINT	Value in DWORD
VAR_OUTPUT	DONE	BOOL	End of operation
	OUT	STRING	Value in STRING

Operation

When this block has a TRUE value in EN, it interprets the IN value as DWORD and converts it into STRING, storing in Result.

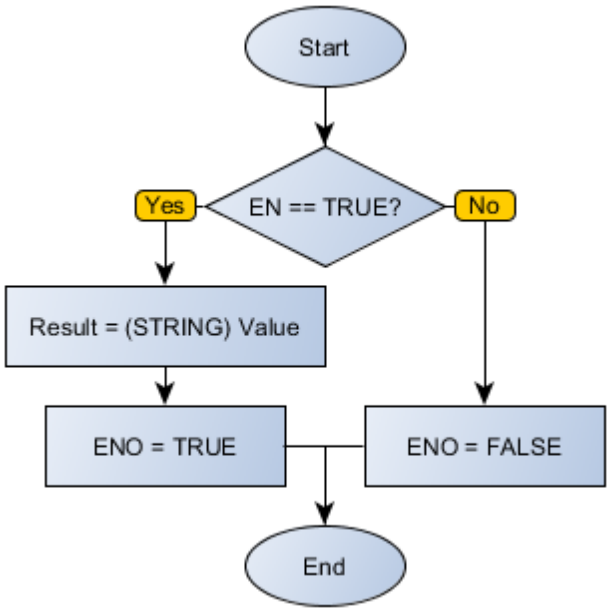
NOTE!
If the number represented has more digits than the capacity of the STRING, the value will be truncated.

When EN has FALSE value, OUT remains unchanged and DONE remains FALSE.

Compatibility

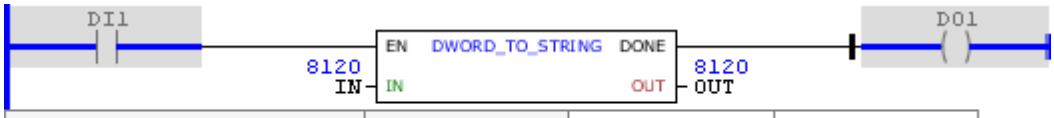
Device	Version
PLC300	2.10 or higher

Block Flowchart



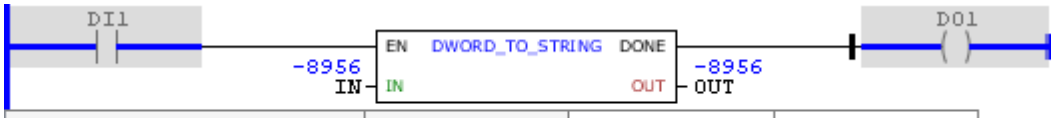
Example

The following examples show various conversions of DWORD type values into STRING. All conversions enable the DONE output to the end of the operation.



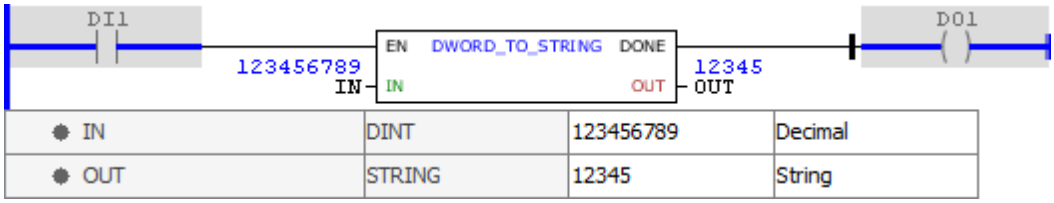
IN	DINT	8120	Decimal
OUT	STRING	8120	String

The conversion above was successfully completed.



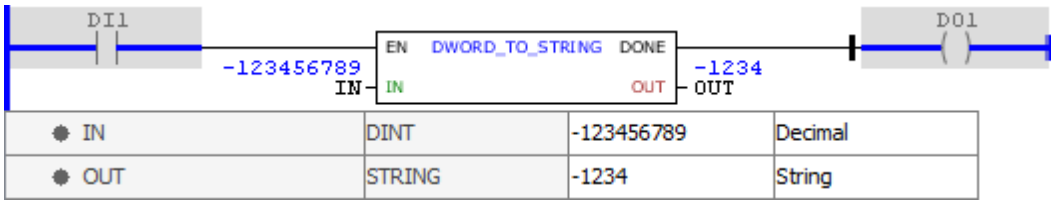
IN	DINT	-8956	Decimal
OUT	STRING	-8956	String

The conversion above was successfully completed.



IN	DINT	123456789	Decimal
OUT	STRING	12345	String

The above conversion was successful, but the size of the result of the conversion is greater than the OUT size, and this has been truncated.



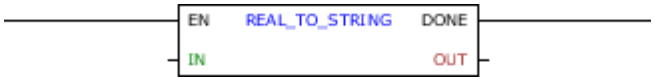
IN	DINT	-123456789	Decimal
OUT	STRING	-1234	String

The above conversion was successful, but the size of the result of the conversion is greater than the OUT size, and this has been truncated.

11.8.5.6.7.2 REAL_TO_STRING

Block that performs the conversion of a REAL value into a STRING value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	IN	REAL	Value in REAL
VAR_OUTPUT	DONE	BOOL	End of operation
	OUT	STRING	Value in STRING

Operation

When this block has a TRUE value in EN, it interprets the IN value as REAL and converts it into STRING, storing in OUT and sending TRUE to the DONE output.

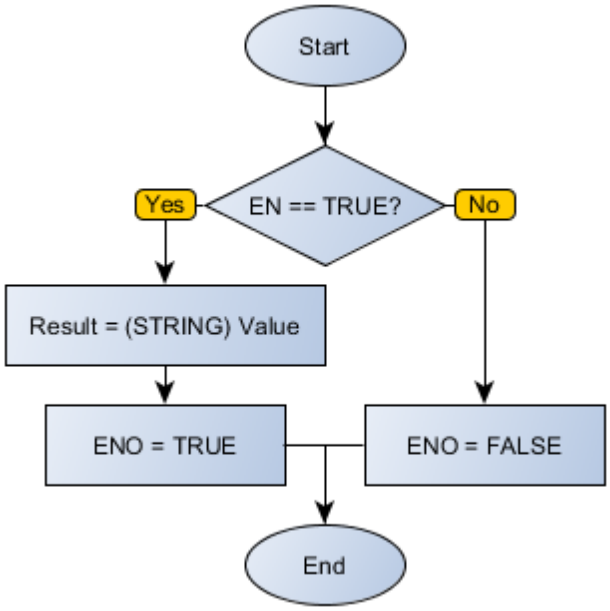
NOTE!
If the number represented has more digits than the capacity of the STRING, the value will be truncated.

When EN has FALSE value, OUT remains unchanged and DONE remains FALSE.

Compatibility

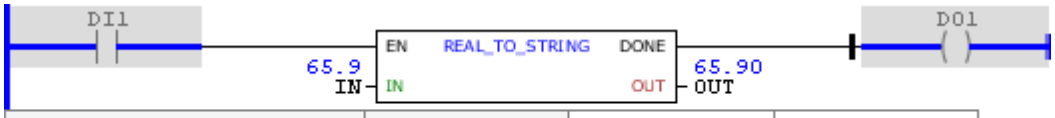
Device	Version
PLC300	2.10 or higher

Block Flowchart



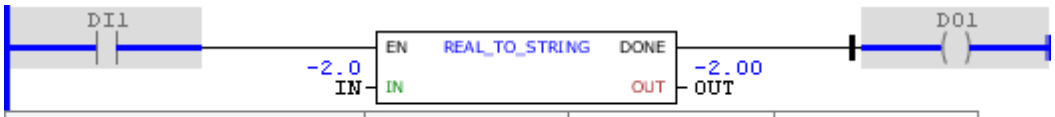
Example

The following examples show various conversions of REAL type values into STRING. All conversions enable the DONE output to the end of the operation.



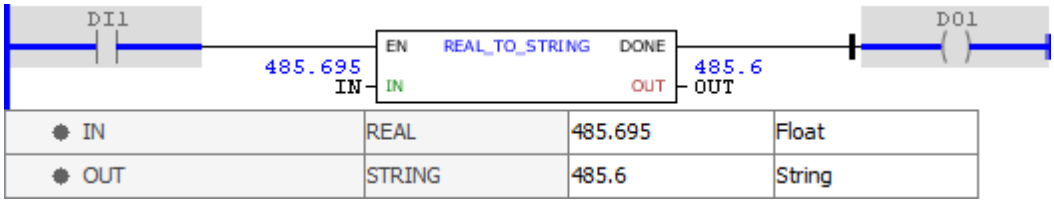
IN	REAL	65.9	Float
OUT	STRING	65.90	String

The conversion above was successfully completed.



IN	REAL	-2.0	Float
OUT	STRING	-2.00	String

The conversion above was successfully completed.



IN	REAL	485.695	Float
OUT	STRING	485.6	String

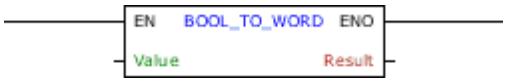
The above conversion was successful, but the size of the result of the conversion is greater than the OUT size, and this has been truncated.

11.8.5.6.8 WORD

11.8.5.6.8.1 BOOL_TO_WORD

Block that performs the conversion of a BOOL value into a WORD value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BOOL	Value in BOOL
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	WORD UINT INT	Value in WORD

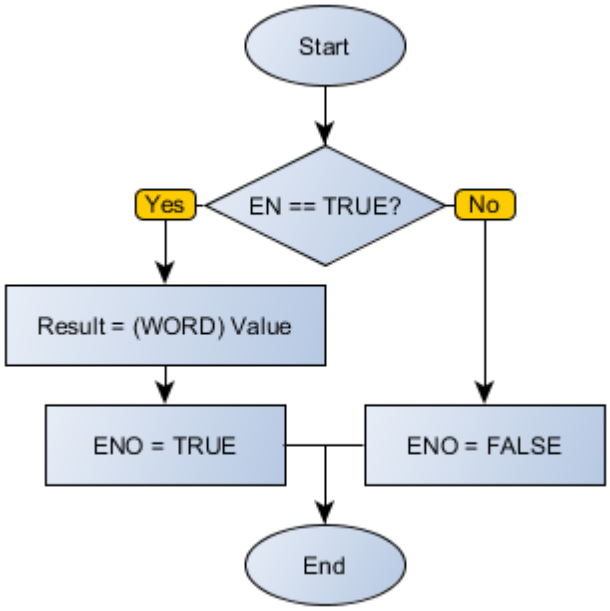
Operation

When this block has a TRUE value in EN, it interprets the Value value as BOOL and converts it into WORD, storing in Result.

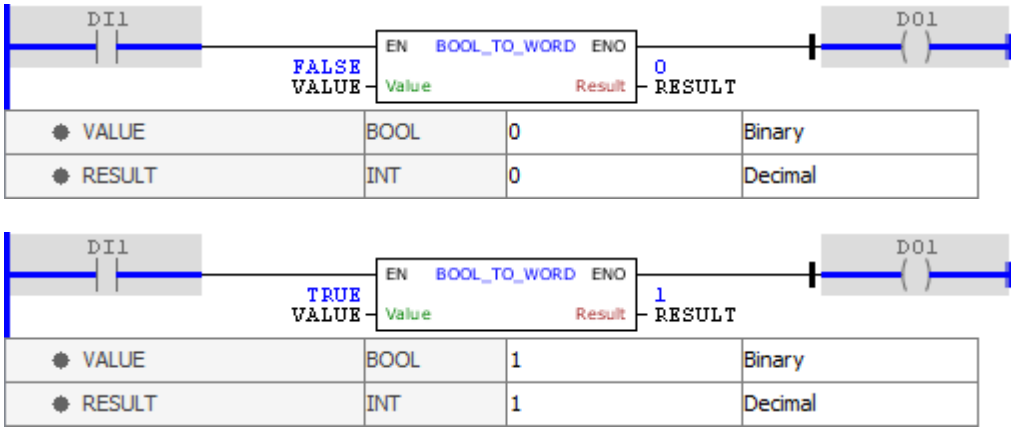
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example

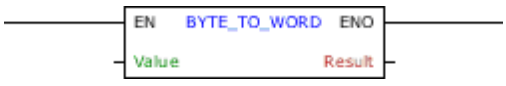


The examples above perform the conversion of VALUE variable, in BOOL, into a WORD value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.8.5.6.8.2 BYTE_TO_WORD

Block that performs the conversion of a BYTE value into a WORD value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT	Value in BYTE
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	WORD UINT INT	Value in WORD

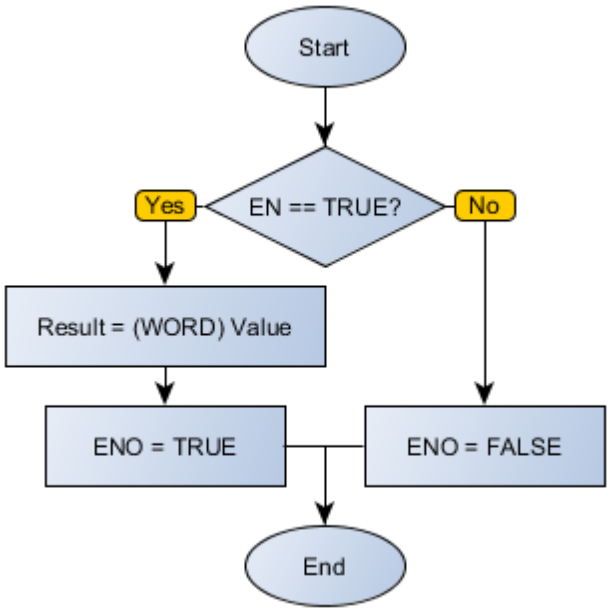
Operation

When this block has a TRUE value in EN, it interprets the Value value as BYTE and converts it into WORD, storing in Result.

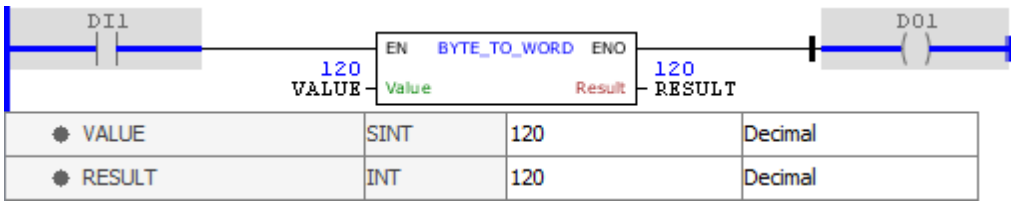
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

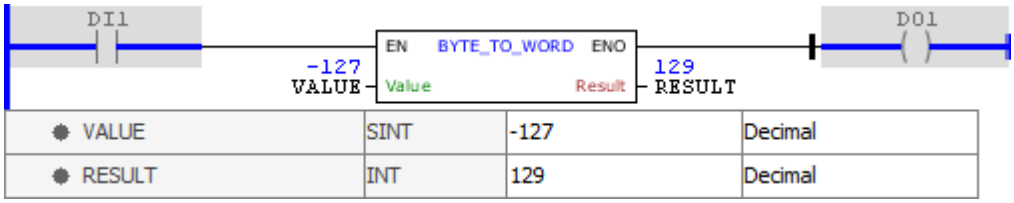
Block Flowchart



Example



VALUE	SINT	120	Decimal
RESULT	INT	120	Decimal



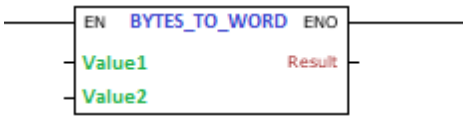
VALUE	SINT	-127	Decimal
RESULT	INT	129	Decimal

The examples above perform the conversion of variable VALUE, in BYTE, into a WORD value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.8.5.6.8.3 BYTES_TO_WORD

Block that performs the conversion of two 8 bits (BYTE) values into a 16 bits (WORD) value.

Ladder Representation



Block Structure

Variable Type	Name	Data type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT	1st BYTE (LSB)
	Value2	BYTE USINT SINT	2nd BYTE (MSB)
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	WORD UINT INT	Value in WORD

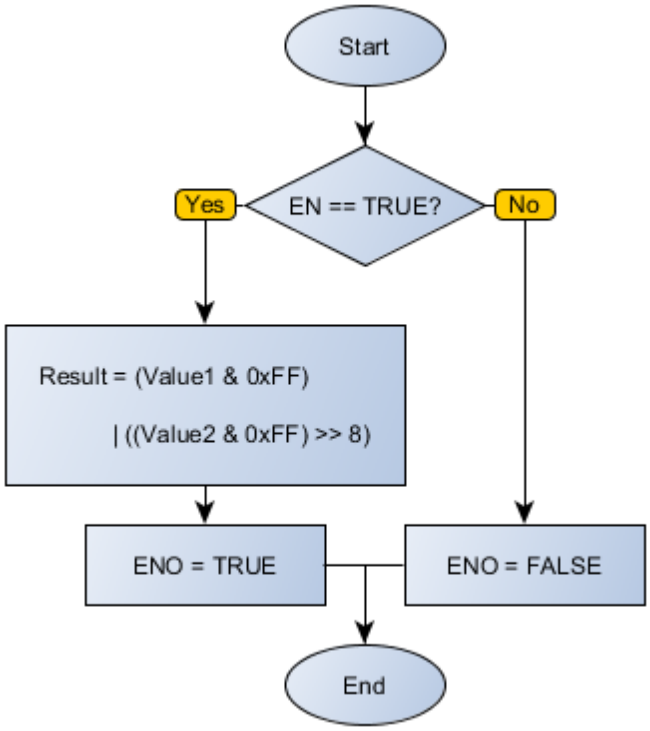
Operation

When this block has a TRUE value in EN, it interprets the Value1 and Value2 values as BYTE and converts it into a WORD variable, storing in Result.

When EN has FALSE value, Result remains unchanged.

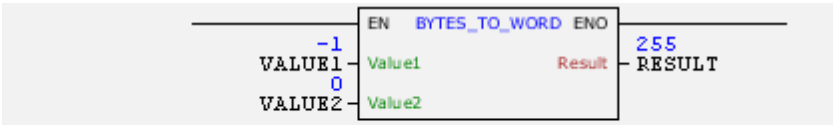
The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example

VALUE1	SINT	0	5	Decimal
VALUE2	SINT	0	10	Decimal



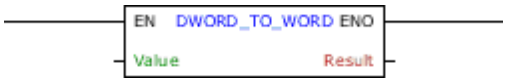
VALUE1	SINT	0	-1	Decimal
VALUE2	SINT	0	0	Decimal

The examples above perform the conversion of two variable VALUE1 and VALUE2, in BYTE, into a WORD value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.8.5.6.8.4 DWORD_TO_WORD

Block that performs the conversion of a DWORD value into a WORD value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	DWORD UDINT DINT	Value in DWORD
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	WORD UINT INT	Value in WORD

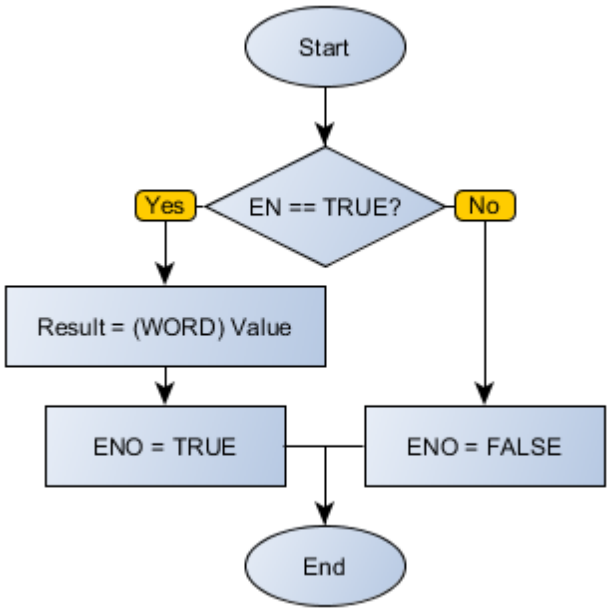
Operation

When this block has a TRUE value in EN, it interprets the Value value as DWORD and converts it into WORD, storing in Result.

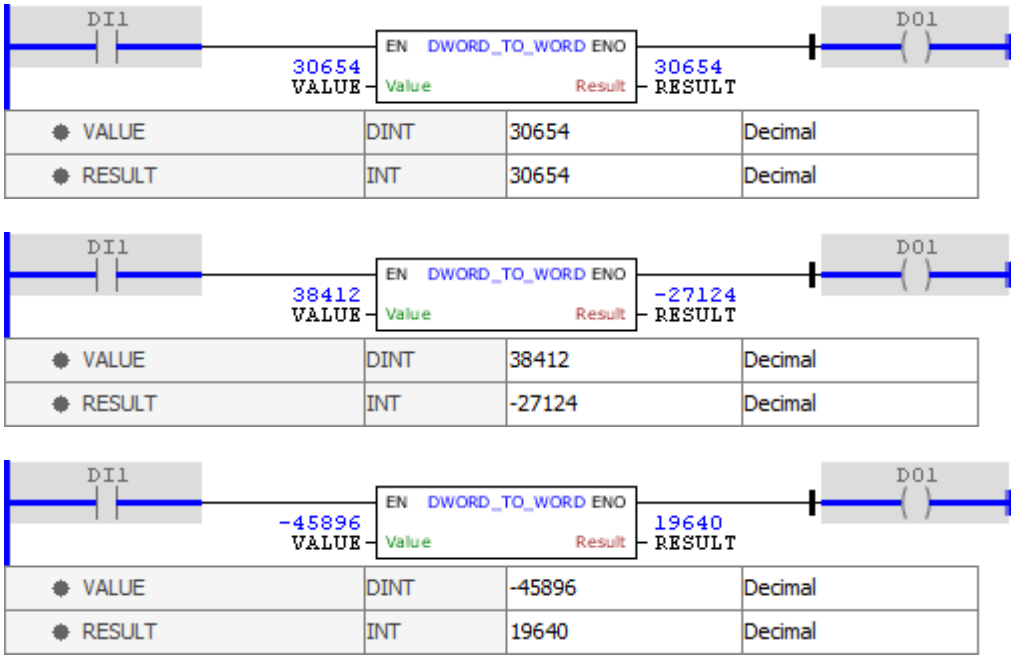
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example

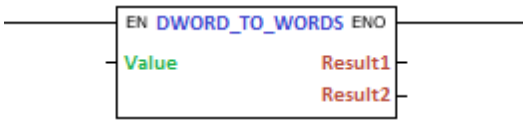


The examples above convert the VALUE variable, in DWORD, into a WORD value storing the final result in RESULT. The block ends with success and ENO output is activated. Notice that only the sixteen least significant bits are taken into account.

11.8.5.6.8.5 DWORD_TO_WORDS

Block that performs the conversion of a 32 bits (DWORD) value in two 16 bits (2 WORD) value.

Ladder Representation



Block Structure

Variable Type	Name	Data type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	DWORD UDINT DINT	Value in DWORD
VAR_OUTPUT	ENO	BOOL	End of operation
	Result1	WORD UINT INT	Value in WORD (Less Significant Word)
	Result2	WORD UINT INT	Value in WORD (More Significant Word)

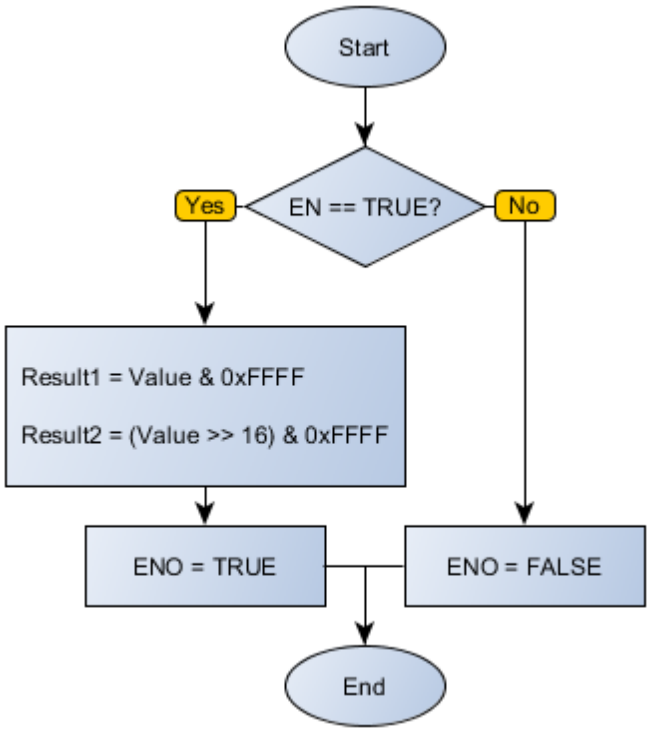
Operation

When this block has a TRUE value in EN, it interprets the value as DWORD and converts it in two WORD variables (Result1 and Result2), storing in Result.

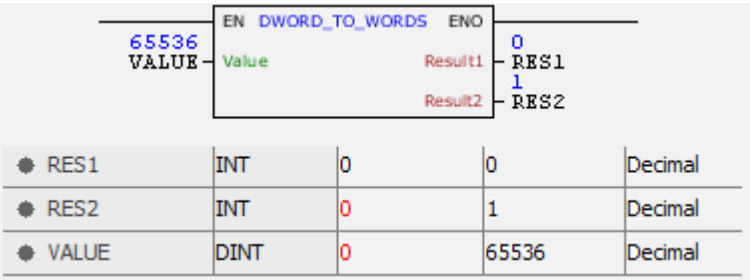
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

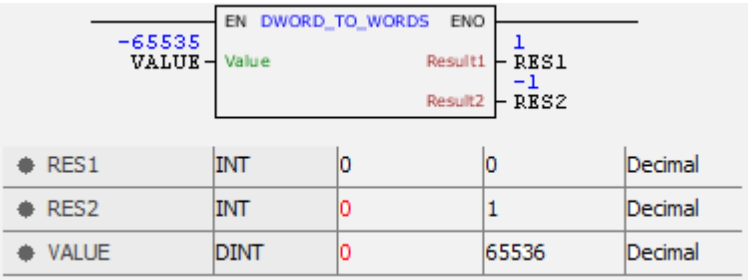
Block Flowchart



Example



RES1	INT	0	0	Decimal
RES2	INT	0	1	Decimal
VALUE	DINT	0	65536	Decimal



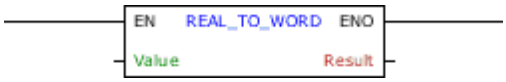
RES1	INT	0	0	Decimal
RES2	INT	0	1	Decimal
VALUE	DINT	0	65536	Decimal

The examples above perform the conversion of a variable VALUE, in DWORD, in two WORD values storing the final result in RESULT1 and RESULT2. The block ends with success and ENO output is activated.

11.8.5.6.8.6 REAL_TO_WORD

Block that performs the conversion of a REAL value into a WORD value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Value in REAL
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	WORD UINT INT	Value in WORD

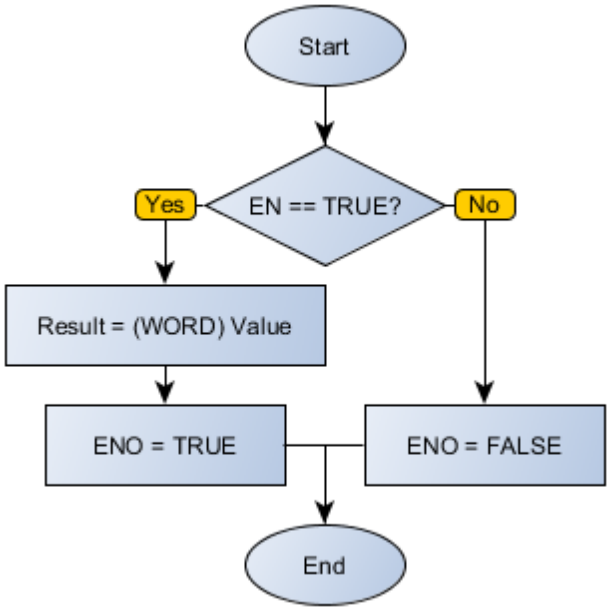
Operation

When this block has a TRUE value in EN, it interprets the Value value as REAL and converts it into WORD, storing in Result.

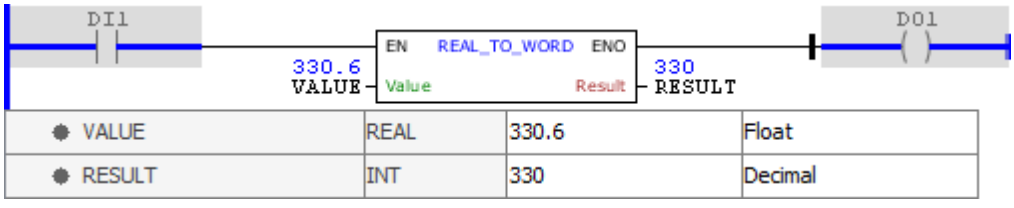
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

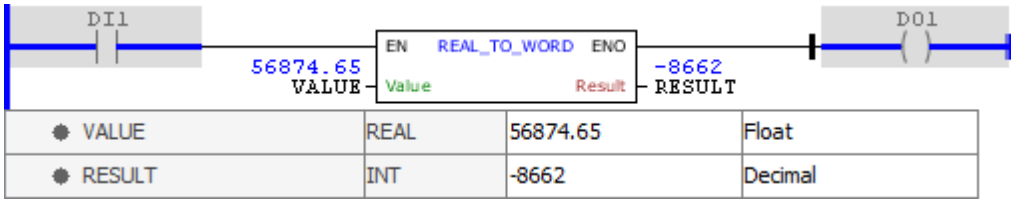
Block Flowchart



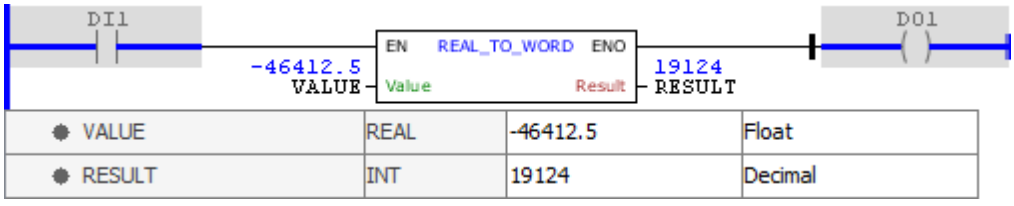
Example



VALUE	REAL	330.6	Float
RESULT	INT	330	Decimal



VALUE	REAL	56874.65	Float
RESULT	INT	-8662	Decimal



VALUE	REAL	-46412.5	Float
RESULT	INT	19124	Decimal

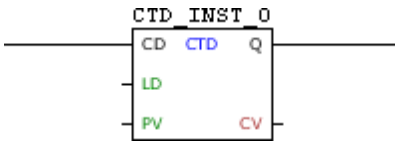
The examples above convert the VALUE variable, in DWORD, into a WORD value storing the final result in RESULT. The block ends with success and ENO output is activated. Note that the results are truncated in decimal and only the sixteen least significant bits are taken into account.

11.8.5.7 Counter

11.8.5.7.1 CTD

Countdown block of input pulses.

Ladder Representation



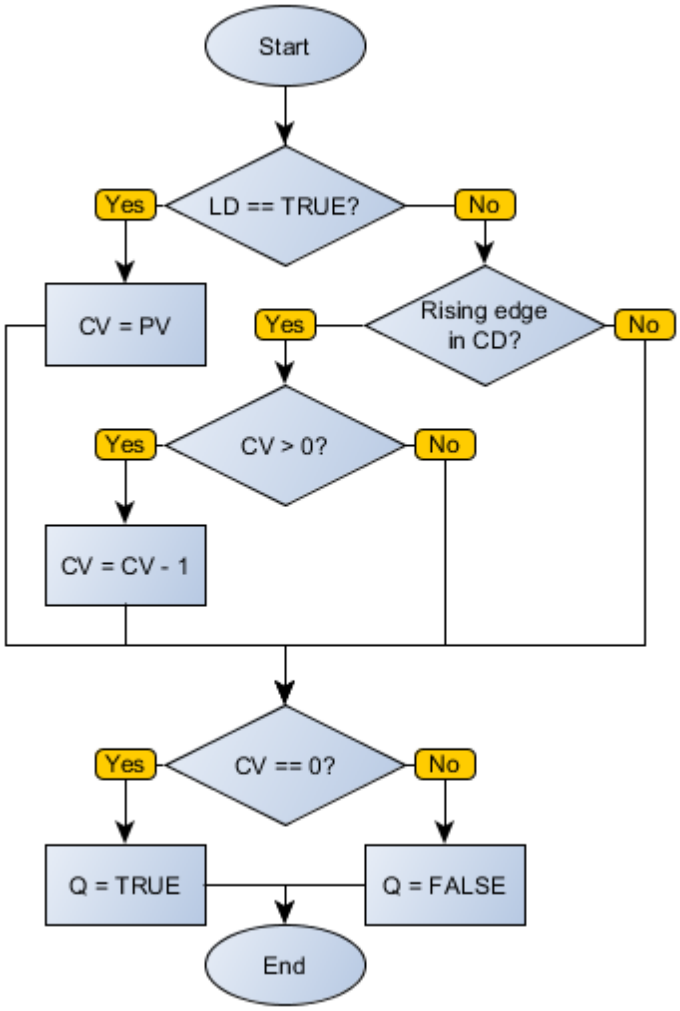
Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	CD	BOOL	Pulse identifier
	LD	BOOL	Loads the value of PV in CV
	PV	WORD UINT	Value of initial configuration
VAR_OUTPUT	Q	BOOL	Counter zeroed flag
	CV	WORD UINT	Current count value
VAR	CTD_INST_0	CTD	Instance of access to block structure

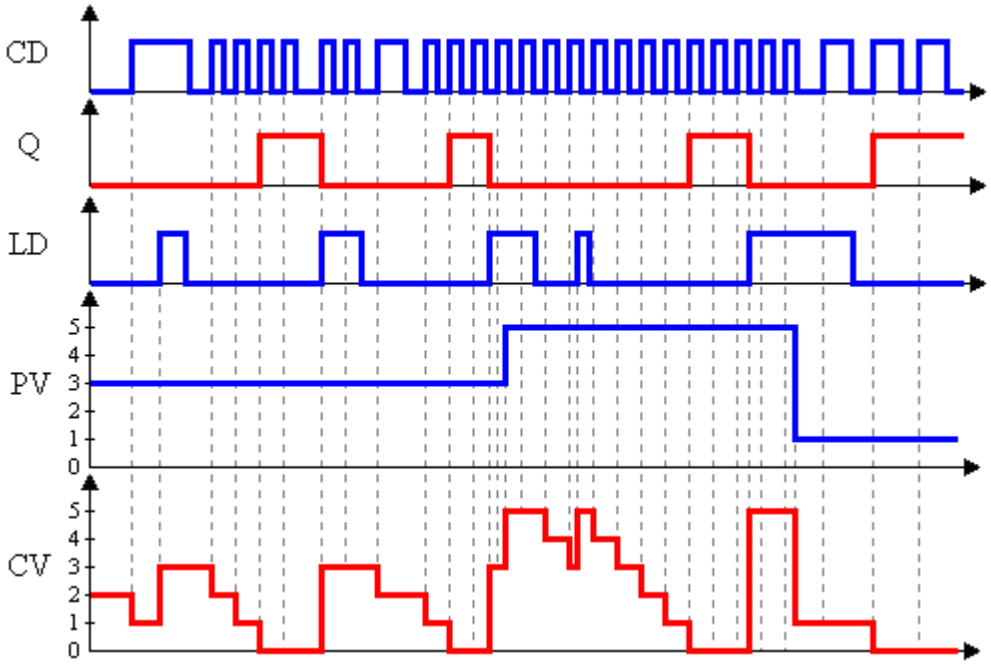
Operation

When this block identifies a leading edge in CD, it decrements the CV variable until it is zero. While CV equals zero, the output Q remains at TRUE level. By detecting high-level LD, the block loads the PV value in CV.

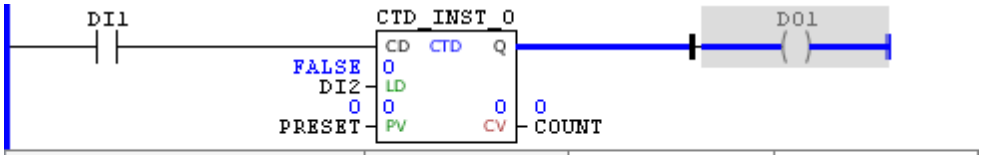
Block Flowchart



Operation Diagram

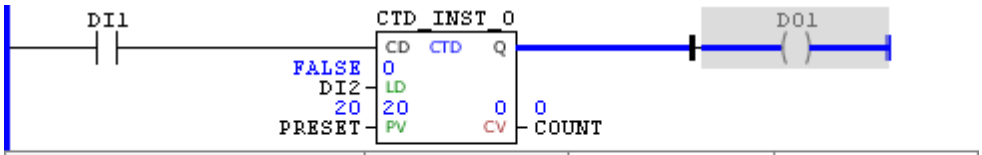


Example



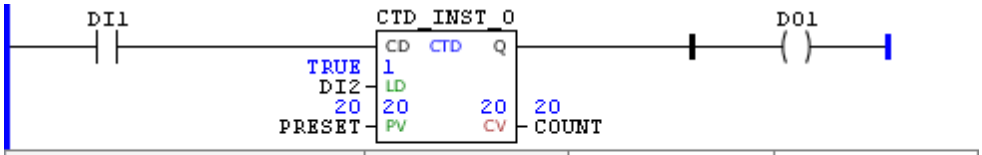
PRESET	UINT	0	Decimal
COUNT	UINT	0	Decimal

The above example shows the initial conditions of routine. As CV has a value of zero, the Q output is enabled.



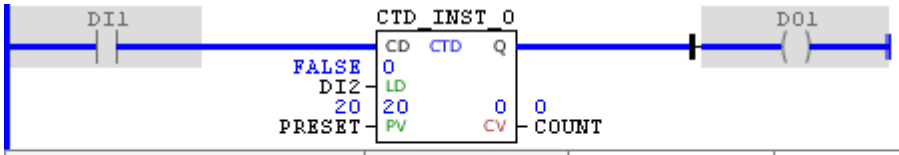
PRESET	UINT	20	Decimal
COUNT	UINT	0	Decimal

The value of the PV variable was changed to 20, but not yet loaded.



PRESET	UINT	20	Decimal
COUNT	UINT	20	Decimal

By identifying TRUE level in LD, the block loads the PV value to CV. Since this value is greater than zero, the Q output is disabled.



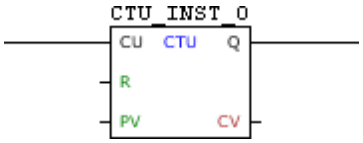
PRESET	UINT	20	Decimal
COUNT	UINT	0	Decimal

At each leading edge identified in CD, the value of COUNT is decremented until it reaches zero, when the Q output is enabled.

11.8.5.7.2 CTU

Block for gradual count of input pulses.

Ladder Representation



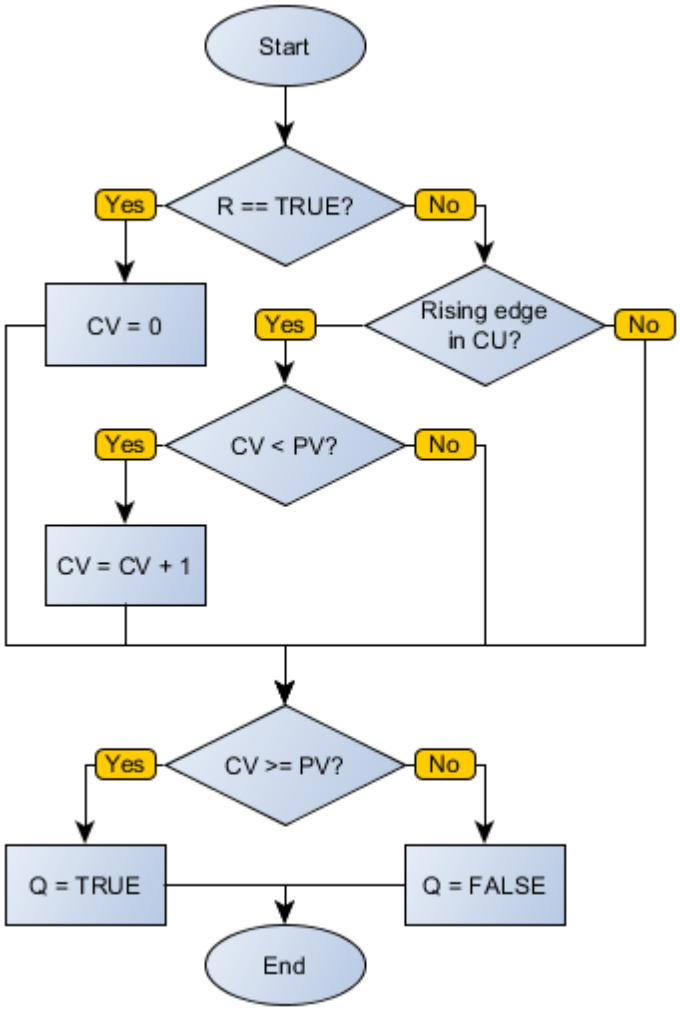
Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	CU	BOOL	Pulse identifier
	R	BOOL	Loads the zero value in CV
	PV	WORD UINT	Maximum count value
VAR_OUTPUT	Q	BOOL	Counter overrun flag
	CV	WORD UINT	Current count value
VAR	CTU_INST_0	CTU	Instance of access to block structure

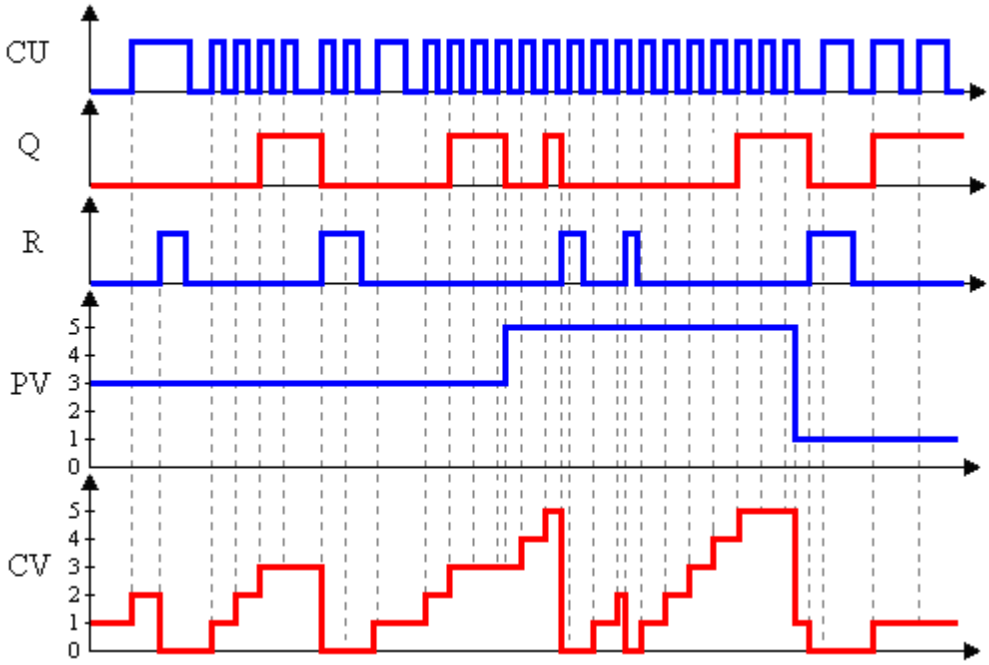
Operation

When this block identifies a leading edge in CD, it increments the CV variable until it is equal to PV. While CV equals PV, the output Q remains at TRUE level. By detecting high-level R, the block loads the zero value in CV.

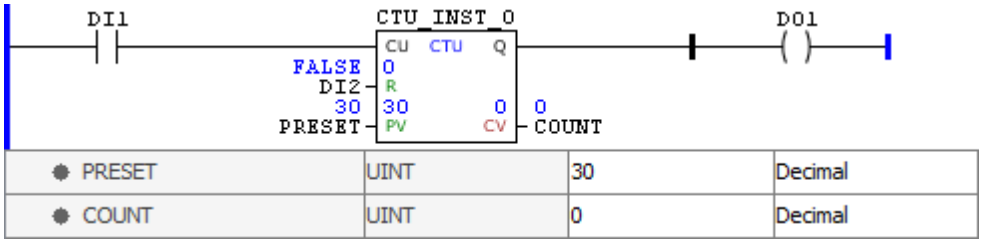
Block Flowchart



Operation Diagram

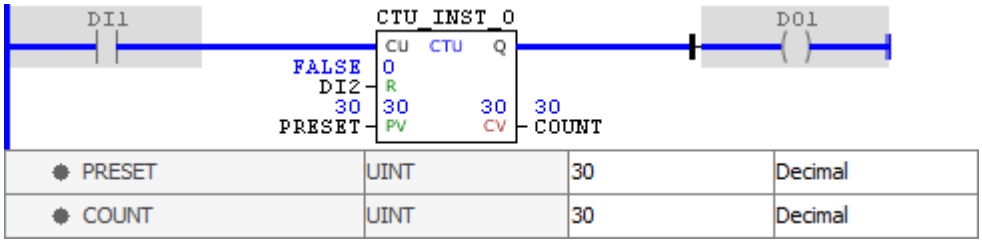


Example



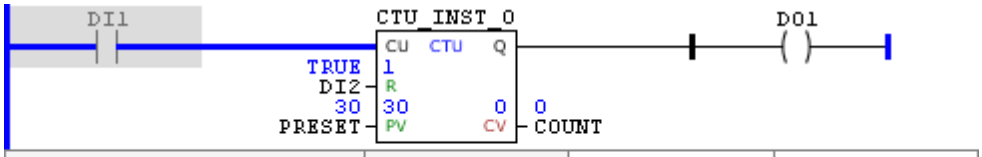
PRESET	UINT	30	Decimal
COUNT	UINT	0	Decimal

The above example shows the initial conditions of routine. Since CV has a lower value than of PV, the Q output is disabled.



PRESET	UINT	30	Decimal
COUNT	UINT	30	Decimal

At each leading edge identified in CU, the value of CV is incremented until it reaches the PV value, when the Q output is enabled.



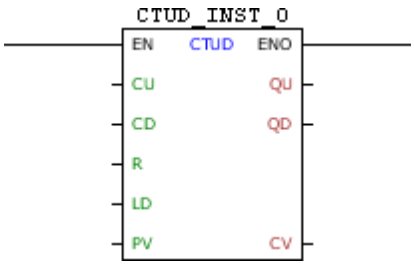
PRESET	UINT	30	Decimal
COUNT	UINT	0	Decimal

By identifying TRUE level in R, the block loads the zero value to CV. Since this value is lower than of PV, the Q output is disabled.

11.8.5.7.3 CTUD

Block for gradual count and countdown of input pulses.

Ladder Representation



Block Structure

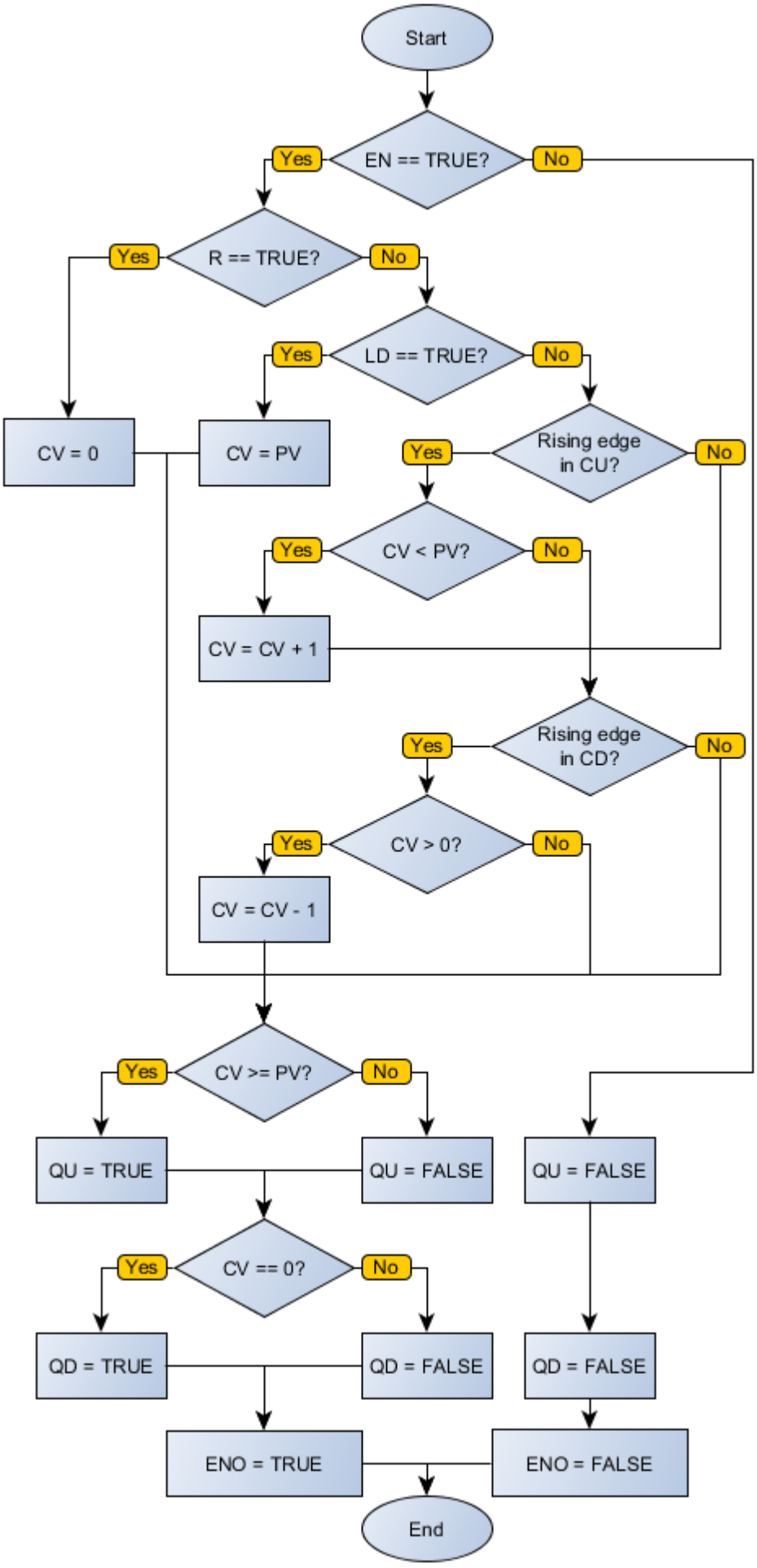
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	CU	BOOL	Pulse identifier for incremental
	CD	BOOL	Pulse identifier for decremental
	R	BOOL	Loads the zero value in CV
	LD	BOOL	Loads the value of PV in CV
	PV	WORD UINT	Reference value
VAR_OUTPUT	ENO	BOOL	Output enabling
	QU	BOOL	Counter overrun flag
	QD	BOOL	Counter zeroed flag
	CV	WORD UINT	Current count value
VAR	CTUD_INST_0	CTUD	Instance of access to block structure

Operation

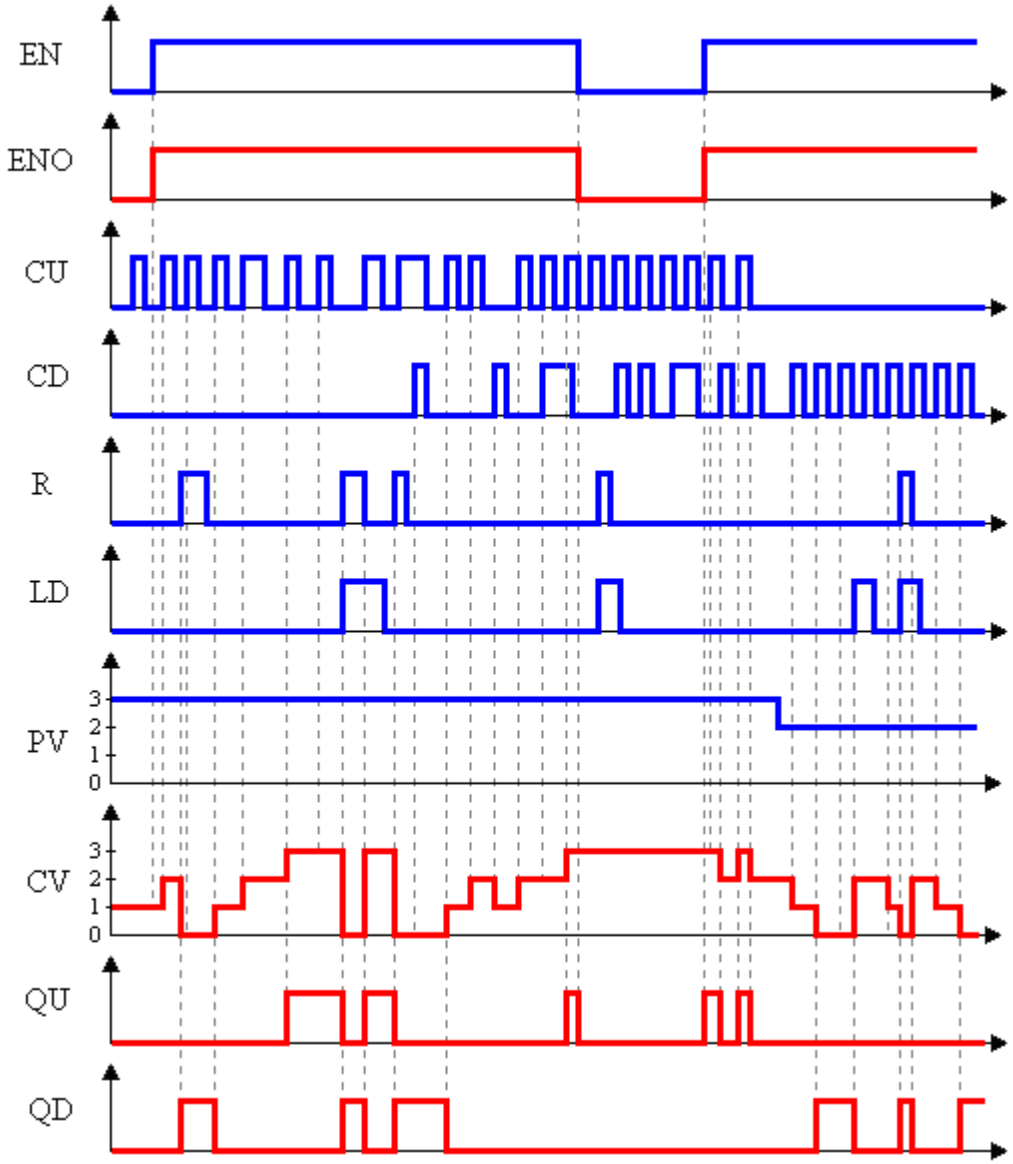
When this block has a TRUE value in EN, it acts as a CTD block and block CTU at the same time acting on the same CV counter. That is: increments CV until it reaches PV to the leading edges in CU and decrements CV until it reaches zero to the leading edges in CD. A positive transition in R carries zero in CV, while a leading edge in LD loads the PV value in CV. If CV has zero value, QD receives TRUE, and if CV has value equal to PV, QU receives TRUE.

The ENO value forwards to the next Ladder block the EN value.

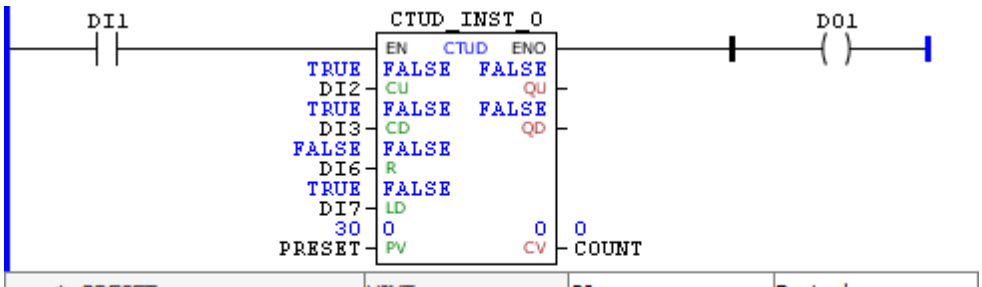
Block Flowchart


Start
EN == TRUE?
Yes
No
R == TRUE?
Yes
No
LD == TRUE?
Yes
No
CV = 0
CV = PV
Rising edge in CU?
Yes
No
CV < PV?
Yes
No
CV = CV + 1
Rising edge in CD?
Yes
No
CV > 0?
Yes
No
CV = CV - 1
CV >= PV?
Yes
No
QU = TRUE
QU = FALSE
QU = FALSE
CV == 0?
Yes
No
QD = TRUE
QD = FALSE
QD = FALSE
ENO = TRUE
ENO = FALSE
End

Operation Diagram

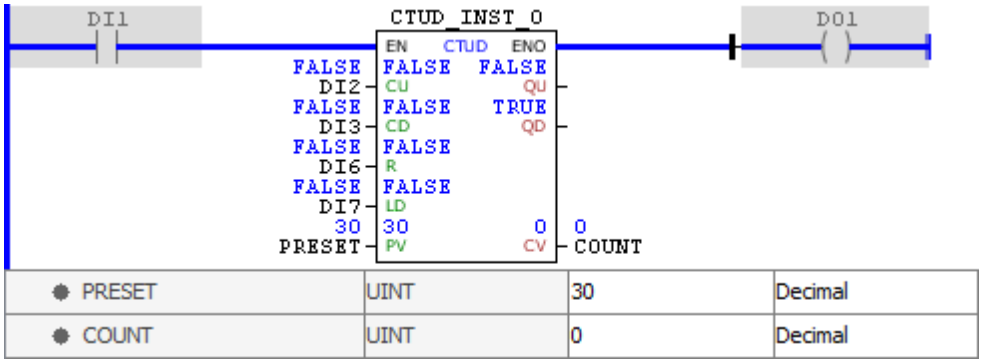


Example



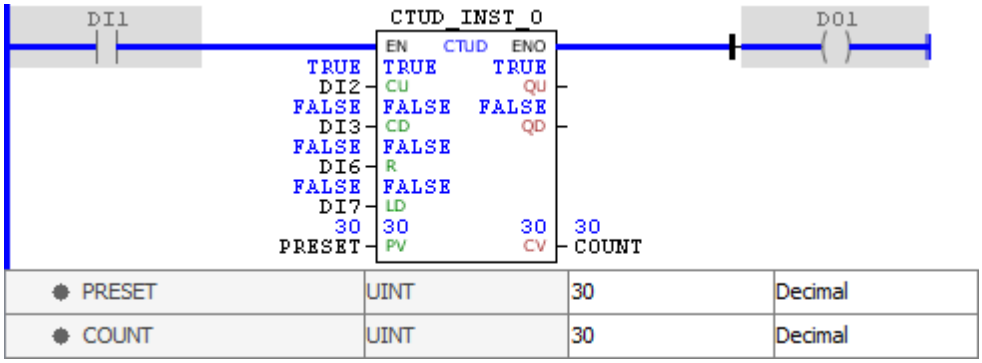
PRESET	UINT	30	Decimal
COUNT	UINT	0	Decimal

The example above shows the disabled block, with all its internal variables zeroed. Although the external controls are activated, these values are not forwarded to the instance of the block.



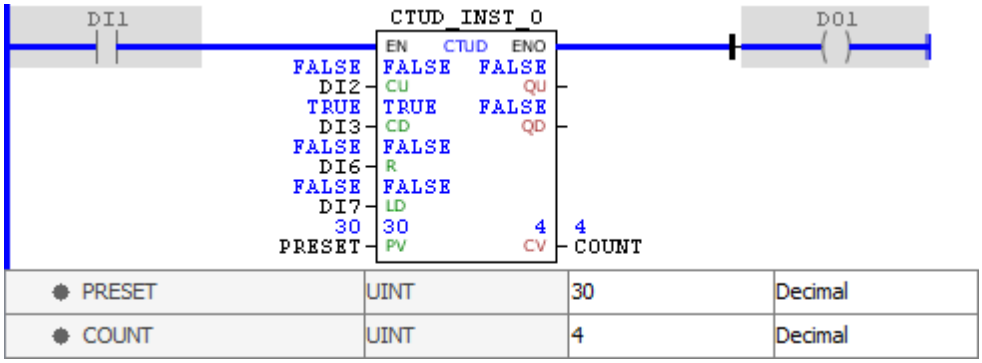
PRESET	UINT	30	Decimal
COUNT	UINT	0	Decimal

When activated, the block identifies the value of PRESET, loading it in PV, and identifies that the output is at zero, enabling the QD output. When execution is completed, the ENO output is activated.



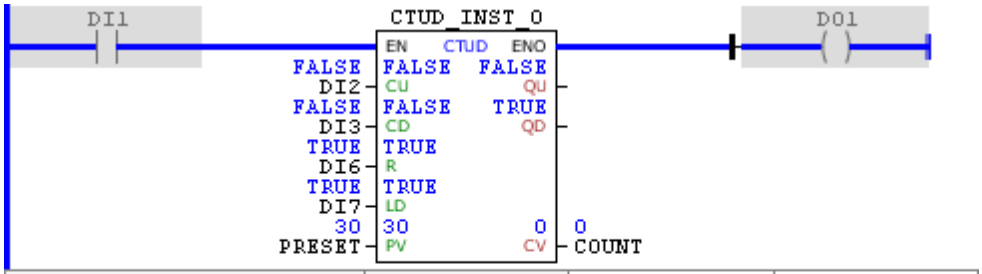
PRESET	UINT	30	Decimal
COUNT	UINT	30	Decimal

At each leading edge identified in CU, the value of CV is incremented until it reaches the PV value, when the QU output is enabled. When execution is completed, the ENO output is activated.



PRESET	UINT	30	Decimal
COUNT	UINT	4	Decimal

At each leading edge detected in CD, the CV value is decremented. When CV is a value between zero and PV, both QD and QU outputs are deactivated. When execution is completed, the ENO output is activated.



PRESET	UINT	30	Decimal
COUNT	UINT	0	Decimal

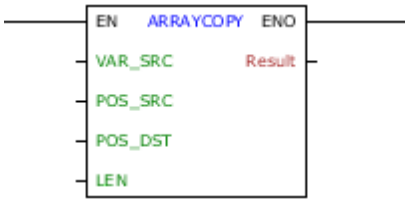
A TRUE value in R resets CV, while a TRUE value in LD loads the value of PV to CV. As we can see, R prevails over LD, leaving CV and enabling the QD output. When execution is completed, the ENO output is activated.

11.8.5.8 Data Transfer

11.8.5.8.1 ARRAYCOPY

Block that copies an array from a certain position to another array or to itself.

Ladder Representation



Block Structure

Variable Type	Name	Data type	Description
VAR_INPUT	EN	BOOL	Block enabling
	VAR_SRC	Array: BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Input Array
	POS_SRC	BYTE USINT WORD UINT	Position of the input array from which the copy will be made
	POS_DST	BYTE USINT WORD UINT	Position of the output array from which it will be replaced
	LEN	BYTE USINT WORD UINT	Number of array positions to be copied
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	Array: BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Output Array

Operation

This block, when it has a value of TRUE in EN, copies LEN values from the POS_SRC position from the input array (VAR_SRC) to the position POS_DST into the destination array (Result).

Comments:
- POS_SRC, POS_DST and LEN input variables only accept positive integers. If a negative value is assigned to any of them, the value zero will be considered.
- The Input Array can be repeated on the output without worrying about data being overwritten.
- If the amount of data to be copied defined by LEN exceeds the last position of the input array, only valid data will be copied to the last position of the input array, thus avoiding any garbage being assigned to the output array.
- If the amount of data to be copied defined by LEN exceeds the last position of the output array, only the data required to complete it will be copied, preventing subsequent memory from receiving unwanted values.
- The block will not execute if LEN has a value greater than the size of the input array.

When EN has FALSE value, Result remains unchanged.

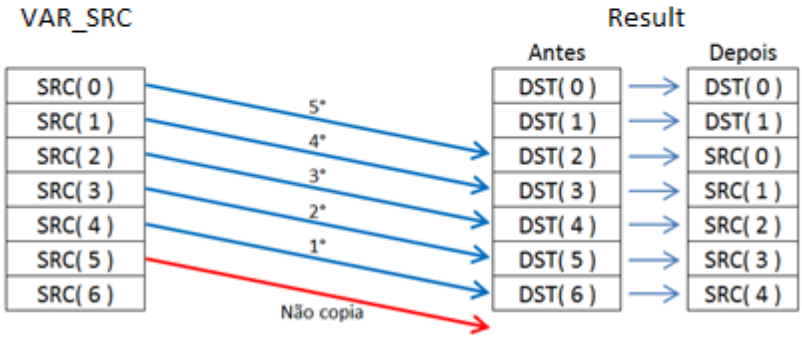
The ENO value forwards to the next Ladder block the EN value after the operation is completed.

NOTE!

- It is important to notice that not only LEN but also POS_SRC will not exceed the VAR_SRC array's size. The same must be noticed when setting values to POS_DST, related to the output array **Result**.
- To learn how to create arrays please go to: Ladder > Editor > Variables > Editing in the Rung

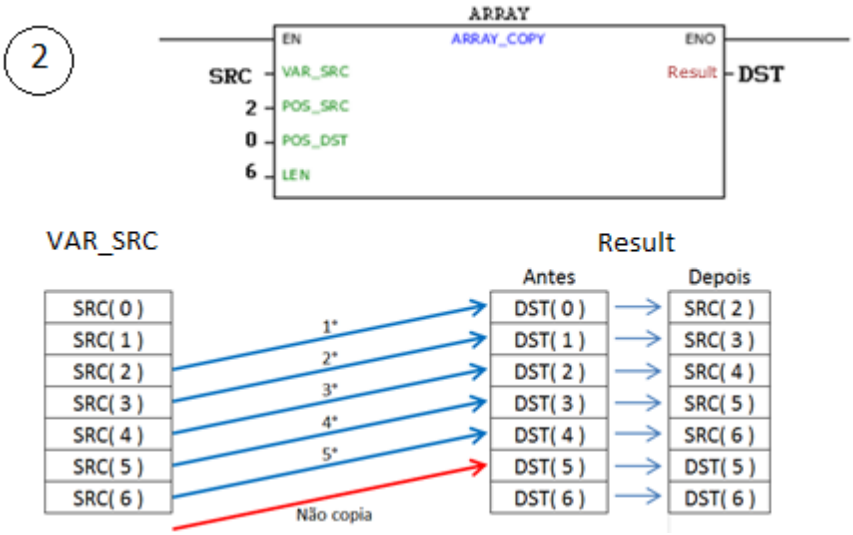
Block Flowchart

Example



1

Main Ladder				
SRC				
SRC[0]	INT	1	1	Decimal
SRC[1]	INT	2	2	Decimal
SRC[2]	INT	3	3	Decimal
SRC[3]	INT	4	4	Decimal
SRC[4]	INT	5	5	Decimal
SRC[5]	INT	6	6	Decimal
SRC[6]	INT	7	7	Decimal
DST				
DST[0]	INT	0	0	Decimal
DST[1]	INT	0	0	Decimal
DST[2]	INT	0	3	Decimal
DST[3]	INT	0	4	Decimal
DST[4]	INT	0	5	Decimal
DST[5]	INT	0	6	Decimal
DST[6]	INT	0	7	Decimal



2

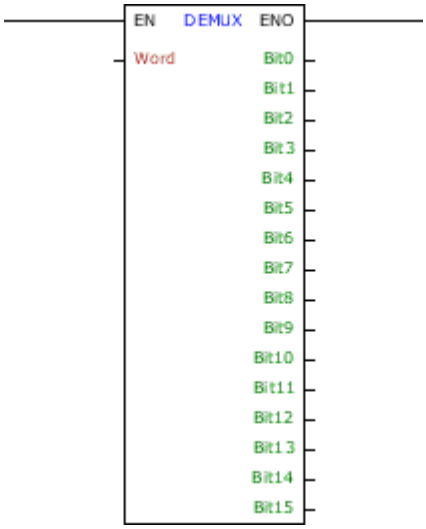
Main Ladder				
SRC				
SRC[0]	INT	1	1	Decimal
SRC[1]	INT	2	2	Decimal
SRC[2]	INT	3	3	Decimal
SRC[3]	INT	4	4	Decimal
SRC[4]	INT	5	5	Decimal
SRC[5]	INT	6	6	Decimal
SRC[6]	INT	7	7	Decimal
DST				
DST[0]	INT	0	3	Decimal
DST[1]	INT	0	4	Decimal
DST[2]	INT	0	5	Decimal
DST[3]	INT	0	6	Decimal
DST[4]	INT	0	7	Decimal
DST[5]	INT	0	0	Decimal
DST[6]	INT	0	0	Decimal

In the examples above the value of the variable SRC is copied to DST array, according to source position (POS_SRC), destination (POS_DST) and the lenght to be copied (LEN). The block ends with success and ENO output is activated.

11.8.5.8.2 DEMUX

Block that creates 16 new BOOL variables from the decomposition of a WORD variable.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Word	WORD UINT INT	Input variable of 15 bits
VAR_OUTPUT	ENO	BOOL	End of operation
	Bit0 – Bit15	BOOL	Bit of the corresponding position of Word

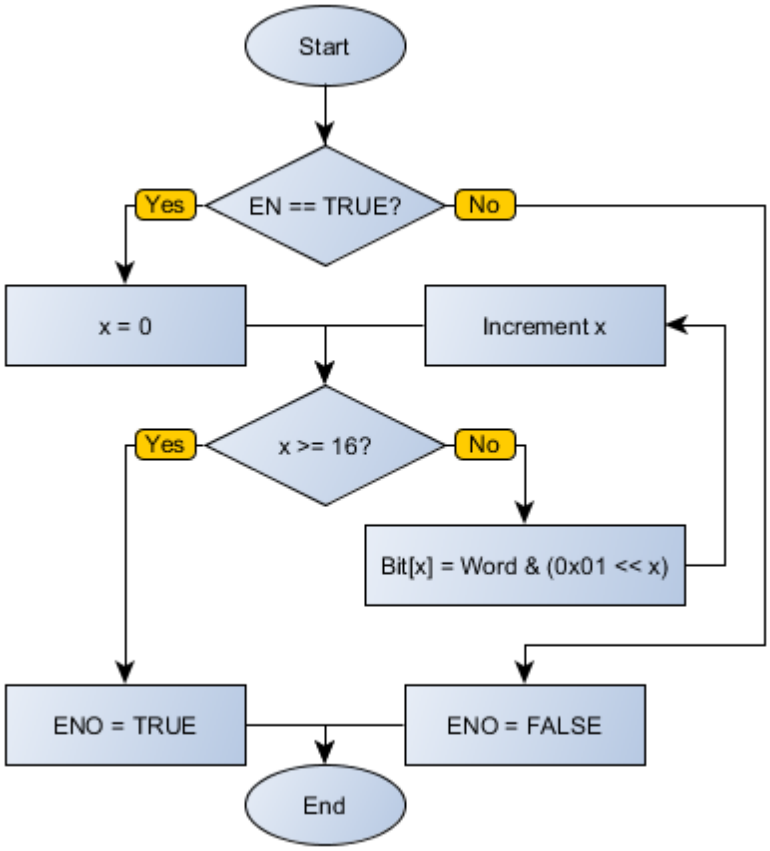
Operation

When this block has a TRUE value in EN, it decomposes the input variable in Word 15 Boolean values stored in Bit0 to Bit15 variables. Bit0 corresponds to the LSB (least significant bit) and Bit15 corresponds to the MSB (most significant bit).

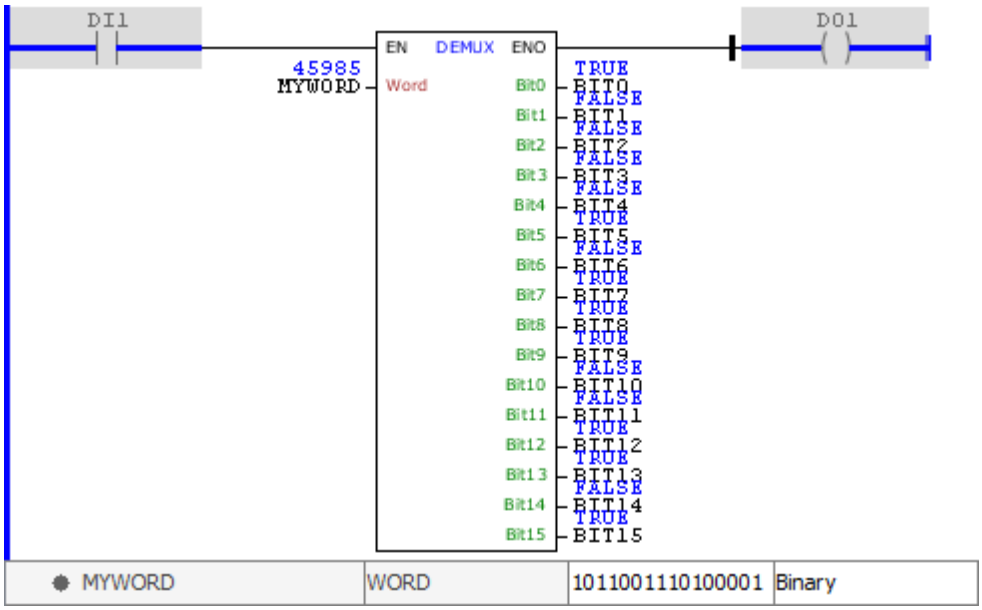
When EN has FALSE value, output variables remain unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example

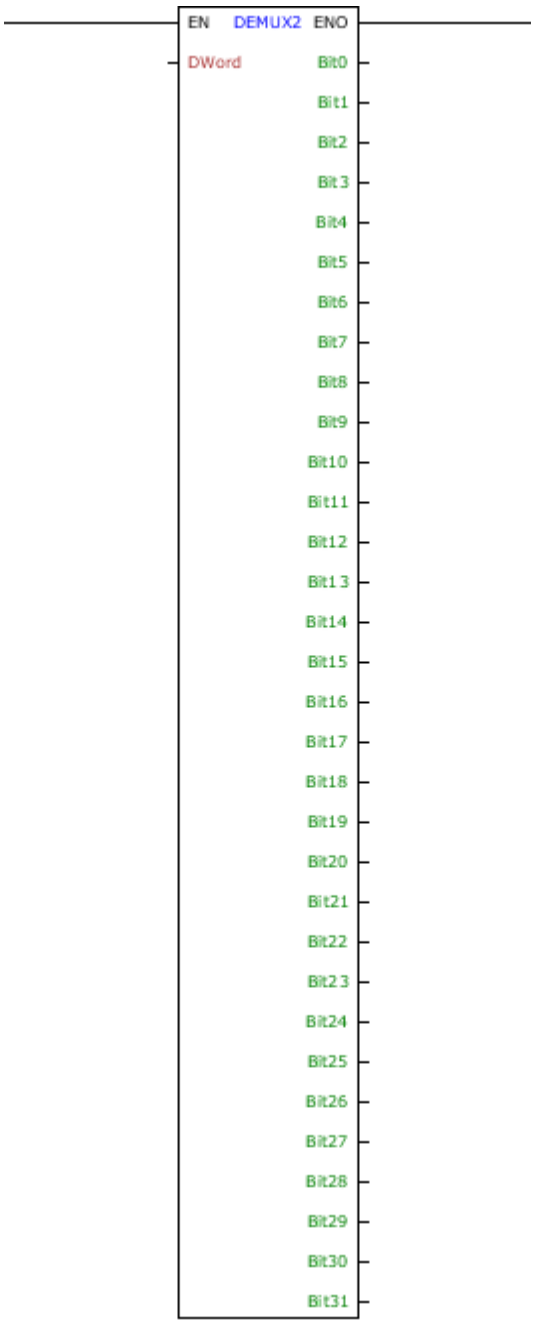


The example above decomposes the value of MYWORD in Boolean values, which are stored in the output variables BIT0 to Bit15. The block ends successfully and the ENO output is activated.

11.8.5.8.3 DEMUX2

Block that creates 32 new BOOL variables from the decomposition of a DWORD variable.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	DWord	DWORD UDINT DINT	Input variable of 15 bits
VAR_OUTPUT	ENO	BOOL	End of operation
	Bit0 – Bit31	BOOL	Bit of the corresponding position of Word

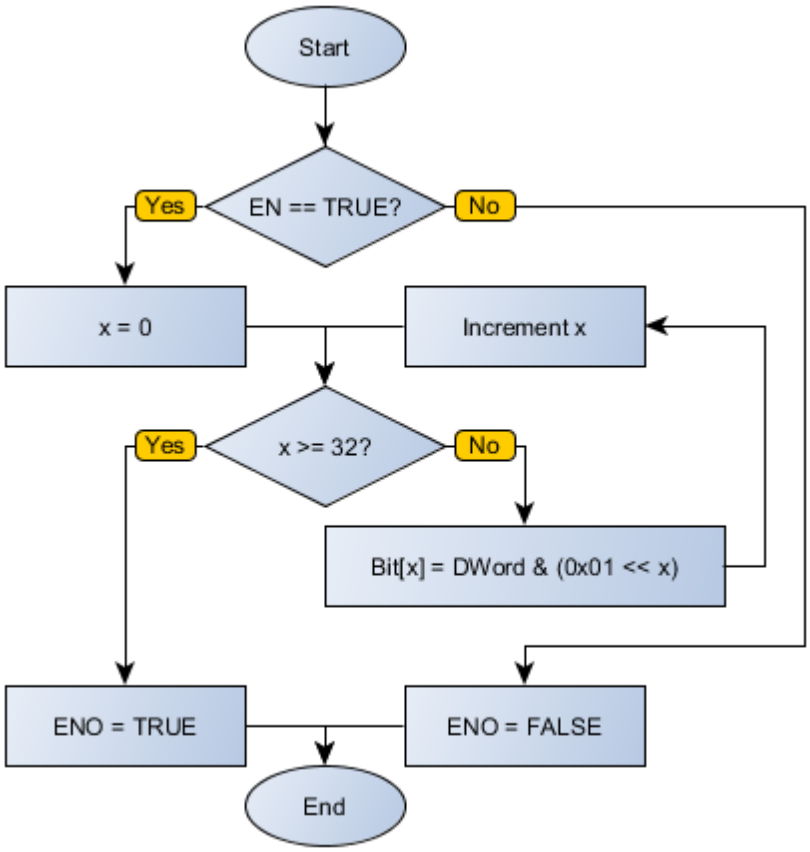
Operation

When this block has a TRUE value in EN, it decomposes the input variable in DWord 32 Boolean values stored in Bit0 to Bit31 variables. Bit0 corresponds to the LSB (least significant bit) and Bit15 corresponds to the MSB (most significant bit).

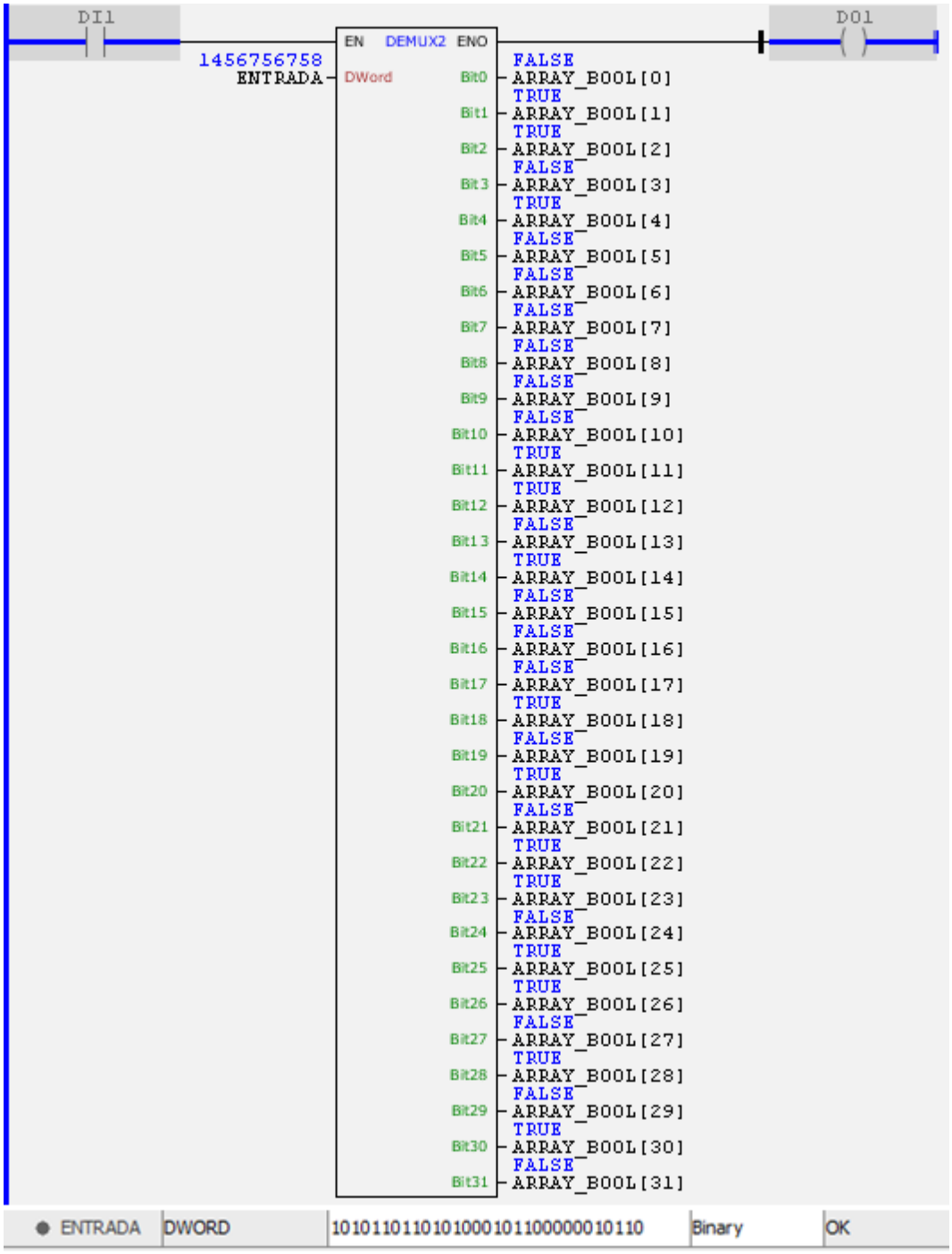
When EN has FALSE value, output variables remain unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example

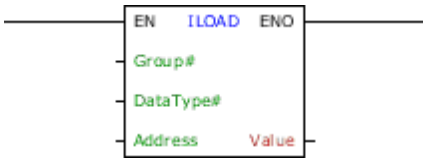


The example above decomposes the value of MYDWORD in Boolean values, which are stored in the output variables BIT0 to Bit31. The block ends successfully and the ENO output is activated

11.8.5.8.4 ILOAD

Block which indirectly loads the value of a variable and transfers it to Value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Group#	BYTE	Group w here the variable is stored
	DataType#	BYTE	Data type of the selected variable
	Address	DWORD UDINT DINT	Address of the global variable, as its group
VAR_OUTPUT	ENO	BOOL	End of operation
	Value	As selected in DataType#	Value of the selected variable

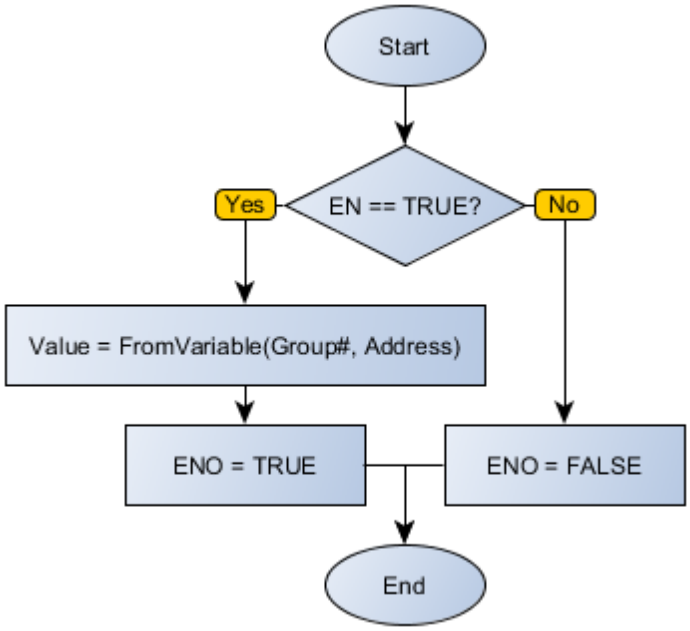
Operation

When this block has a TRUE value in EN, it loads, in Value, the of the Address variable belonging to the Group# group, as the selected DataType#.

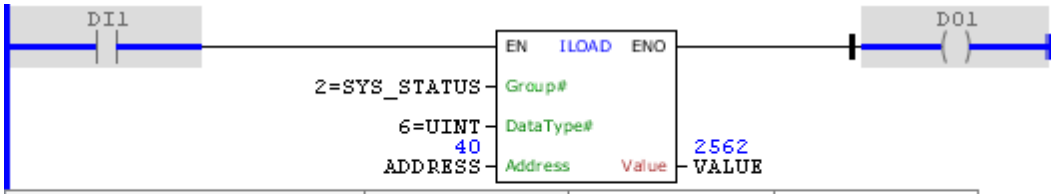
When EN has FALSE value, Value remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



ADDRESS	DINT	40	Decimal
VALUE	UINT	2562	Decimal

The above example loads the value of the address 40 of group 2 (GLOBAL_SYSTEM%S), which represents the status of ESC key in UINT format for the VALUE variable. The block ends with success and ENO output is activated.

11.8.5.8.5 ILOADBOOL

Block that indirectly loads the value of a bit in a global variable address.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Group#	BYTE	Group where the variable is stored
	Address	DWORD UDINT DINT	Address of the global variable, as its group
	Bit	BYTE USINT SINT	Position of the bit to be checked
VAR_OUTPUT	ENO	BOOL	End of operation
	Value	BOOL	Value of the bit selected by the input arguments

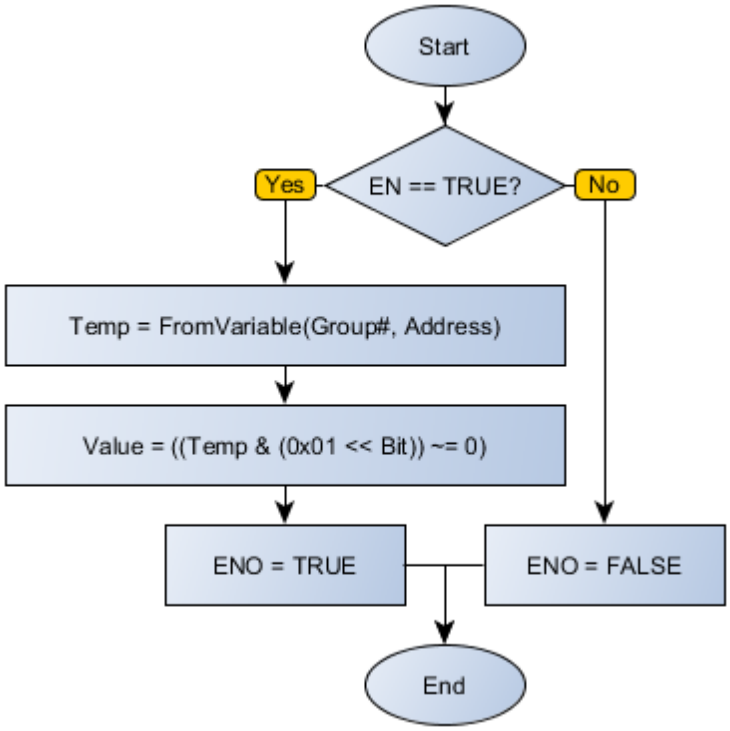
Operation

When this block has a TRUE value in EN, it loads, in Value, the Bit contents of the Address variable belonging to the Group# group.

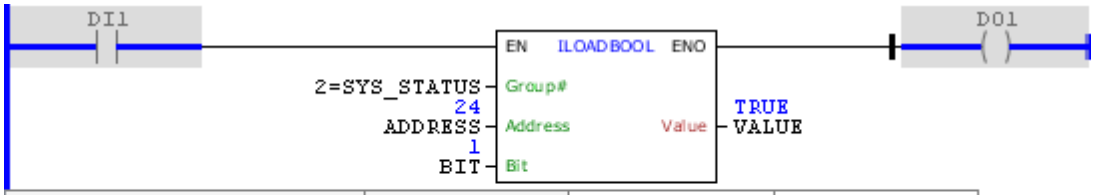
When EN has FALSE value, Value remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



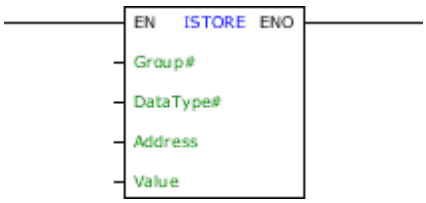
ADDRESS	DINT	24	Decimal
BIT	BYTE	1	Decimal
VALUE	BOOL	1	Binary

The above example loads the value of bit 1 of the address 24 of group 2 (S GLOBAL_SYSTEM%), which represents the status of ESC key for the VALUE variable. The block ends with success and ENO output is activated.

11.8.5.8.6 ISTORE

Block that indirectly loads the Value value in a variable.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Group#	BYTE	Group where the variable is stored
	DataType#	BYTE	Data type of the selected variable
	Address	DWORD UDINT DINT	Address of the global variable, as its group
	Value	Depending on the selection of the DataType#	Value to be written in the selected variable
VAR_OUTPUT	ENO	BOOL	End of operation

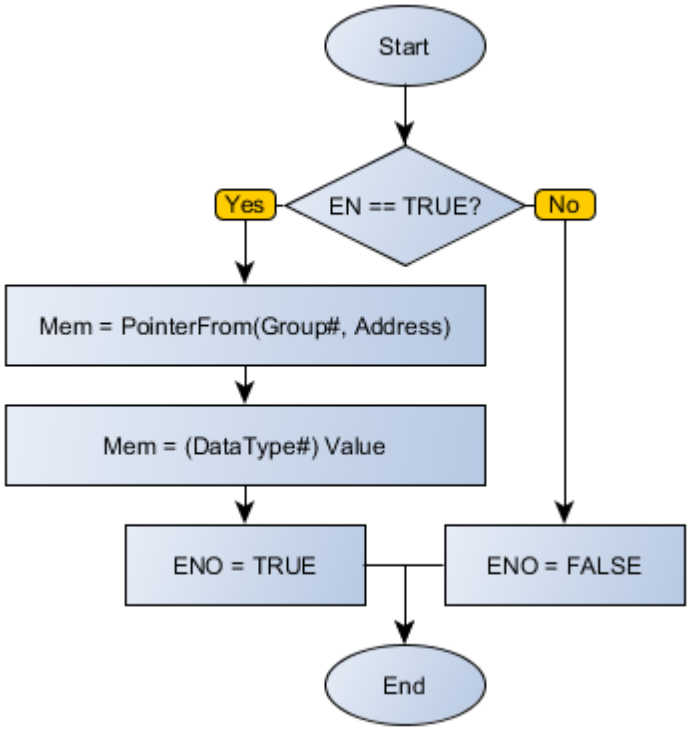
Operation

When this block has a TRUE value in EN, it loads the Value value in the contents of the Address variable belonging to the Group# group, as the selected DataType#.

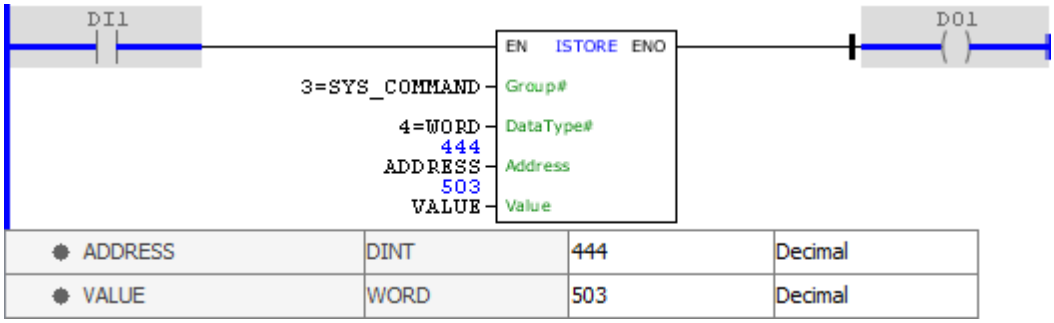
When EN has FALSE value, Value remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



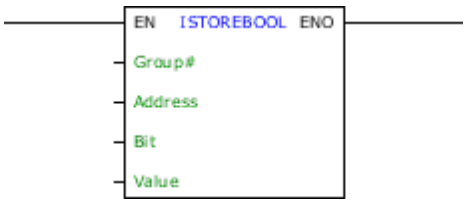
ADDRESS	DINT	444	Decimal
VALUE	WORD	503	Decimal

The example above stores the VALUE value in WORD format in address 444 of group 3 (GLOBAL_SYSTEM% C), which represents the index of the communication port Modbus TCP. The block ends with success and ENO output is activated.

11.8.5.8.7 ISTOREBOOL

Block that indirectly loads the Value value in a bit in a global variable address.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Group#	BYTE	Group where the variable is stored
	Address	DWORD UDINT DINT	Address of the global variable, as its group
	Bit	BYTE USINT SINT	Position of the bit to be modified
	Value	BOOL	New value of the selected bit
VAR_OUTPUT	ENO	BOOL	End of operation

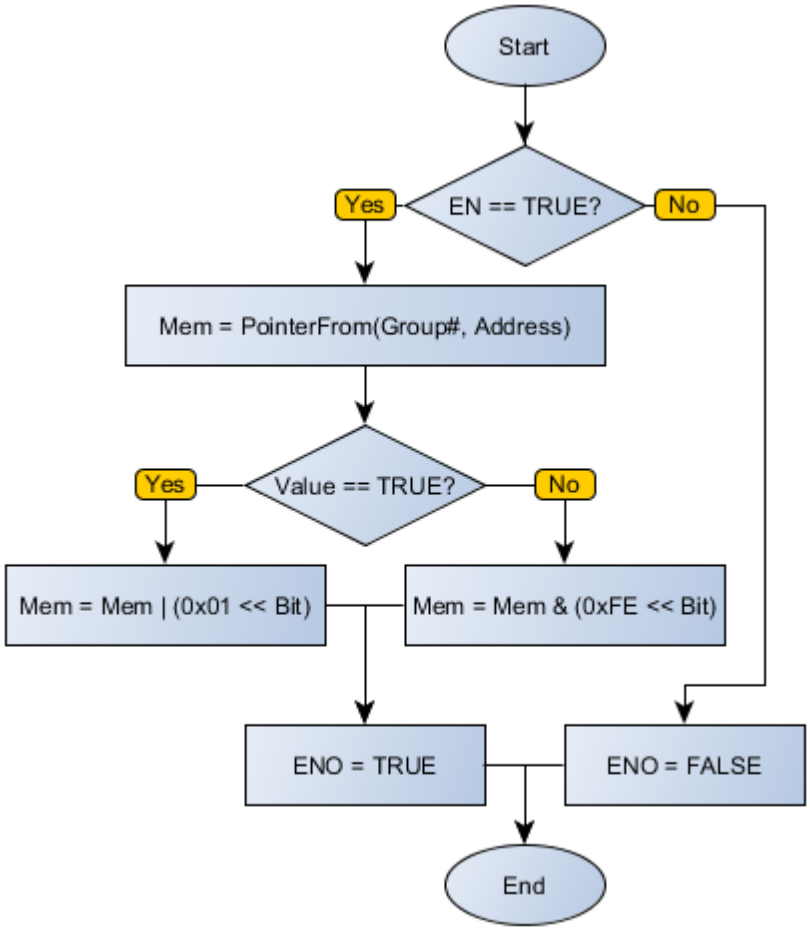
Operation

When this block has a TRUE value in EN, it loads the Value value in the Bit contents of the Address variable belonging to the Group# group.

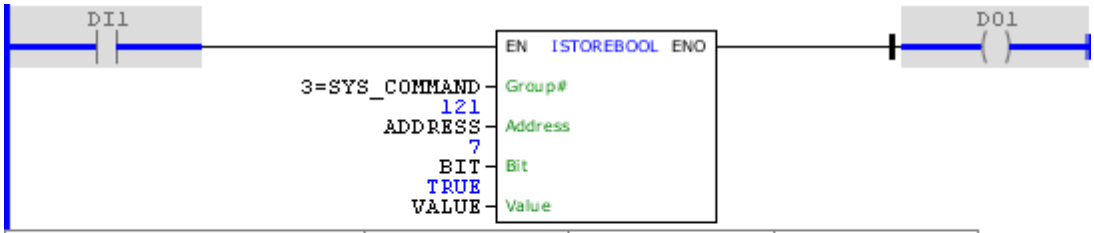
When EN has FALSE value, Value remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



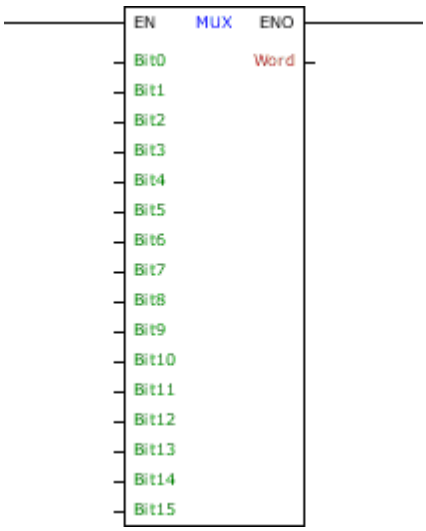
ADDRESS	DINT	121	Decimal
BIT	BYTE	7	Decimal
VALUE	BOOL	1	Binary

The example above stores the value of VALUE in bit 7 of the address 121 in group 3 (GLOBAL_SYSTEM% C), which represents the disable command of CANopen communication. The block ends with success and ENO output is activated.

11.8.5.8.8 MUX

Block that creates a new WORD variable from the concatenation of 16 BOOL variables.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Bit0 – Bit15	BOOL	Bit of the corresponding position in the new word
VAR_OUTPUT	ENO	BOOL	End of operation
	Word	WORD UINT INT	New word formed from the input bits

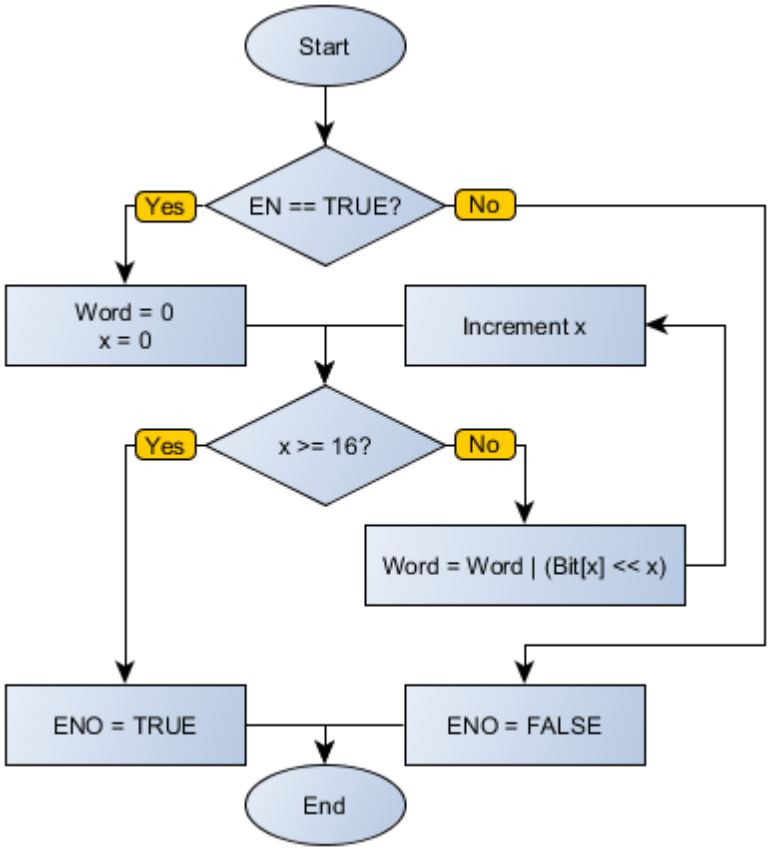
Operation

When this block has a TRUE value in EN, it concatenates Boolean values of the input variables and stores this value in the variable Word. Bit0 corresponds to LSB (least significant bit) and Bit15 corresponds to the MSB (most significant bit).

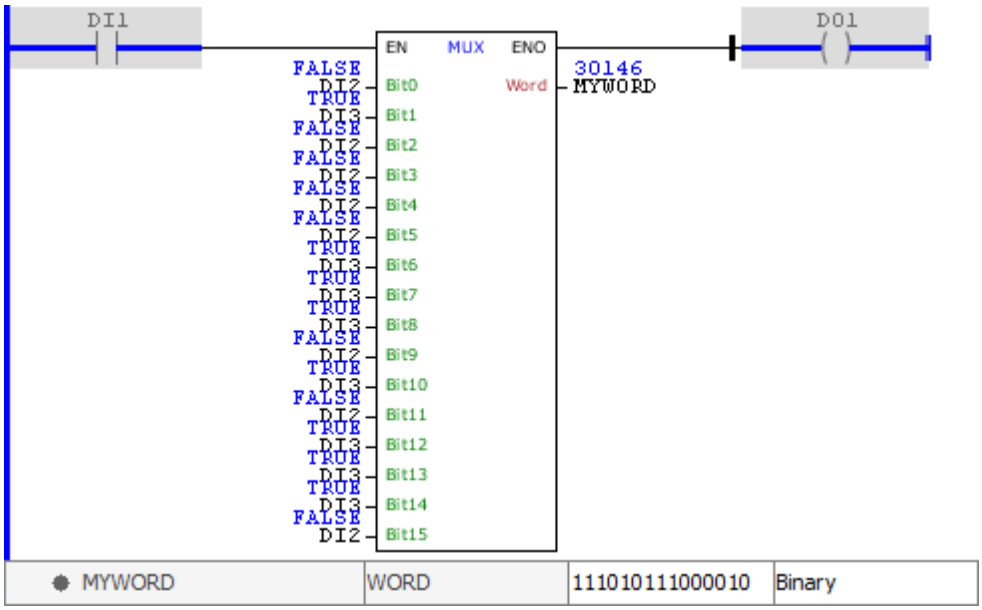
When EN has FALSE value, Word remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example

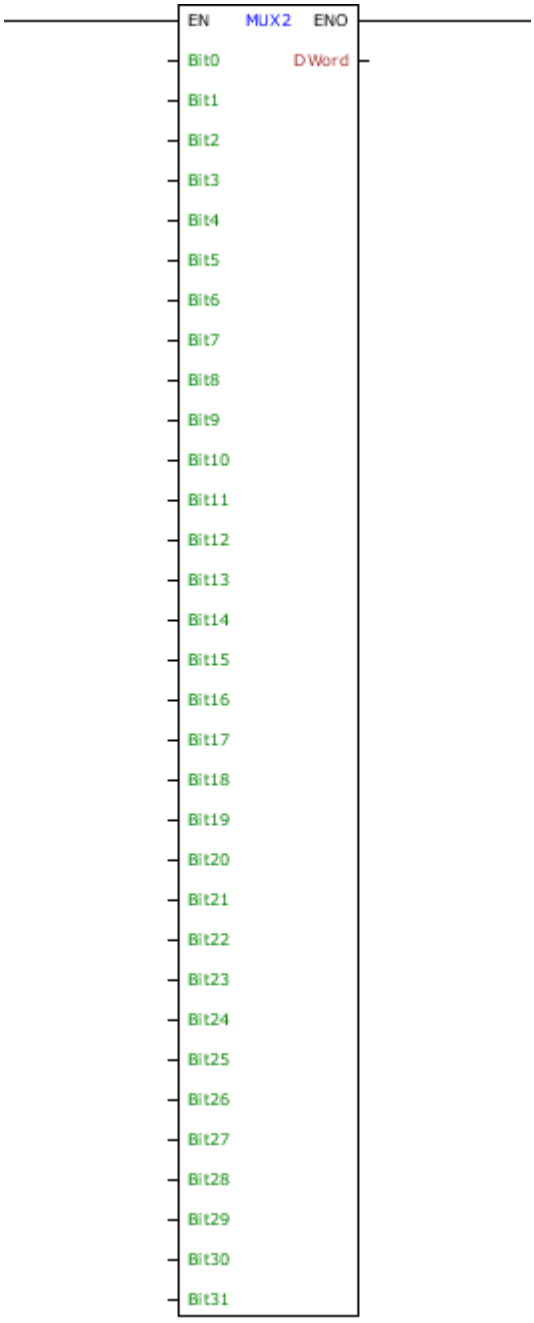


The above example concatenates the Boolean values of the input bits of the block to form the output word stored in MYWORD. The block ends with success and ENO output is activated.

11.8.5.8.9 MUX2

Block that creates a new DWORD variable from the concatenation of 32 BOOL variables.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Bit0 – Bit31	BOOL	Bit of the corresponding position in the new word
VAR_OUTPUT	ENO	BOOL	End of operation
	DWord	DWORD UDINT DINT	New word formed from the input bits

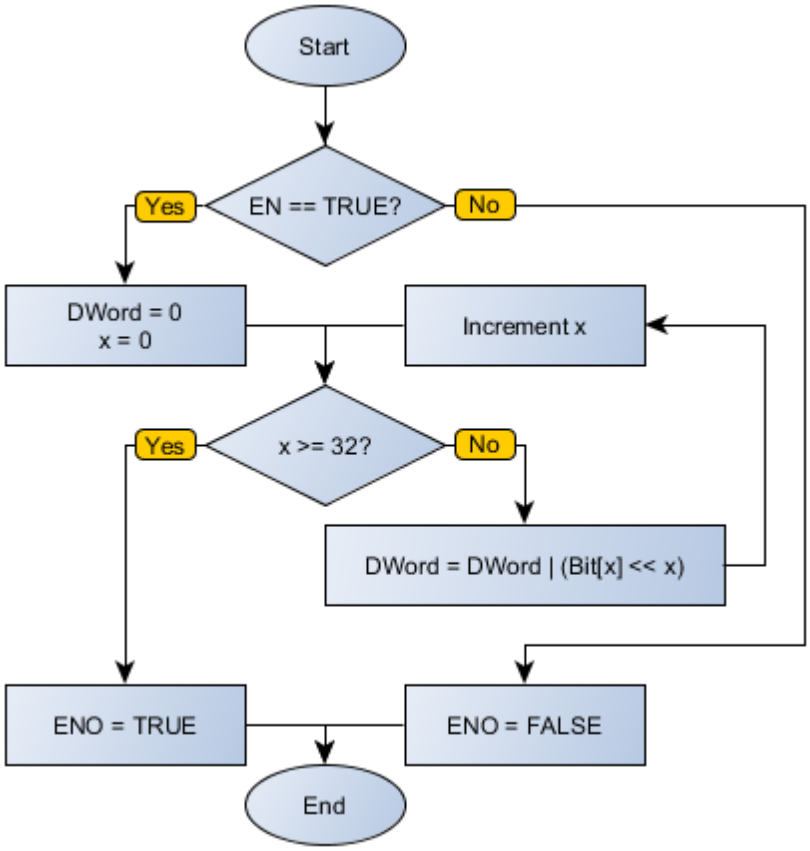
Operation

When this block has a TRUE value in EN, it concatenates Boolean values of the input variables and stores this value in the variable DWord. Bit0 corresponds to LSB (least significant bit) and Bit31 corresponds to the MSB (most significant bit).

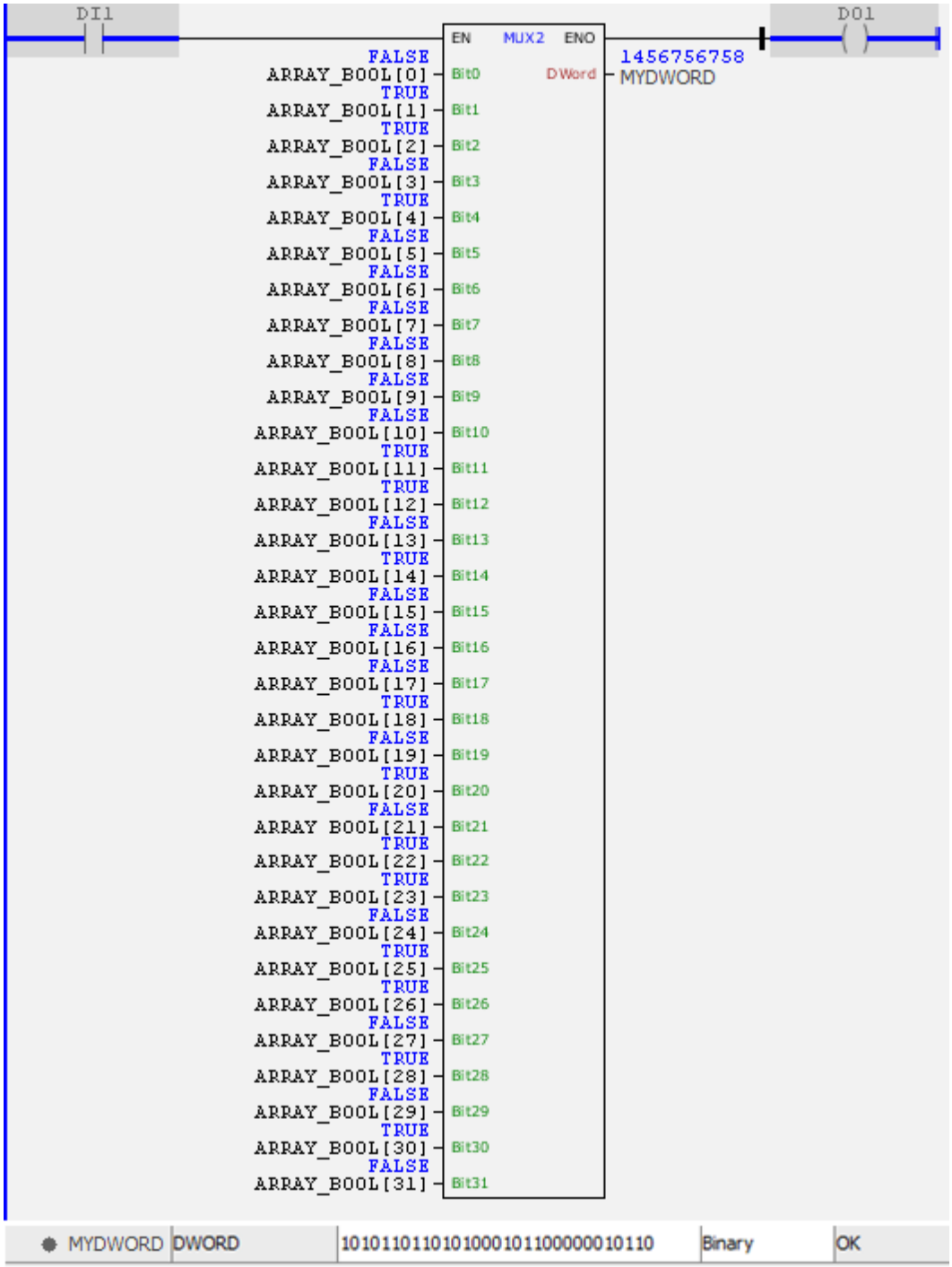
When EN has FALSE value, Word remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



The above example concatenates the Boolean values of the input bits of the block to form the output word stored in MYDWORD. The block ends with success and ENO output is activated.

11.8.5.8.10 ReadRecipe

Block that gets the recipe from a recipe file and sends it to a variable.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	Execute	BOOL	Block enabling
	FILENAME#	STRING	Name of the recipe file
	INDEX	WORD UINT	Recipe index to be read
VAR_OUTPUT	Q	BOOL	End of operation
	ERROR	BOOL	Error occurrence flag
	ERRORID	BYTE USINT	Identifier of the occurred error
	DST	STRUCT	Variable where the data read will be saved
VAR	READRECIPE_INST_0	READRECIPE	Instance of access to block structure

Operation

When this block identifies a leading edge in Execute, it gets the data from the selected recipe by the INDEX index in the # FILENAME file and sends them to the DST. If everything goes successfully, Q receives TRUE and remains so while Execute is TRUE.

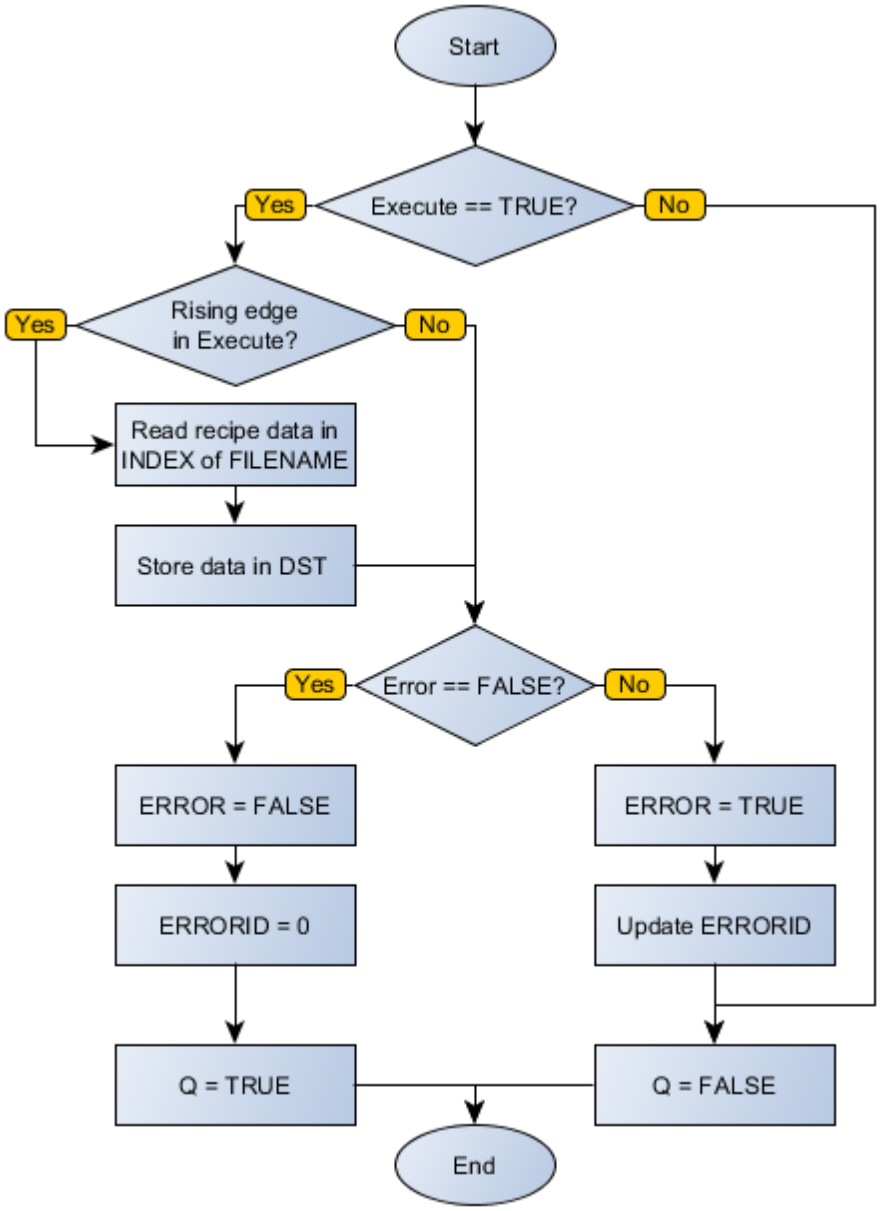
NOTE!
Recipes stored in RAM is identified by 'RECIPE_NAME'. Recipes stored on the SD card are identified by 'RECIPE_NAME.CSV'.

When Execute has FALSE value, DST remains unchanged.

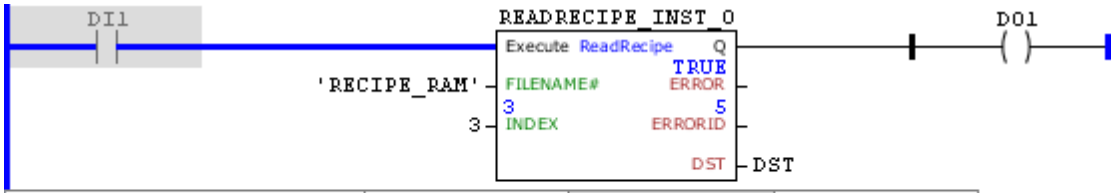
If there is any error in the execution, the Error output is activated and ErrorID displays an error code according to the table below.

Code	Description
1	Incomplete recipe
2	Invalid structure
3	Nonexistent recipe
4	Invalid file
5	Invalid file or nonexistent SD card
6	SD card blocked for writing
7	SD card busy (log or other use)

Block Flowchart

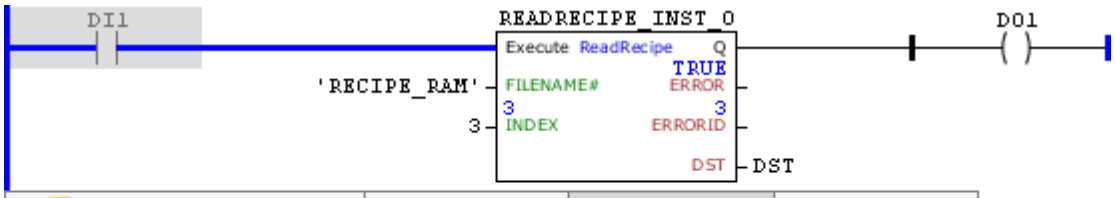

Start
Execute == TRUE?
Yes
No
Rising edge in Execute?
Yes
No
Read recipe data in INDEX of FILENAME
Store data in DST
Error == FALSE?
Yes
No
ERROR = FALSE
ERROR = TRUE
ERRORID = 0
Update ERRORID
Q = TRUE
Q = FALSE
End

Example



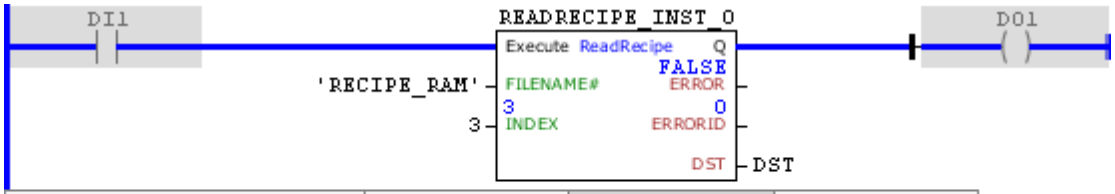
DST		0	
MYSINT	SINT	0	Decimal
MYSTR	STRING	valor_inicial	String
MYINT	INT	0	Decimal

The above example searches the index 3 of the recipe file stored in RAM 'RECIPE_RAM'. The block does not find the specified file, enabling the ERROR output with ERRORID with value 5 and disabling the Q output.



DST		0	
MYSINT	SINT	0	Decimal
MYSTR	STRING	valor_inicial	String
MYINT	INT	0	Decimal

The above example searches the index 3 of the recipe file stored in RAM 'RECIPE_RAM'. The block finds the specified file, but does not find the index 3, enabling the ERROR output with ERRORID with value 3 and disabling the Q output.



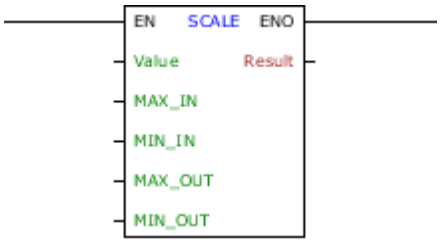
DST		0	
MYSINT	SINT	65	Decimal
MYSTR	STRING	valor_receita	String
MYINT	INT	250	Decimal

The above example searches the index 3 of the recipe file stored in RAM 'RECIPE_RAM'. The block finds the file and the specified index, stores values in DST, disables the ERROR output and enables Q output.

11.8.5.8.11 SCALE

Block that converts a value from a scale to another one.

Ladder Representation



Block Structure

Variable Type	Name	Data type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Input value to be converted
	MAX_IN	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Maximum value of input scale
	MIN_IN	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Minimum value of input scale
	MAX_OUT	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Maximum value of output scale
	MIN_OUT	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Minimum value of output scale
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Output value on new scale

Operation

This block, when it has a TRUE value in EN, by setting the minimum and maximum values of the variable to be converted and the minimum and maximum values of the new scale variable, defined by the user, performs the Scale function for the conversion of the variable according to equation:

$$VAR_{OUT} = a \times (Value - MIN_{in}) + b$$

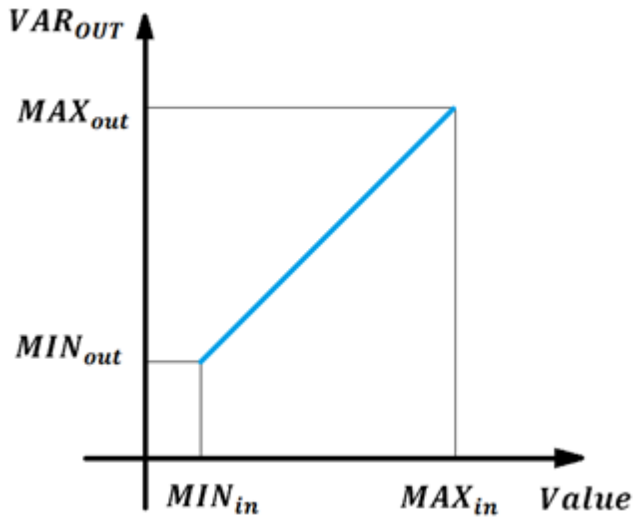
Where:

$$a = \left(\frac{MAX_{out} - MIN_{out}}{MAX_{in} - MIN_{in}}\right)$$

and

$$b = MIN_{out}$$

The graph below represents the straight linearized:



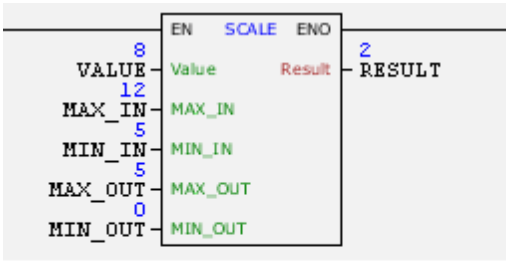
When EN has FALSE value, DST remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

NOTE!
- The value in MAX_{in} must be greater than value in MIN_{in};
- The value in MAX_{out} must be greater than value in MIN_{out};
- Value in **Value** according to: **MIN_{in} = Value = MAX_{in}.**

Block Flowchart

Example



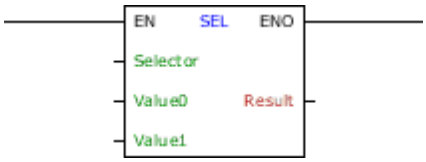
VALUE	INT	0	8
RESULT	INT	0	2

The example above stores the value of the variable VALUE in Result. The block considers the equation described above and ends with success and ENO output is activated.

11.8.5.8.12 SEL

Block that replicates to the output the value of an input variable (Value0 or Value1) according to the Selector selection.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Selector	BOOL	Variable that selects the input
	Value0	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Multiplexed input number 1
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Multiplexed input number 2
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Output value selected

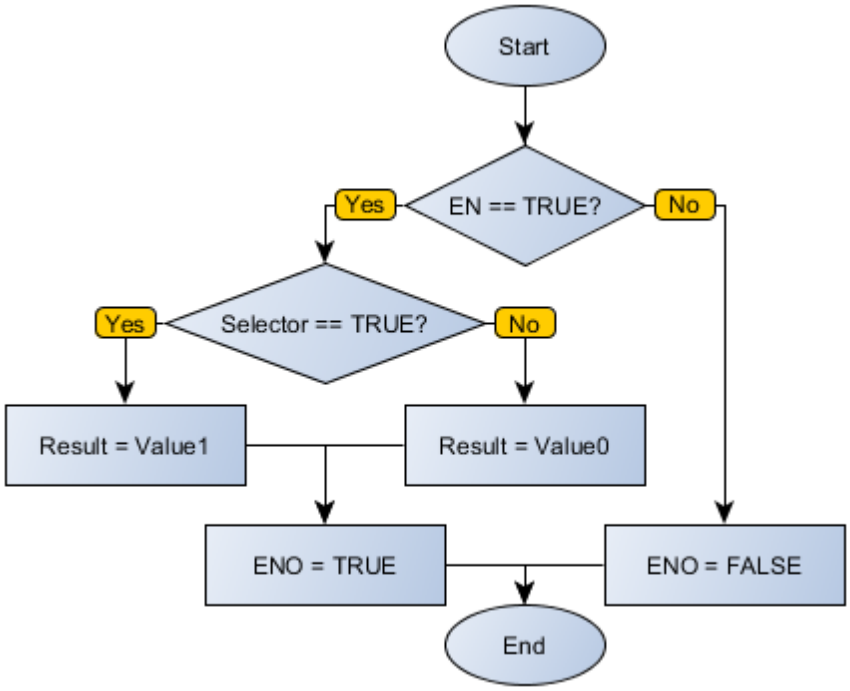
Operation

When this block has a TRUE value in EN, it replicates to the Result variable the Value0 value if selector is FALSE or the Value1 value if Selector is TRUE.

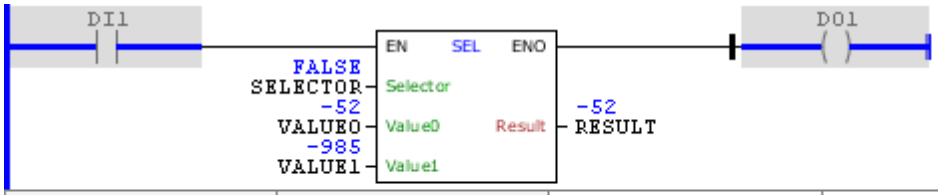
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart

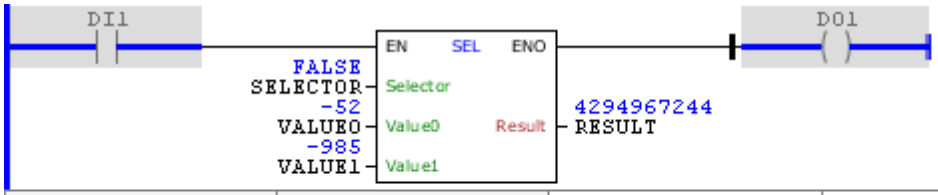


Example



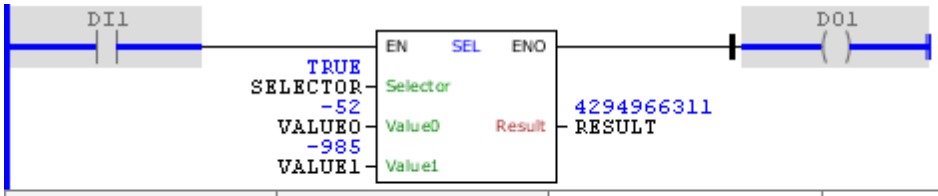
SELECTOR	BOOL	0	Binary
VALUE0	SINT	-52	Decimal
VALUE1	INT	-985	Decimal
RESULT	INT	-52	Decimal

The above example uses the SELECTOR variable as multiplexing channel selector. When it is at FALSE level, the RESULT output gets the value of VALUE0. The block ends successfully and the ENO output is activated.



SELECTOR	BOOL	0	Binary
VALUE0	SINT	-52	Decimal
VALUE1	INT	-985	Decimal
RESULT	DWORD	4294967244	Decimal

The above example uses the SELECTOR variable as multiplexing channel selector. When it is at FALSE level, the RESULT output gets the value of VALUE0. The block ends successfully and the ENO output is activated. Note that the binary pattern has been maintained even though the decimal representation is corrupted, since DWORD does not accept negative values.



SELECTOR	BOOL	1	Binary
VALUE0	SINT	-52	Decimal
VALUE1	INT	-985	Decimal
RESULT	DWORD	4294966311	Decimal

The above example uses the SELECTOR variable as multiplexing channel selector. When it is at TRUE level, the RESULT output gets the value of VALUE1. The block ends successfully and the ENO output is activated. Note that the binary pattern has been maintained even though the decimal representation is corrupted, since DWORD does not accept negative values.

11.8.5.8.13 STORE

Block that performs direct storage of data from a source to a destination.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	SRC	BYTE USINT SINT WORD UINT INT DWORD DINT DINT REAL	Data source
VAR_OUTPUT	ENO	BOOL	End of operation
	DST	BYTE USINT SINT WORD UINT INT DWORD DINT DINT REAL	Data destination

Operation

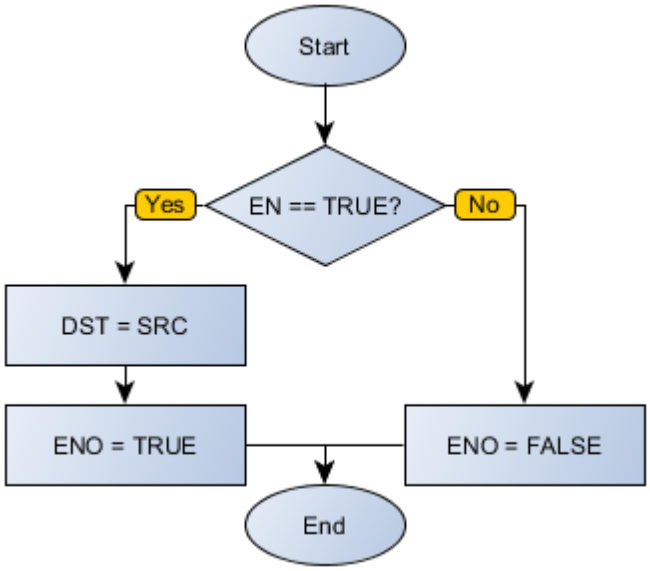
When this block has a TRUE value in EN, it stores the contents from SRC into DST.

NOTE!
SRC and DST must have data types of the same size.

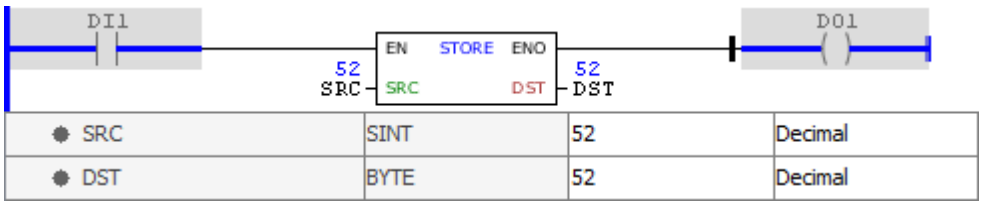
When EN has FALSE value, DST remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

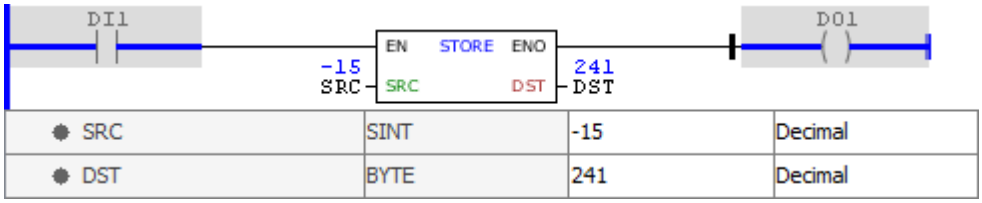
Block Flowchart



Example



The example above stores the value of the variable SRC in DST. The block ends with success and ENO output is activated.



SRC	SINT	-15	Decimal
DST	BYTE	241	Decimal

The example above stores the value of the variable SRC in DST. The block ends with success and ENO output is activated. Note that the binary pattern is maintained between variables of different types.

11.8.5.8.14 SWAP

Block that performs a swap between the odd bytes and consecutive even bytes into Value and sends the value to Result.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	WORD UINT INT DWORD UDINT DINT	Input variable to be swapped
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	WORD UINT INT DWORD UDINT DINT REAL(*)	Output value

Operation

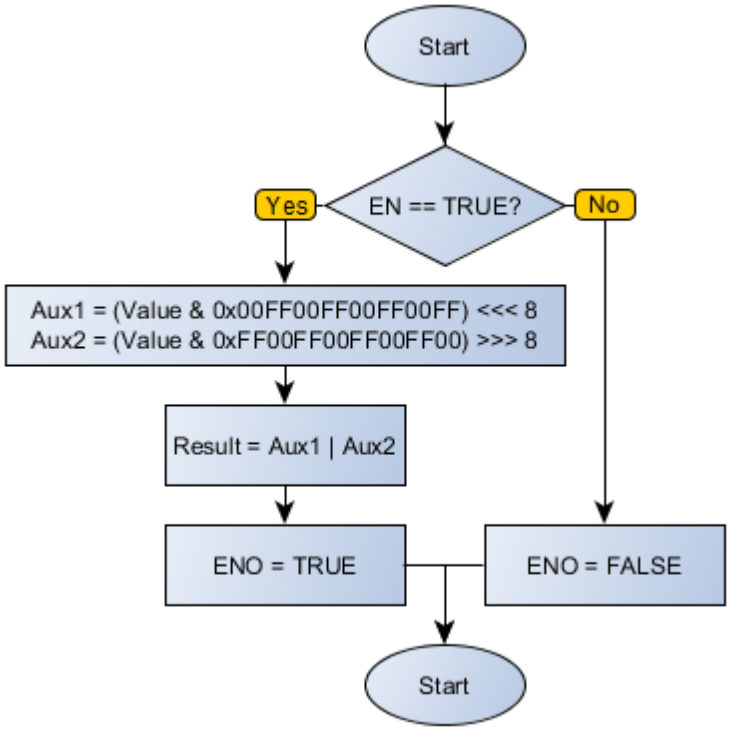
When this block has a TRUE value in EN, it changes the values of the odd bytes (1, 3, 5 and 7) and the consecutive even bytes (2, 4, 6 and 8, respectively) of the Value variable, storing the result in Result.

When EN has FALSE value, Result remains unchanged.

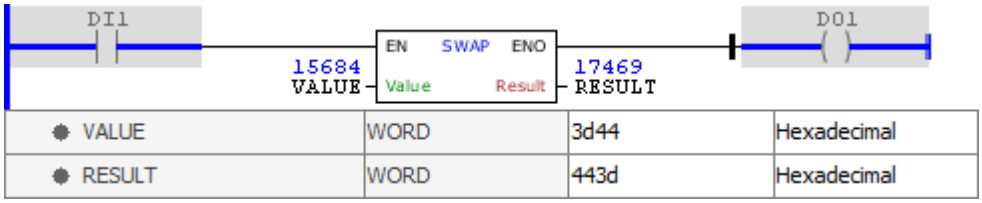
The ENO value forwards to the next Ladder block the EN value after the operation is completed.

NOTA!
Caution when using in Result a variable of REAL type, because the block does not perform type conversion, that is, it only reverses the bytes in memory.

Block Flowchart

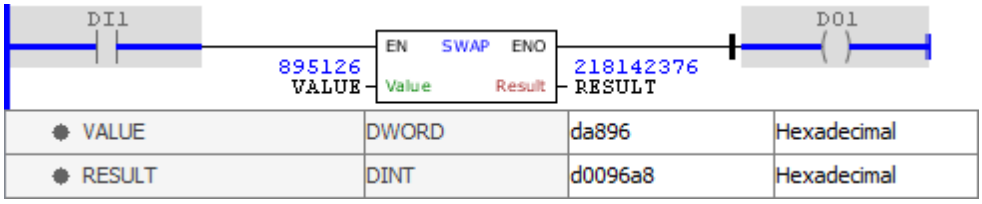


Example



VALUE	WORD	3d44	Hexadecimal
RESULT	WORD	443d	Hexadecimal

The example changes the position of byte 1 value of VALUE (0x44) with byte 2 of VALUE (0x3D), storing the final result (0x44_3D) in RESULT. The block ends with success and ENO output is activated.



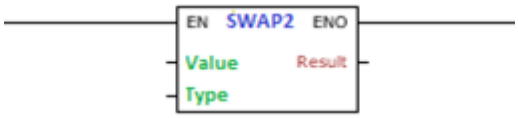
VALUE	DWORD	da896	Hexadecimal
RESULT	DINT	d0096a8	Hexadecimal

The example changes the position of byte 1 value of VALUE (0x96) with byte 2 of VALUE (0xA8) and byte 3 of VALUE (0x0D) with byte 4 of VALUE (0x00), storing the final result (0x0D_00_96_A8) in RESULT. The block ends with success and ENO output is activated.

11.8.5.8.15 SWAP2

Block that rearranges the bytes of a variable.

Ladder Representation



Block Structure

Variable type	Name	Data type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	WORD UINT INT DWORD UDINT DINT	Input variable to be rearranged
	Type	BYTE	Variable that defines the conversion type according to Table 2
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL(*)	Output value

Table 1. Block variables.

Type	WORD UINT INT	ENO	DWORD UDINT DINT	ENO
0	AB->AB*	TRUE	ABCD->ABCD	TRUE
1	AB->BA	TRUE	ABCD->DCBA	TRUE
2	-	FALSE	ABCD->CDAB	TRUE
3	-	FALSE	ABCD->BADC	TRUE
4 ...	-	FALSE	-	FALSE

Table 2. Conversion type (*characters A, B, C and D represents BYTES).

Operation

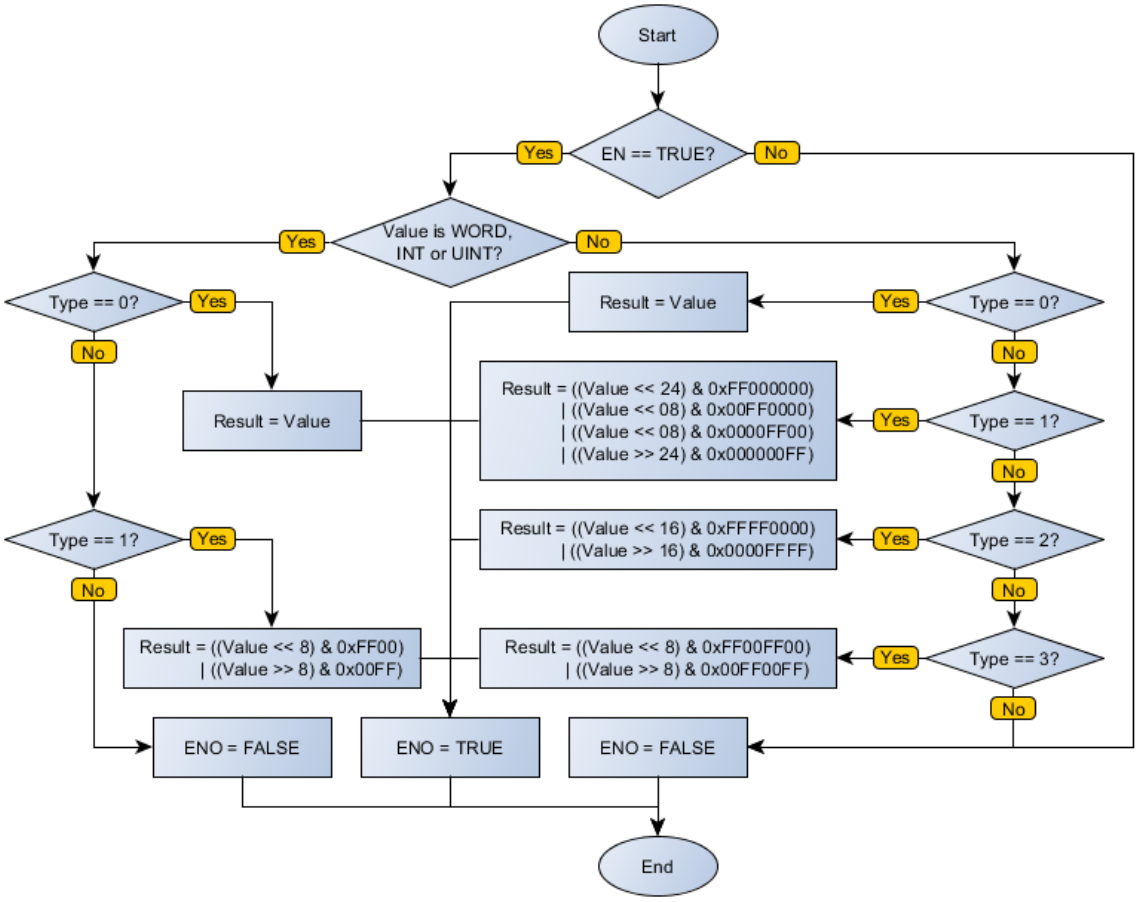
When this block has a TRUE value in EN, it rearranges the bytes from Value variable, storing the result in Result.
Type defines how the bytes will be rearranged, as shown in Table 2.
Note that for 16-bit variables, only options 0 and 1 are valid.
For 32-bit variables the options 0, 1, 2, and 3 are valid.

When EN has FALSE value, Result remains unchanged.

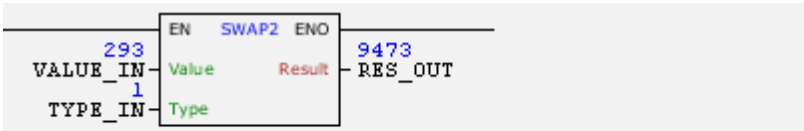
The ENO value forwards to the next Ladder block the EN value after the operation is completed.
Invalid TYPE options assigns FALSE to ENO, and Result value is not changed.

NOTA!
Caution when using in Result a variable of REAL type, because the block does not perform type conversion, that is, it only reverses the bytes in memory.

Block Flowchart



Example



RES_OUT	INT	2501	Hexadecimal	OK
TYPE_IN	BYTE	1	Decimal	OK
VALUE_IN	INT	125	Hexadecimal	OK

The example rearranges the position of value VALUE_IN according to the type set in TYPE_IN = 1 (AB->BA), storing the final result in RESULT. The block ends with success and ENO output is activated.

11.8.5.8.16 WriteRecipe

Block that writes a recipe into a recipe file from a variable.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	Execute	BOOL	Block enabling
	FILENAME#	STRING	Name of the recipe file
	INDEX	WORD UINT	Recipe index to be written
	SRC	STRUCT	Variable where the data is read
VAR_OUTPUT	Q	BOOL	End of operation
	ERROR	BOOL	Error occurrence flag
	ERRORID	BYTE USINT	Identifier of the occurred error
VAR	WRITERECIPE_INST_0	WRITERECIPE	Instance of access to block structure

Operation

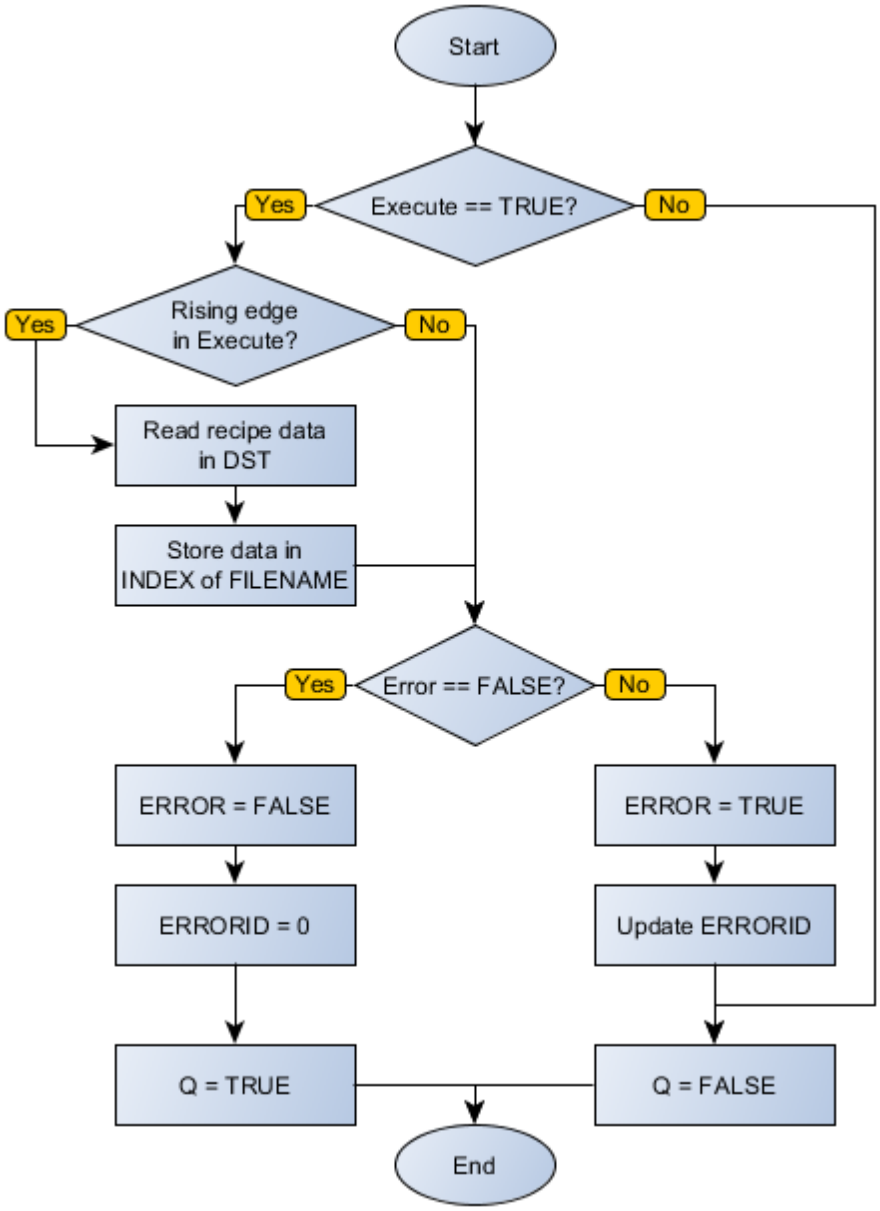
When this block identifies a leading edge in Execute, it gets the data from SRC file and sends them to the recipe file selected by the INDEX index in the FILENAME#. If everything goes successfully, Q receives TRUE and remains so while Execute is TRUE.

NOTE!
Recipes stored in RAM is identified by 'RECIPE_NAME'. Recipes stored on the SD card are identified by 'RECIPE_NAME.CSV'.

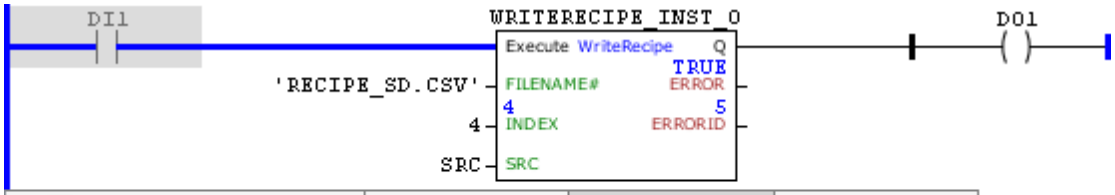
If there is any error in the execution, the Error output is activated and ErrorID displays an error code according to the table below.

Code	Description
1	Incomplete recipe
2	Invalid structure
3	Nonexistent recipe
4	Invalid file
5	Invalid file or nonexistent SD card
6	SD card blocked for writing

Block Flowchart

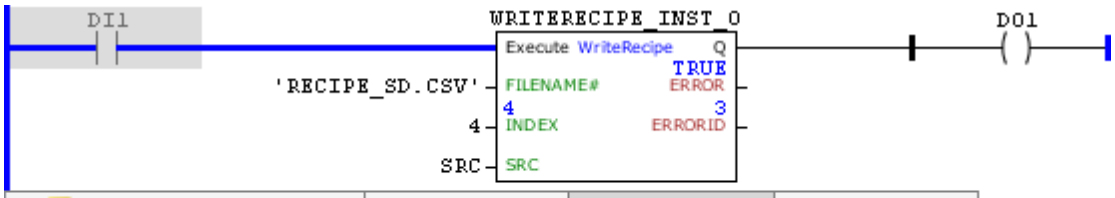

Start
Execute == TRUE?
Yes
No
Rising edge in Execute?
Yes
No
Read recipe data in DST
Store data in INDEX of FILENAME
Error == FALSE?
Yes
No
ERROR = FALSE
ERROR = TRUE
ERRORID = 0
Update ERRORID
Q = TRUE
Q = FALSE
End

Example



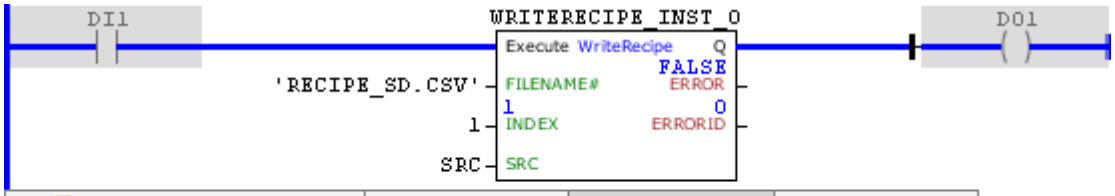
SRC		0	
MYSINT	SINT	0	Decimal
MYSTR	STRING	valor_inicial	String
MYINT	INT	0	Decimal

The above example attempts to write the contents of SRC in the index 4 of the recipes file stored in SD card 'RECIPE_SD.CSV'. The block does not find the specified file, enabling the ERROR output with ERRORID with value 5 and disabling the Q output.



SRC		0	
MYSINT	SINT	0	Decimal
MYSTR	STRING	valor_inicial	String
MYINT	INT	0	Decimal

The above example attempts to write the contents of SRC in the index 4 of the recipes file stored in SD card 'RECIPE_SD.CSV'. The block finds the specified file, but does not find the index 4, enabling the ERROR output with ERRORID with value 3 and disabling the Q output.



SRC		0	
MYSINT	SINT	65	Decimal
MYSTR	STRING	valor_receita	String
MYINT	INT	250	Decimal

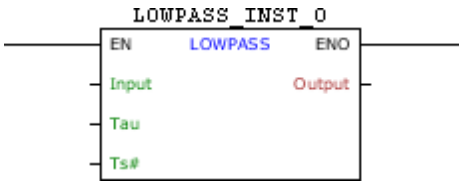
The above example attempts to write the contents of SRC in the index 4 of the recipes file stored in SD card 'RECIPE_SD.CSV'. The block finds the file and the specified index, stores values in the recipe, disables the ERROR output and enables Q output.

11.8.5.9 Filter

11.8.5.9.1 LOWPASS

Block that filters the input using a low pass filter of first order and inserts the result in the output.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Input	REAL	Input signal
	Tau	REAL	Filter time constant
	Ts#	UINT	Sampling time [ms]
VAR_OUTPUT	ENO	BOOL	Output enabling
	Output	REAL	Filter output
VAR	LOWPASS_INST_0	LOWPASS	Instance of access to block structure

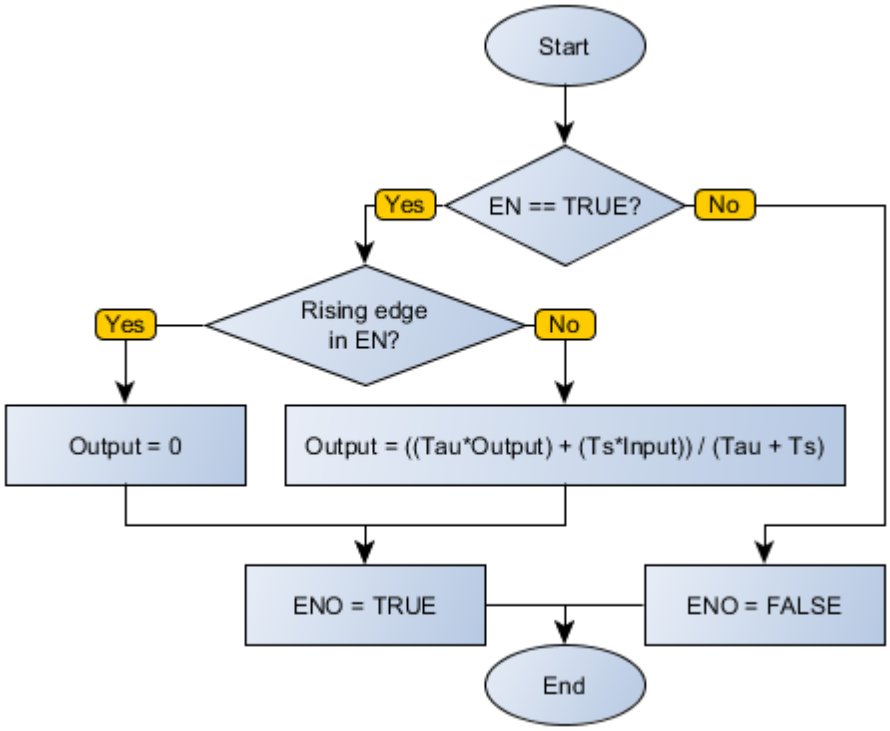
Operation

When this block has a TRUE value in EN, filters the input value of Input using a low pass first order filter described by Tau and Ts#, inserting the result in Output. On the leading edge of EN, Output receives zero.

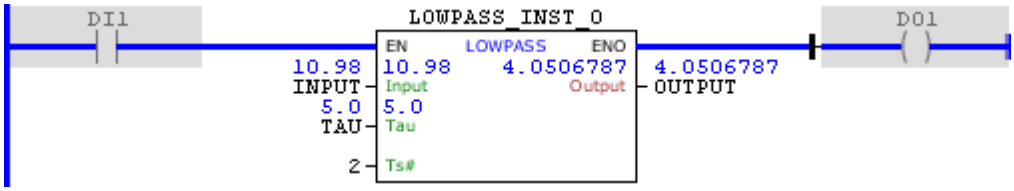
When EN has FALSE value, Output remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

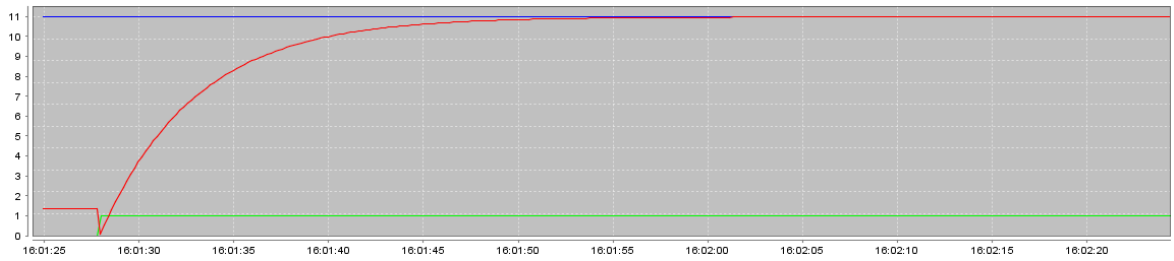
Block Flowchart



Example



The above example causes OUTPUT, by identifying a leading edge in EN, to display a behavior of first order with time constant equal to Tau and the sampling time of 2 ms, in order to achieve the reference set to INPUT. At each calculation completed successfully, the ENO output is activated.

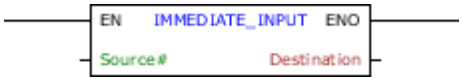


11.8.5.10 Hardware

11.8.5.10.1 IMMEDIATE_INPUT

Block that performs an instantaneous reading of the selected input value, without changing the value of images (GLOBAL_IO variables).

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Source#	BYTE	Inputs to be read (digital or analog)
VAR_OUTPUT	ENO	BOOL	Output enabling
	Destination	WORD INT UINT	Variable mapped w ith the values of the inputs selected

Operation

When this block has a TRUE value in EN, it gets the immediate value of the selected input in Source#. If selected the analog input AI1, its value is passed on to Destination. If the digital input is selected, its bits are concatenated so that DI1 be the least significant bit and DI10 be the most significant bit and the result is sent to Destination.

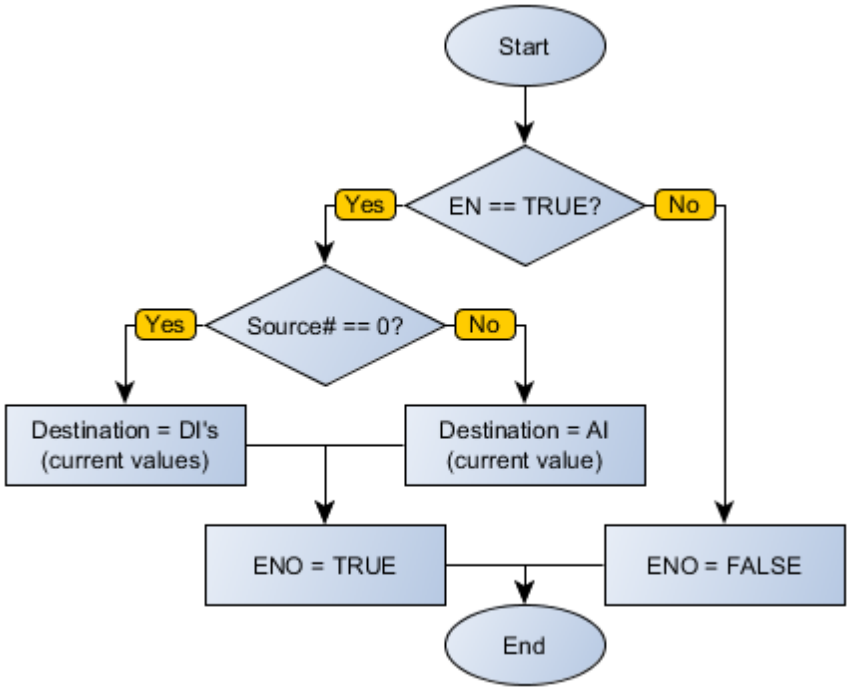
When EN has FALSE value, Destination remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

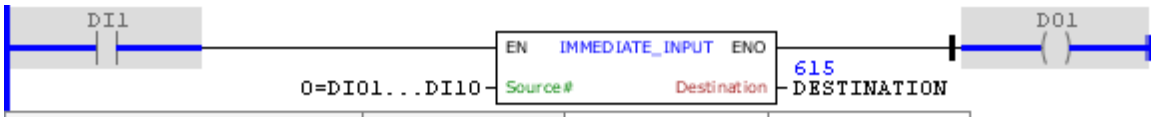
Compatibility

Device	Version
PLC300	1.20 or higher
SCA06	2.00 or higher

Block Flowchart



Example



DESTINATION	WORD	1001100111	Binary
DI1	BOOL	1	Binary
DI2	BOOL	1	Binary
DI3	BOOL	1	Binary
DI4	BOOL	0	Binary
DI5	BOOL	0	Binary
DI6	BOOL	1	Binary
DI7	BOOL	1	Binary
DI8	BOOL	0	Binary
DI9	BOOL	0	Binary
DI10	BOOL	1	Binary

The example above is an immediate reading of the signs of the digital inputs DI1 to DI10 of the PLC300. This reading is then interpreted as a binary sequence with DI1 being the least significant bit and the result is sent to the DESTINATION variable. The block ends with success, ENO output is activated.

11.8.5.10.2 IMMEDIATE_OUTPUT

Block that performs an instantaneous reading of the selected output port, without changing the value of images (GLOBAL_IO variables).

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Source	WORD INT UINT	Variable mapped with the values to be sent to the selected outputs
VAR_OUTPUT	ENO	BOOL	Output enabling
	Destination#	BYTE	Outputs to be written (digital or analog)

Operation

When this block has a TRUE value in EN, it writes immediately in the selected output the value of Source. If selected analog output AO1, the Source value is passed on to it. If the digital outputs are selected, DO1 will receive the zero bit of Source, DO2 bit one, DO3 bit two, and so on.

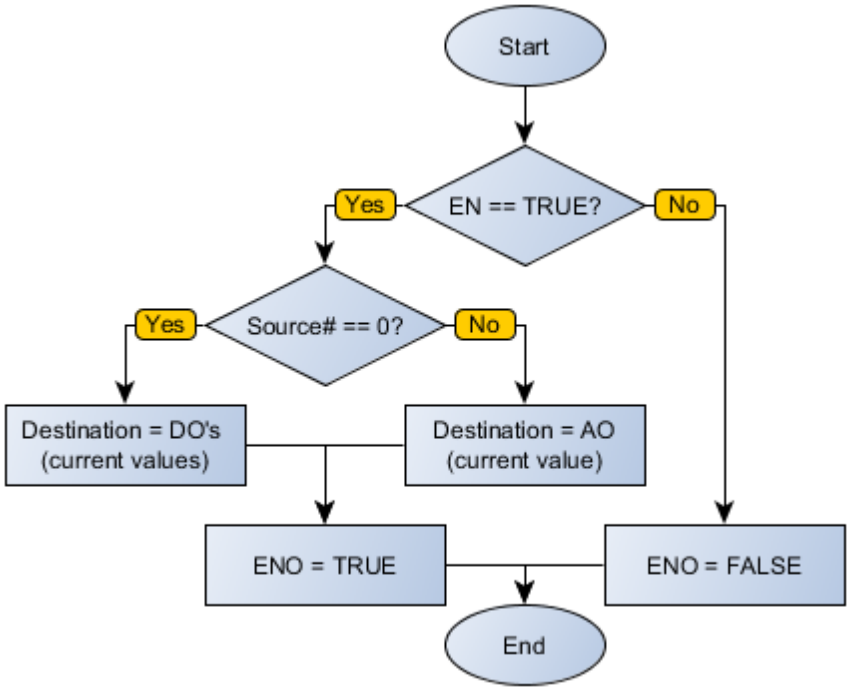
When EN has FALSE value, Destination# remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

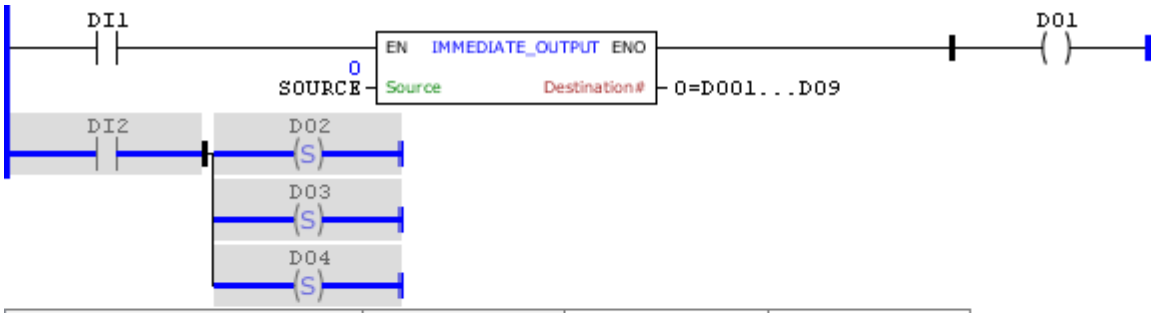
Compatibility

Device	Version
PLC300	1.20 or higher
SCA06	2.00 or higher

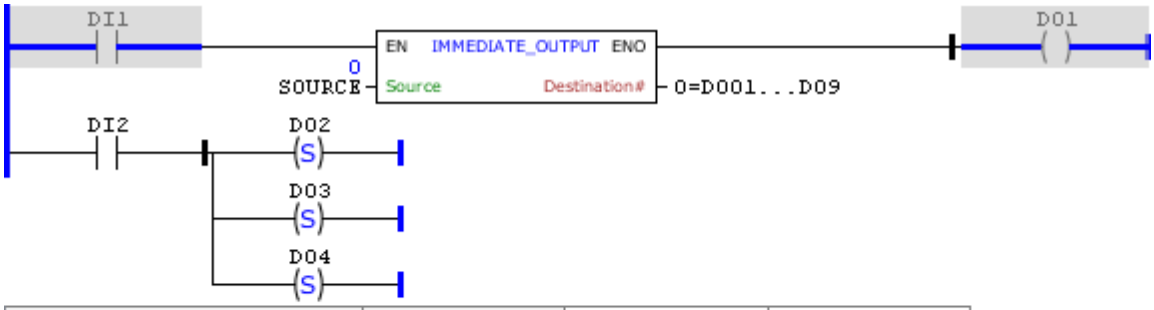
Block Flowchart



Example



SOURCE	WORD	0	Binary
DO1	BOOL	0	Binary
DO2	BOOL	1	Binary
DO3	BOOL	1	Binary
DO4	BOOL	1	Binary
DO5	BOOL	0	Binary
DO6	BOOL	0	Binary
DO7	BOOL	0	Binary
DO8	BOOL	0	Binary
DO9	BOOL	0	Binary



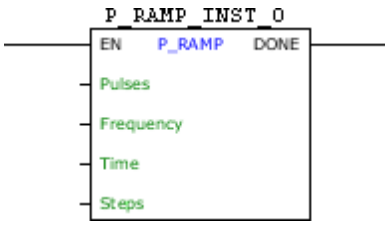
SOURCE	WORD	0	Binary
DO1	BOOL	1	Binary
DO2	BOOL	0	Binary
DO3	BOOL	0	Binary
DO4	BOOL	0	Binary
DO5	BOOL	0	Binary
DO6	BOOL	0	Binary
DO7	BOOL	0	Binary
DO8	BOOL	0	Binary
DO9	BOOL	0	Binary

The above example is for immediate SOURCE written value, interpreted as a binary sequence, the digital outputs DO1 to DO9 of the PLC300 and DO1 receives the value of the least significant bit. The block ends with success, ENO output is activated. Note that the immediate writing does not prevail over direct coil DO1 or over enabling coils in DO2, DO3 and DO4.

11.8.5.10.3 P_RAMP

Block that generates a PWM signal with a certain numer of pulses at the digital output DO9, respecting a ramp of rise and fall in frequency..

Ladder Representation



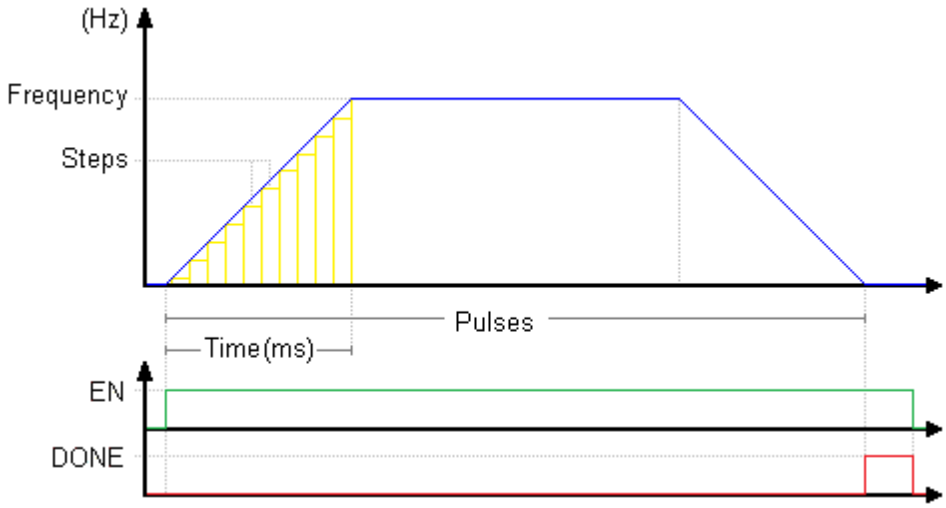
Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Pulses	DWORD UDINT	Total number of pulses of the block execution
	Frequency	DWORD UDINT	Maximum frequency to be reached
	Time	DWORD UDINT	Ramp time [ms]
	Steps	BYTE USINT	Steps to increase frequency
VAR_OUTPUT	DONE	BOOL	Output enabling
VAR	P_RAMP_INST_0	P_RAMP	Instance of access to block structure

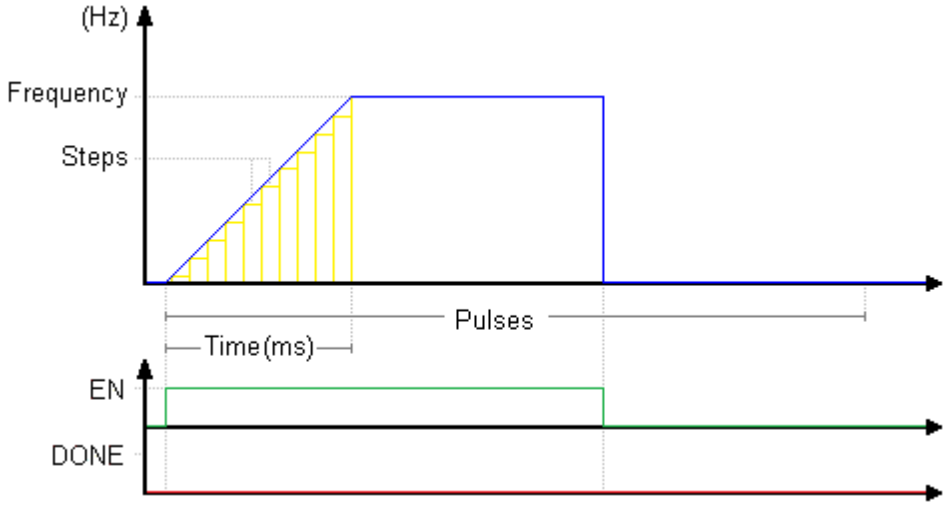
Operation

When the EN input is TRUE, the block generates a PWM signal at output DO9. Initially, it generates a rise ramp, ranging from zero frequency to Frquency value (between 0 and 200 KHz), based on the Time and Steps settings. At the appropriate time, the block generates a descent ramp, with the same profile as the rise ramp, until the frequency is zero and the Pulses value is reached.

When the Pulses value is reached, the DONE output goes to TRUE.



If the EN input goes to FALSE before reaching the Pulses number, the PWM generation stops immediately.



The duty cycle of the signal remains constant at 0.5 throught the block execution.

For wich the frequency ramp is generated, it is necessary thet the number of Steps is less or equal than to a quarter of the Frequency value, that is:
Steps = (1/4) . Frequency

If this limitation is not respected, the frequency ramp will not be generated. It means that PWM signal goes directly to the Frequency value.

Example: If the Frequency value is 100 Hz, for the frequency ramp to be generated, the number of Steps must be less or equal than 25.

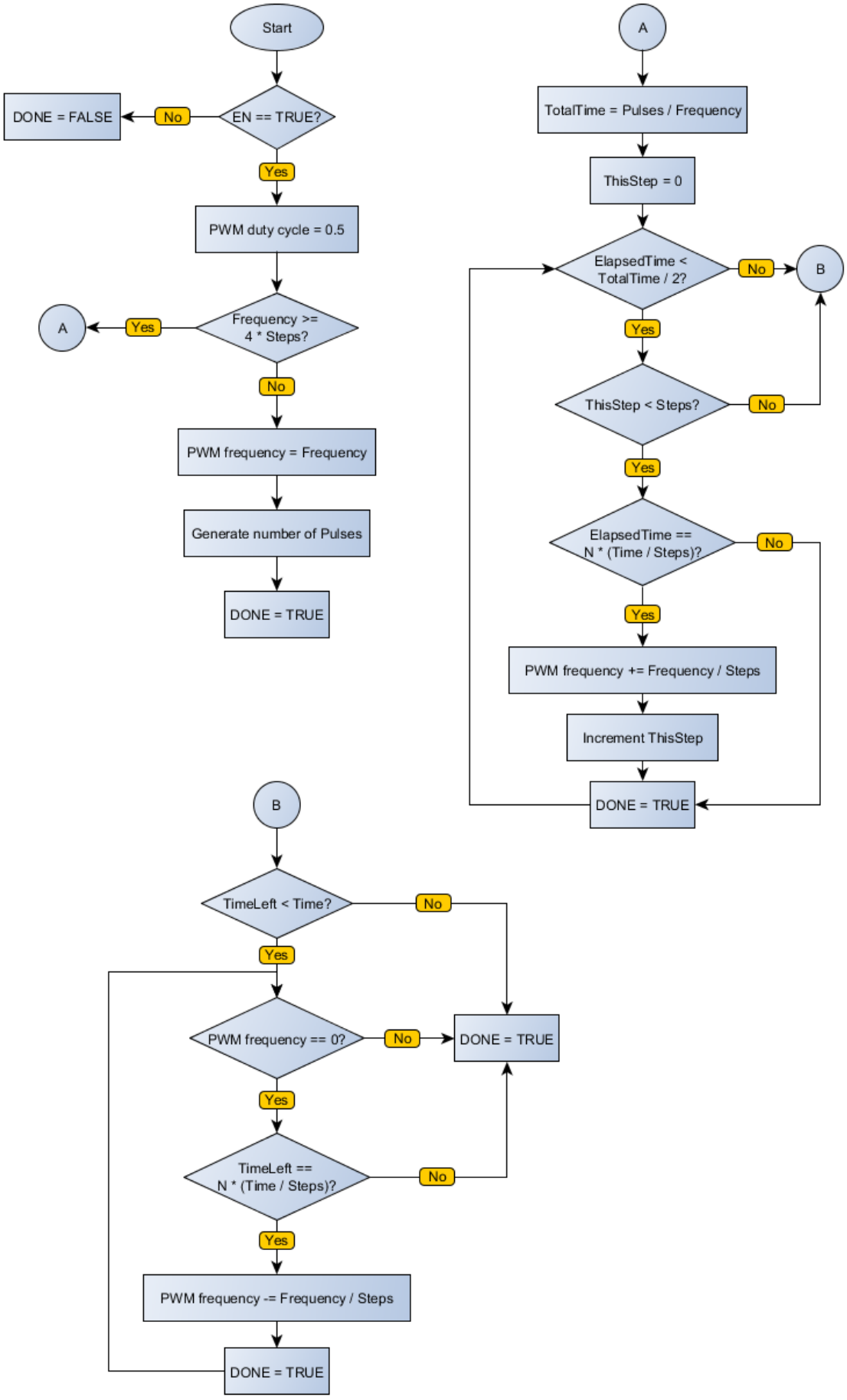
NOTE!
Having other PWM blocks running, whichever is called first will have priority.

NOTE!
This block has priority over any coil that is writing data in DO9.

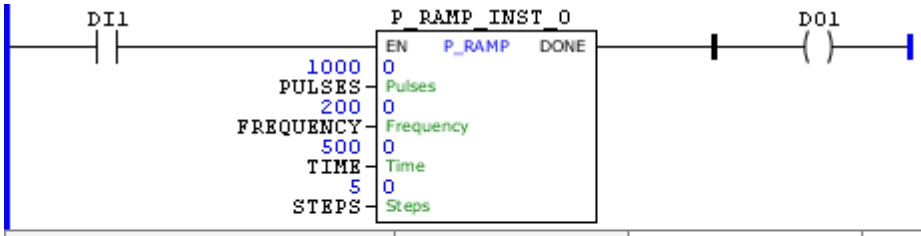
Compatibility

Device	Version
PLC300	2.00 or higher

Block Flowchart

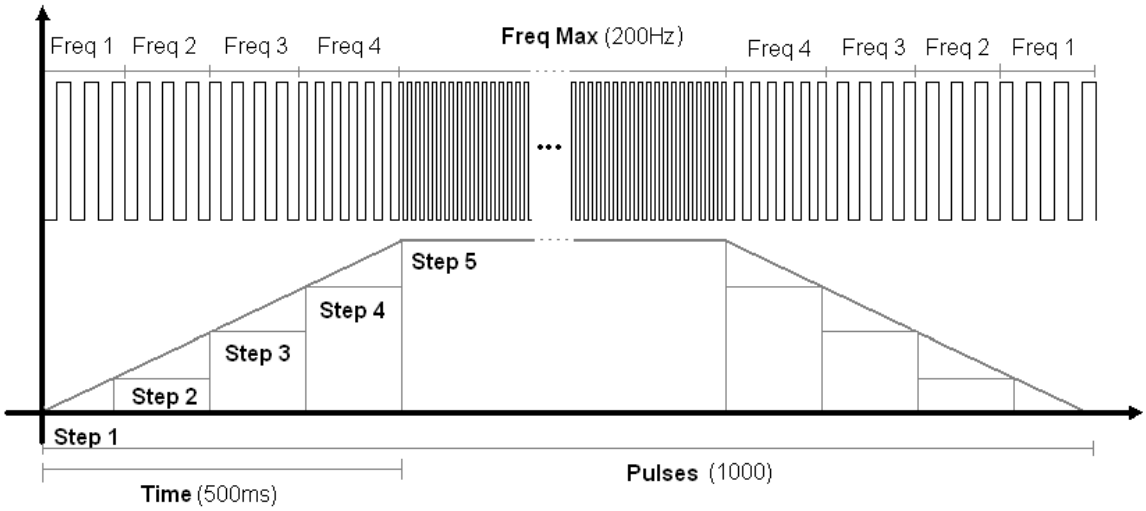

Start
EN == TRUE?
No
DONE = FALSE
Yes
PWM duty cycle = 0.5
Frequency >= 4 * Steps?
Yes
A
No
PWM frequency = Frequency
Generate number of Pulses
DONE = TRUE
A
TotalTime = Pulses / Frequency
ThisStep = 0
ElapsedTime < TotalTime / 2?
No
B
Yes
ThisStep < Steps?
No
Yes
ElapsedTime == N * (Time / Steps)?
No
Yes
PWM frequency += Frequency / Steps
Increment ThisStep
DONE = TRUE
B
TimeLeft < Time?
No
Yes
PWM frequency == 0?
No
DONE = TRUE
Yes
TimeLeft == N * (Time / Steps)?
No
Yes
PWM frequency -= Frequency / Steps
DONE = TRUE

Example



PULSES	DWORD	1000	Decimal
FREQUENCY	DWORD	200	Decimal
TIME	DWORD	500	Decimal
STEPS	BYTE	5	Decimal

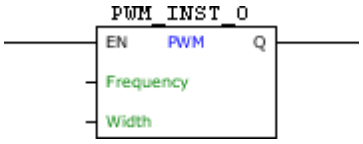
The above example enables a PWM signal in DO9 output with initial frequency 0 Hz and duty cycle of 50%. The rise time of the ramp frequency is 500 ms, and the ramp has 5 frequency steps (0, 50, 100, 150 and 200 Hz). In total executing the block, 1000 pulses will be sent to the output. The block ends with success, Q output is activated.



11.8.5.10.4 PWM

Block that inserts a PWM signal on digital output DO9.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Frequency	DWORD UDINT	Frequency of the PWM
	Width	WORD	Pulse Width
VAR_OUTPUT	Q	BOOL	Output enabling
VAR	PWM_INST_0	PWM	Instance of access to block structure

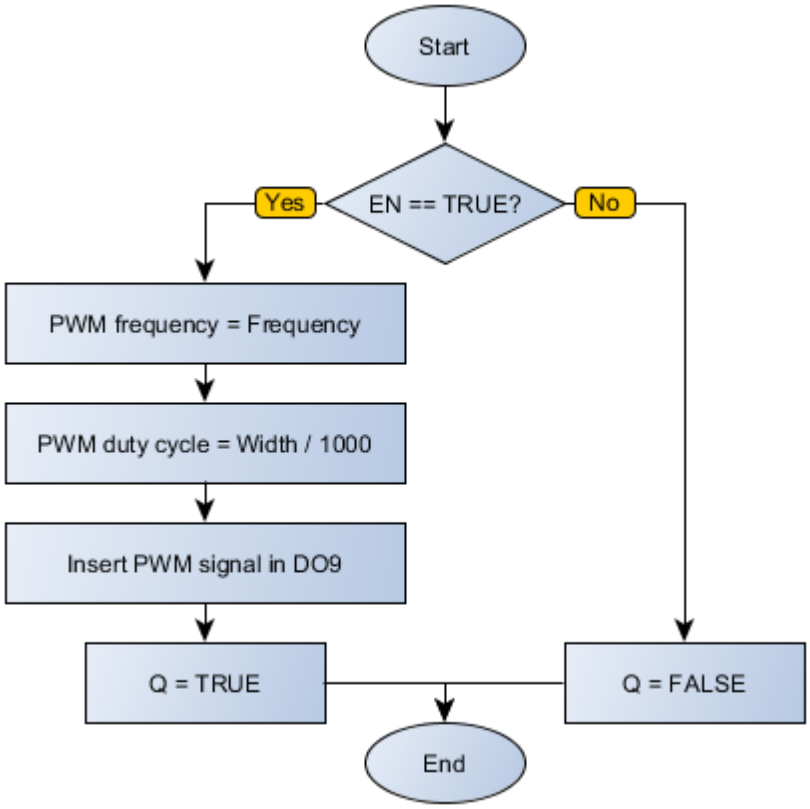
Operation

When this block has a TRUE value in EN, it inserts into DO9 a PWM signal with a given frequency in Frequency (between 0 and 300 kHz) and pulse width determined by Width (between 0 and 1000, where 1000 would be 100% active cycle).

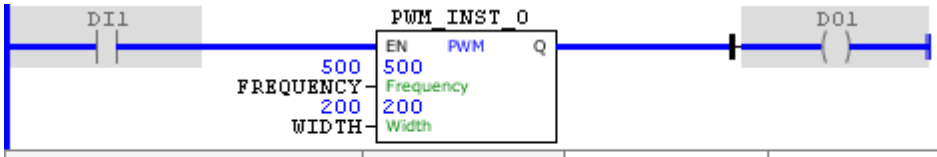
NOTE!
This block has priority over any coil that is writing data in DO9.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



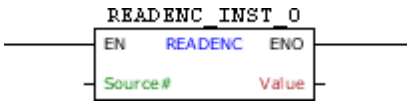
FREQUENCY	UDINT	500	Decimal
WIDTH	UINT	200	Decimal

The above example enables a PWM signal in DO9 output with frequency 500 Hz and duty cycle of 20%. The block ends with success, Q output is activated.

11.8.5.10.5 READENC

Block that continuously reads the pulse value of an encoder according to a type of reading chosen.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Source#	BYTE	Counting source
VAR_OUTPUT	ENO	BOOL	Output enabling
	Value	DINT DWORD UDINT	Counting value
VAR	READENC_INST_0	READENC	Instance of access to block structure

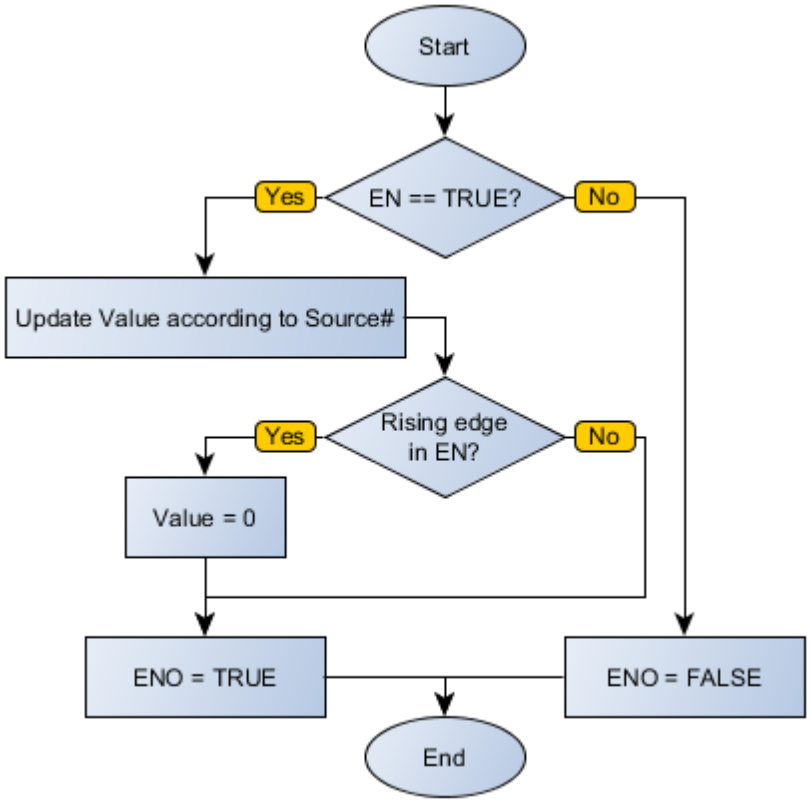
Operation

When this block has a TRUE value in EN, it gets the number of pulses counted in the encoder, as selected in Source #, and transfers them to Value. At each leading edge, Value is reset to zero.

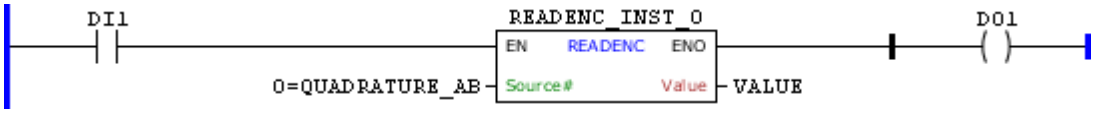
When EN has FALSE value, Value remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example

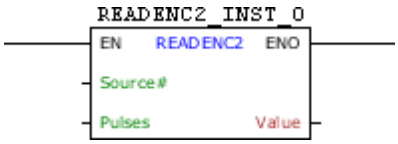


The above example, when identifying a leading edge on DI1, starts counting the number of pulses in the encoder connected in quadrature conformation AB, storing the value in VALUE. The block ends with success, ENO output is activated.

11.8.5.10.6 READENC2

Block that reads the pulse value of an encoder according to a type of reading chosen, interpreting it as a fraction of revolutions.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Source#	BYTE	Counting source
	Pulses	DWORD UDINT	Maximum number of pulses in a turn
VAR_OUTPUT	ENO	BOOL	Output enabling
	Value	DINT DWORD UDINT	Counting value
VAR	READENC2_INST_0	READENC2	Instance of access to block structure

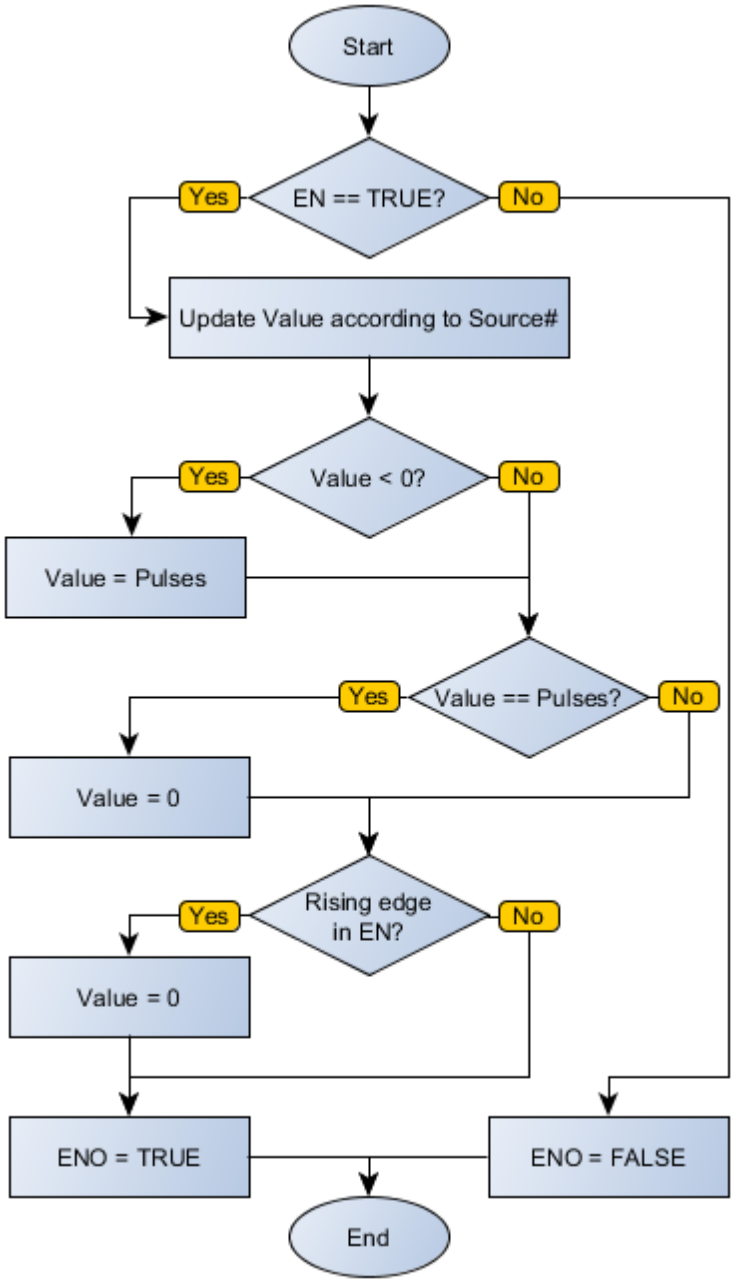
Operation

When this block has a TRUE value in EN, it gets the number of pulses counted in the encoder, as selected in Source #, and transfers them to Value. At each leading edge, Value is reset to zero. When Value increases and reaches the value of Pulses, Value receives zero. When Value decrements and tends to reach a value of -1, Value receives the value in Pulses. This way, Value always displays the result in fraction of revolutions.

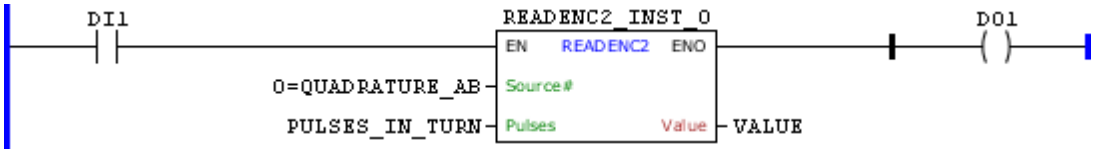
When EN has FALSE value, Value remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



The above example, when identifying a leading edge on DI1, starts counting the number of pulses in the encoder connected in quadrature conformation AB. PULSES_IN_TURN is the value of pulses to complete a turn. If the value read is higher, the counting restarts from zero. If the value read is

negative, counting restarts from PULSES_IN_TURN. The resulting value is stored in VALUE. The block ends with success, ENO output is activated.

11.8.5.10.7 READENC3

Block that reads the value of pulses from an encoder according to a chosen type of reading, calculating differentials between calls to the block.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Source#	BYTE	Counting source
VAR_OUTPUT	ENO	BOOL	Output enabling
	Value	DINT DWORD UDINT	Counting value
	Delta	DINT DWORD UDINT	Difference betw een the previous counting and current counting

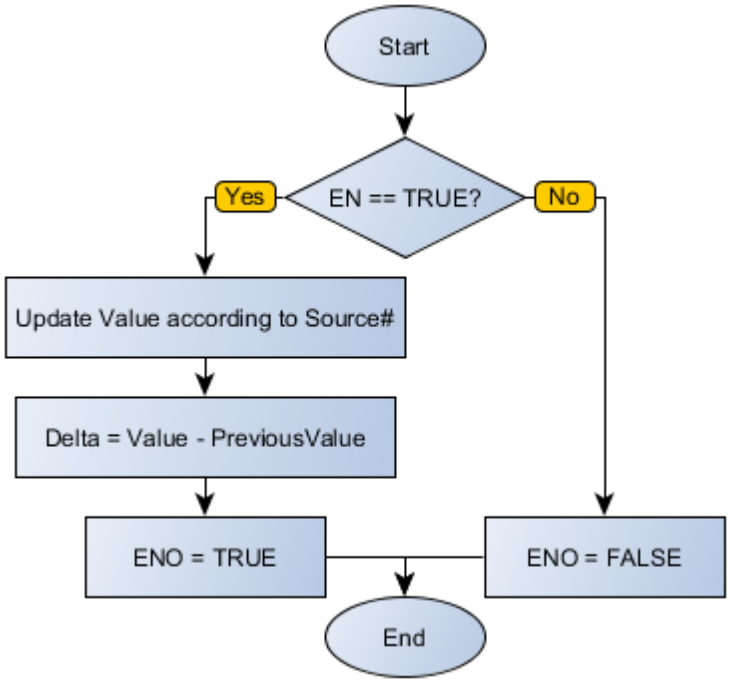
Operation

When this block has a TRUE value in EN, it gets the number of pulses counted in the encoder since energizing of the PLC300, as selected in Source#, and transfers them to Value. The value of Delta is the difference between the pulses counted in this run and the previous run.

When EN has FALSE value, Value and Delta remain unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



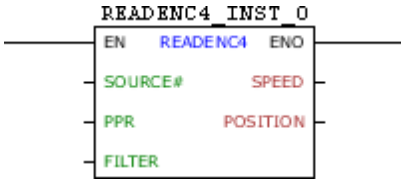
Example

The example above when identifying a leading edge on DI1, gets the number of pulses on input A since the energizing of the equipment, storing the value in VALUE. The difference between the current value and the value previously read is stored in DIFFERENCE. The block ends with success, ENO output is activated.

11.8.5.10.8 READENC4

Block that reads the value of pulses from an encoder according to a chosen type of reading, displaying to the user his current position in revolutions and the speed in RPM.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	SOURCE#	BYTE	Counting source
	PPR	WORD	Number of pulses corresponding to a turn
	FILTER	REAL	Time constant of the input filter
VAR_OUTPUT	ENO	BOOL	Output enabling
	SPEED	REAL	Speed in RPM
	POSITION	REAL	Current position of the encoder, in revolutions
VAR	READENC4_INST_0	READENC4	Instance of access to block structure

Operation

When this block has a TRUE value in EN, it gets the number of encoder pulses counted since energization of the PLC300 as selected on SOURCE #, and calculates the number of absolute revolutions through the PPR argument, inserting the result in POSITION. The value of SPEED is calculated by means of an internal time base of the block. Further, the block allows filtering of the signal with an output filter determined by FILTER.

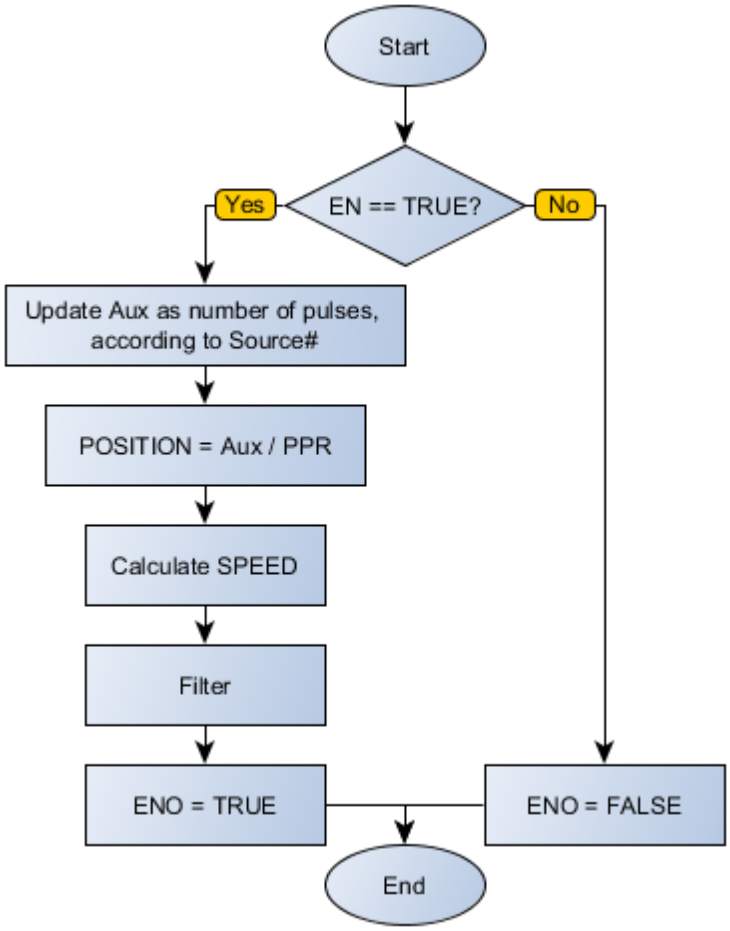
When EN has FALSE value, POSITION and SPEED remain unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

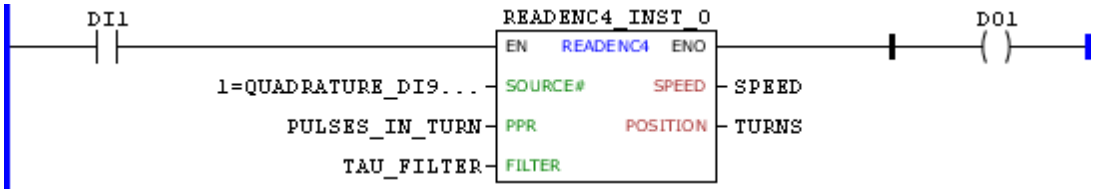
Compatibility

Device	Version
PLC300	2.10 or higher

Block Flowchart



Example



The example above when identifying a leading edge on DI1, gets the number of pulses of the encoder connected in quadrature conformation in the fast inputs since energization equipment. PULSES_IN_TURN is the value of pulses to complete a turn. TURNS stores the value of the pulses interpreted as revolutions, according PULSES_IN_TURN, and SPEED stores the current speed of the encoder. The block ends with success, ENO output is activated.

11.8.5.11 Logic

11.8.5.11.1 Logic Bit

11.8.5.11.1.1 RESETBIT

Logical block used to perform reset of a specific bit in a field.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_IN_OUT	Data	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable whose bit will be changed
VAR_INPUT	EN	BOOL	Block enabling
	Position	BYTE USINT	Position of the bit that will be changed
VAR_OUTPUT	DONE	BOOL	Operation successful

Operation

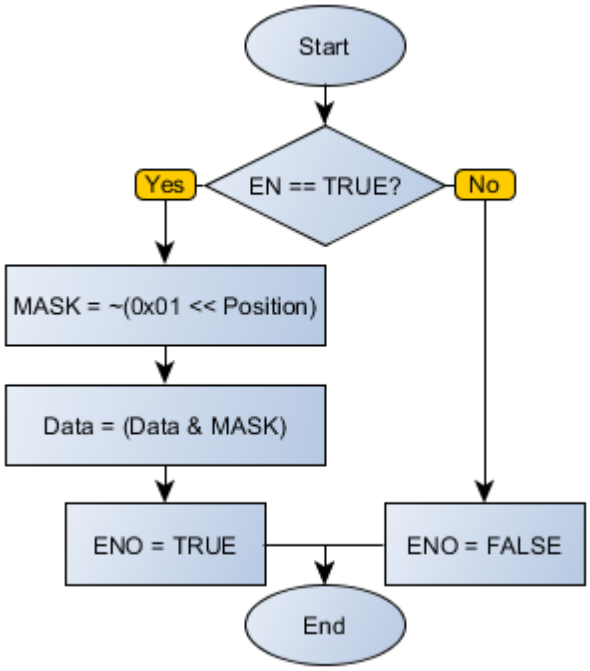
This block when it has a TRUE value in EN, resets the bit indicated in Position in the Data variable that is forwarded to the output already with its updated value.

When EN has FALSE value, Data remains unchanged.

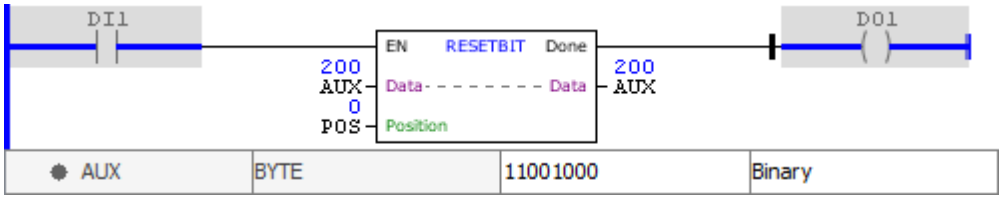
The DONE variable receives the same EN value, except when there is an error in the reset of the bit, then getting a FALSE value.

NOTE!
It is important to notice that Position is within the range of values of bits corresponding to variable type in Data. For example: if Data is a BYTE, it has 8 bits, and Position must contain a value between 0 and 7.

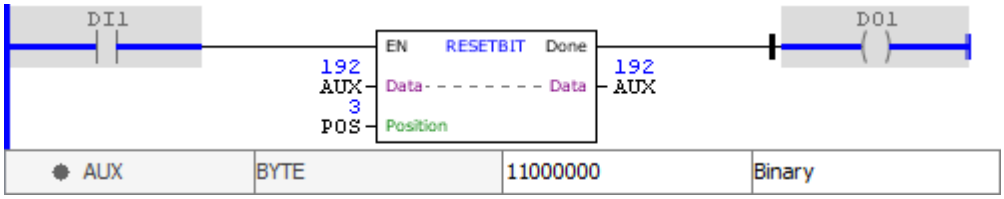
Block Flowchart



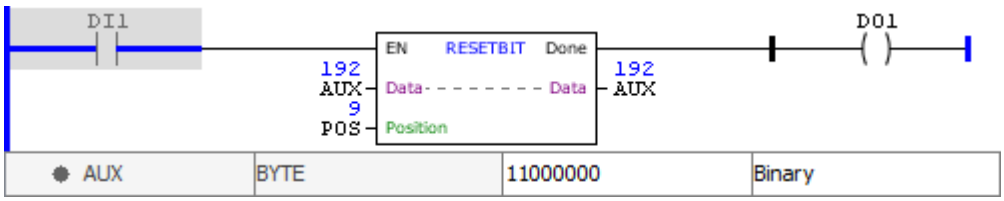
Example



The example above resets the bit of AUX zero position, whose initial value is 200 (1100 1000, in binary). Since this bit already had FALSE value, nothing has changed.



The example above resets the bit in position three of AUX by changing its binary value and, therefore, its decimal representation.



The example above resets the bit in position nine of AUX. Since AUX is a variable BYTE type, it has

only eight bits. Thus, the example above creates a runtime error in the block and therefore the output is not enabled.

11.8.5.11.1.2 SETBIT

Logical block used to perform the set of a specific bit in a field.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_IN_OUT	Data	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable w hose bit w ill be changed
VAR_INPUT	EN	BOOL	Block enabling
	Position	BYTE USINT	Position of the bit that w ill be changed
VAR_OUTPUT	DONE	BOOL	Operation successful

Operation

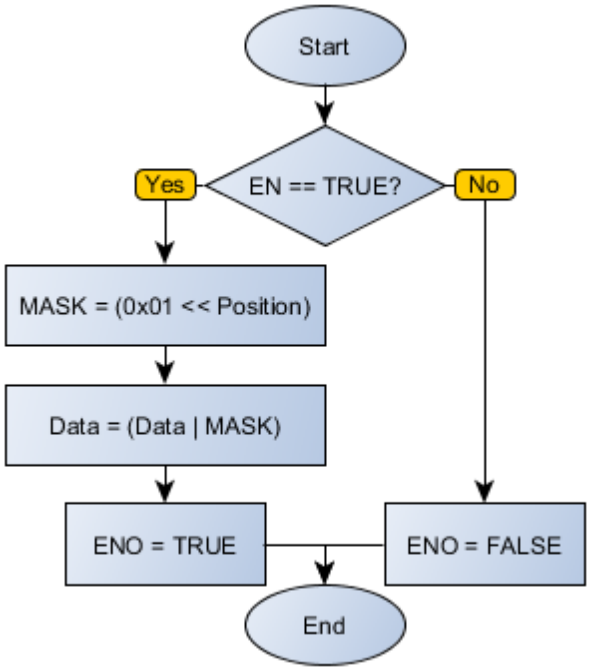
This block when it has a TRUE value in EN, sets the bit indicated in Position in the Data variable that is forwarded to the output already with its updated value.

When EN has FALSE value, Data remains unchanged.

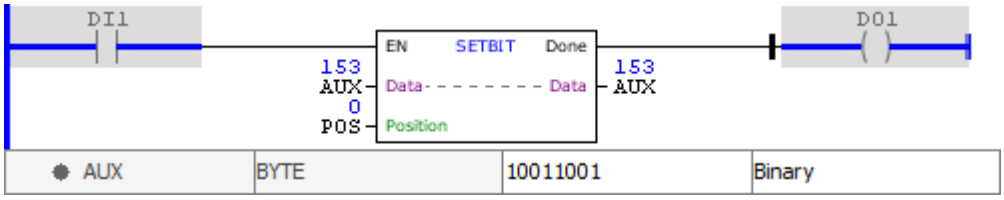
The DONE variable receives the same EN value, except when there is an error in the set of the bit, then getting a FALSE value.

NOTE!
It is important to notice that Position is within the range of values of bits corresponding to variable type in Data. For example: if Data is a BYTE, it has 8 bits, and Position must contain a value between 0 and 7.

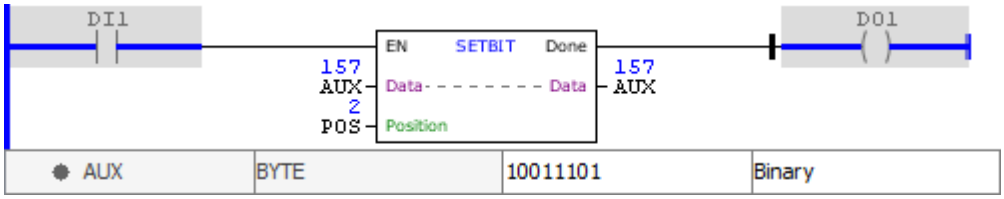
Block Flowchart



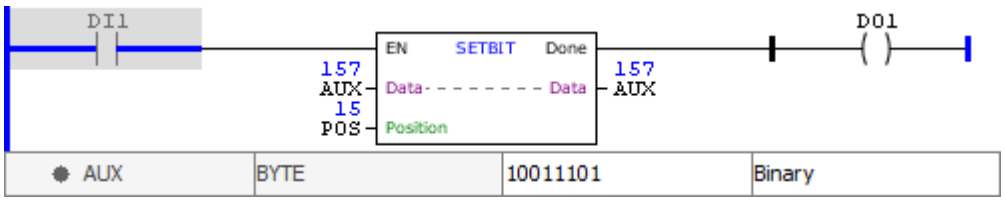
Example



The example above sets the bit of AUX zero position, whose initial value is 153 (1001 1001, in binary). Since this bit already had TRUE value, nothing has changed.



The example above sets the bit in position three of AUX by changing its binary value and, therefore, its decimal representation.



The example above sets the bit in position fifteen of AUX. Since AUX is a variable BYTE type, it has

only eight bits. Thus, the example above creates a runtime error in the block and therefore the output is not enabled.

11.8.5.11.1.3 TESTBIT

Logical block that revolutions the value of a specific bit in a field.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	Data	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable w hose bit w ill be tested
	EN	BOOL	Block enabling
	Position	BYTE USINT	Position of the bit that w ill be changed
VAR_OUTPUT	Q	BOOL	Value of the tested bit

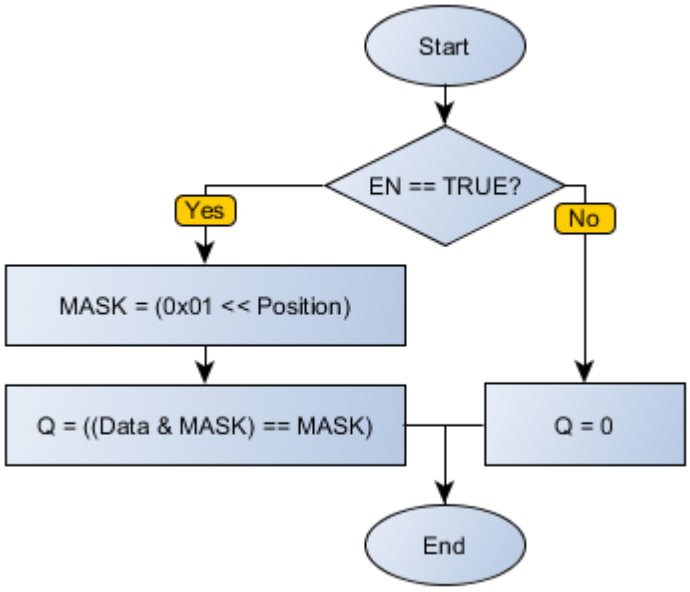
Operation

This block when it has a TRUE value in EN, sends to the output Q the bit value indicated in Position in the Data variable.

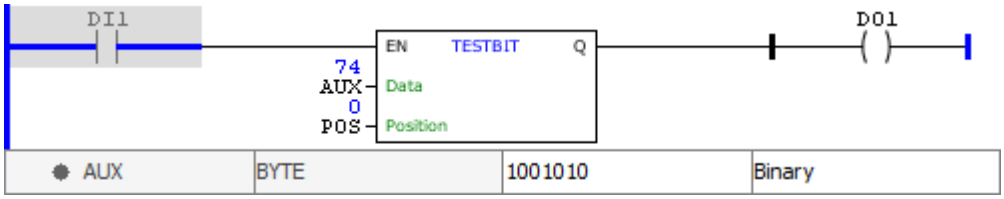
When EN has FALSE value, Q also receives FALSE.

NOTE!
It is important to notice that Position is within the range of values of bits corresponding to variable type in Data. For example: if Data is a BYTE, it has 8 bits, and Position must contain a value between 0 and 7.

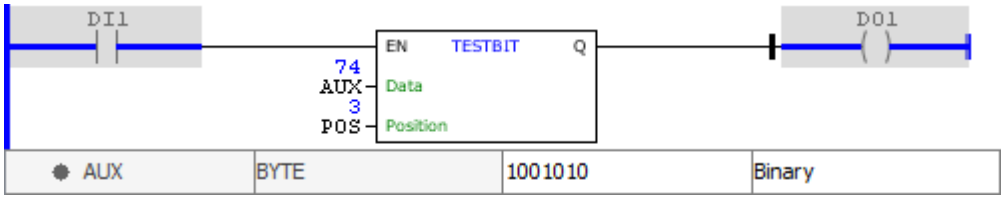
Block Flowchart



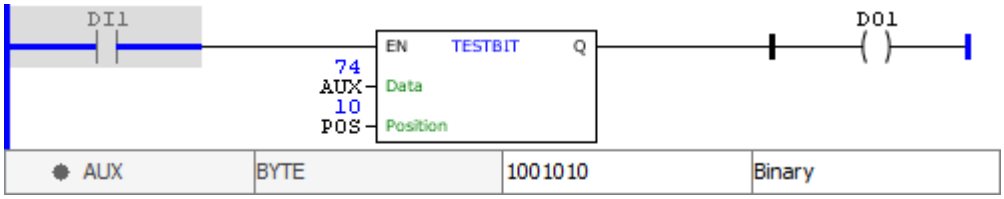
Example



The example above sets the bit value of zero position of AUX, whose initial value is 74 (0100 1010 in binary) to the output Q. Since this bit has value 0, the output is disabled.



The example above sets the value of the bit of position three of AUX to the output Q. Since this bit has value 1, the output is enabled.



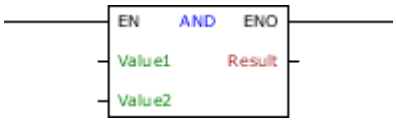
The example above sets the bit value of position ten of AUX to output Q. Since AUX is a variable of BYTE type, it has only eight bits. Thus, the example above creates a runtime error in the block and therefore the output is disabled.

11.8.5.11.2 Logic Boolean

11.8.5.11.2.1 AND

Logical block that performs an boolean "and" operation between two variables, storing the result in a third one.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable for the operation
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable for the operation
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable that stores the result of the operation

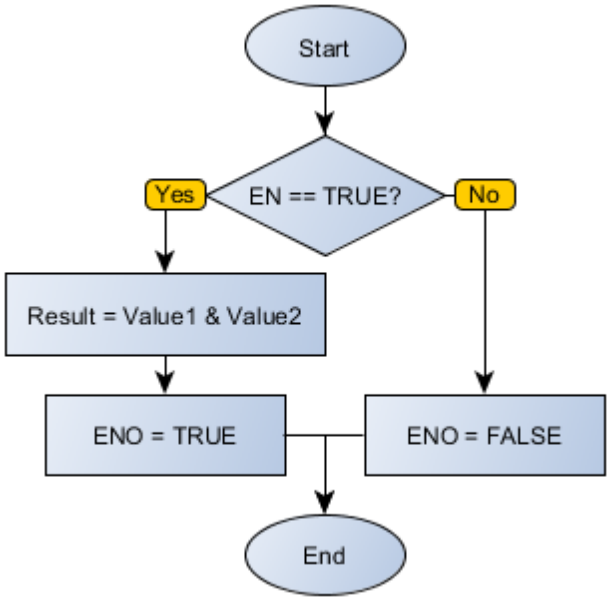
Operation

When this block has a TRUE value in EN, it sends to the Result output the “and” Boolean operation of input variables Value1 and Value2.

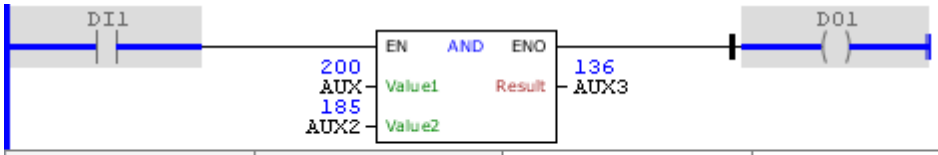
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



AUX	BYTE	11001000	Binary
AUX2	BYTE	10111001	Binary
AUX3	BYTE	10001000	Binary

The example above performs an "and" Boolean operation between AUX and AUX2, storing the result in AUX3.

11.8.5.11.2.2 NOT

Block that performs a logical operation of boolean "not" in a variable, storing the result in another.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Reference variable for the operation
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable that stores the result of the operation

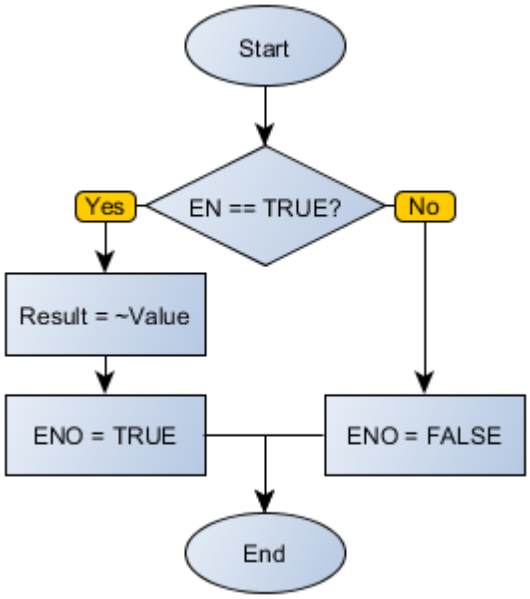
Operation

When this block has a TRUE value in EN, it sends to the Result output the denied Boolean value of the Value input variable.

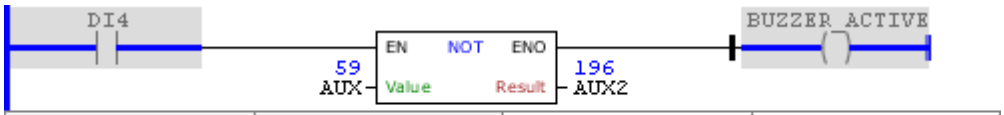
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



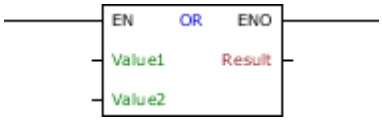
AUX	BYTE	111011	Binary
AUX2	BYTE	11000100	Binary

The example above performs a boolean "not" operation in AUX, storing the result in AUX2.

11.8.5.11.2.3 OR

Logical block that performs an Boolean "or" operation between two variables, storing the result in a third one.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable for the operation
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable for the operation
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable that stores the result of the operation

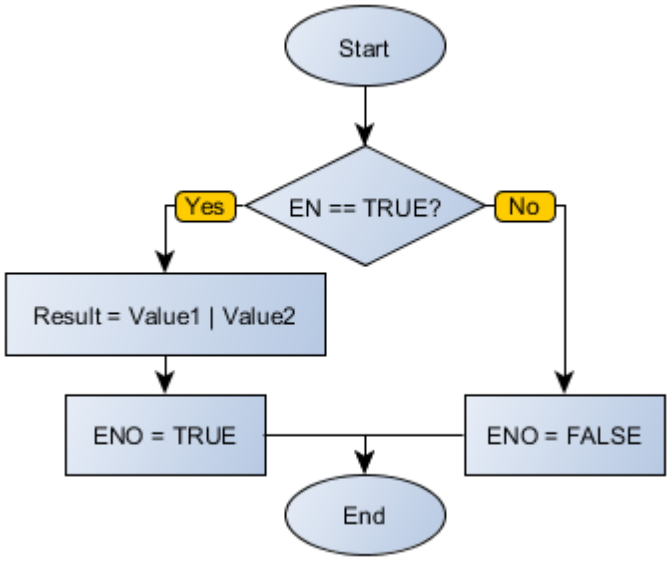
Operation

When this block has a TRUE value in EN, it sends to the Result output the “or” Boolean operation of input variables Value1 and Value2.

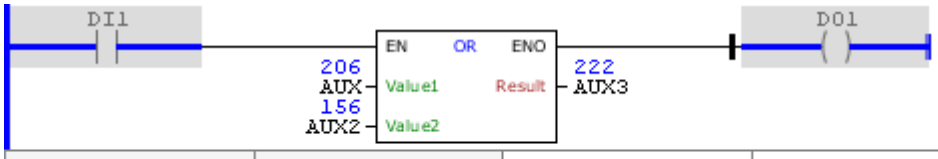
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



AUX	BYTE	11001110	Binary
AUX2	BYTE	10011100	Binary
AUX3	BYTE	11011110	Binary

The example above performs an "or" Boolean operation between AUX and AUX2, storing the result in AUX3.

11.8.5.11.2.4 XNOR

Logical block that performs an Boolean "not exclusive or" operation between two variables, storing the result in a third one.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable for the operation
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable for the operation
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable that stores the result of the operation

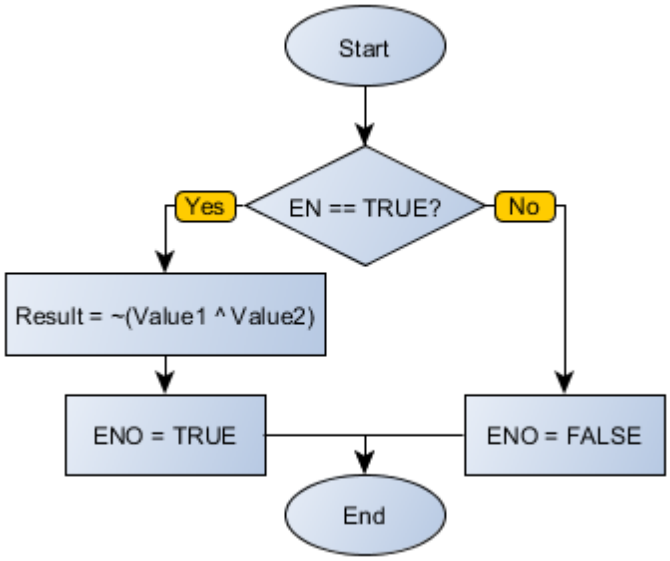
Operation

When this block has a TRUE value in EN, it sends to the Result output the “denied exclusive or” Boolean operation of input variables Value1 and Value2.

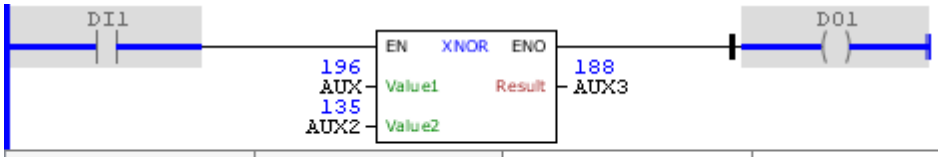
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



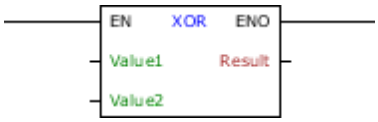
AUX	BYTE	11000100	Binary
AUX2	BYTE	10000111	Binary
AUX3	BYTE	10111100	Binary

The example above performs a "denied exclusive or" Boolean operation between AUX and AUX2, storing the result in AUX3.

11.8.5.11.2.5 XOR

Logical block that performs an Boolean "exclusive or" operation between two variables, storing the result in a third one.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable for the operation
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable for the operation
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable that stores the result of the operation

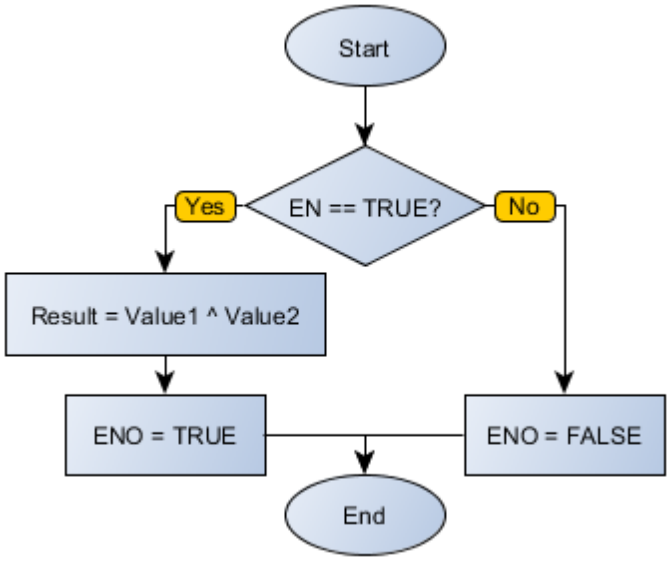
Operation

When this block has a TRUE value in EN, it sends to the Result output the “xor” Boolean operation of input variables Value1 and Value2.

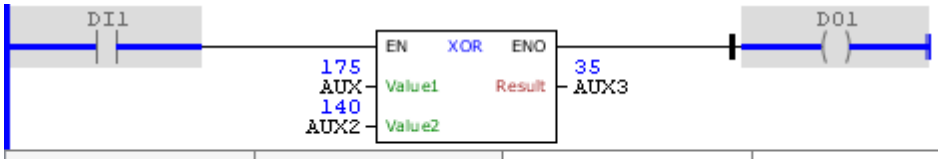
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



AUX	BYTE	10101111	Binary
AUX2	BYTE	10001100	Binary
AUX3	BYTE	100011	Binary

The example above performs a "xor" Boolean operation between AUX and AUX2, storing the result in AUX3.

11.8.5.11.3 Logic Rotate

11.8.5.11.3.1 ROL

Block that performs a logical left rotation operation in a value passed by Value, storing the result in Result.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable to undergo rotation
	Shift	BYTE USINT	Shift index
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable that stores the result of the operation

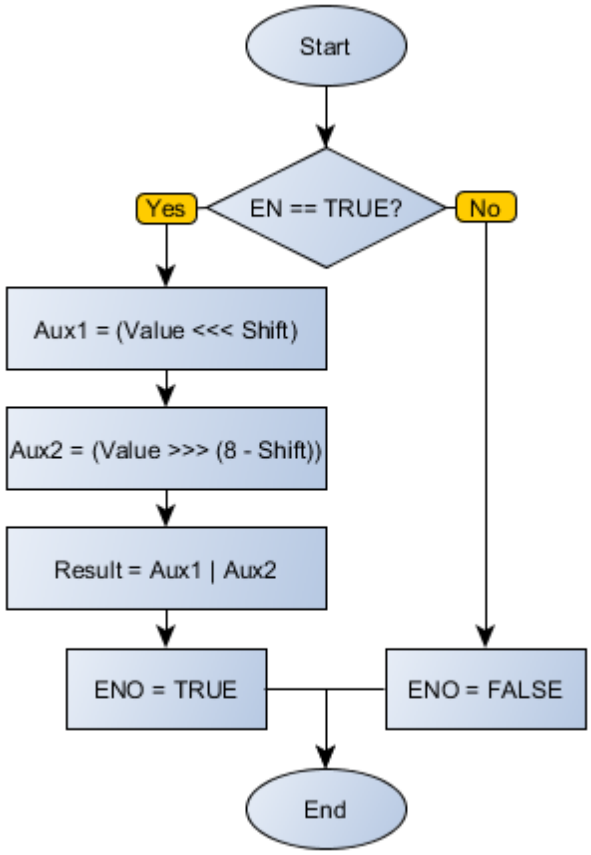
Operation

When this block has a TRUE value in EN, it sends to the Result output the value of the Value variable after performing a number of logical left shifts, according to the Shift value. The most significant bits that are being discarded are returned to the least significant bits, characterizing the rotation.

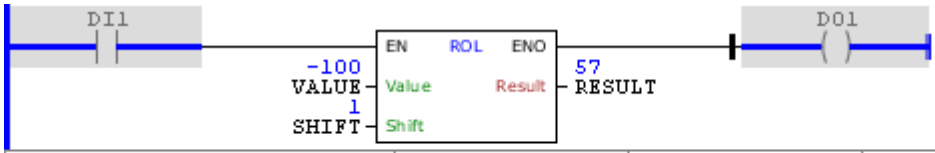
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart

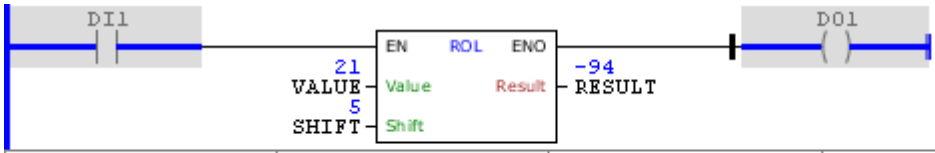


Example



VALUE	SINT	10011100	Binary
SHIFT	BYTE	1	Decimal
RESULT	SINT	111001	Binary

The above example performs a logical left shift by one position in the VALUE variable whose initial value is -100 (1001 1100 in binary). The discarded bits on the left are reinserted on the right. The final result (0011 1001 in binary) is stored in RESULT.



VALUE	SINT	10101	Binary
SHIFT	BYTE	5	Decimal
RESULT	SINT	10100010	Binary

The above example performs a logical left rotation by five positions in the VALUE variable whose initial value is 21 (0001 0101 in binary). The discarded bits on the left are reinserted on the right. The final result (1010 0010 in binary) is stored in RESULT.

11.8.5.11.3.2 ROR

Block that performs a logical right rotation operation in a value passed by Value, storing the result in Result.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable to undergo rotation
	Shift	BYTE USINT	Shift index
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable that stores the result of the operation

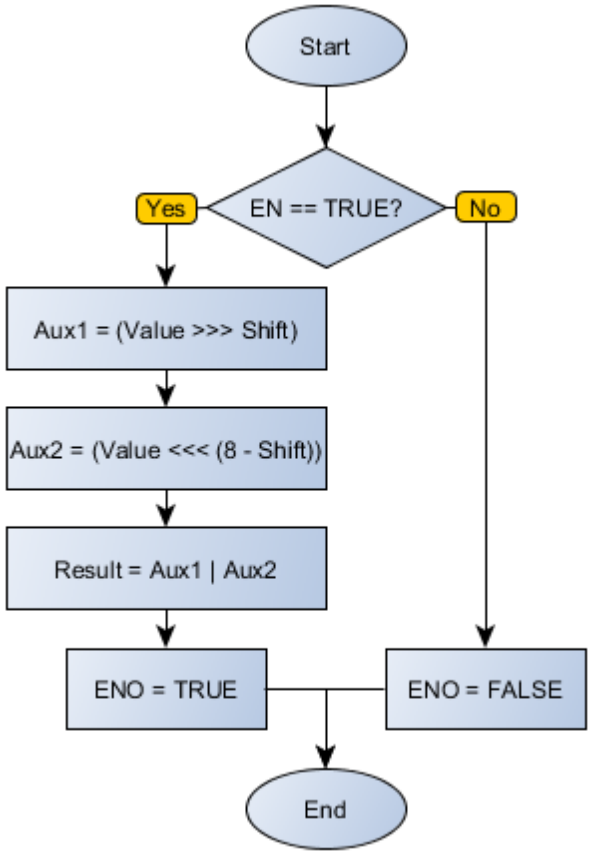
Operation

When this block has a TRUE value in EN, it sends to the Result output the value of the Value variable after performing a number of logical right shifts, according to the Shift value. The least significant bits that are being discarded are returned to the most significant bits, characterizing the rotation.

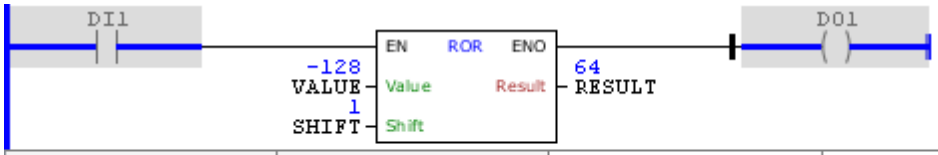
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart

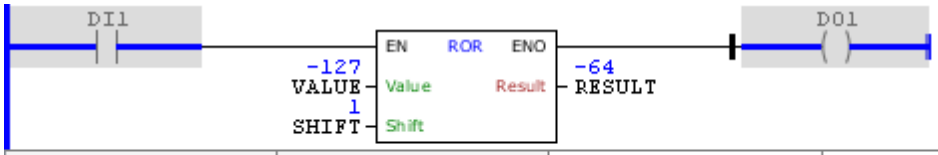


Example



VALUE	SINT	10000000	Binary
SHIFT	BYTE	1	Decimal
RESULT	SINT	1000000	Binary

The above example performs a logic right shift by one position in the VALUE variable whose initial value is -128 (1000 0000 in binary). The discarded bits on the right are reinserted on the left. The final result (0100 0000 in binary) is stored in RESULT. Notice that the sign is not preserved in this operation.



VALUE	SINT	10000001	Binary
SHIFT	BYTE	1	Decimal
RESULT	SINT	11000000	Binary

The above example performs a logical right rotation by one position in the VALUE variable whose initial value is -127 (1000 0001 in binary). The discarded bits on the right are reinserted on the left. The final result (1100 0000 in binary) is stored in RESULT.

11.8.5.11.4 Logic Shift

11.8.5.11.4.1 ASHL

Block that performs a binary left shift operation in a value passed by Value, storing the result in Result.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	SINT INT DINT	Variable to undergo shift
	Shift	BYTE USINT	Shift index
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	SINT INT DINT	Variable that stores the result of the operation

Operation

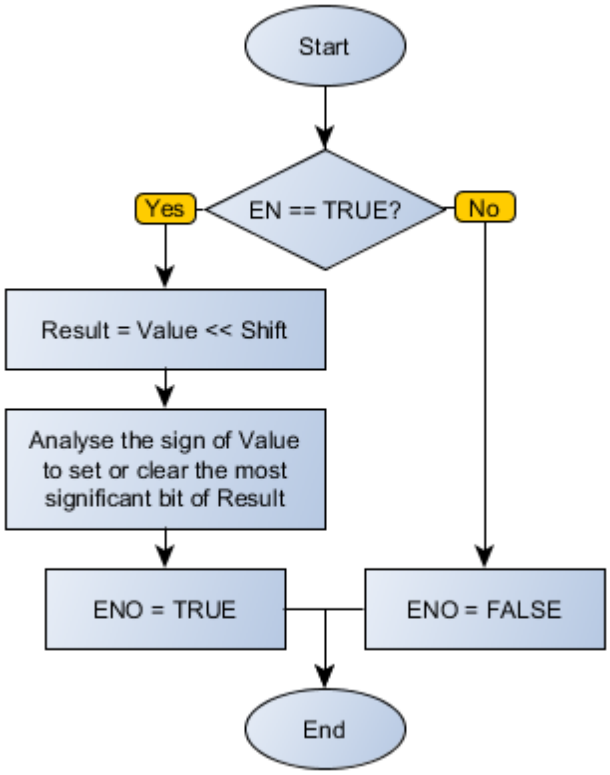
When this block has a TRUE value in EN, it sends to the Result output the value of the Value variable after performing a number of arithmetic left shifts, according to the Shift value.

NOTE!
All arithmetic shifts implemented maintain the sign of the variable.

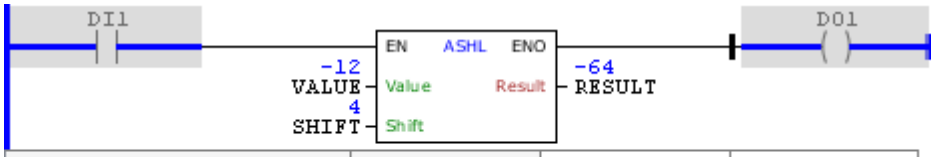
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart

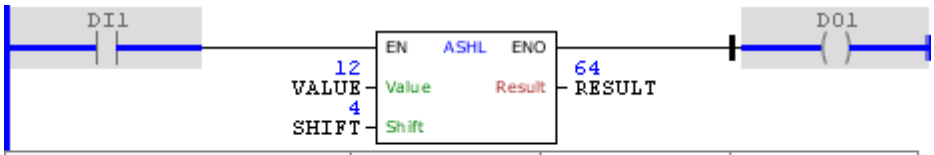


Example



VALUE	SINT	11110100	Binary
SHIFT	USINT	100	Binary
RESULT	SINT	11000000	Binary

Description of exemple.



VALUE	SINT	1100	Binary
SHIFT	USINT	100	Binary
RESULT	SINT	1000000	Binary

Description of exemple.

11.8.5.11.4.2 ASHR

Block that performs arithmetic left shift operation in a value passed by Value, storing the result in Result.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	SINT INT DINT	Variable to undergo shift
	Shift	BYTE USINT	Shift index
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	SINT INT DINT	Variable that stores the result of the operation

Operation

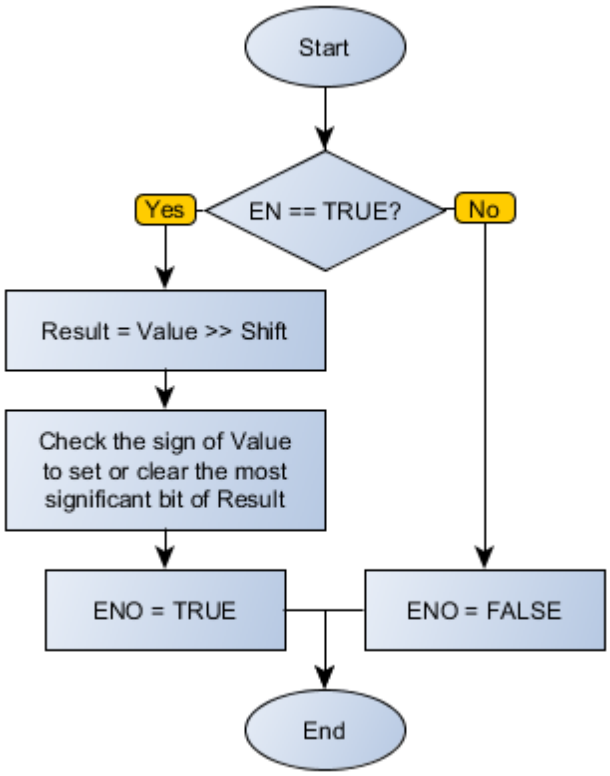
When this block has a TRUE value in EN, it sends to the Result output the value of the Value variable after performing a number of arithmetic right shifts, according to the Shift value.

NOTE!
All arithmetic shifts implemented maintain the sign of the variable.

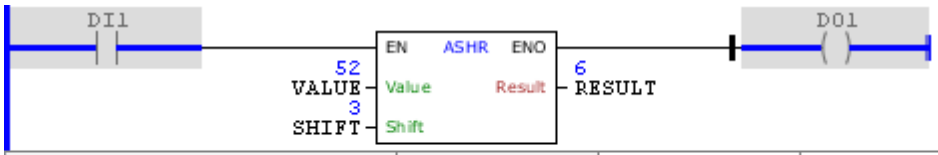
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart

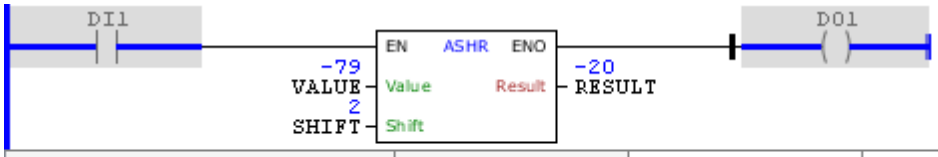


Example



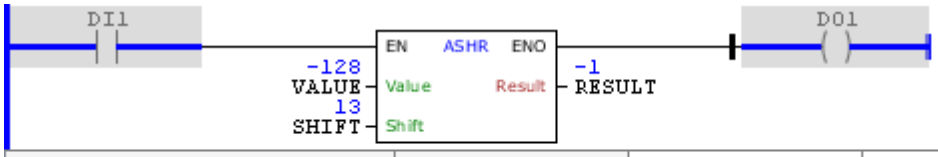
VALUE	SINT	110100	Binary
SHIFT	BYTE	3	Decimal
RESULT	INT	110	Binary

The above example performs an arithmetic right shift by three positions in the VALUE variable whose initial value is 52 (0011 0100 in binary). The bits on the right are being discarded, and on the left new zeros are inserted. The final result (0000 0110 in binary) is stored in RESULT.



VALUE	SINT	10110001	Binary
SHIFT	BYTE	2	Decimal
RESULT	INT	1111111111101100	Binary

The above example performs an arithmetic right shift by two positions in the VALUE variable whose initial value is -79 (1011 0001 in binary). The bits on the right will be discarded and new ones on the left are inserted, since the arithmetic right shifts preserve the sign of the variable. The final result (1111 0110 in binary) is stored in RESULT.



VALUE	SINT	10000000	Binary
SHIFT	BYTE	13	Decimal
RESULT	INT	1111111111111111	Binary

The above example performs an arithmetic right shift by thirteen positions in the VALUE variable whose initial value is -128 (1000 0000 in binary). The bits on the right are being discarded, and on the left new ones are inserted. The final result (1111 1111 in binary) is stored in RESULT.

11.8.5.11.4.3 SHL

Block that performs a binary logical left shift operation in a value passed by Value, storing the result in Result.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable to undergo shift
	Shift	BYTE USINT	Shift index
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable that stores the result of the operation

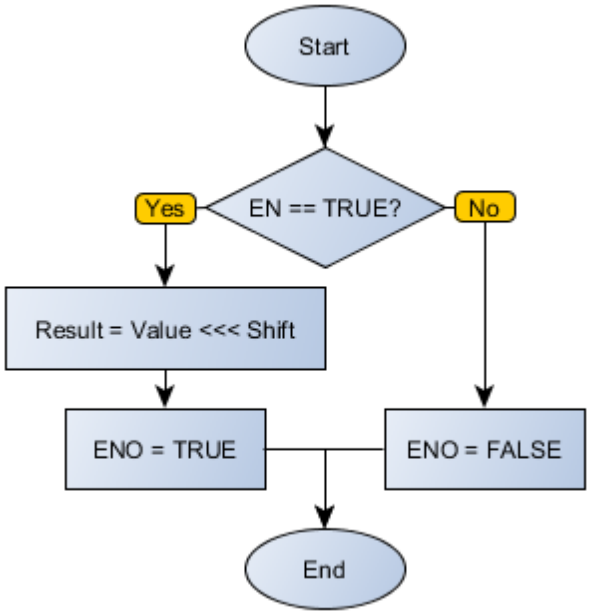
Operation

When this block has a TRUE value in EN, it sends to the Result output the value of the Value variable after performing a number of logical shifts left, according to the Shift value.

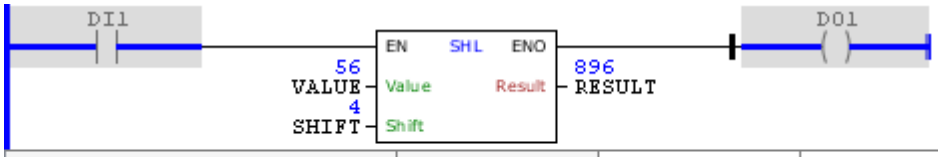
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart

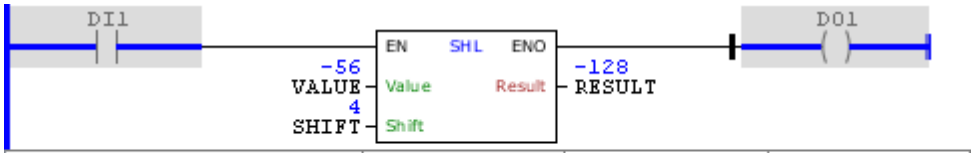


Example



VALUE	INT	111000	Binary
SHIFT	BYTE	4	Decimal
RESULT	INT	1110000000	Binary

The above example performs a logical right shift by four positions in the VALUE variable whose initial value is 56 (0011 1000 in binary). The bits on the left are being discarded, and on the left new zeros are inserted. The final result (0011 1000 0000 in binary) is stored in RESULT.



VALUE	SINT	11001000	Binary
SHIFT	USINT	4	Decimal
RESULT	SINT	10000000	Binary

The above example performs a logical right shift by four positions in the VALUE variable whose initial value is -56 (1100 1000 in binary). The bits on the left are being discarded, and on the left new zeros are inserted. The final result (1100 1000 0000 in binary) is stored in RESULT. Since RESULT is SINT type, it only accepts the first eight bits (1000 0000).

11.8.5.11.4.4 SHR

Block that performs a binary logical right shift operation in a value passed by Value, storing the result in Result.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable to undergo shift
	Shift	BYTE USINT	Shift index
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable that stores the result of the operation

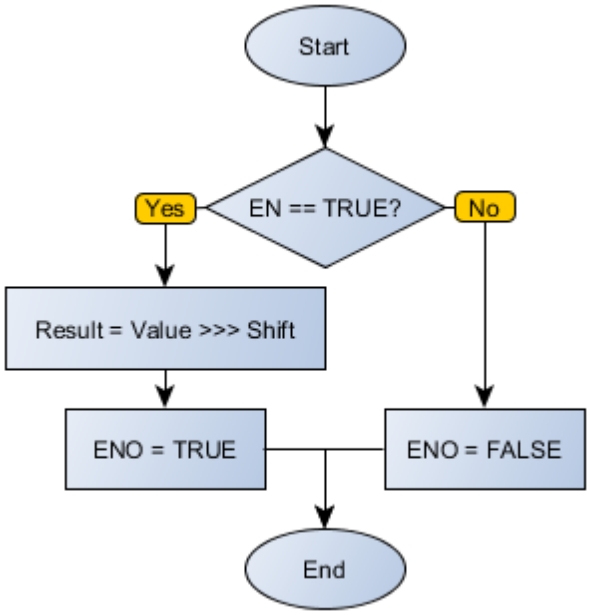
Operation

When this block has a TRUE value in EN, it sends to the Result output the value of the Value variable after performing a number of logical shifts right, according to the Shift value.

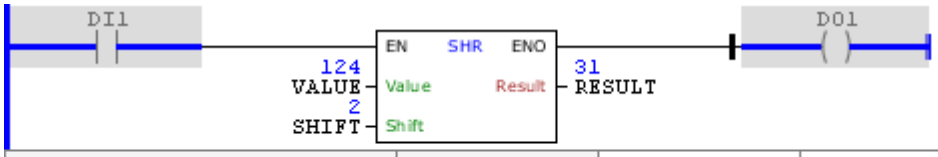
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart

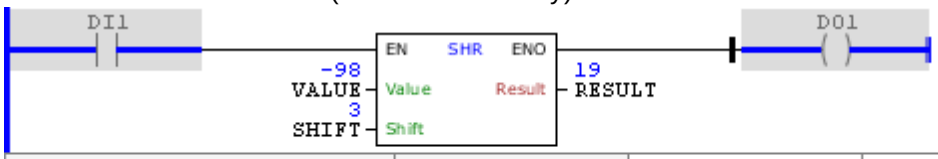


Example



VALUE	SINT	1111100	Binary
SHIFT	BYTE	2	Decimal
RESULT	SINT	11111	Binary

The above example performs a logical right shift by two positions in the VALUE variable whose initial value is 124 (0111 1100 in binary). The bits on the right are being discarded, and on the left new zeros are inserted. The final result (0001 1111 in binary) is stored in RESULT.



VALUE	SINT	10011110	Binary
SHIFT	BYTE	3	Decimal
RESULT	INT	10011	Binary

The above example performs a logical right shift by three positions in the VALUE variable whose initial value is -98 (1001 1110 in binary). The bits on the right are being discarded, and on the left new zeros are inserted. The final result (0001 0011 in binary) is stored in RESULT.

11.8.5.12 Math

11.8.5.12.1 Math Basic

11.8.5.12.1.1 ABS

Block that calculates the Value module, storing the result in Result.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Reference variable for the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Variable that stores the result of the operation

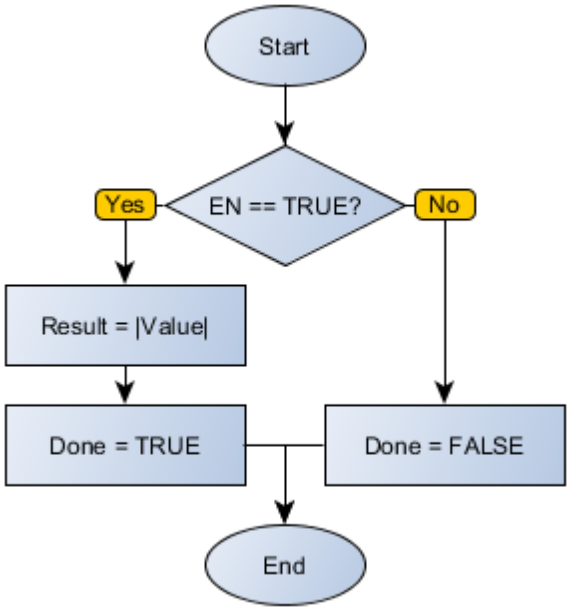
Operation

When this block has a TRUE value in EN, it sends to the Result output the absolute value of the

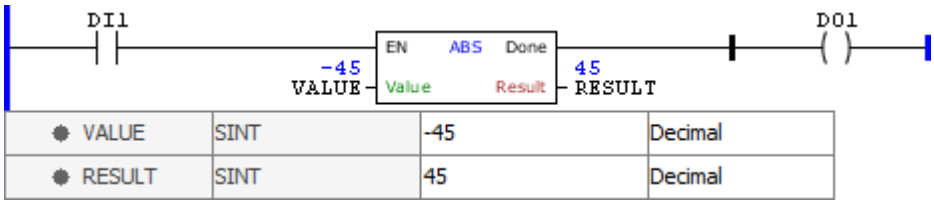
Value variable. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

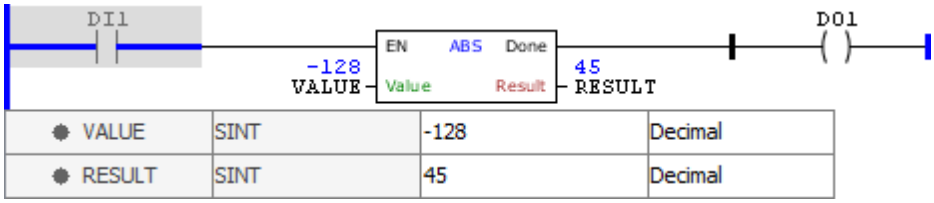


Example



VALUE	SINT	-45	Decimal
RESULT	SINT	45	Decimal

The above example calculates the absolute value of the VALUE variable whose initial value is -45, storing the final result, 45, in RESULT.



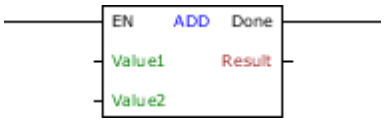
VALUE	SINT	-128	Decimal
RESULT	SINT	45	Decimal

The above example calculates the absolute value of the VALUE variable whose initial value is -45. The final result, 128, cannot be stored in RESULT, because it is outside the limits of accepted values by SINT type. Therefore, RESULT remains unchanged and the output is disabled.

11.8.5.12.1.2 ADD

Block that calculates the sum of the values of Value1 and Value2, storing the result in Result.

Ladder Representation



Block Structure

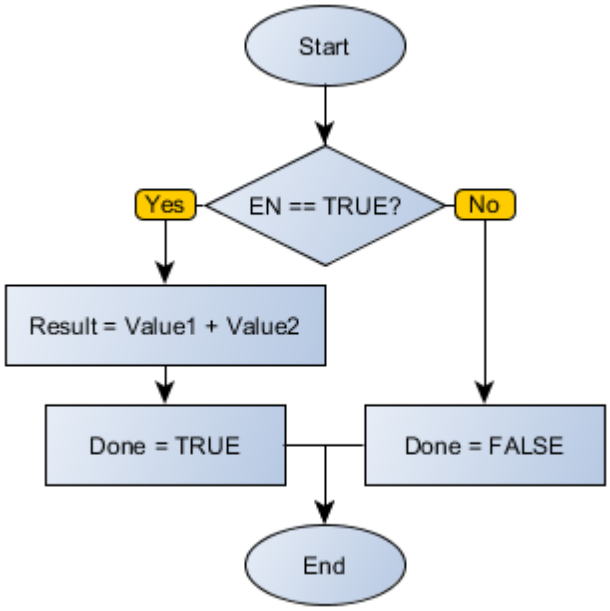
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	First addend of the operation
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Second addend of the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Variable that stores the result of the operation

Operation

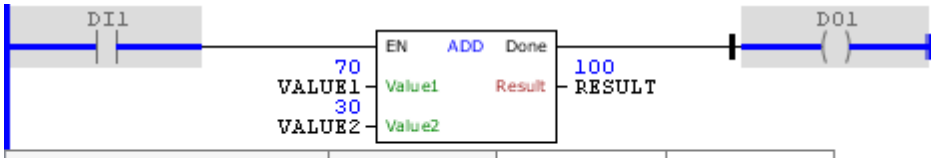
When this block has a TRUE value in EN, it sends to the Result output the sum of Value1 and Value2 variables. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

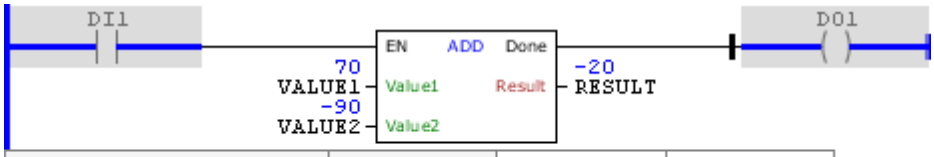


Example



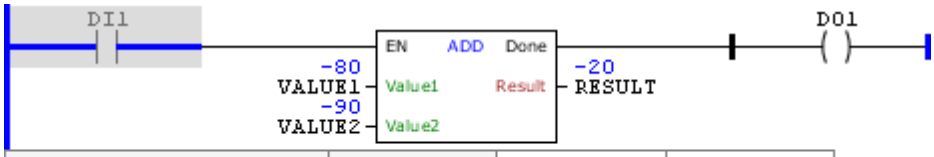
VALUE1	SINT	70	Decimal
VALUE2	SINT	30	Decimal
RESULT	SINT	100	Decimal

The above example calculates the sum of VALUE 1 and VALUE2 variables, storing the final result in RESULT.



VALUE1	SINT	70	Decimal
VALUE2	SINT	-90	Decimal
RESULT	SINT	-20	Decimal

The above example calculates the sum of VALUE 1 and VALUE2 variables, storing the final result in RESULT. Notice that the block accepts arguments of both signs.



VALUE1	SINT	-80	Decimal
VALUE2	SINT	-90	Decimal
RESULT	SINT	-20	Decimal

The above example calculates the sum of VALUE1 and VALUE2 variables. The final result -170 cannot be stored in RESULT, because it is outside the limits of accepted values by SINT type. Therefore, RESULT remains unchanged and the output is disabled.

11.8.5.12.1.3 DIV

Block that calculates the division of the values of Value1 and Value2, storing the result in Result.

Ladder Representation

Block Structure

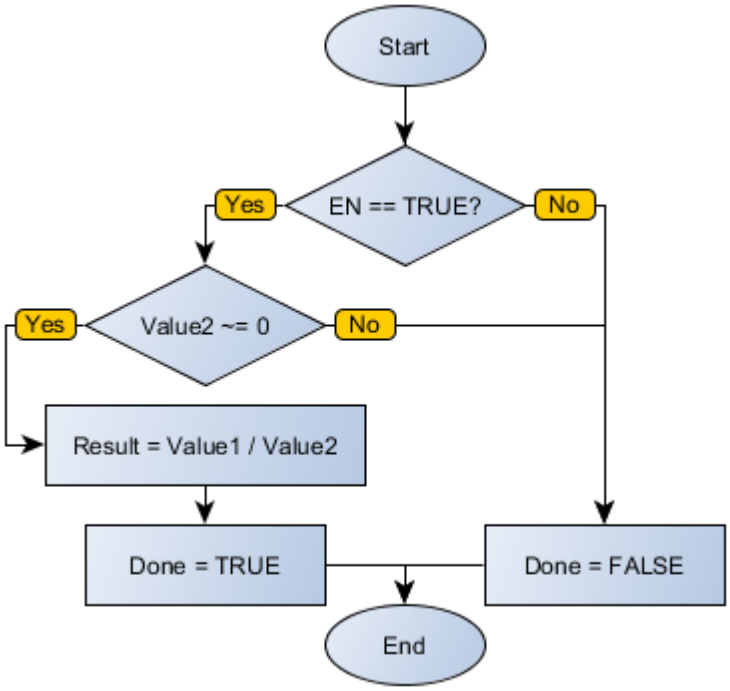
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Dividend of the operation
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Divisor of the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Variable that stores the result of the operation

Operation

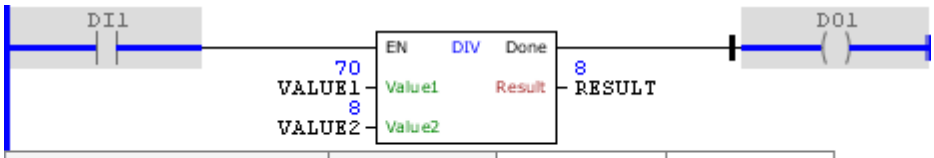
When this block has a TRUE value in EN, it sends to the Result output the division of Value1 and Value2 variables. The value stored will be the exact division if Result is REAL, or, in other cases, only the quotient. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

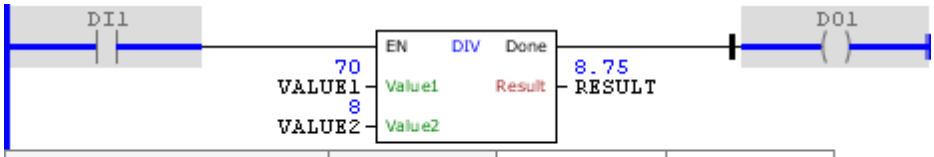


Example



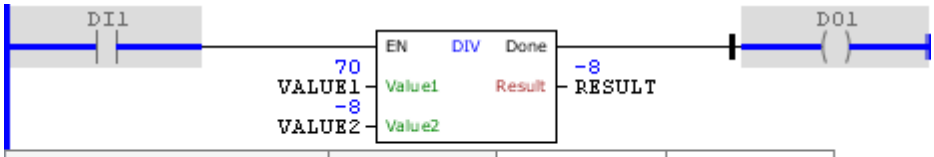
VALUE1	SINT	70	Decimal
VALUE2	SINT	8	Decimal
RESULT	SINT	8	Decimal

The above example calculates the division of VALUE 1 and VALUE2 variables, storing the final result in RESULT. Since RESULT is SINT type, only the quotient is stored in it.



VALUE1	SINT	70	Decimal
VALUE2	SINT	8	Decimal
RESULT	REAL	8.75	Float

The above example calculates the division of VALUE 1 and VALUE2 variables, storing the final result in RESULT. Since RESULT is of REAL type, the exact value of the division is stored in it.



VALUE1	SINT	70	Decimal
VALUE2	SINT	-8	Decimal
RESULT	SINT	-8	Decimal

The above example calculates the division of VALUE 1 and VALUE2 variables, storing the final result in RESULT. Since RESULT is SINT type, only the quotient is stored in it. Notice that the block accepts arguments of both signs.

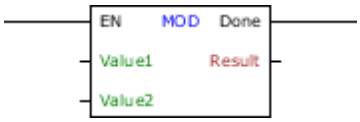
VALUE1	SINT	70	Decimal
VALUE2	SINT	0	Decimal
RESULT	SINT	-8	Decimal

The above example calculates the division of VALUE 1 and VALUE2 variables, storing the final result in RESULT. Since VALUE2 is zero, the block generates a runtime error, RESULT remains unchanged and the output is disabled.

11.8.5.12.1.4 MOD

Block that calculates the remainder of the values of Value1 and Value2, storing the result in Result.

Ladder Representation



Block Structure

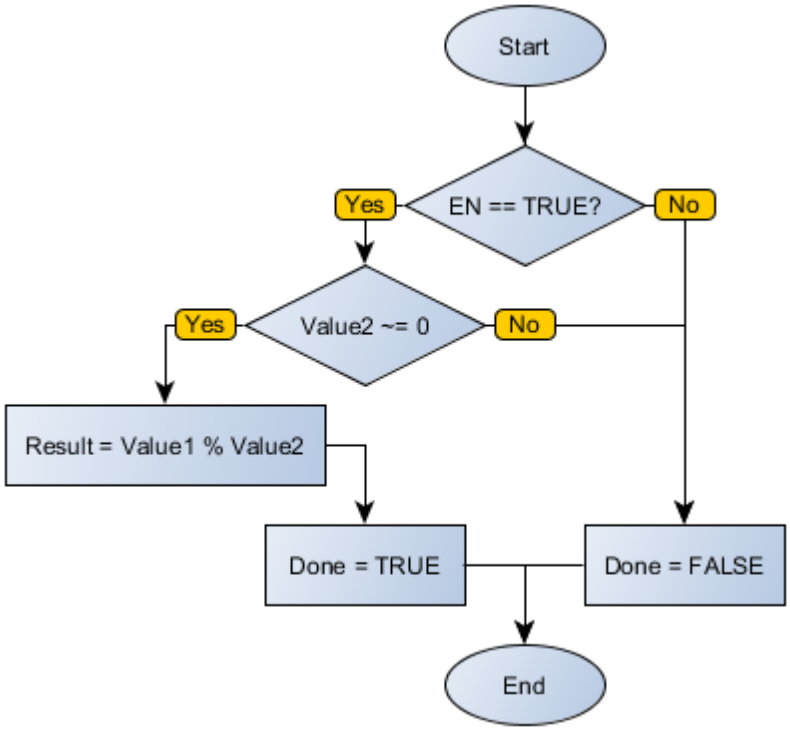
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Dividend of the operation
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Divisor of the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable that stores the result of the operation

Operation

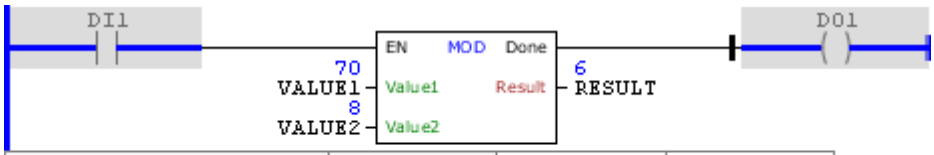
When this block has a TRUE value in EN, it sends to the Result output the remainder of Value1 and Value2 variables. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

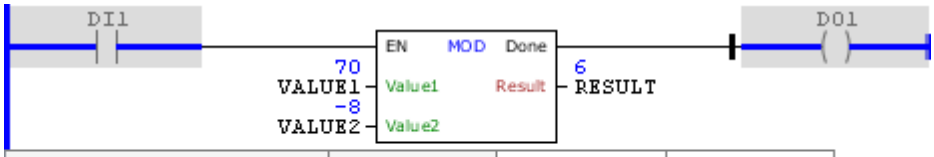


Example



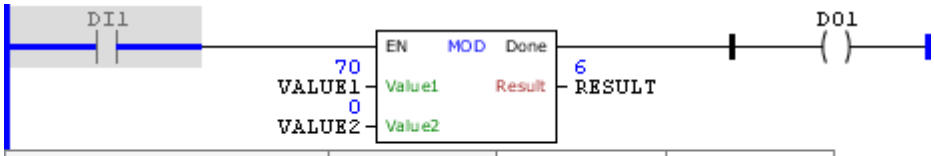
VALUE1	SINT	70	Decimal
VALUE2	SINT	8	Decimal
RESULT	SINT	6	Decimal

The above example calculates the remainder of VALUE 1 and VALUE2 variables, storing the final result in RESULT.



VALUE1	SINT	70	Decimal
VALUE2	SINT	-8	Decimal
RESULT	SINT	6	Decimal

The above example calculates the remainder of VALUE 1 and VALUE2 variables, storing the final result in RESULT. Notice that the block accepts arguments of both signs.



VALUE1	SINT	70	Decimal
VALUE2	SINT	0	Decimal
RESULT	SINT	6	Decimal

The above example calculates the remainder of VALUE 1 and VALUE2 variables, storing the final result in RESULT. Since VALUE2 is zero, the block generates a runtime error, RESULT remains unchanged and the output is disabled.

11.8.5.12.1.5 MUL

Block that calculates the multiplication of the values of Value1 and Value2, storing the result in Result.

Ladder Representation

Block Structure

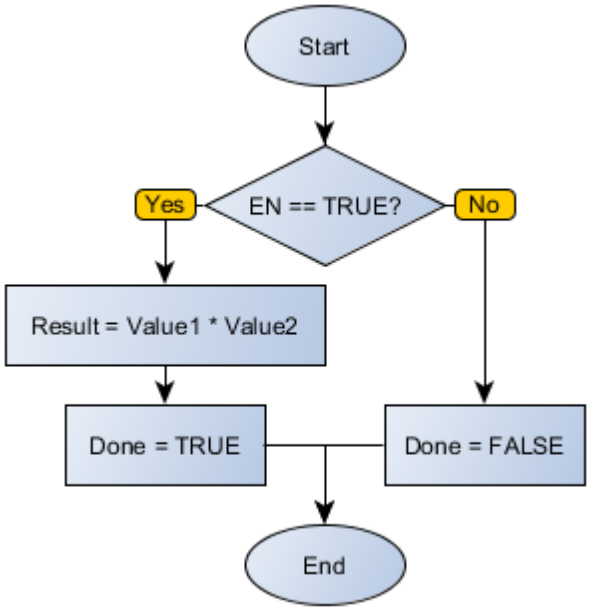
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	First factor of the operation
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Second factor of the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Variable that stores the result of the operation

Operation

When this block has a TRUE value in EN, it sends to the Result output the multiplication of Value1 and Value2 variables. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

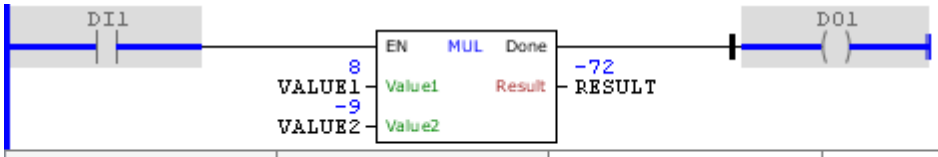
Block Flowchart



Example

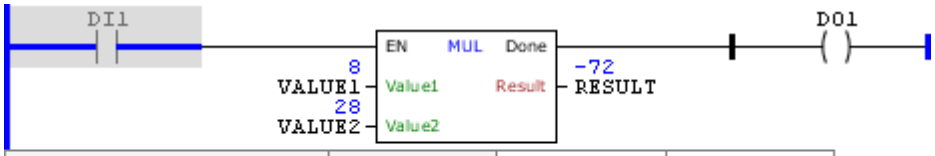
VALUE1	SINT	8	Decimal
VALUE2	SINT	12	Decimal
RESULT	SINT	96	Decimal

The above example calculates the product of VALUE 1 and VALUE2 variables, storing the final result in RESULT.



VALUE1	SINT	8	Decimal
VALUE2	SINT	-9	Decimal
RESULT	SINT	-72	Decimal

The above example calculates the product of VALUE 1 and VALUE2 variables, storing the final result in RESULT. Notice that the block accepts arguments of both signs.



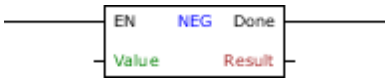
VALUE1	SINT	8	Decimal
VALUE2	SINT	28	Decimal
RESULT	SINT	-72	Decimal

The above example calculates the product of VALUE1 and VALUE2 variables. The final result 224 cannot be stored in RESULT, because it is outside the limits of accepted values by SINT type. Therefore, RESULT remains unchanged and the output is disabled.

11.8.5.12.1.6 NEG

Block that calculates the opposite (i.e., the product with -1) of a value passed by Value, storing the result in Result.

Ladder Representation



Block Structure

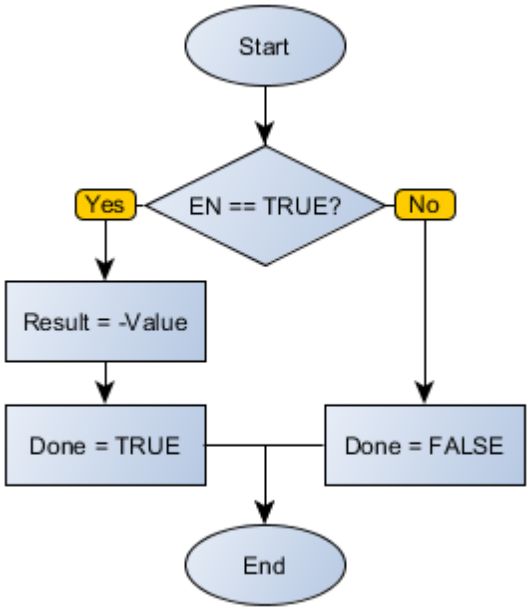
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Reference variable for the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Variable that stores the result of the operation

Operation

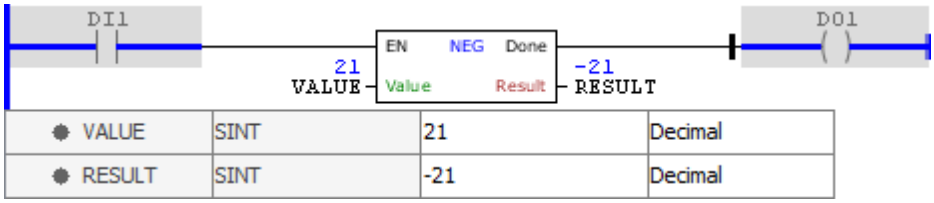
When this block has a TRUE value in EN, it sends to the Result output the opposite of the Value variable. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

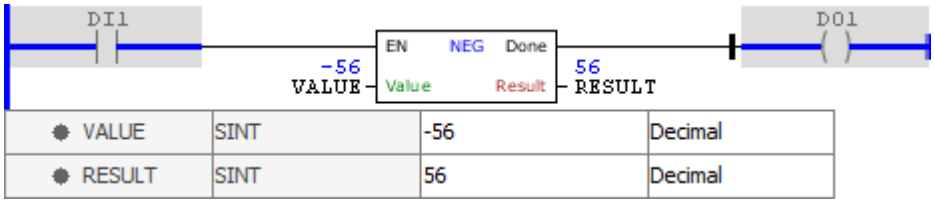


Example



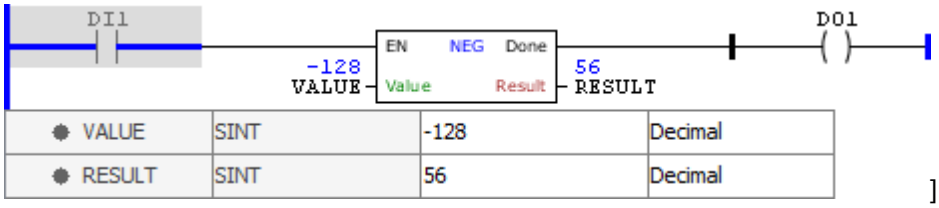
VALUE	SINT	21	Decimal
RESULT	SINT	-21	Decimal

The above example calculates the opposite of the VALUE variable whose initial value is 21, storing the final result, -21, in RESULT.



VALUE	SINT	-56	Decimal
RESULT	SINT	56	Decimal

The above example calculates the opposite of the VALUE variable whose initial value is -56, storing the final result, 56, in RESULT.



VALUE	SINT	-128	Decimal
RESULT	SINT	56	Decimal

]

The above example calculates the opposite of the VALUE variable whose initial value is -128. The final result, 128, cannot be stored in RESULT, because it is outside the limits of accepted values by SINT type. Therefore, RESULT remains unchanged and the output is disabled.

11.8.5.12.1.7 SUB

Block that calculates the subtraction between the Value1 and Value2 values, storing the result in Result.

Ladder Representation

Block Structure

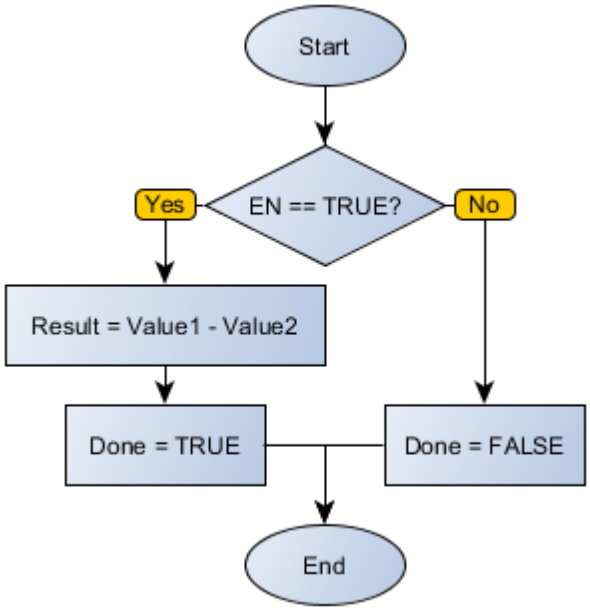
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Minuend of operation
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Subtrahend of operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Variable that stores the result of the operation

Operation

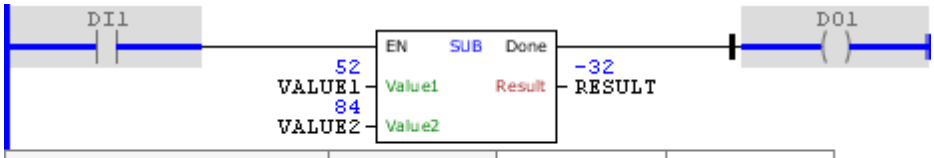
When this block has a TRUE value in EN, it sends to the Result output the subtraction of Value1 and Value2 variables. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

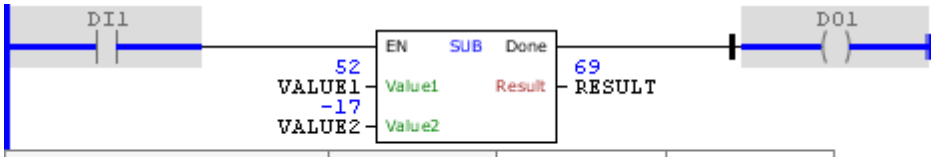


Example



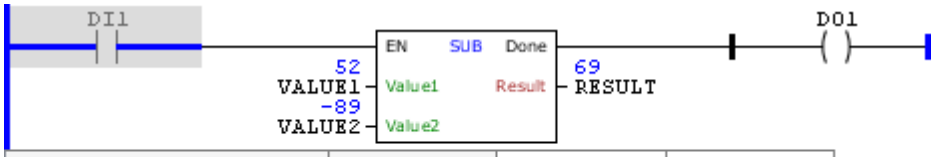
VALUE1	SINT	52	Decimal
VALUE2	SINT	84	Decimal
RESULT	SINT	-32	Decimal

The above example calculates the subtraction of VALUE 1 and VALUE2 variables, storing the final result in RESULT.



VALUE1	SINT	52	Decimal
VALUE2	SINT	-17	Decimal
RESULT	SINT	69	Decimal

The above example calculates the subtraction of VALUE 1 and VALUE2 variables, storing the final result in RESULT. Notice that the block accepts arguments of both signs.



VALUE1	SINT	52	Decimal
VALUE2	SINT	-89	Decimal
RESULT	SINT	69	Decimal

The above example calculates the subtraction of VALUE1 and VALUE2 variables. The final result 141 cannot be stored in RESULT, because it is outside the limits of accepted values by SINT type. Therefore, RESULT remains unchanged and the output is disabled.

11.8.5.12.2 Math Extended

11.8.5.12.2.1 ALOG10

Block that calculates the antilogarithm (exponent with base 10) of the Value value, storing the result in Result.

Ladder Representation

Block Structure

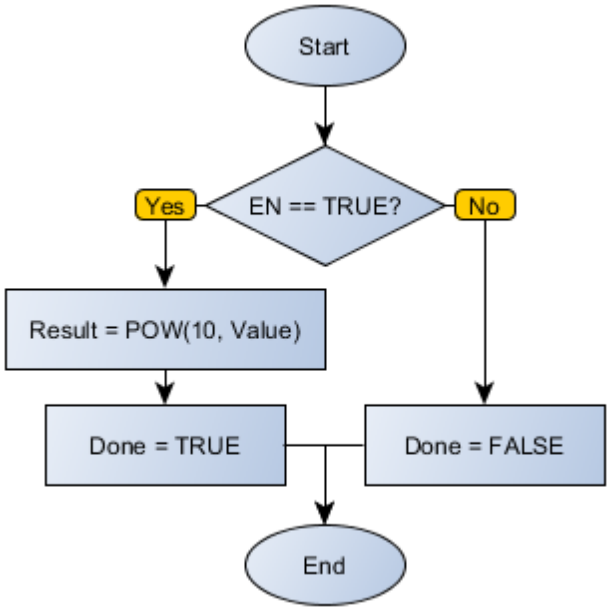
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Reference variable for the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	REAL	Variable that stores the result of the operation

Operation

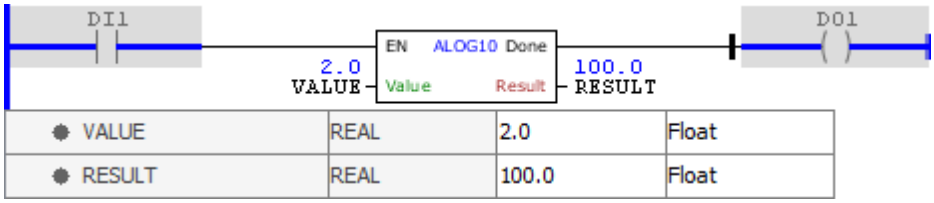
When this block has a TRUE value in EN, it sends to the Result output the antilogarithm of the Value variable. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

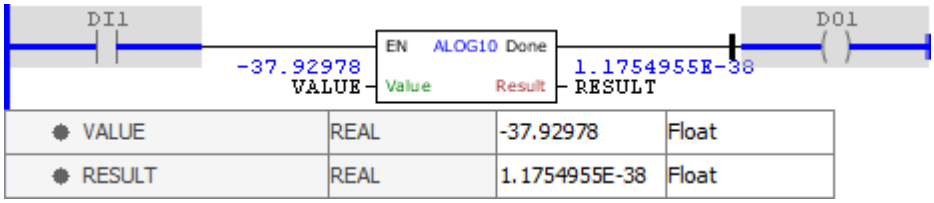


Example



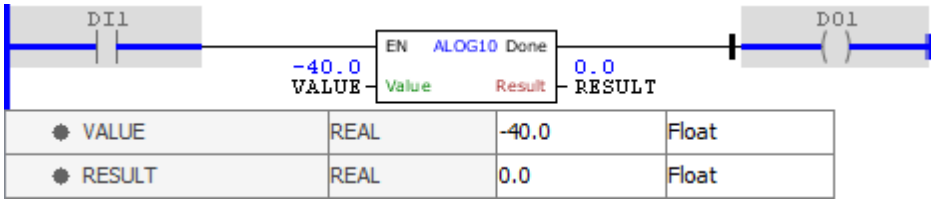
VALUE	REAL	2.0	Float
RESULT	REAL	100.0	Float

The above example calculates the antilogarithm of the VALUE variable, storing the final result in RESULT. The block ends with success and Done output is activated.



VALUE	REAL	-37.92978	Float
RESULT	REAL	1.1754955E-38	Float

The above example calculates the antilogarithm of the VALUE variable, storing the final result in RESULT. The indicated value is the minimum input value for which the block revolutions a nonzero result. The block ends with success and Done output is activated.



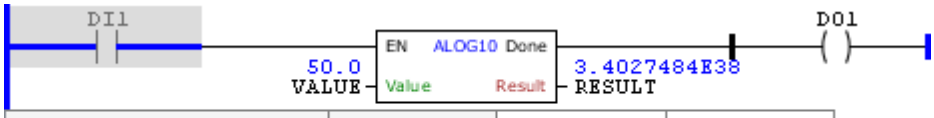
VALUE	REAL	-40.0	Float
RESULT	REAL	0.0	Float

The above example calculates the antilogarithm of the VALUE variable, storing the final result in

RESULT. Below the minimum values cause the block to return a null value. The block ends with success and Done output is activated.

VALUE	REAL	38.53183	Float
RESULT	REAL	3.4027484E38	Float

The above example calculates the antilogarithm of the VALUE variable, storing the final result in RESULT. The indicated value is the maximum input value for which the block revolutions a valid result. The block ends with success and Done output is activated.



VALUE	REAL	50.0	Float
RESULT	REAL	3.4027484E38	Float

The above example calculates the antilogarithm of the VALUE variable, storing the final result in RESULT. Values higher than the maximum cause the block to generate an error, the RESULT output remains unchanged and Done output is disabled.

11.8.5.12.2.2 EXP

Block that calculates the exponential of the Euler number "and" raised to the value of Value, storing the result in Result.

Ladder Representation

Block Structure

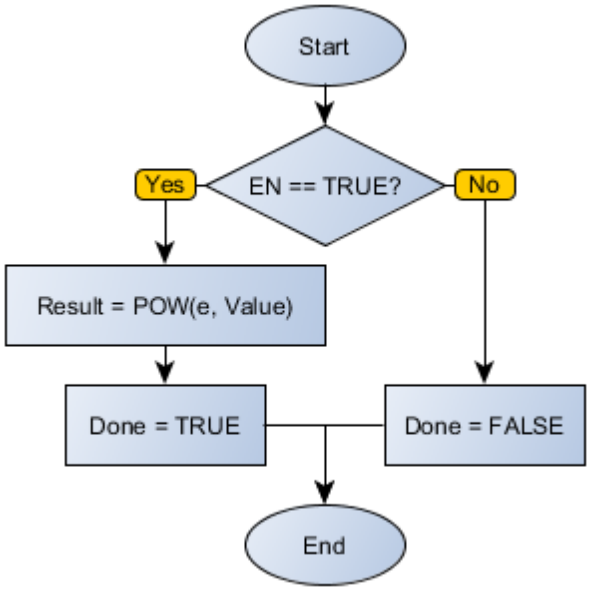
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Reference variable for the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	REAL	Variable that stores the result of the operation

Operation

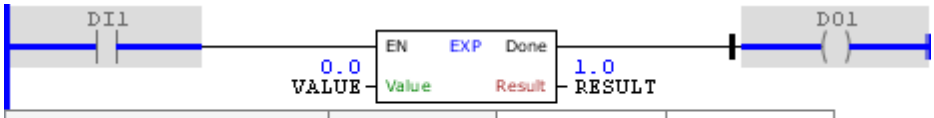
When this block has a TRUE value in EN, it sends to the Result output the exponent of the Euler number "and" raised to the Value variable. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart



Example



VALUE	REAL	0.0	Float
RESULT	REAL	1.0	Float

The above example calculates the exponent of the VALUE variable, storing the final result in RESULT. The block ends with success and Done output is activated.

VALUE	REAL	-87.33654	Float
RESULT	REAL	1.1754997E-38	Float

The above example calculates the exponent of the VALUE variable, storing the final result in RESULT. The indicated value is the minimum input value for which the block revolutions a nonzero result. The block ends with success and Done output is activated.

VALUE	REAL	-50000.0	Float
RESULT	REAL	0.0	Float

The above example calculates the exponent of the VALUE variable, storing the final result in RESULT. Values below the minimum cause the block to return to a null value. The block ends with success and Done output is activated.

VALUE	REAL	88.72283	Float
RESULT	REAL	3.4027985E38	Float

The above example calculates the exponent of the VALUE variable, storing the final result in RESULT. The indicated value is the maximum input value for which the block revolutions a valid result. The block ends with success and Done output is activated.

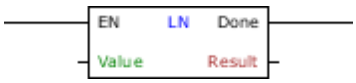
VALUE	REAL	90.0	Float
RESULT	REAL	3.4027985E38	Float

The above example calculates the exponent of the VALUE variable, storing the final result in RESULT. Values higher than the maximum cause the block to generate an error, the RESULT output remains unchanged and Done output is disabled.

11.8.5.12.2.3 LN

Block that calculates the natural logarithm of the Value value, storing the result in Result.

Ladder Representation



Block Structure

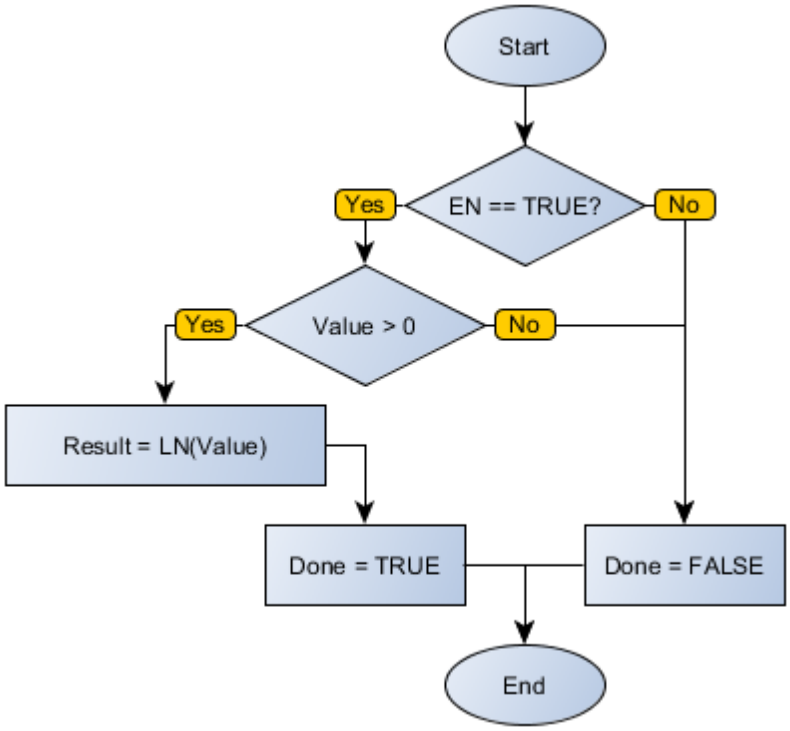
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Reference variable for the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	REAL	Variable that stores the result of the operation

Operation

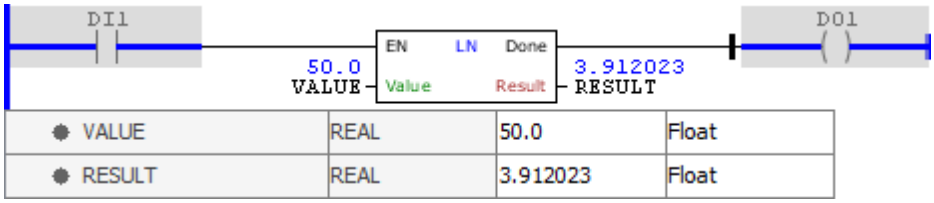
When this block has a TRUE value in EN, it sends to the Result output the natural logarithm of the Value variable. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart



Example



VALUE	REAL	50.0	Float
RESULT	REAL	3.912023	Float

The above example calculates the natural logarithm of the VALUE variable, storing the final result in RESULT. The block ends with success and Done output is activated.

VALUE	REAL	0.0	Float
RESULT	REAL	3.912023	Float

The above example calculates the natural logarithm of the VALUE variable, storing the final result in RESULT. The block generates a runtime error, since VALUE has value zero, and Done output is disabled.

11.8.5.12.2.4 LOG10

Block that calculates the common logarithm (base 10) of the Value value, storing the result in Result.

Ladder Representation

Block Structure

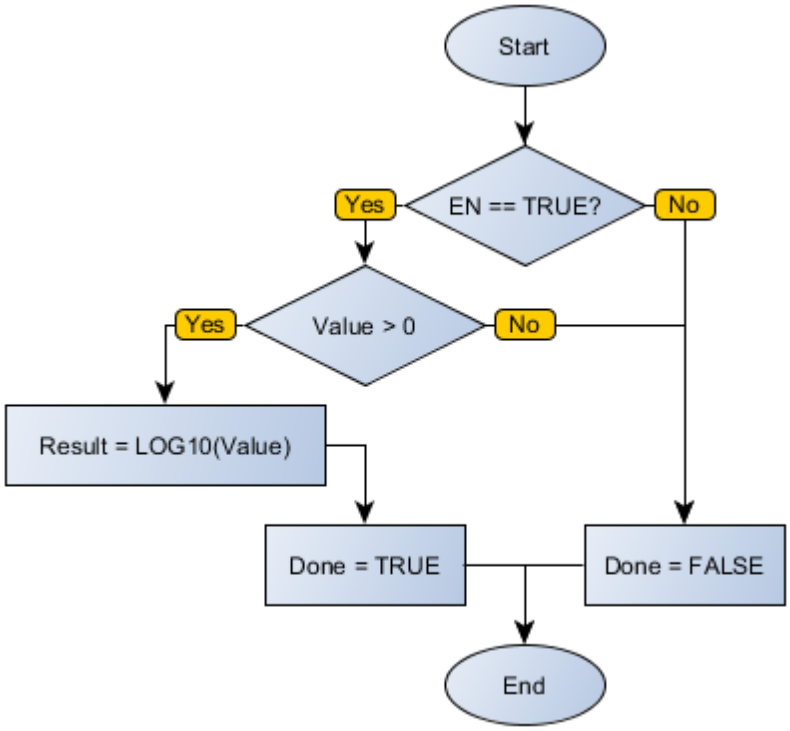
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Reference variable for the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	REAL	Variable that stores the result of the operation

Operation

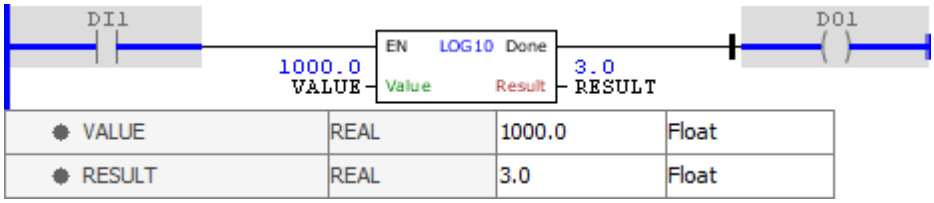
When this block has a TRUE value in EN, it sends to the Result output the common logarithm of the Value variable. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

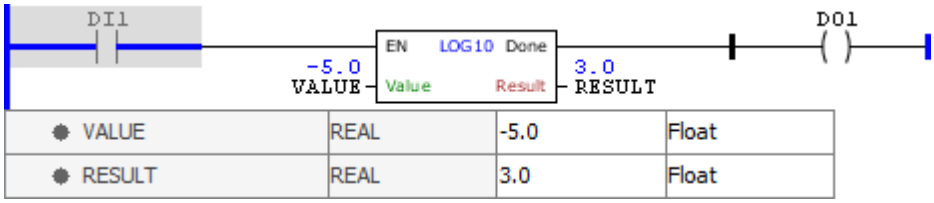


Example



VALUE	REAL	1000.0	Float
RESULT	REAL	3.0	Float

The above example calculates the common logarithm of the VALUE variable, storing the final result in RESULT. The block ends with success and Done output is activated.



VALUE	REAL	-5.0	Float
RESULT	REAL	3.0	Float

The above example calculates the common logarithm of the VALUE variable, storing the final result in RESULT. The block generates a runtime error, since VALUE has negative value, and Done output is disabled.

11.8.5.12.2.5 POW

Block that calculates the value of Value raised to the exponent Power, storing the result in Result.

Ladder Representation

Block Structure

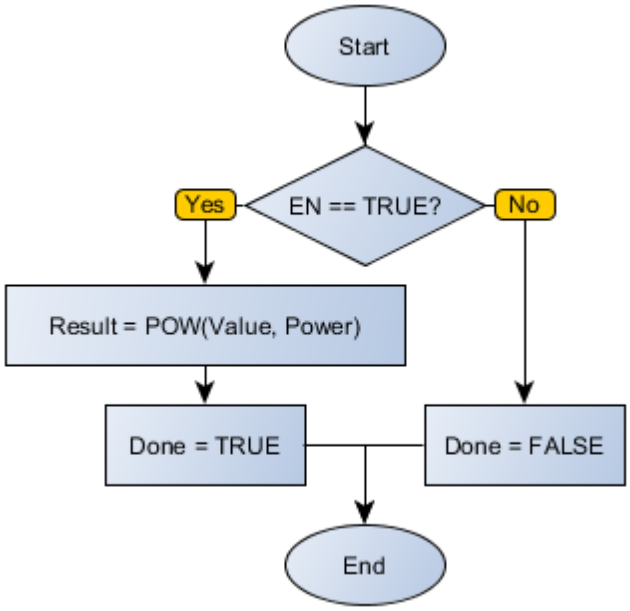
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Base of the operation
	Power	REAL	Exponent of the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	REAL	Variable that stores the result of the operation

Operation

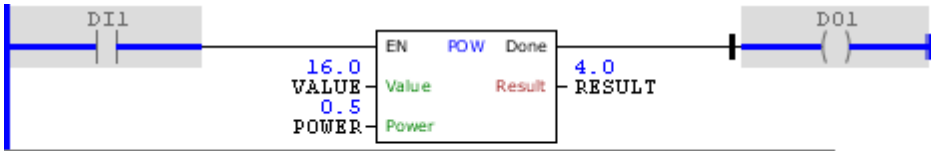
When this block has a TRUE value in EN, it sends to the Result output the value of Value raised to the exponent Power. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

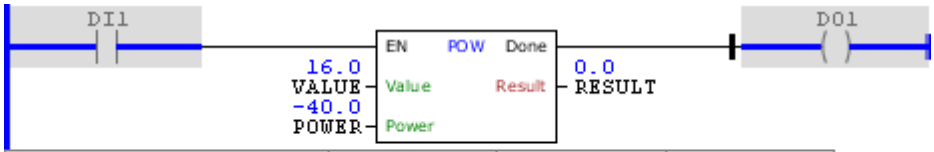


Example



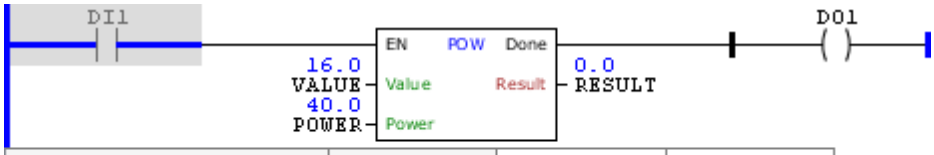
VALUE	REAL	16.0	Float
POWER	REAL	0.5	Float
RESULT	REAL	4.0	Float

The above example calculates the value of VALUE raised to the POWER variable, storing the final result in RESULT. The block ends with success and Done output is activated.



VALUE	REAL	16.0	Float
POWER	REAL	-40.0	Float
RESULT	REAL	0.0	Float

The above example calculates the value of VALUE raised to the POWER variable, storing the final result in RESULT. The block ends with success and Done output is activated.



VALUE	REAL	16.0	Float
POWER	REAL	40.0	Float
RESULT	REAL	0.0	Float

The above example calculates the value of VALUE raised to the POWER variable, storing the final result in RESULT. Since the result is higher than the maximum supported by REAL type, the block generates an error and Done output is disabled.

11.8.5.12.2.6 ROUND

Block that rounds the value of Value, storing the result in Result.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Reference variable for the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	REAL	Variable that stores the result of the operation

Operation

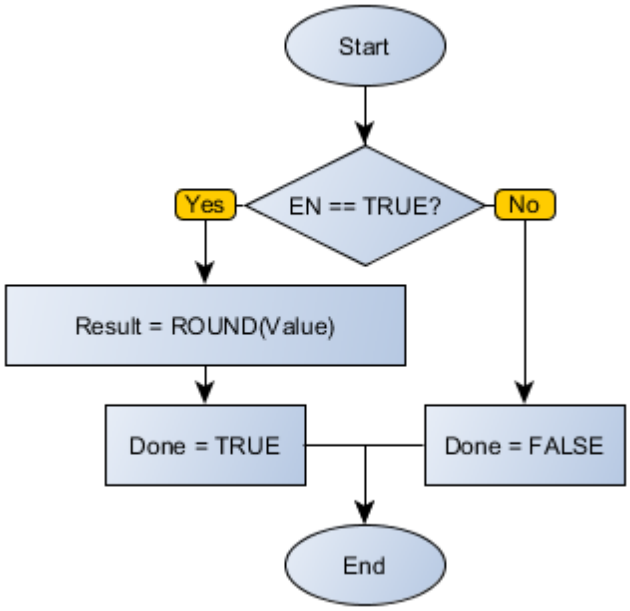
When this block has a TRUE value in EN, it sends to the Result output the rounded value of Value. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

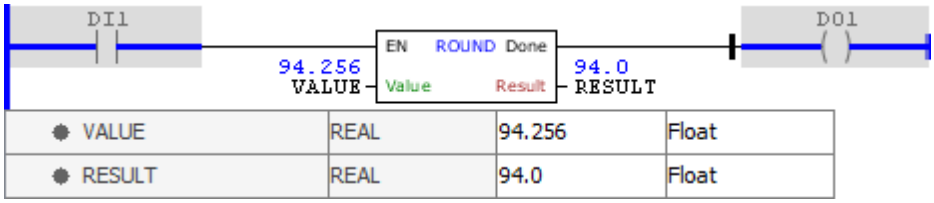
Compatibility

Device	Version
PLC300	2.10 or higher
SCA06	2.00 or higher

Block Flowchart

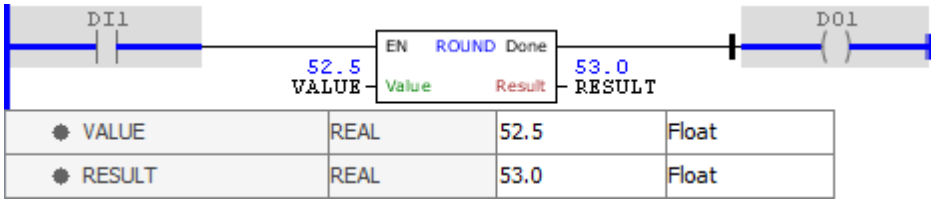


Example



VALUE	REAL	94.256	Float
RESULT	REAL	94.0	Float

The above example rounds the value of the VALUE variable, storing the final result in RESULT. Decimals less than 0.5 are discarded. The block ends with success and Done output is activated.



VALUE	REAL	52.5	Float
RESULT	REAL	53.0	Float

The above example rounds the value of the VALUE variable, storing the final result in RESULT. Decimals greater than or equal to 0.5 promote unity value immediately above. The block ends with success and Done output is activated.

11.8.5.12.2.7 SQRT

Block that calculates the square root value of Value, storing the result in Result.

Ladder Representation

Block Structure

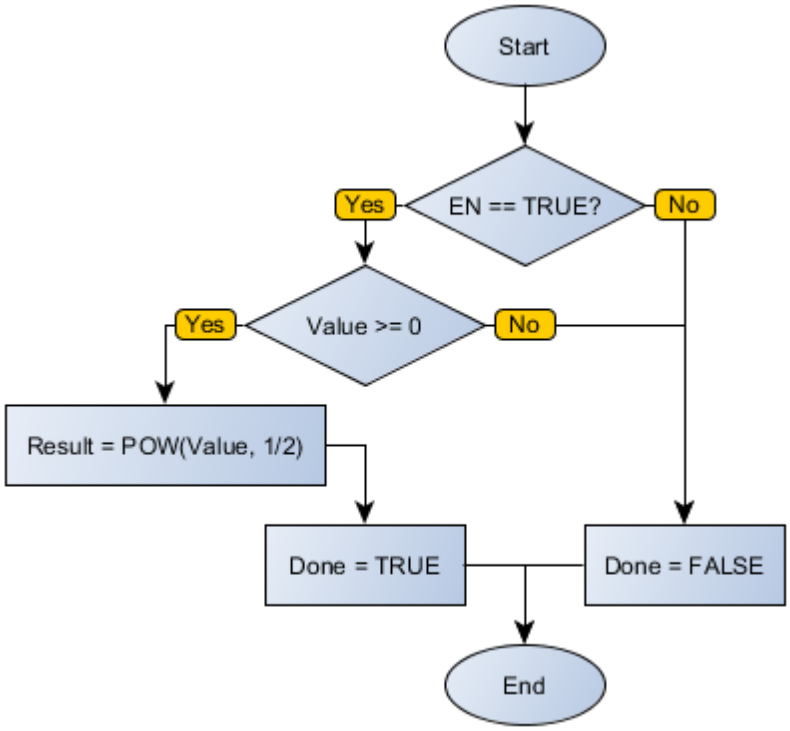
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Reference variable for the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	REAL	Variable that stores the result of the operation

Operation

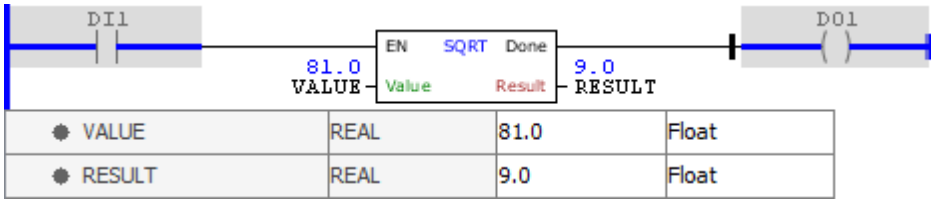
When this block has a TRUE value in EN, it sends to the Result output the square root value of Value. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

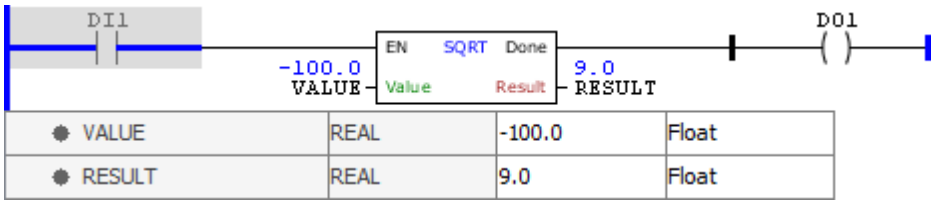


Example



VALUE	REAL	81.0	Float
RESULT	REAL	9.0	Float

The above example calculates the square root value of the VALUE variable, storing the final result in RESULT. The block ends with success and Done output is activated.



VALUE	REAL	-100.0	Float
RESULT	REAL	9.0	Float

The above example calculates the square root value of the VALUE variable, storing the final result in RESULT. The block generates a runtime error, since VALUE has negative value, and Done output is disabled.

11.8.5.12.2.8 TRUNC

Block that truncates the value of Value, storing the result in Result.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Reference variable for the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	REAL	Variable that stores the result of the operation

Operation

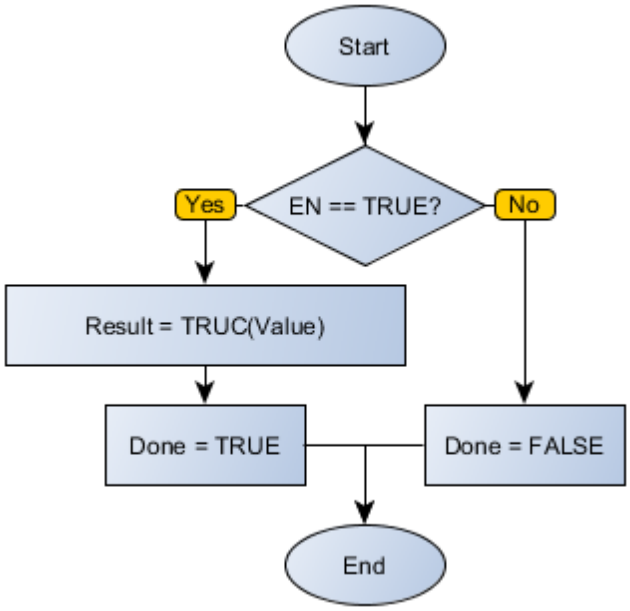
When this block has a TRUE value in EN, it sends to the Result output the truncated value of Value. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

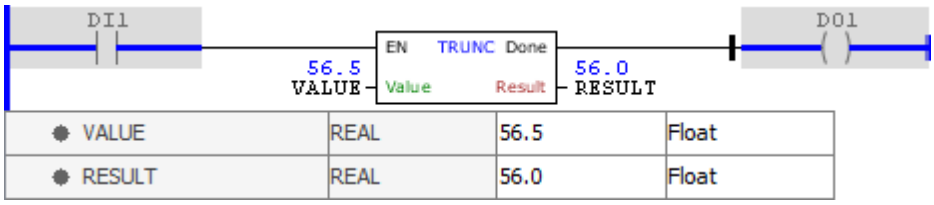
Compatibility

Device	Version
PLC300	2.10 or higher
SCA06	2.00 or higher

Block Flowchart



Example



VALUE	REAL	56.5	Float
RESULT	REAL	56.0	Float

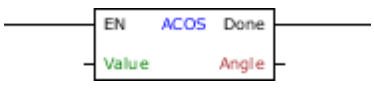
The above example truncates the value of the VALUE variable, storing the final result in RESULT. Decimals are discarded. The block ends with success and Done output is activated.

11.8.5.12.3 Math Trigonometry

11.8.5.12.3.1 ACOS

Block that calculates the arccosine of Value, storing the result in Angle.

Ladder Representation



Block Structure

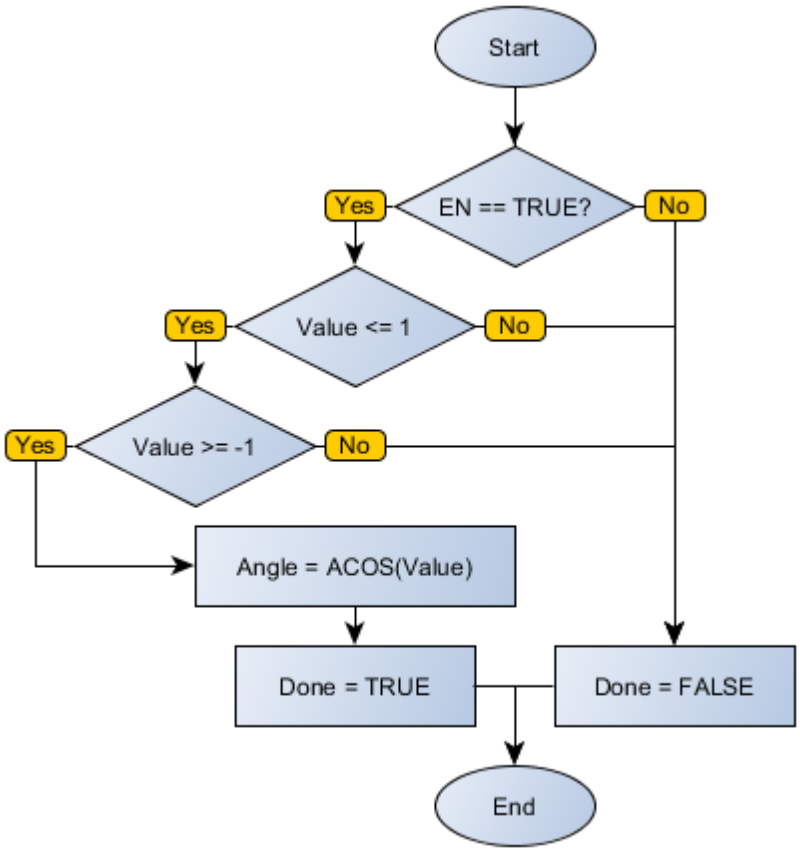
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Value of cosine
VAR_OUTPUT	Done	BOOL	End of operation
	Angle	REAL	Value of the angle w hose cosine is equal to Value (in radians)

Operation

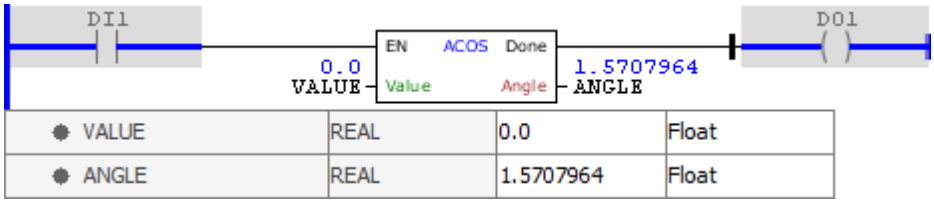
When this block has a TRUE value in EN, it sends to the Angle output the arccosine of Value. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Angle remains with its value unchanged.

When EN has FALSE value, Angle remains unchanged and Done remains in FALSE.

Block Flowchart

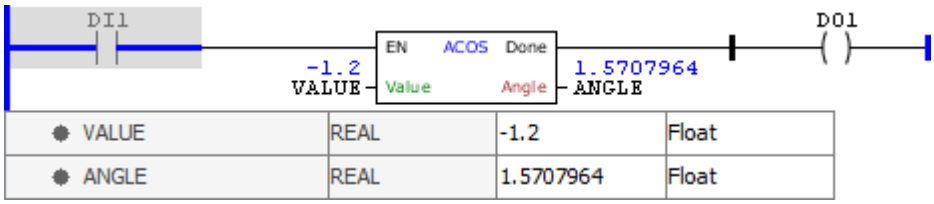


Example



VALUE	REAL	0.0	Float
ANGLE	REAL	1.5707964	Float

The above example calculates the arc, in radians, whose cosine is the VALUE variable, storing the final result in RESULT. The block ends with success and Done output is activated.



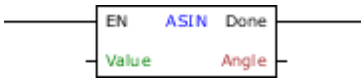
VALUE	REAL	-1.2	Float
ANGLE	REAL	1.5707964	Float

The above example calculates the arc, in radians, whose cosine is the VALUE variable, storing the final result in RESULT. The block generates a runtime error, since VALUE has value inferior to 1, and Done output is disabled.

11.8.5.12.3.2 ASIN

Block that calculates the arcsine of Value, storing the result in Angle.

Ladder Representation



Block Structure

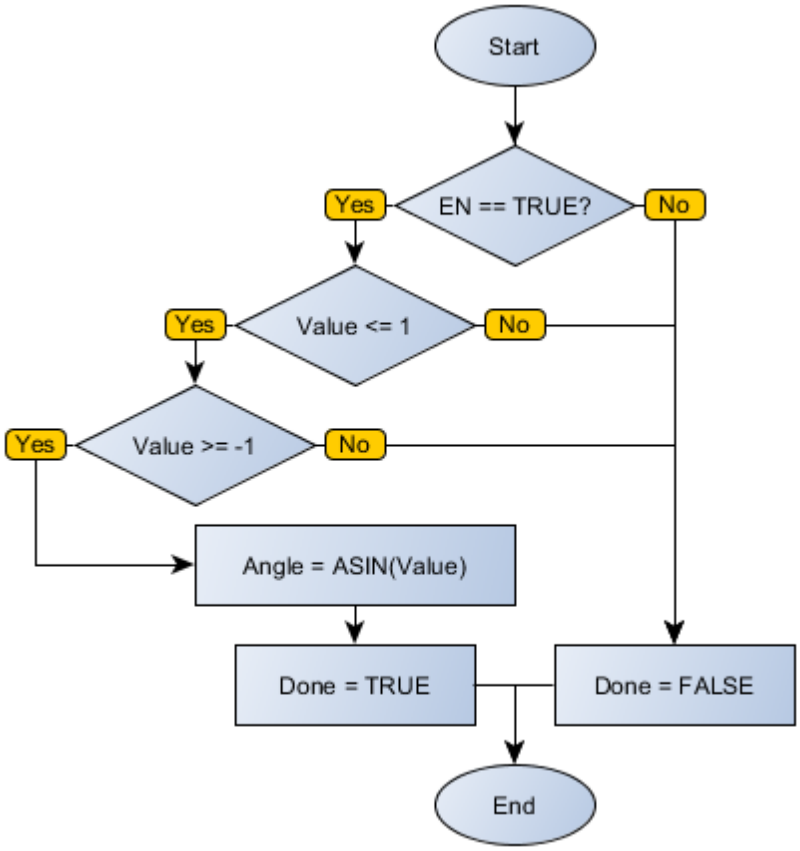
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Value of sine
VAR_OUTPUT	Done	BOOL	End of operation
	Angle	REAL	Value of the angle whose sine is equal to Value (in radians)

Operation

When this block has a TRUE value in EN, it sends to the Angle output the arcsine of Value. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Angle remains with its value unchanged.

When EN has FALSE value, Angle remains unchanged and Done remains in FALSE.

Block Flowchart



Example

VALUE	REAL	-1.0	Float
ANGLE	REAL	-1.5707964	Float

The above example calculates the arc, in radians, whose sine is the VALUE variable, storing the final result in RESULT. The block ends with success and Done output is activated.

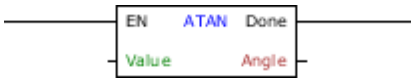
VALUE	REAL	2.0	Float
ANGLE	REAL	-1.5707964	Float

The above example calculates the arc, in radians, whose sine is the VALUE variable, storing the final result in RESULT. The block generates a runtime error, since VALUE has value superior to 1, and Done output is disabled.

11.8.5.12.3.3 ATAN

Block that calculates the arctangent of Value, storing the result in Angle.

Ladder Representation



Block Structure

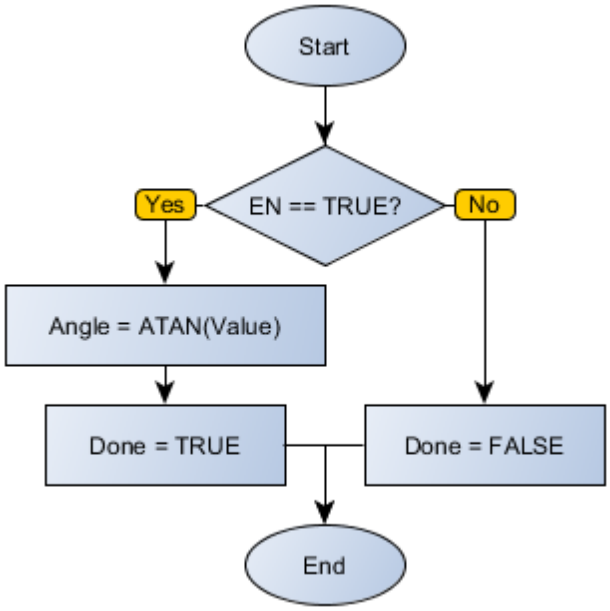
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Value of tangent
VAR_OUTPUT	Done	BOOL	End of operation
	Angle	REAL	Value of the angle whose tangent is equal to Value (in radians)

Operation

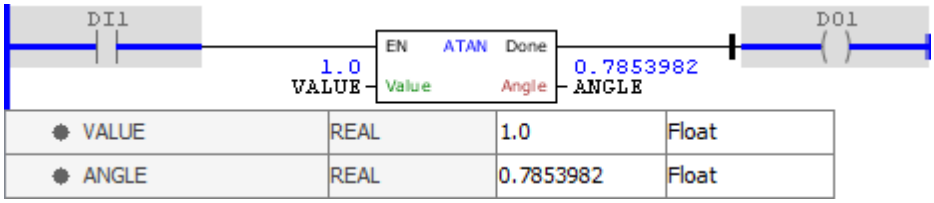
When this block has a TRUE value in EN, it sends to the Angle output the arctangent of Value. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Angle remains with its value unchanged.

When EN has FALSE value, Angle remains unchanged and Done remains in FALSE.

Block Flowchart

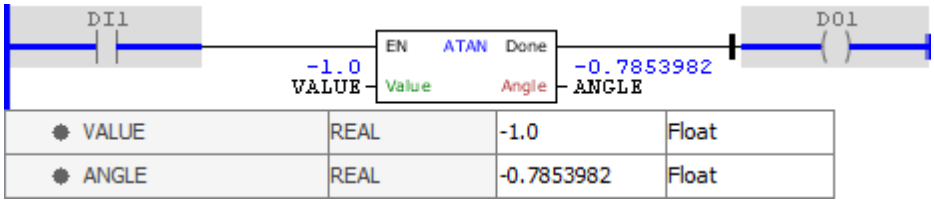


Example



VALUE	REAL	1.0	Float
ANGLE	REAL	0.7853982	Float

The above example calculates the arc, in radians, whose tangent is the VALUE variable, storing the final result in RESULT. The arc, for positive values, is always in the first quadrant. The block ends with success and Done output is activated.



VALUE	REAL	-1.0	Float
ANGLE	REAL	-0.7853982	Float

The above example calculates the arc, in radians, whose tangent is the VALUE variable, storing the final result in RESULT. The arc, for negative values, is always in the fourth quadrant. The block ends with success and Done output is activated.

11.8.5.12.3.4 ATAN2

Block that calculates the arctangent of Y/X, storing the result in Angle.

Ladder Representation

Block Structure

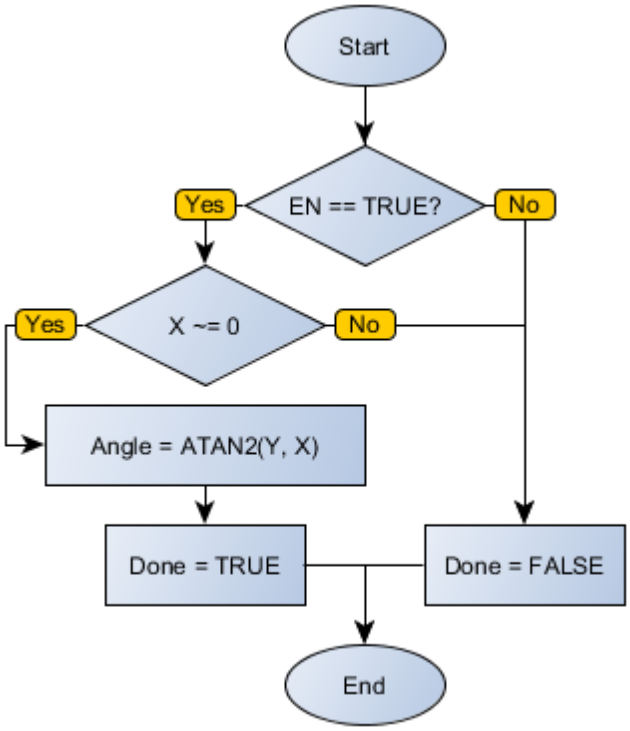
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	X	REAL	Parameter X of the function
	Y	REAL	Parameter Y of the function
VAR_OUTPUT	Done	BOOL	End of operation
	Angle	REAL	Value of the angle whose tangent is equal to (Y/X) (in radians)

Operation

When this block has a TRUE value in EN, it sends to the Angle output the arctangent of Y/X. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Angle remains with its value unchanged.

When EN has FALSE value, Angle remains unchanged and Done remains in FALSE.

Block Flowchart



Example

X	REAL	1.0	Float
Y	REAL	1.0	Float
ANGLE	REAL	0.7853982	Float

The above example calculates the arc, in radians, whose tangent is the Y/X variable, storing the final result in RESULT. The arc, for positive values of X and Y, is always in the first quadrant. The block ends with success and Done output is activated.

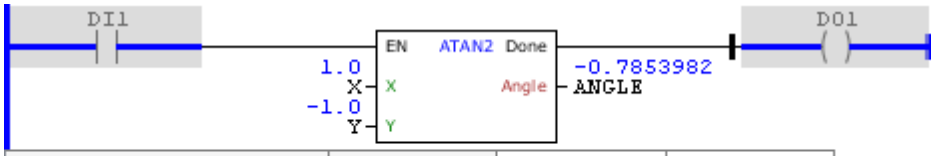
X	REAL	-1.0	Float
Y	REAL	1.0	Float
ANGLE	REAL	2.3561945	Float

The above example calculates the arc, in radians, whose tangent is the Y/X variable, storing the final

result in RESULT. The arc, for negative values of X and positive values of Y, is always in the second quadrant. The block ends with success and Done output is activated.

X	REAL	-1.0	Float
Y	REAL	-1.0	Float
ANGLE	REAL	-2.3561945	Float

The above example calculates the arc, in radians, whose tangent is the Y/X variable, storing the final result in RESULT. The arc, for negative values of X and Y, is always in the third quadrant. The block ends with success and Done output is activated.



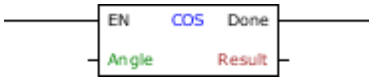
X	REAL	1.0	Float
Y	REAL	-1.0	Float
ANGLE	REAL	-0.7853982	Float

The above example calculates the arc, in radians, whose tangent is the Y/X variable, storing the final result in RESULT. The arc, for positive values of X and negative values of Y, is always in the fourth quadrant. The block ends with success and Done output is activated.

11.8.5.12.3.5 COS

Block that calculates the cosine of Angle, storing the result in Result.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Angle	REAL	Angle (in radians)
VAR_OUTPUT	Done	BOOL	End of operation
	Result	REAL	Variable that stores the result of the operation

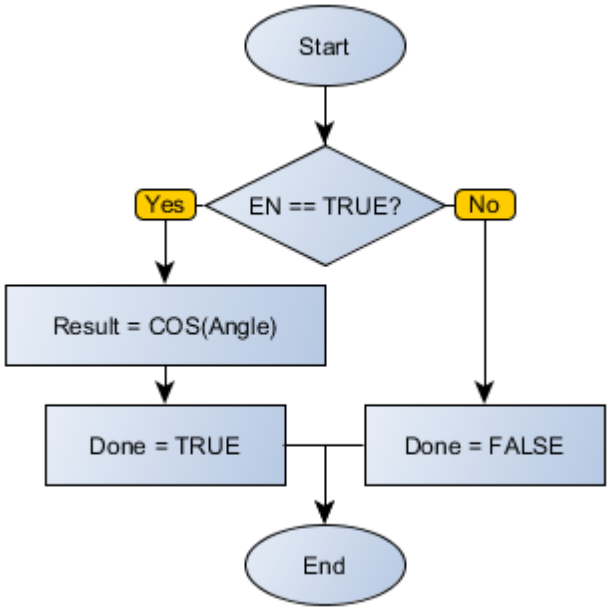
Operation

When this block has a TRUE value in EN, it sends to the Result output the cosine of Angle. If no

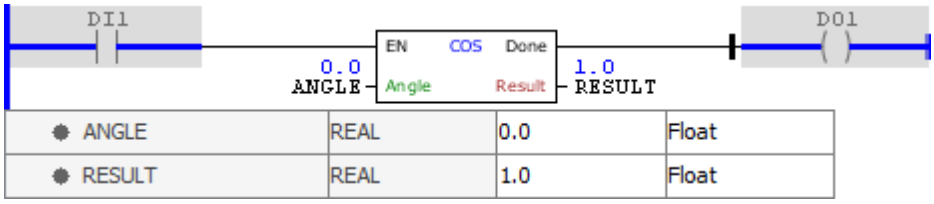
errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

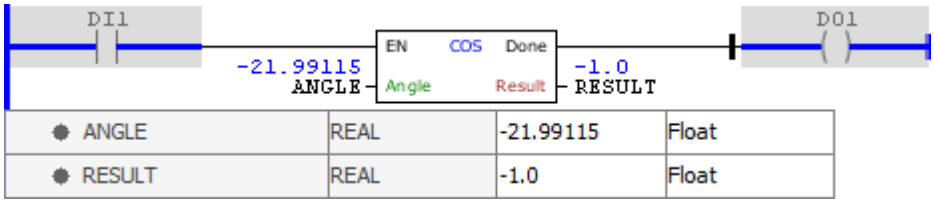


Example



ANGLE	REAL	0.0	Float
RESULT	REAL	1.0	Float

The above example calculates the cosine of the VALUE variable, interpreted in radians, storing the final result in RESULT. The block ends with success and Done output is activated.



ANGLE	REAL	-21.99115	Float
RESULT	REAL	-1.0	Float

The above example calculates the cosine of the VALUE variable, interpreted in radians, storing the final result in RESULT. The block ends with success and Done output is activated. Notice that the block accepts negative input values and greater than one turn.

11.8.5.12.3.6 SIN

Block that calculates the sine of Angle, storing the result in Result.

Ladder Representation

Block Structure

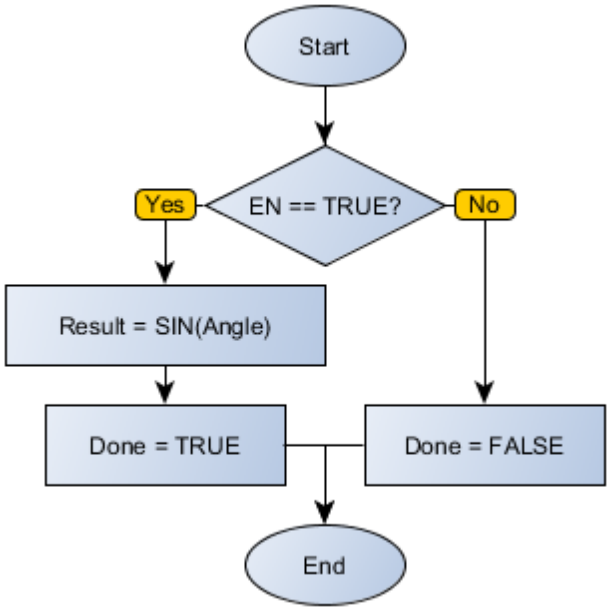
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Angle	REAL	Angle (in radians)
VAR_OUTPUT	Done	BOOL	End of operation
	Result	REAL	Variable that stores the result of the operation

Operation

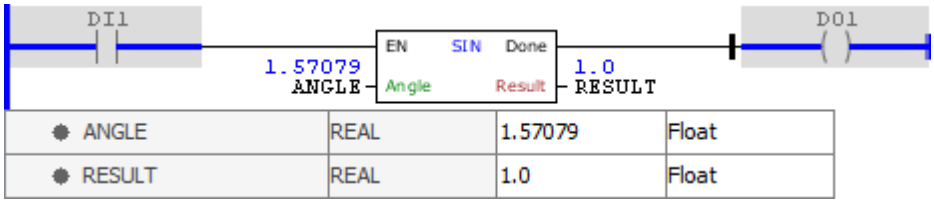
When this block has a TRUE value in EN, it sends to the Result output the sine of Angle. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

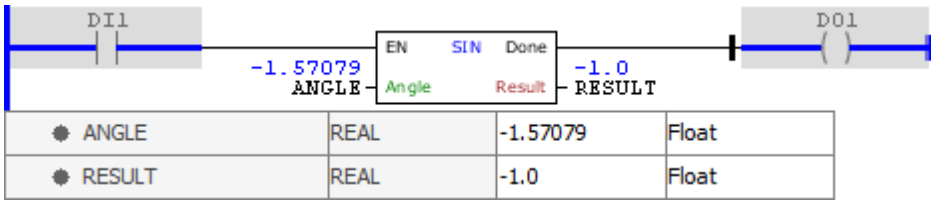


Example



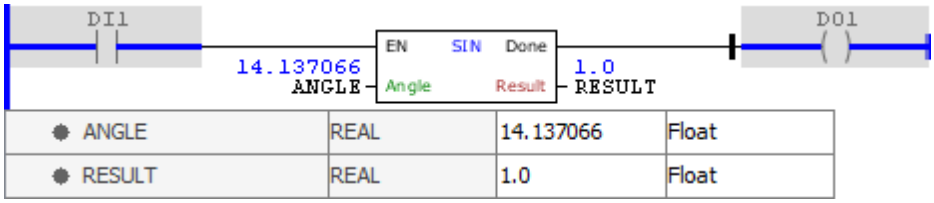
ANGLE	REAL	1.57079	Float
RESULT	REAL	1.0	Float

The above example calculates the sine of the VALUE variable, interpreted in radians, storing the final result in RESULT. The block ends with success and Done output is activated.



ANGLE	REAL	-1.57079	Float
RESULT	REAL	-1.0	Float

The above example calculates the sine of the VALUE variable, interpreted in radians, storing the final result in RESULT. The block ends with success and Done output is activated. Notice that the block accepts negative input values.



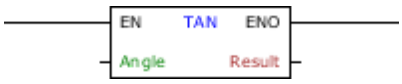
ANGLE	REAL	14.137066	Float
RESULT	REAL	1.0	Float

The above example calculates the sine of the VALUE variable, interpreted in radians, storing the final result in RESULT. The block ends with success and Done output is activated. Notice that the block accepts values greater than one full turn.

11.8.5.12.3.7 TAN

Block that calculates the tangent of Angle, storing the result in Result.

Ladder Representation



Block Structure

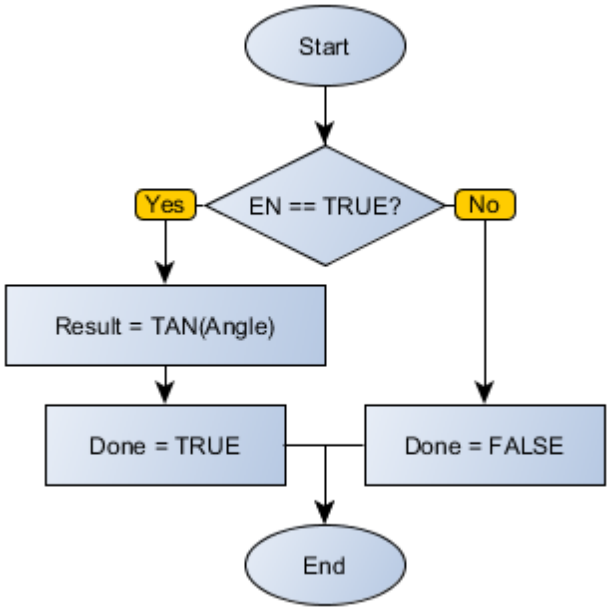
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Angle	REAL	Angle (in radians)
VAR_OUTPUT	Done	BOOL	End of operation
	Result	REAL	Variable that stores the result of the operation

Operation

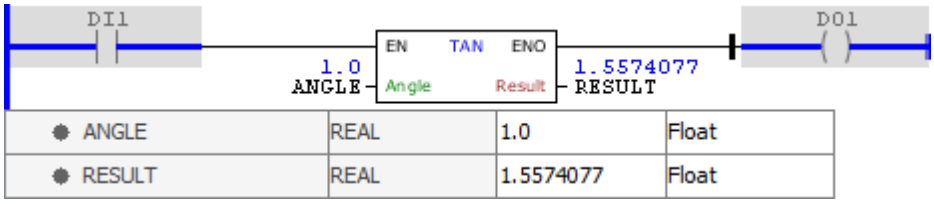
When this block has a TRUE value in EN, it sends to the Result output the tangent of Angle. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

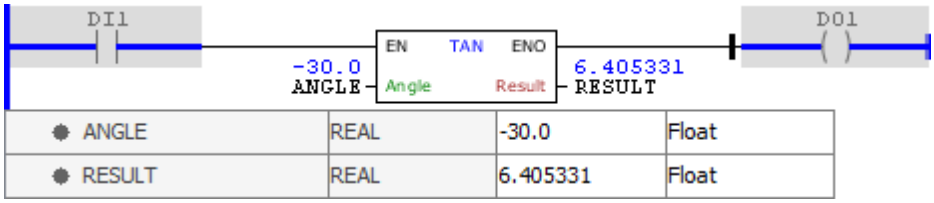


Example



ANGLE	REAL	1.0	Float
RESULT	REAL	1.5574077	Float

The above example calculates the tangent of the VALUE variable, interpreted in radians, storing the final result in RESULT. The block ends with success and Done output is activated.



ANGLE	REAL	-30.0	Float
RESULT	REAL	6.405331	Float

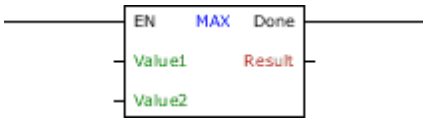
The above example calculates the tangent of the VALUE variable, interpreted in radians, storing the final result in RESULT. The block ends with success and Done output is activated. Notice that the block accepts negative input values and greater than one turn.

11.8.5.12.4 Math Util

11.8.5.12.4.1 MAX

Block that compares the values of Value1 and Value2 and stores the highest of them in Result.

Ladder Representation



Block Structure

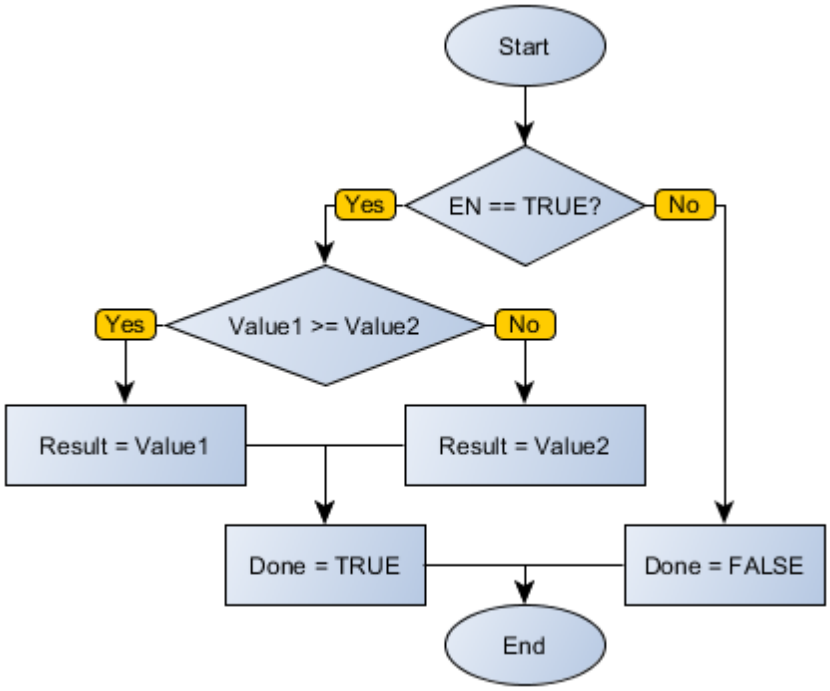
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	First value of comparison
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Second value of comparison
VAR_OUTPUT	Done	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Highest of the values compared

Operation

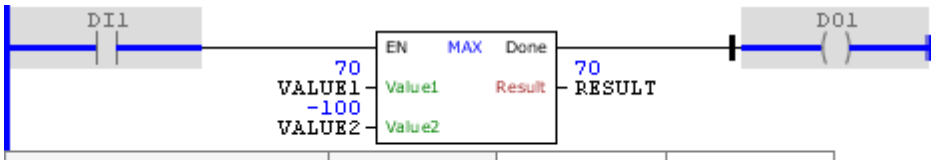
When this block has a TRUE value in EN, it sends to the Result output the highest value in the comparison between Value1 and Value2. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

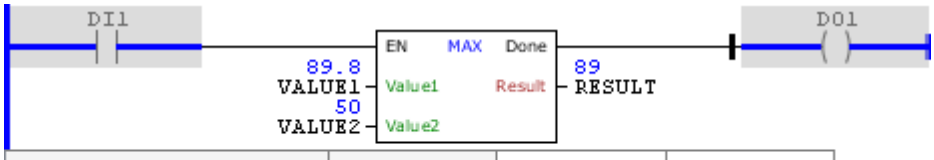


Example



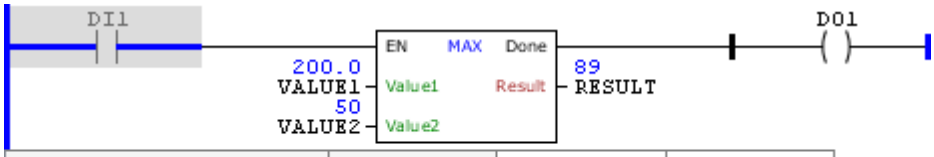
VALUE1	SINT	70	Decimal
VALUE2	SINT	-100	Decimal
RESULT	SINT	70	Decimal

The above example calculates the maximum value between VALUE 1 and VALUE2 variables, storing the final result in RESULT. The block ends with success and Done output is activated.



VALUE1	REAL	89.8	Float
VALUE2	SINT	50	Decimal
RESULT	SINT	89	Decimal

The above example calculates the maximum value between VALUE 1 and VALUE2 variables, storing the final result in RESULT. The block ends with success and Done output is activated. Notice that the types of the input variables can be different without causing execution problems.



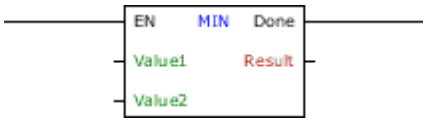
VALUE1	REAL	200.0	Float
VALUE2	SINT	50	Decimal
RESULT	SINT	89	Decimal

The above example calculates the maximum value between VALUE 1 and VALUE2 variables, storing the final result in RESULT. Since the result is higher than the maximum supported by SINT type, the block generates an error and Done output is disabled.

11.8.5.12.4.2 MIN

Block that compares the values of Value1 and Value2 and stores the lowest of them in Result.

Ladder Representation



Block Structure

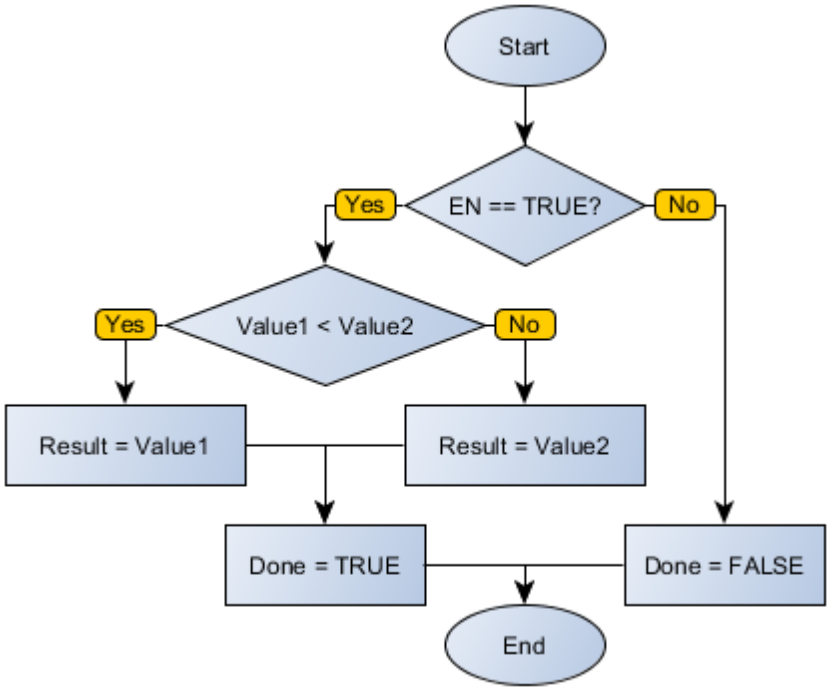
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	First value of comparison
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Second value of comparison
VAR_OUTPUT	Done	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Low est of the values compared

Operation

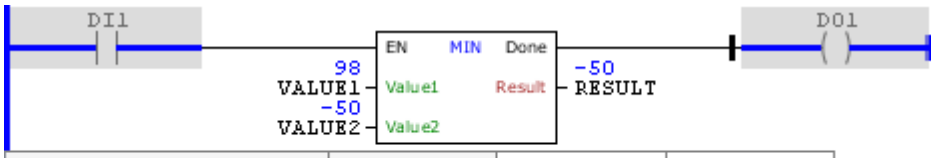
When this block has a TRUE value in EN, it sends to the Result output the lowest value in the comparison between Value1 and Value2. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

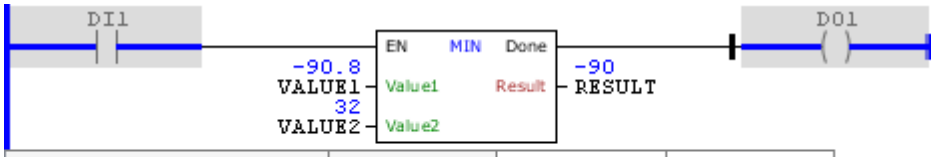


Example



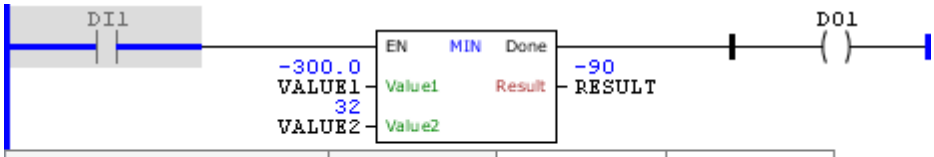
VALUE1	SINT	98	Decimal
VALUE2	SINT	-50	Decimal
RESULT	SINT	-50	Decimal

The above example calculates the minimum value between VALUE 1 and VALUE2 variables, storing the final result in RESULT. The block ends with success and Done output is activated.



VALUE1	REAL	-90.8	Float
VALUE2	SINT	32	Decimal
RESULT	SINT	-90	Decimal

The above example calculates the minimum value between VALUE 1 and VALUE2 variables, storing the final result in RESULT. The block ends with success and Done output is activated. Notice that the types of the input variables can be different without causing execution problems.



VALUE1	REAL	-300.0	Float
VALUE2	SINT	32	Decimal
RESULT	SINT	-90	Decimal

The above example calculates the minimum value between VALUE 1 and VALUE2 variables, storing the final result in RESULT. Since the result is lower than the minimum supported by SINT type, the block generates an error and Done output is disabled.

11.8.5.12.4.3 SAT

Block that performs a routine for saturation of the value found in Value in accordance with the limits for Minimum and Maximum, storing the result in Result.

Ladder Representation

Block Structure

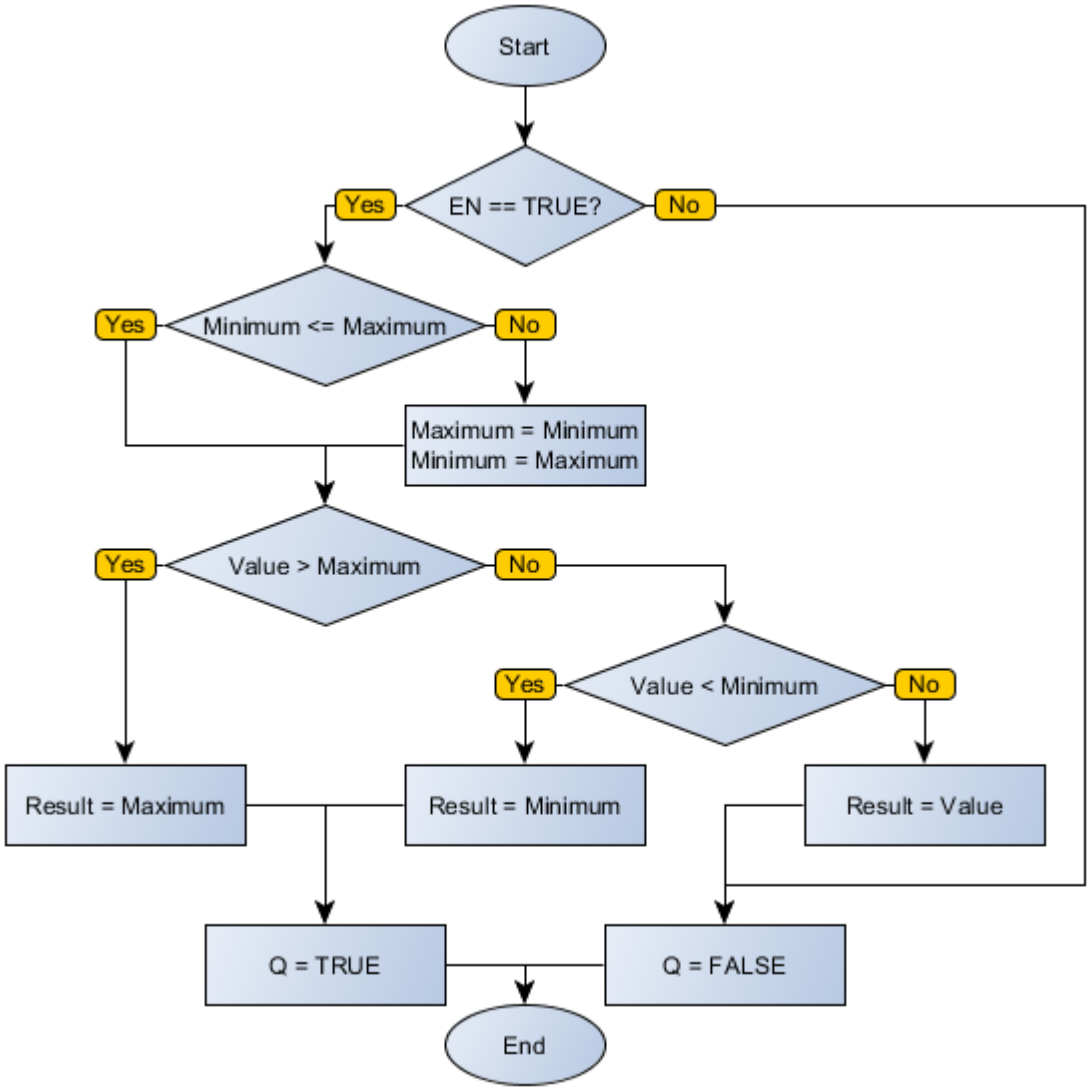
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Reference value
	Minimum	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Inferior saturation value
	Maximum	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Superior saturation value
VAR_OUTPUT	Q	BOOL	Indicator that there w as saturation in the process
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Result of operation

Operation

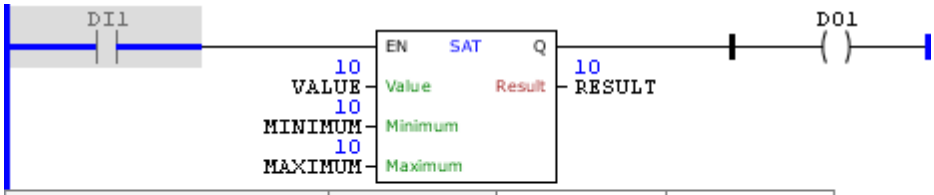
When this block has a TRUE value in EN, it performs a comparison between Value and Minimum and Maximum. If Value is in the range between Minimum and Maximum, Result receives the value of Value and Q remains FALSE. If Value is higher than Maximum, Result receives Maximum and Q receives TRUE. If Value is lower than Minimum, Result receives Minimum and Q receives TRUE. If there is any error in the operation, Q is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Q remains in FALSE.

Block Flowchart

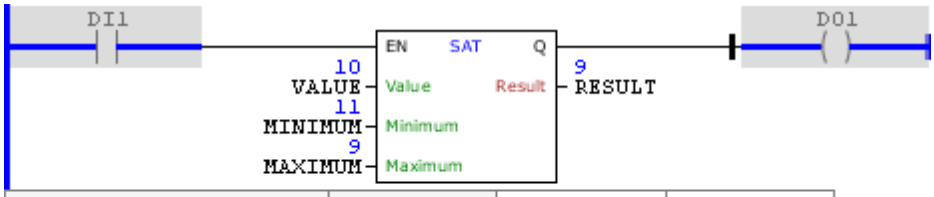


Example



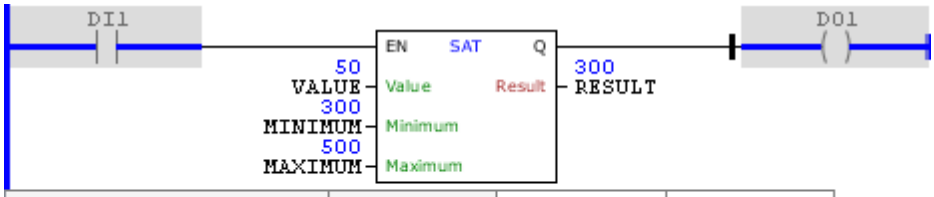
VALUE	INT	10	Decimal
MINIMUM	INT	10	Decimal
MAXIMUM	INT	10	Decimal
RESULT	INT	10	Decimal

The above example passes the VALUE value to RESULT, since it is not lower than MINIMUM or higher than MAXIMUM. The block ends successfully and the Q output is disabled, since there was no saturation.



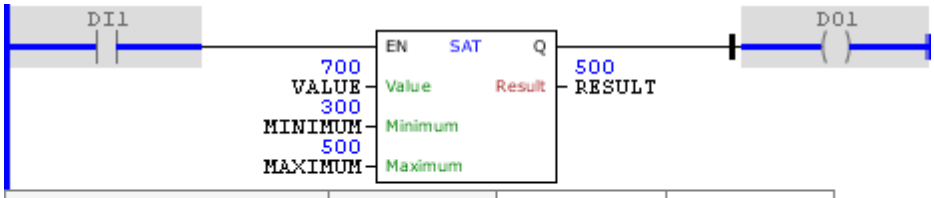
VALUE	INT	10	Decimal
MINIMUM	INT	11	Decimal
MAXIMUM	INT	9	Decimal
RESULT	INT	9	Decimal

The above example passes the MAXIMUM to RESULT, since VALUE is higher than MAXIMUM. The block ends successfully and the Q output is activated, since there was saturation.



VALUE	INT	50	Decimal
MINIMUM	INT	300	Decimal
MAXIMUM	INT	500	Decimal
RESULT	INT	300	Decimal

The above example passes the MINIMUM to RESULT, since VALUE is lower than MINIMUM. The block ends successfully and the Q output is activated, since there was saturation.



VALUE	INT	700	Decimal
MINIMUM	INT	300	Decimal
MAXIMUM	INT	500	Decimal
RESULT	INT	500	Decimal

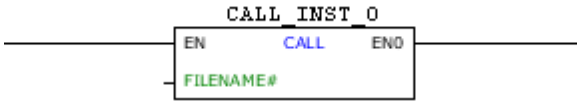
The above example passes the MAXIMUM value to RESULT, since VALUE is higher than MAXIMUM. The block ends successfully and the Q output is activated, since there was saturation.

11.8.5.13 Module

11.8.5.13.1 CALL

Block that loads a file and do a ladder call.

Ladder Representation



Block Structure

Variable Type	Name	Data type	Description
VAR_INPUT	EN	BOOL	Block enabling
	FILENAME#	STRING	Ladder file name (POU) enclosed in gingle quotation marks
VAR_OUTPUT	ENO	BOOL	End of operation
VAR	CALL_INST_0	CALL	Instance of access to block structure

Operation

When this block has a TRUE value in EN, it updates the values of internal fields with the input variables, performs the Ladder routine loading the file and updates the values of the outputs after completing routine.

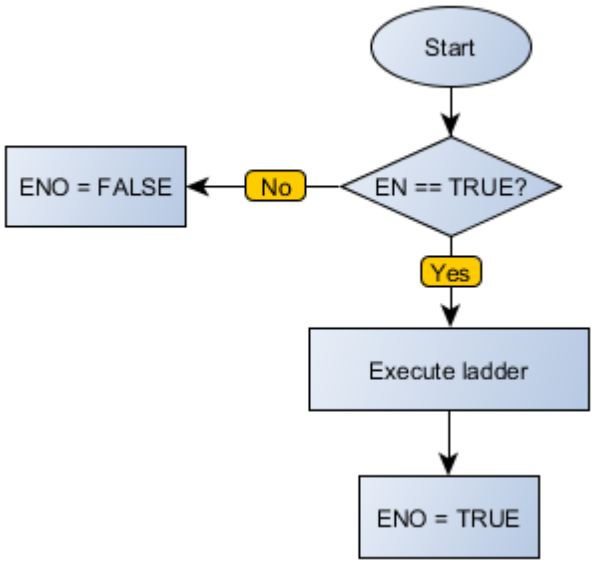
When EN has FALSE value, outputs remain unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Compatibility

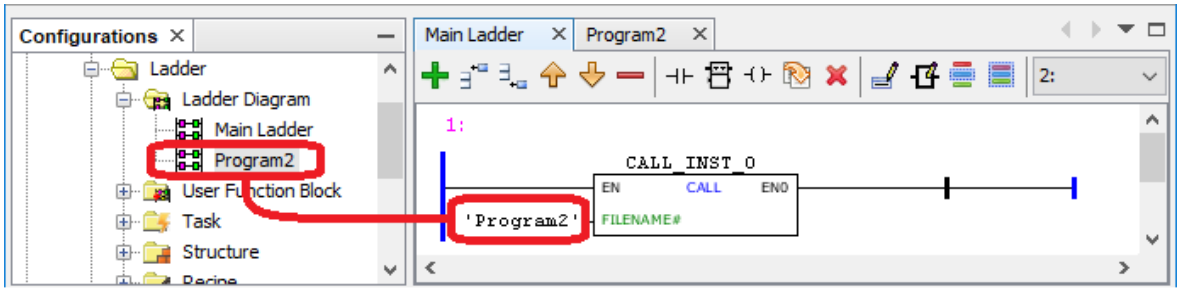
Device	Version
PLC300	4.03 or higher

Block Flowchart



Example

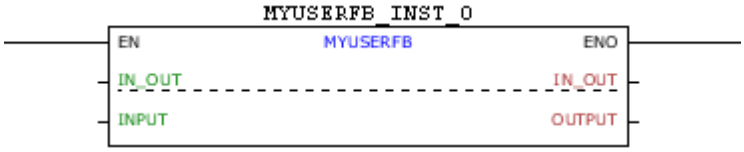
In the example below, the POU 'Program2' will be executed through the 'Main Ladder'.



11.8.5.13.2 USERFB

Block that performs a subroutine programmed by the user.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	INPUT	According to user programming	Block inputs
VAR_OUTPUT	ENO	BOOL	End of operation
	OUTPUT	According to user programming	Block outputs
VAR_IN_OUT	IN_OUT	According to user programming	Block inputs/outputs
VAR	MYUSERFB_INST_0	MYUSERFB	Instance of access to block structure

Operation

When this block has a TRUE value in EN, it updates the values of internal fields with the input variables, performs the Ladder routine programmed by the user and updates the values of the outputs after completing routine.

When EN has FALSE value, outputs remain unchanged.

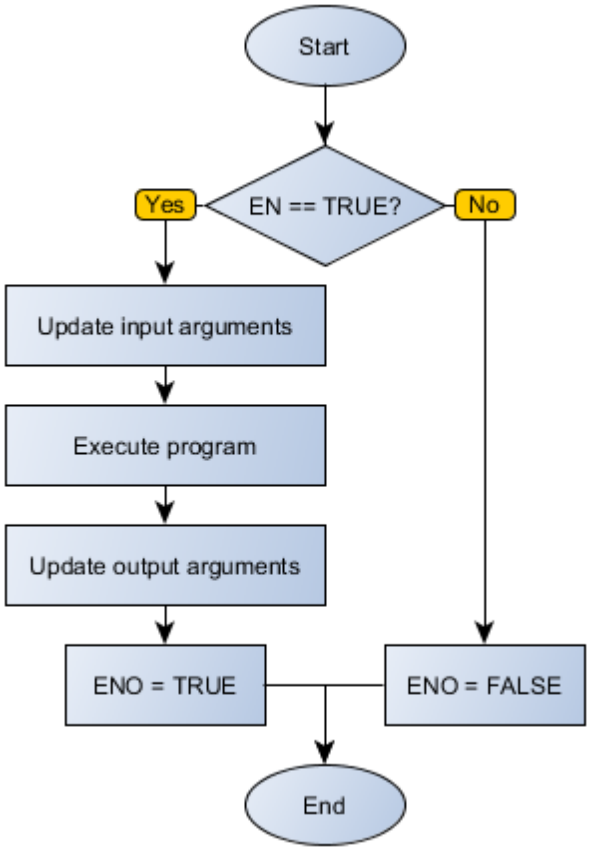
The ENO value forwards to the next Ladder block the EN value after the operation is completed.

NOTE!
Refer to section Working with USERFBs for further information.

Compatibility

Device	Version
PLC300	1.50 or higher
SCA06	2.00 or higher

Block Flowchart

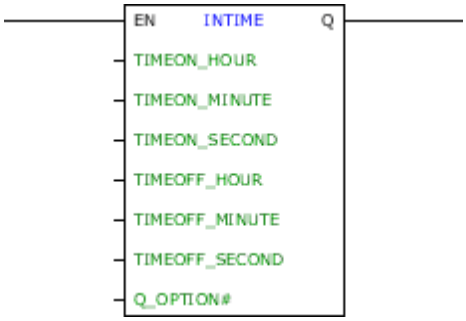


11.8.5.14 RTC

11.8.5.14.1 INTIME

Block that performs a programmed enabling for a time based on RTC (Real Time Clock).

Ladder Representation



Block Structure

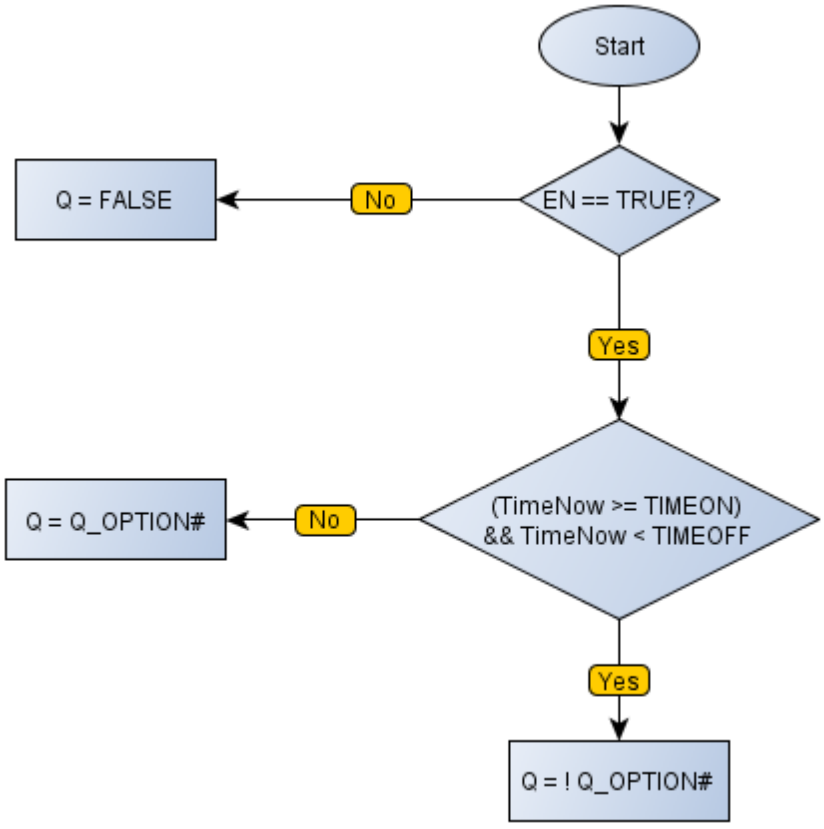
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	TIMEON_HOUR	WORD UINT	Enabling hour
	TIMEON_MINUTE	WORD UINT	Enabling minute
	TIMEON_SECOND	WORD UINT	Enabling second
	TIMEOFF_HOUR	WORD UINT	Disabling hour
	TIMEOFF_MINUTE	WORD UINT	Disabling minute
	TIMEOFF_SECOND	WORD UINT	Disabling second
	Q_OPTION#	BYTE	Output operation
VAR_OUTPUT	Q	BOOL	Block output

Operation

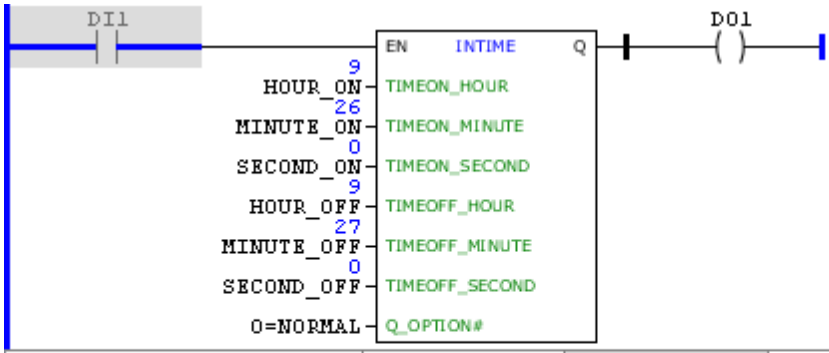
When this block has a TRUE value in EN, it has two modes of operation. If Q_OPTION# is Normal, Q is enabled when the internal clock's time is equal to that defined by the parameters TIMEON and disabled when the internal clock's time is equal to the parameters set by TIMEOFF. If Q_OPTION# is Inverted, Q is disabled when the internal clock's time is equal to that defined by the parameters TIMEON and enabled when the internal clock's time is equal to the parameters set by TIMEOFF.

When EN has FALSE value, Q remains FALSE.

Block Flowchart

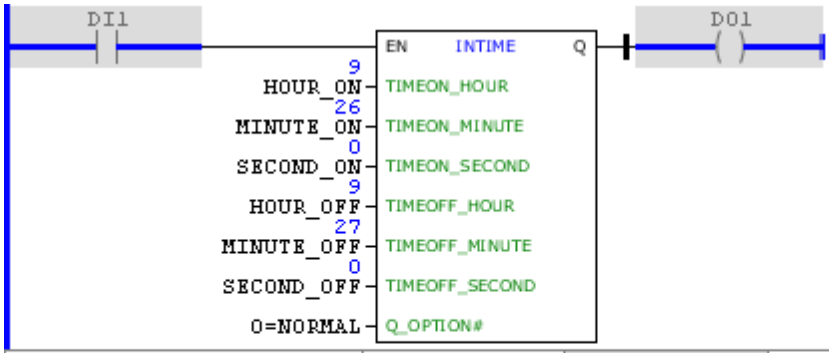


Example



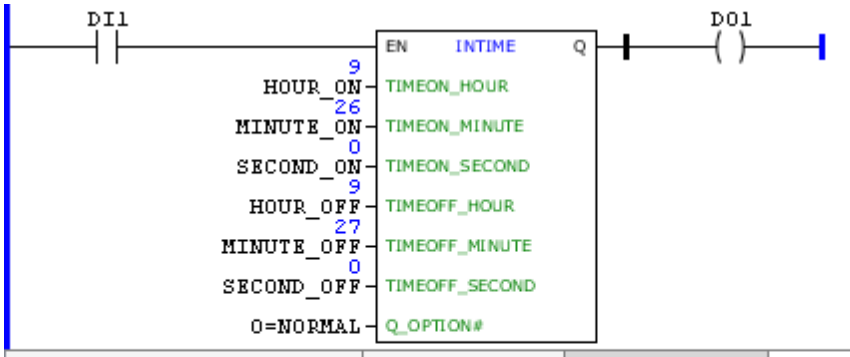
HOUR	UINT	9	Decimal
MINUTE	UINT	25	Decimal
SECOND	UINT	56	Decimal

In the example above, the INTIME block is enabled, the Q_OPTION# input is enabled for NORMAL operation and the current time of the internal clock of the device is lower than the registered enabling inputs of the block (HOUR_ON, MINUTE_ON and SECOND_ON). This way, the Q output is disabled.



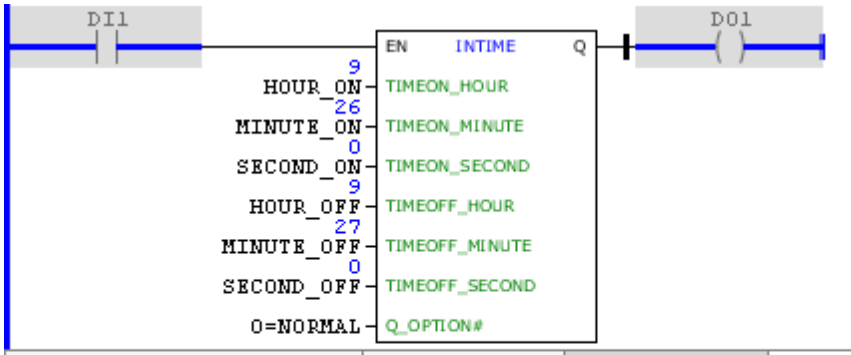
HOUR	UINT	9	Decimal
MINUTE	UINT	26	Decimal
SECOND	UINT	0	Decimal

In the example above, the INTIME block is enabled, the Q_OPTION# input is enabled for NORMAL operation and the current time of the internal clock of the device is equal to the registered in the enabling inputs of the block (HOUR_ON, MINUTE_ON and SECOND_ON). This way, the Q output is disabled.



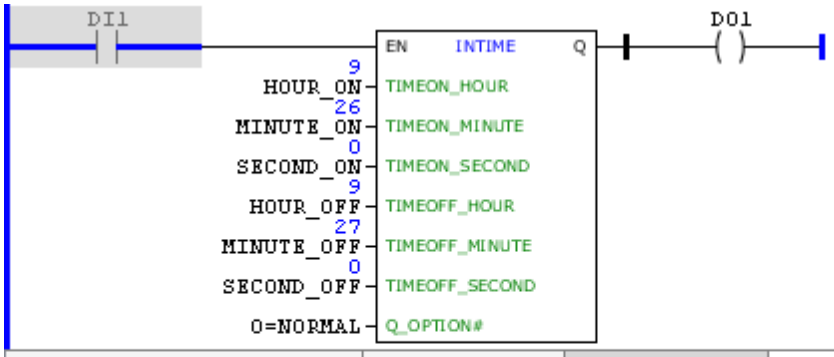
HOUR	UINT	9	Decimal
MINUTE	UINT	26	Decimal
SECOND	UINT	28	Decimal

In the above example, the INTIME block is disabled. This way, regardless of the input, the Q output is disabled.



HOUR	UINT	9	Decimal
MINUTE	UINT	27	Decimal
SECOND	UINT	0	Decimal

In the example above, the INTIME block is enabled, the Q_OPTION# input is enabled for NORMAL operation and the current time of the internal clock of the device is equal to the registered in the disabling inputs of the block (HOUR_OFF, MINUTE_OFF and SECOND_OFF). This way, the Q output is enabled.



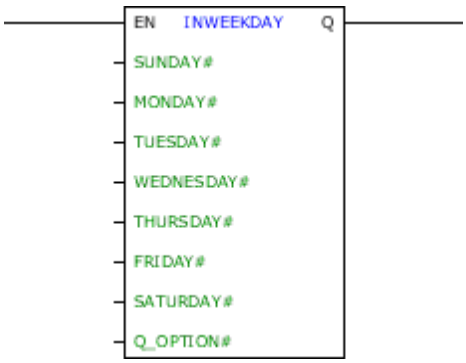
HOUR	UINT	9	Decimal
MINUTE	UINT	27	Decimal
SECOND	UINT	1	Decimal

In the example above, the INTIME block is enabled, the Q_OPTION# input is enabled for NORMAL operation and the current time of the internal clock of the device is superior to the registered in the disabling inputs of the block (HOUR_OFF, MINUTE_OFF and SECOND_OFF). Thus, the Q output is disabled.

11.8.5.14.2 INWEEKDAY

Block that performs a programmed enabling for weekdays based on RTC (Real Time Clock).

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	SUNDAY#	BOOL	Enabled on Sundays
	MONDAY#	BOOL	Enabled on Mondays
	TUESDAY#	BOOL	Enabled on Tuesdays
	WEDNESDAY#	BOOL	Enabled on Wednesdays
	THURSDAY#	BOOL	Enabled on Thursdays
	FRIDAY#	BOOL	Enabled on Fridays
	SATURDAY#	BOOL	Enabled on Saturdays
	Q_OPTION#	BYTE	Output operation
VAR_OUTPUT	Q	BOOL	Block output

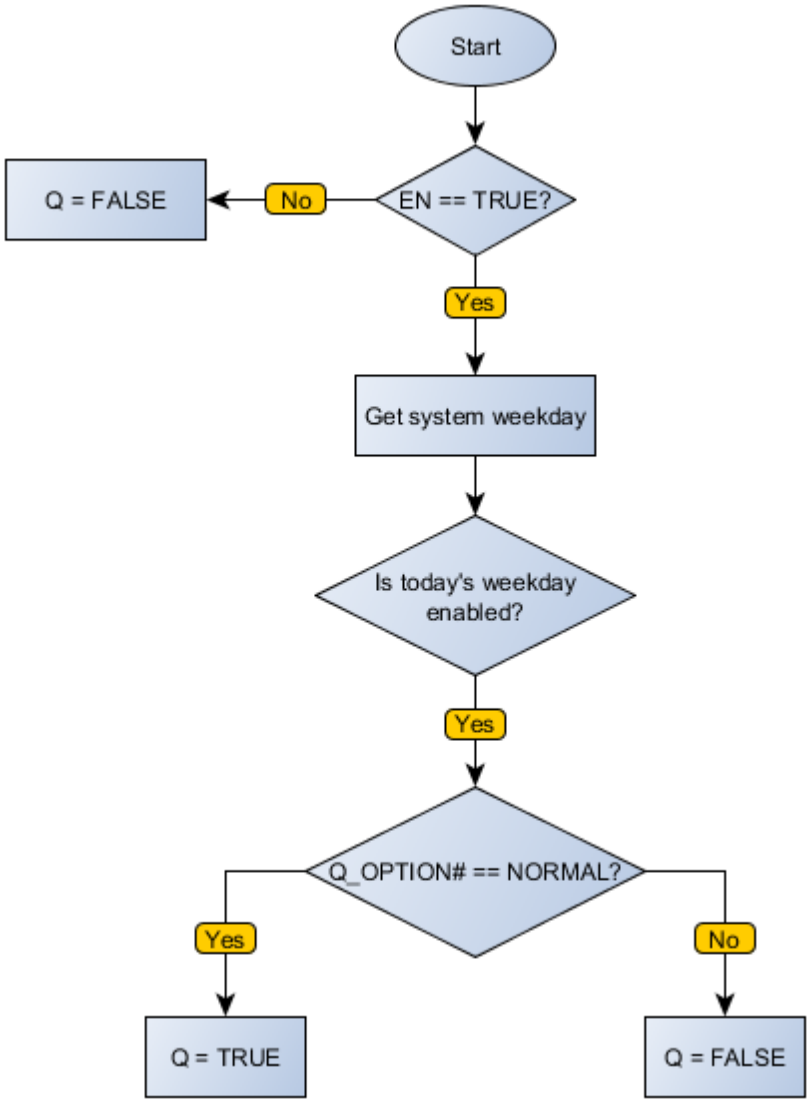
Operation

When this block has a TRUE value in EN, it has two modes of operation. If Q_OPTION# is Normal, Q is enabled if the day of week of the internal clock has Enabled parameter in the block. If Q_OPTION# is Inverted, Q is disabled if the day of week of the internal clock has Enabled parameter in the block.

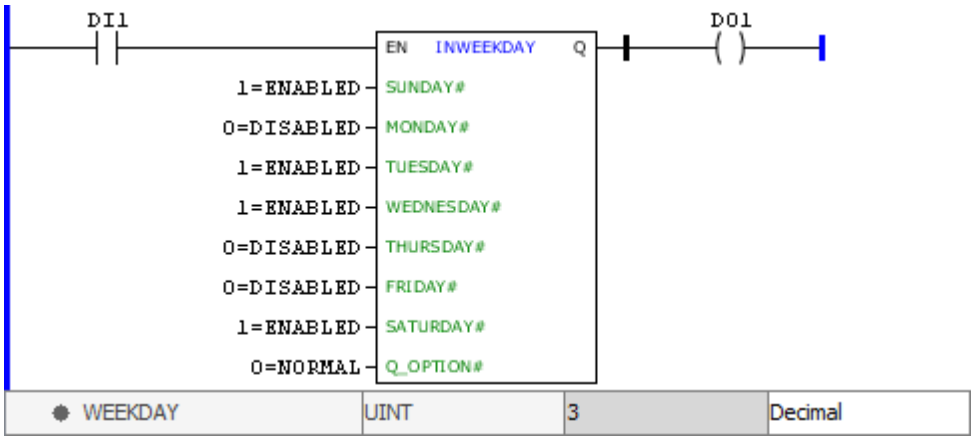
When EN has FALSE value, Q remains FALSE.

NOTE!
The weekdays are identified by numbers, with Sunday being day 0 and Saturday day 6.

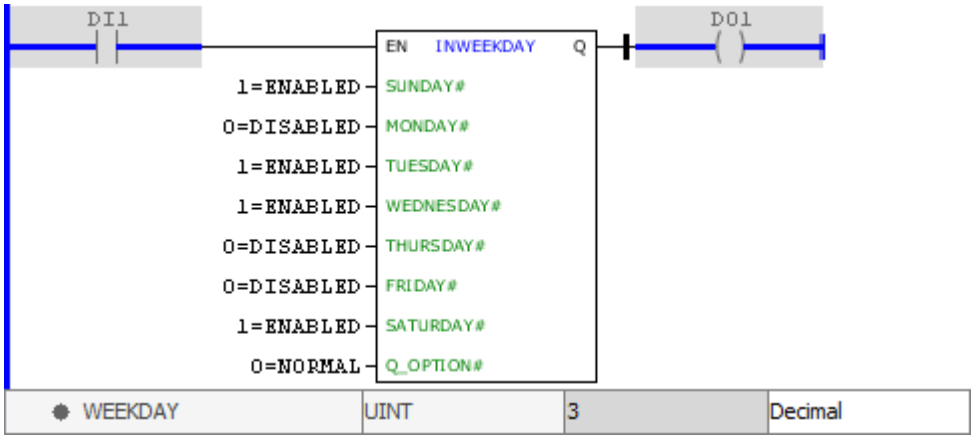
Block Flowchart


Start
EN == TRUE?
No
Q = FALSE
Yes
Get system weekday
Is today's weekday enabled?
Yes
Q_OPTION# == NORMAL?
Yes
No
Q = TRUE
Q = FALSE

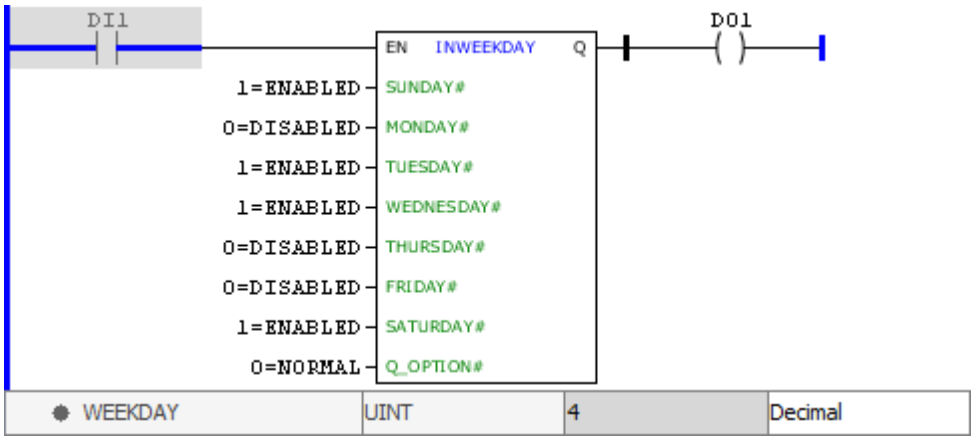
Example


DI1
EN
INWEEKDAY
Q
DO1
1=ENABLED
SUNDAY#
0=DISABLED
MONDAY#
1=ENABLED
TUESDAY#
1=ENABLED
WEDNESDAY#
0=DISABLED
THURSDAY#
0=DISABLED
FRIDAY#
1=ENABLED
SATURDAY#
0=NORMAL
Q_OPTION#
WEEKDAY
UINT
3
Decimal

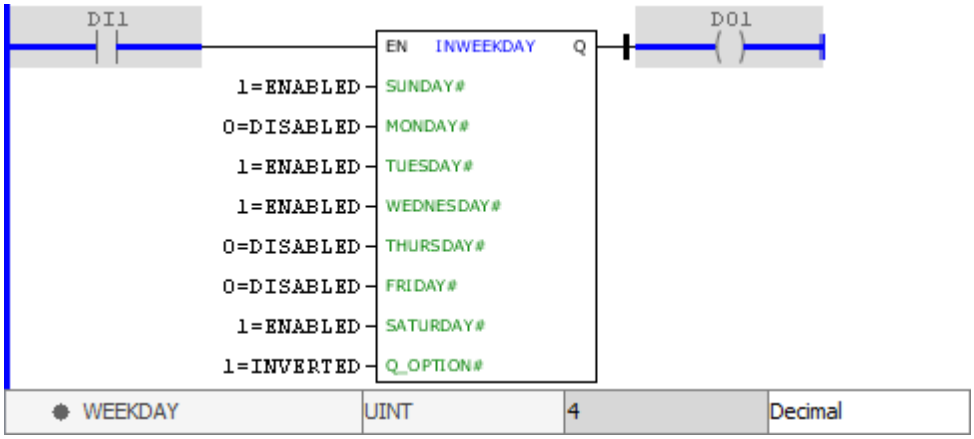
In the above example, the INWEEKDAY block is disabled. This way, regardless of the input, the Q output is disabled.



In the example above, the INWEEKDAY block is enabled and Q_OPTION# input is enabled for NORMAL operation. The current day of the week of the device's internal clock is Wednesday (value 3), which has ENABLED status in the programming. This way, the Q output is enabled.



In the example above, the INWEEKDAY block is enabled and Q_OPTION# input is enabled for NORMAL operation. The current day of the week of the device's internal clock is Thursday (value 4), which has DISABLED status in the programming. Thus, the Q output is disabled.



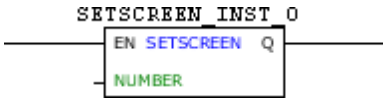
In the example above, the INWEEKDAY block is enabled and Q_OPTION# input is enabled for INVERTED operation. The current day of the week of the device's internal clock is Thursday (value 4), which has DISABLED status in the programming. This way, the Q output is enabled.

11.8.5.15 Screen

11.8.5.15.1 SETSCREEN

Block that displays a particular screen in the HMI.

Ladder Representation



Block Structure

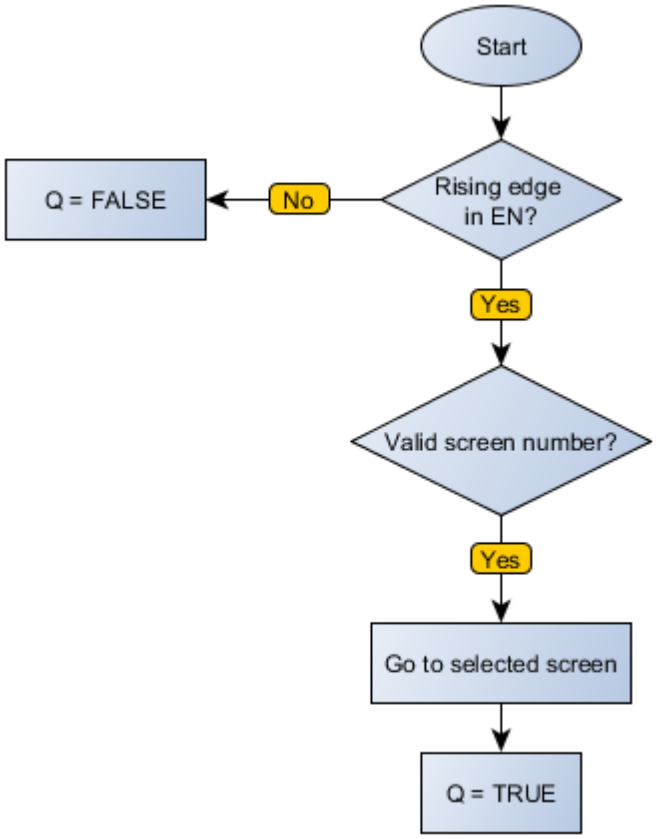
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	NUMBER	BYTE UINT USINT WORD	Screen number to be displayed
VAR_OUTPUT	Q	BOOL	Block output
VAR	SETSCREEN_INST_0	SETSCREEN	Instance of access to block structure

Operation

When this block detects a leading edge on EN, it displays the screen represented by the HMI NUMBER.
Q receives TRUE for one scan cycle if the screen number is valid.

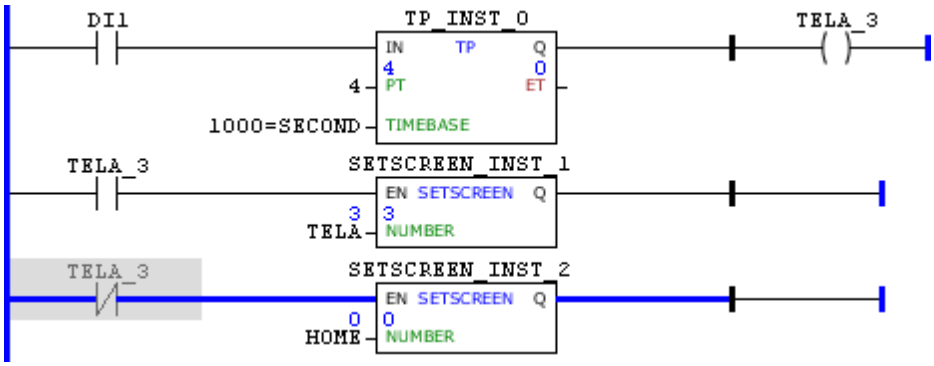
When EN has FALSE value, Q remains FALSE.

Block Flowchart

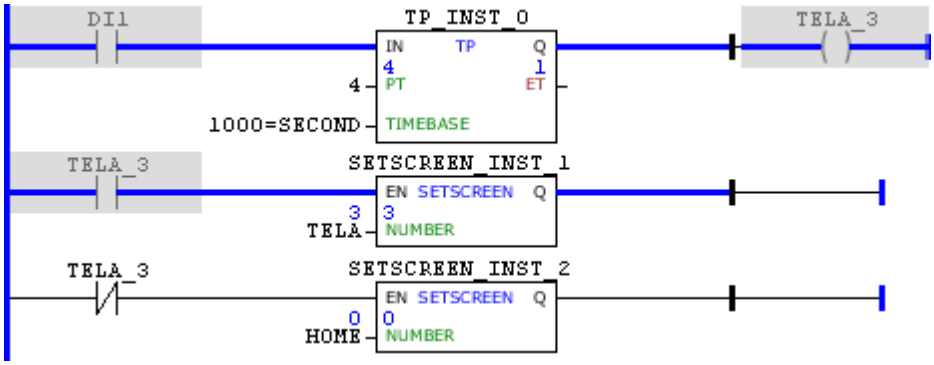


Example

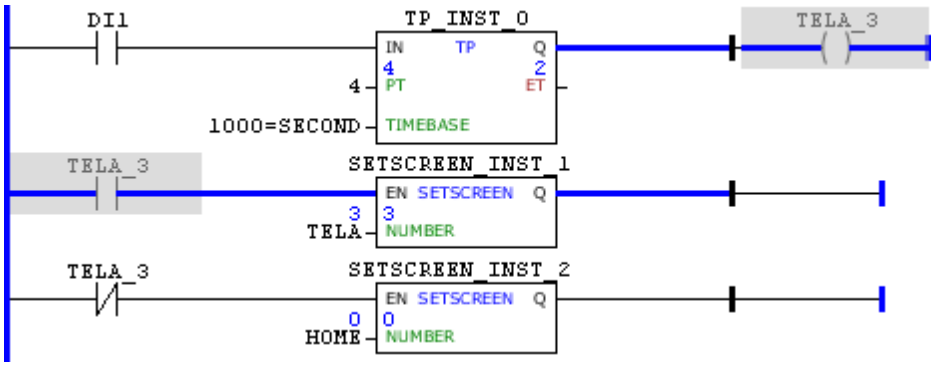
The following example enables screen 3 by 4 seconds at each pulse in DI1.



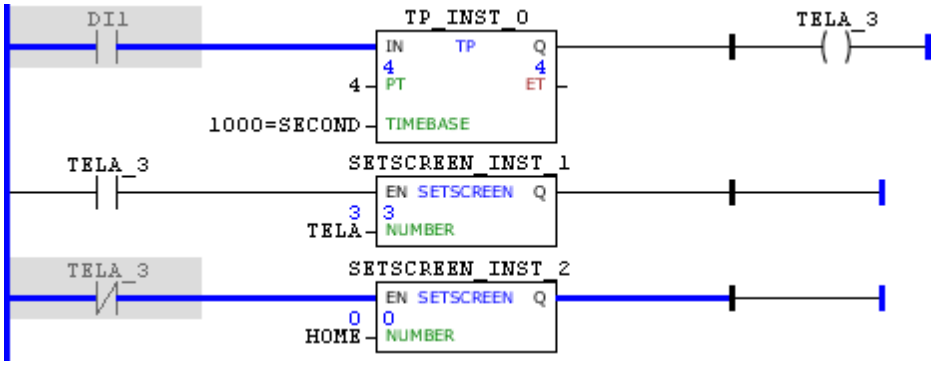
In the beginning, it is considered TELA_3 with FALSE value, activating the input SETSCREEN_INST_0 block that displays the HOME.



By identifying a pulse in DI1, TELA_3 receives TRUE value, activating the input SETSCREEN_INST_1 block that displays screen 3.



Even with the signal DI1 removed, TELA_3 continues with the TRUE value, that is, the screen 3 remains displayed.



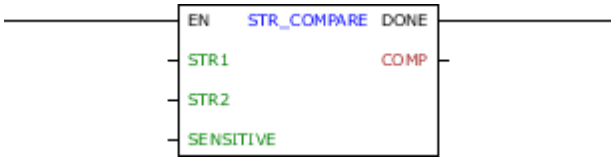
After the four seconds if DI1 is still TRUE, the HOME screen is still displayed. Screen 3 will only be displayed when there is a new leading edge in DI1.

11.8.5.16 String

11.8.5.16.1 STR_COMPARE

Block that performs the comparison between STRINGs.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	STR1	STRING	First STRING of the comparison
	STR2	STRING	Second STRING of the comparison
	SENSITIVE	BOOL	Selects w hether the comparison w ill be case insensitive or not
VAR_OUTPUT	DONE	BOOL	Output enabling
	COMP	BYTE	Value of comparison

Operation

This block remains active as long as EN is at TRUE level by updating the value of COMP according to the input STRINGs. The SENSITIVE input with value TRUE forces the comparison to consider uppercase and lowercase letters, while a FALSE value ignores this differentiation. The values that COMP can take are:

- 0, if both are the same;
- -1, if STR1 comes first in alphabetical order;
- 1, if STR1 comes after in alphabetical order.

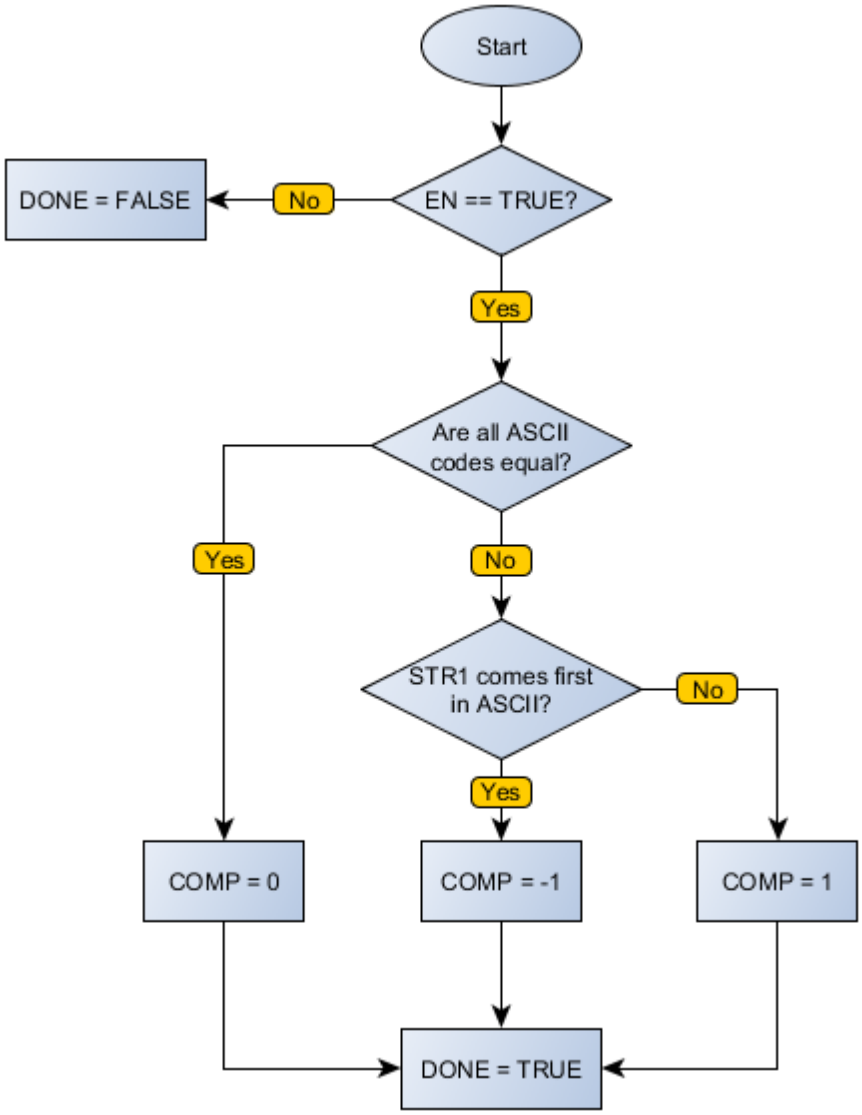
The comparison takes into account the values of the ASCII table for the characters.

The DONE value forwards to the next Ladder block the EN value when the operation is completed.

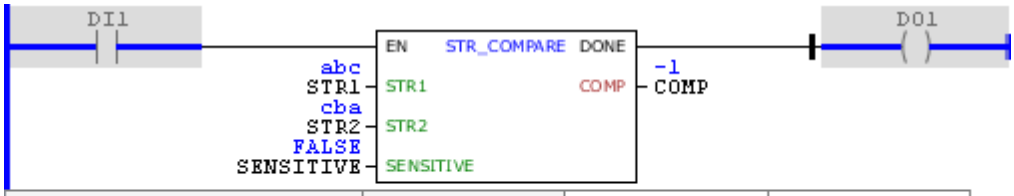
Compatibility

Device	Version
PLC300	2.10 or higher

Block Flowchart



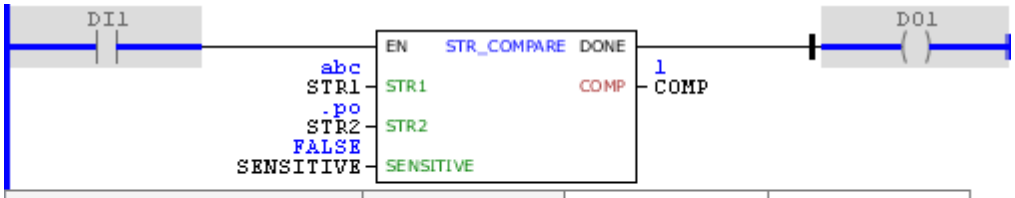
Example



STR1	STRING	abc	String
STR2	STRING	cba	String
SENSITIVE	BOOL	0	Binary
COMP	SINT	-1	Decimal

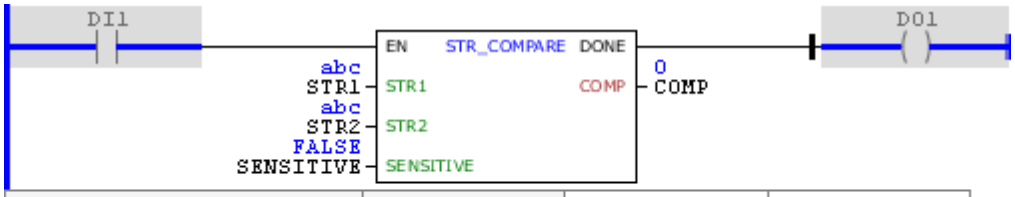
In the above example, the ASCII code for 'a', first character of STR1, is smaller than that of 'c', first

character of STR2. Thus, COMP receives the value -1. When the block is finished successfully, the DONE output is activated.



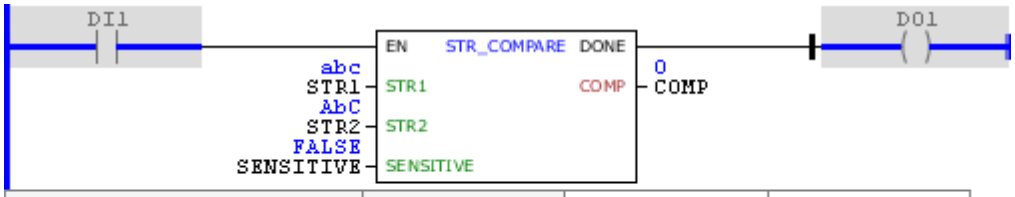
STR1	STRING	abc	String
STR2	STRING	.po	String
SENSITIVE	BOOL	0	Binary
COMP	SINT	1	Decimal

In the above example, the ASCII code for 'a', first character of STR1, is greater than that of '.', first character of STR2. Thus, COMP receives the value 1. When the block is ended successfully, the DONE output is activated.



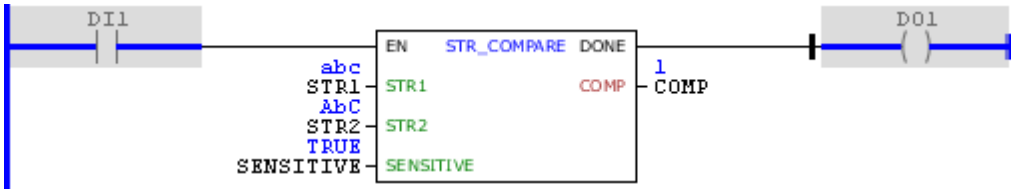
STR1	STRING	abc	String
STR2	STRING	abc	String
SENSITIVE	BOOL	0	Binary
COMP	SINT	0	Decimal

In the example above, STR1 and STR2 are the same. Therefore, COMP receives the value of 0. When the block ends successfully, Done output is activated.



STR1	STRING	abc	String
STR2	STRING	AbC	String
SENSITIVE	BOOL	0	Binary
COMP	SINT	0	Decimal

In the example above, STR1 and STR2 are the same, disregarding uppercase and lowercase (SENSITIVE with FALSE value). Therefore, COMP receives the value of 0. When the block ends successfully, Done output is activated.



In the example above, STR1 and STR2 are the different, considering uppercase and lowercase (SENSITIVE with FALSE value). The ASCII code for 'a', first character of STR1, is greater than that of 'A', first character of STR2. Thus, COMP receives the value 1. When the block is ended successfully, the DONE output is activated.

11.8.5.16.2 STR_CONCAT

Block that performs concatenation of STR1 and STR2, storing the result in DST.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	STR1	STRING	First STRING
	STR2	STRING	Second STRING
VAR_OUTPUT	DONE	BOOL	Output enabling
	DST	BYTE	Variable that receives the new STRING formed from the junction of STR1 and STR2

Operation

This block remains active as long as EN is at TRUE level, updating the value of DST according to the input STRINGs. DST receives STR1 value concatenated with STR2 value at its end.

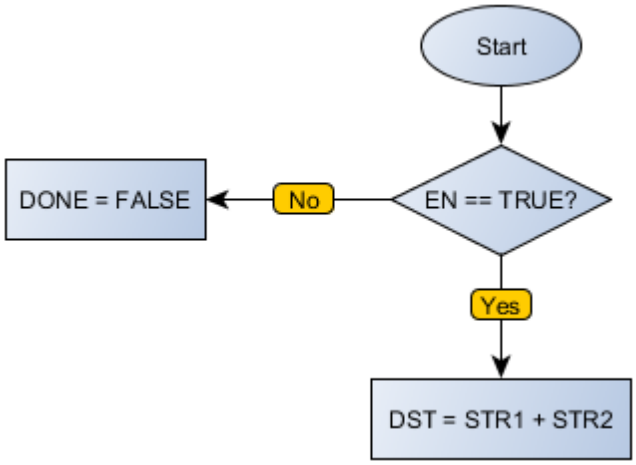
The DONE value forwards to the next Ladder block the EN value when the operation is completed.

NOTE!
If the size of DST is less than the sum of the number of characters STR1 and STR2, the resulting value will be truncated.

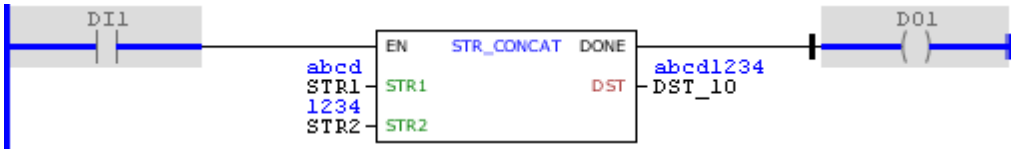
Compatibility

Device	Version
PLC300	2.10 or higher

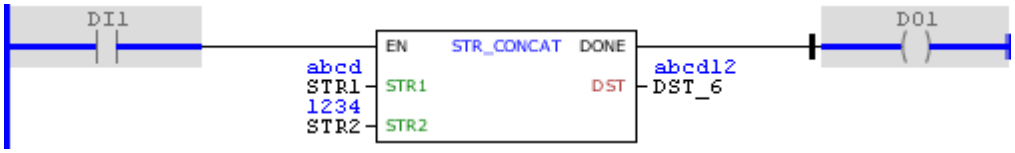
Block Flowchart



Example



In the above example, STR1 and STR2 are concatenated, and the result is sent to DST_10. When the block is ended successfully, the DONE output is activated.



In the above example, STR1 and STR2 are concatenated, and the result is sent to DST_6. Since the size of DST_6 is 6, the last two characters of the concatenation are discarded. The block ends successfully, Done output is activated.

11.8.5.16.3 STR_COPY

Block that performs a copy of a section of STR, storing the result in DST.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	STR	STRING	Original STRING
	POS	BYTE	Position from w hich the copy w ill begin
	LEN	BYTE	Number of characters to be copied from POS
VAR_OUTPUT	DONE	BOOL	Output enabling
	DST	BYTE	Variable that receives the new STRING

Operation

This block remains active as long as EN is at TRUE level, updating the value of DST according to the input parameters. DST receives a number of characters from STR1 defined by LEN from the inserted position in POS.

If POS is outside the allowable range of values (between 1 and the size of STR), DONE receives FALSE and DST remains unchanged.

If successful, the DONE value forwards to the next Ladder block the EN value when the operation is completed.

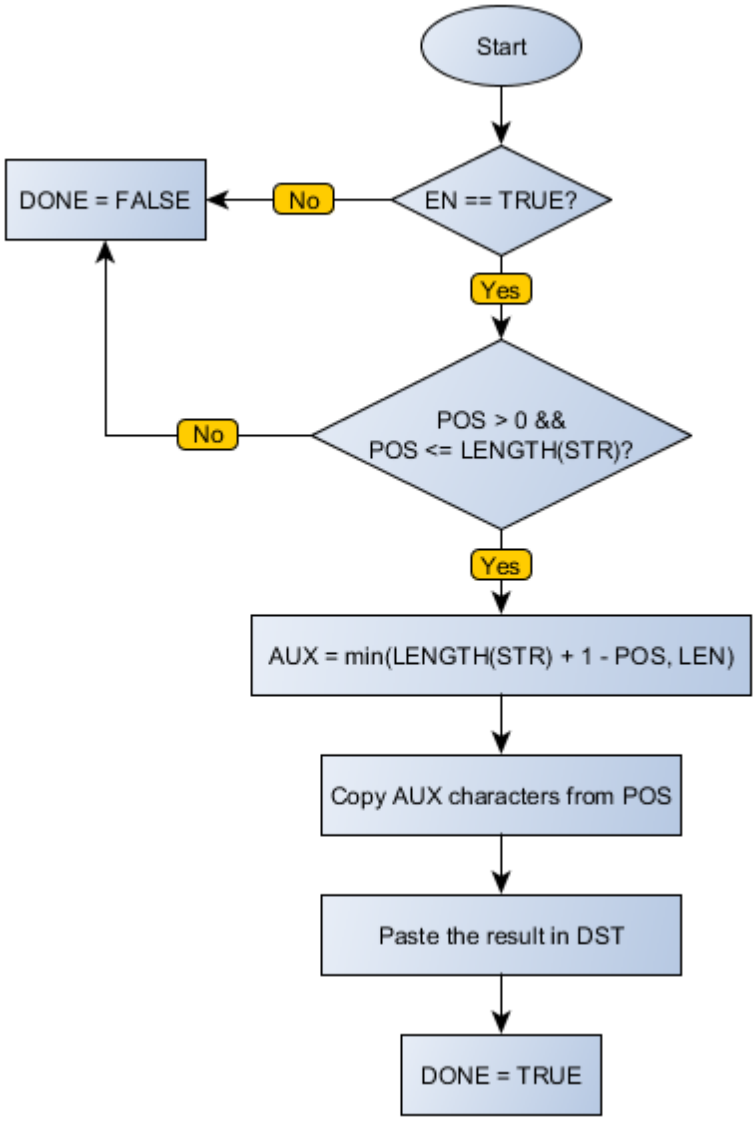
NOTE!
POS is treated with index "based one". That is, POS = 1 references the first position of STR.

NOTE!
If the size of DST is less than the number of characters copied from STR, the resulting value will be truncated.

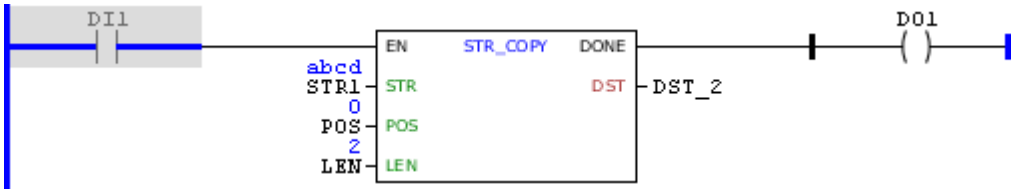
Compatibility

Device	Version
PLC300	2.10 or higher

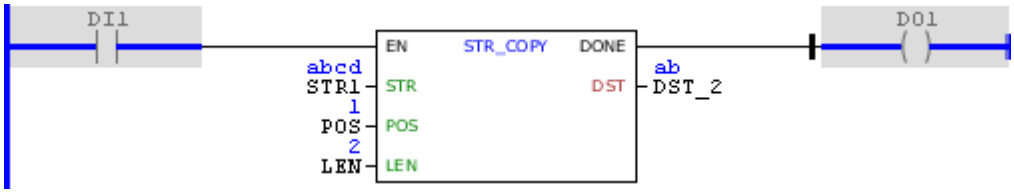
Block Flowchart



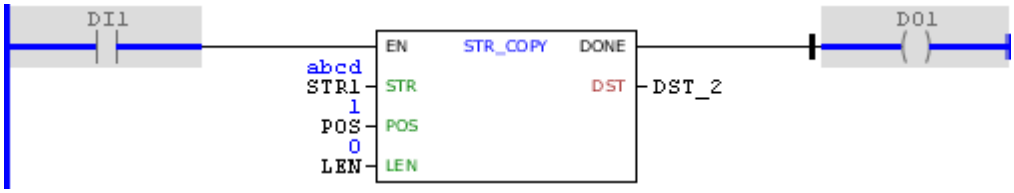
Example



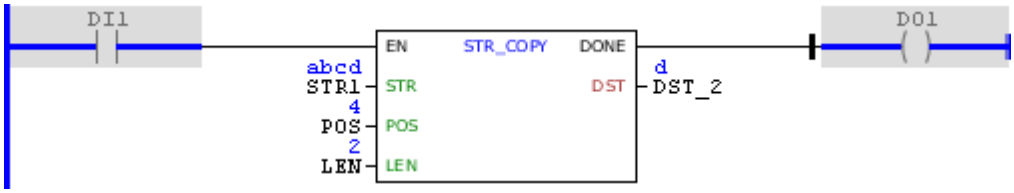
In the above example, as the value of POS is invalid, the block is not completed successfully, and the DONE output is disabled.



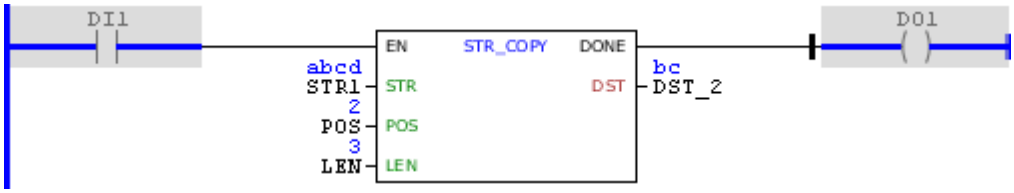
In the example above, two characters are copied from position 1 of STR1, and the result is sent to DST_2. When the block is ended successfully, the DONE output is activated.



In the example above, zero characters are copied from position 1 of STR1, and the result is sent to DST_2. When the block is ended successfully, the DONE output is activated. Note that, when LEN is zero, the output is always built as a null STRING.



In the example above, two characters are copied from position 4 of STR1, and the result is sent to DST_2. When the block is ended successfully, the DONE output is activated. Note that if POS + LEN is greater than the size of STR1, only the remaining characters from STR1 are copied, without generating an error.

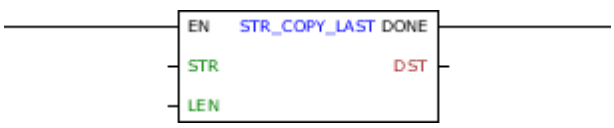


In the example above, three characters are copied from position 2 of STR1, and the result is sent to DST_2. Since the size of DST_2 is 2, the last character copied is discarded. The block ends successfully, Done output is activated.

11.8.5.16.4 STR_COPY_LAST

Block that performs a copy of a final section of STR, storing the result in DST.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	STR	STRING	Original STRING
	LEN	BYTE	Number of characters to be copied from the end of STR
VAR_OUTPUT	DONE	BOOL	Output enabling
	DST	BYTE	Variable that receives the new STRING

Operation

This block remains active as long as EN is at TRUE level, updating the value of DST according to the input parameters. DST receives a number of characters from STR1 defined by LEN from the final position of STR, returning toward the beginning.

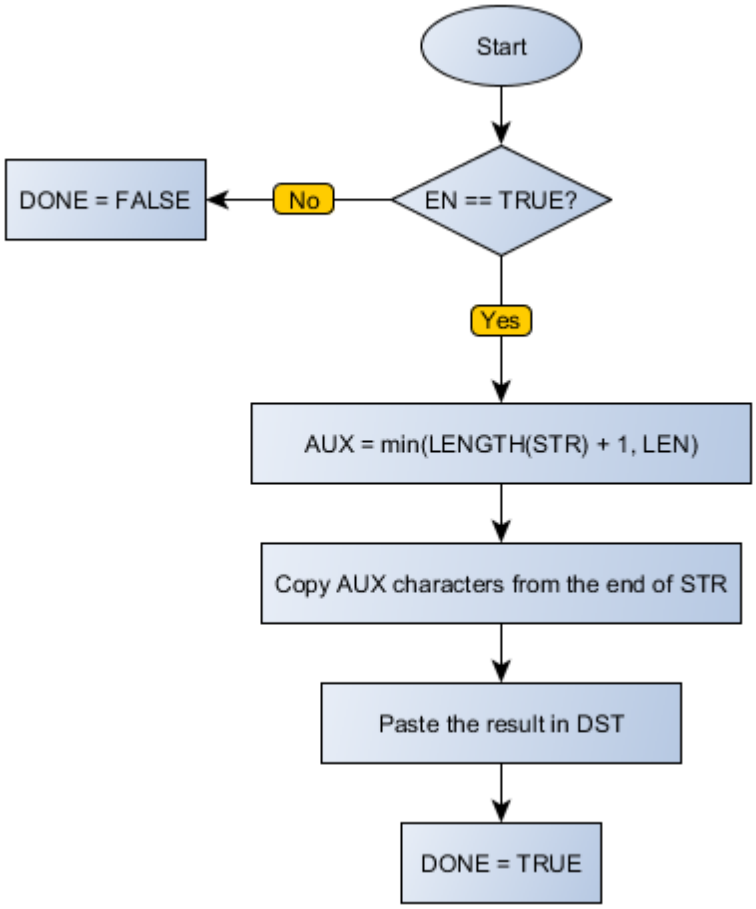
The DONE value forwards to the next Ladder block the EN value when the operation is completed.

NOTE!
If the size of DST is less than the number of characters copied from STR, the resulting value will be truncated.

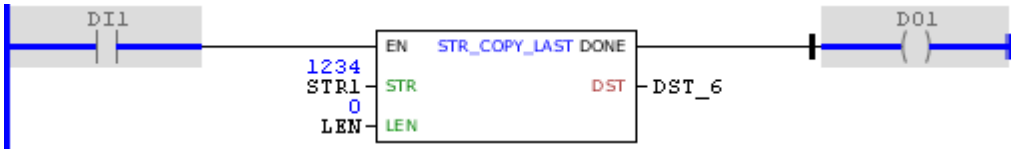
Compatibility

Device	Version
PLC300	2.10 or higher

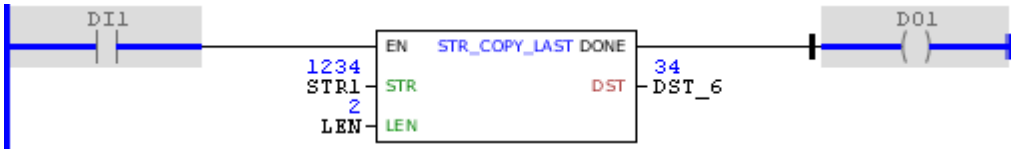
Block Flowchart



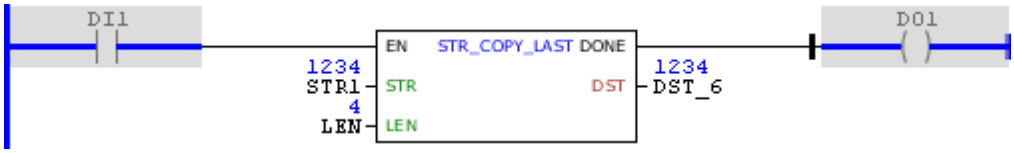
Example



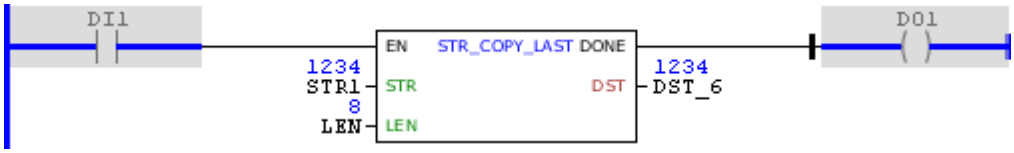
In the example above, zero characters are copied from final position STR1, and the result is sent to DST_6. When the block is ended successfully, the DONE output is activated. Note that, when LEN is zero, the output is always built as a null STRING.



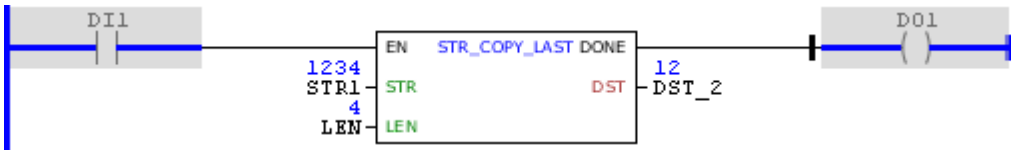
In the example above, two characters are copied from final position STR1, and the result is sent to DST_6. When the block is ended successfully, the DONE output is activated.



In the example above, four characters are copied from final position of STR1, and the result is sent to DST_6. When the block is ended successfully, the DONE output is activated.



In the example above, eight characters are copied from final position STR1, and the result is sent to DST_6. When the block is ended successfully, the DONE output is activated. Note that if LEN is greater than the size of STR1, STR1 is copied in full, without generating an error.

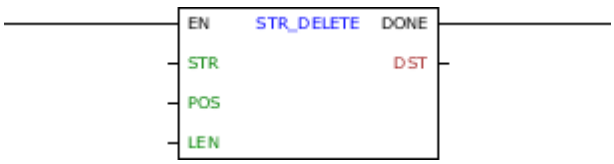


In the example above, four characters are copied from final position of STR1, and the result is sent to DST_2. Since the size of DST_2 is 2, the last two characters copied are discarded. The block ends successfully, Done output is activated.

11.8.5.16.5 STR_DELETE

Block that deletes part of a section of STR, storing the remainder in DST.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	STR	STRING	Original STRING
	POS	BYTE	Position from w hich the removal w ill begin
	LEN	BYTE	Number of characters to be removed from POS
VAR_OUTPUT	DONE	BOOL	Output enabling
	DST	BYTE	Variable that receives the new STRING

Operation

This block remains active as long as EN is at TRUE level, updating the value of DST according to the input parameters. A section defined by the POS initial position is removed from STR and a LEN size, and the final result is stored in DST.

If POS is outside the allowable range of values (between 1 and the size of STR), DONE receives FALSE and DST remains unchanged.

If successful, the DONE value forwards to the next Ladder block the EN value when the operation is completed.

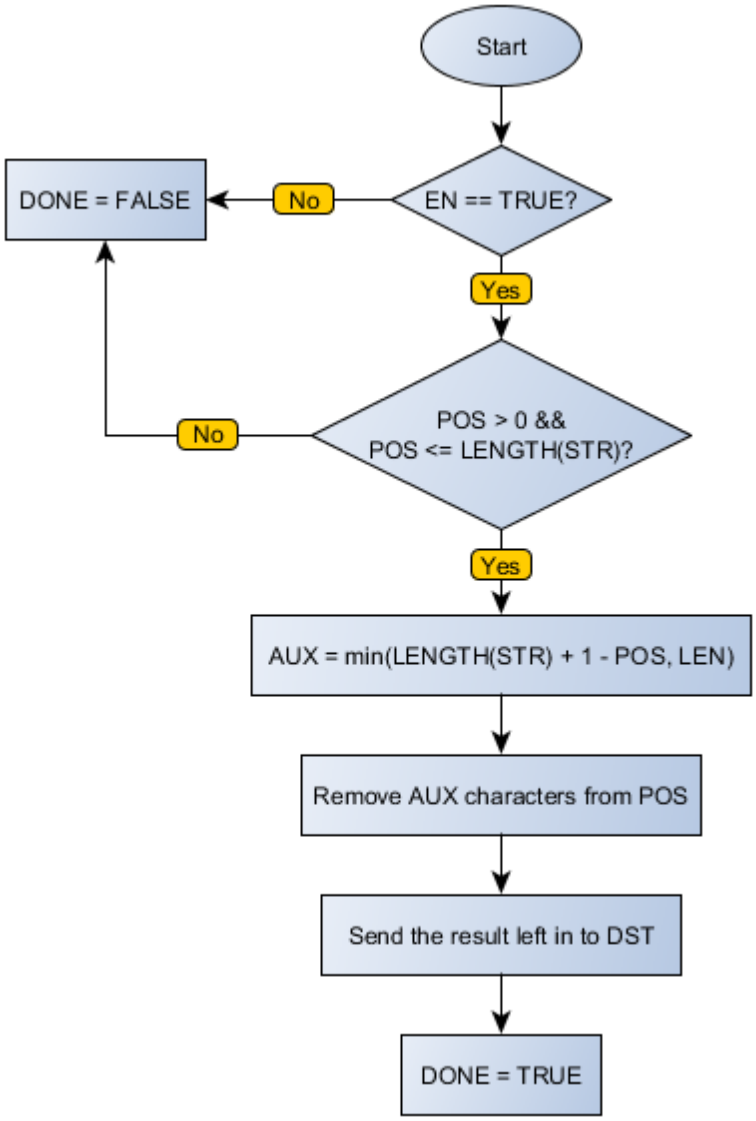
NOTE!
POS is treated with index "based one". That is, POS = 1 references the first position of STR.

NOTE!
If the size of DST is less than the sum of the number of remaining characters in STR, the resulting value will be truncated.

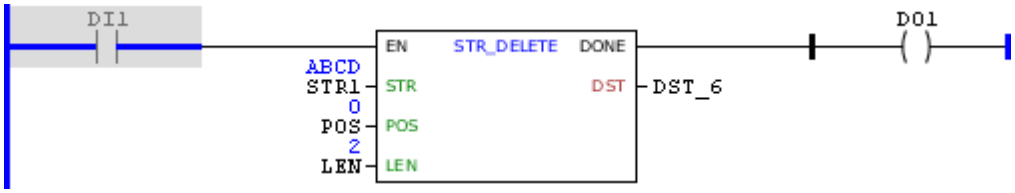
Compatibility

Device	Version
PLC300	2.10 or higher

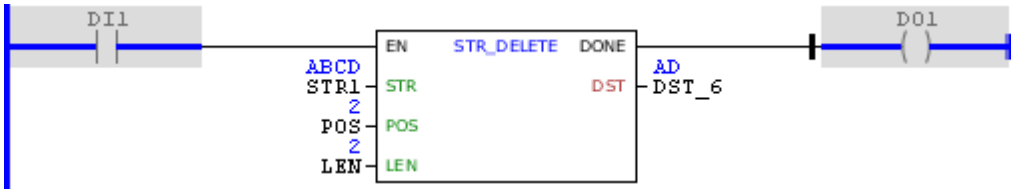
Block Flowchart



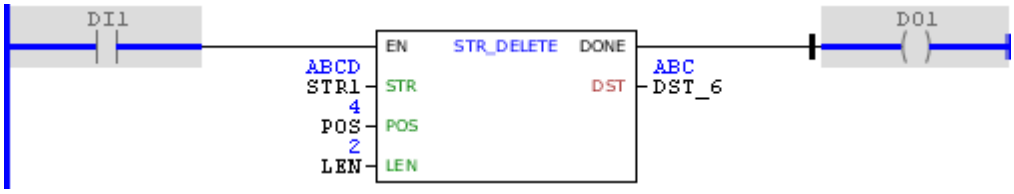
Example



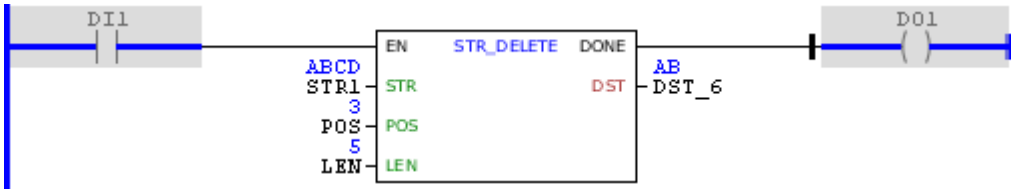
In the above example, as the value of POS is invalid, the block is not completed successfully, and the DONE output is disabled.



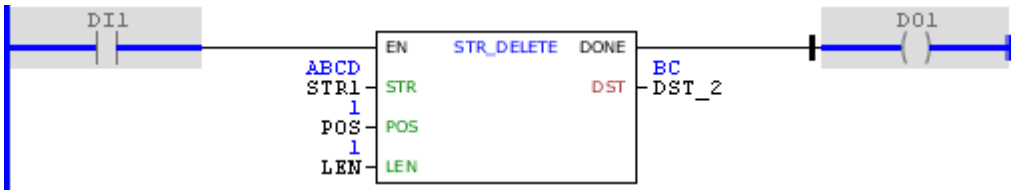
In the example above, two characters are deleted from position 2 of STR1, and the remaining characters are sent to DST_6. When the block is ended successfully, the DONE output is activated.



In the example above, two characters are deleted from position 4 of STR1, and the remaining characters are sent to DST_6. When the block is ended successfully, the DONE output is activated. Note that if POS + LEN is greater than the size of STR1, only the remaining characters are deleted from STR1, without generating an error.



In the example above, five characters are deleted from position 3 of STR1, and the remaining characters are sent to DST_6. When the block is ended successfully, the DONE output is activated. Note that if POS + LEN is greater than the size of STR1, only the remaining characters are deleted from STR1, without generating an error.



In the example above, one character is deleted from position 1 of STR1, and the remaining characters are sent to DST_2. Since the size of DST_2 is 2, the last character copied is discarded. The block ends successfully, Done output is activated.

11.8.5.16.6 STR_FIND

Block that searches for the first occurrence of one STRING in another, returning the position of this occurrence.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	STR1	STRING	STRING where the search will be performed
	STR2	STRING	STRING to be searched
	START	BYTE	Initial search position
VAR_OUTPUT	DONE	BOOL	Output enabling
	POS	BYTE	Position where STR2 was found

Operation

This block remains active as long as EN is at TRUE level, updating the value of POS according to the input parameters.
A search in STR1 is performed, from the START position, by an occurrence of STR2.

If it is found, it inserts into the POS the occurrence of this position, while DONE forwards to the next Ladder block the value of EN when ending the operation.

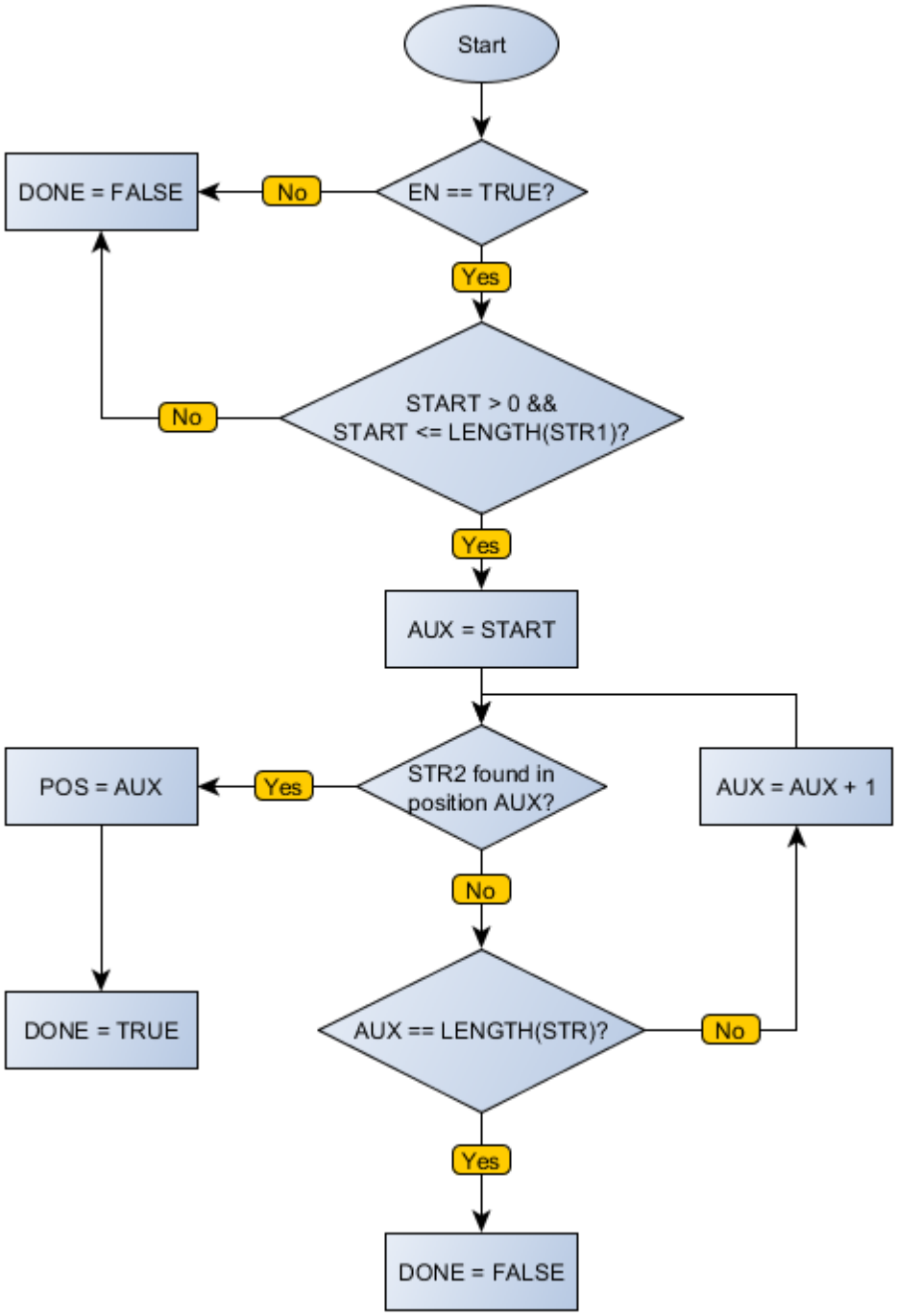
If it is not found or START receives value zero, DONE remains FALSE and POS remains unchanged.

NOTE!
POS is treated with index "based one". That is, POS = 1 references the first position of STR.

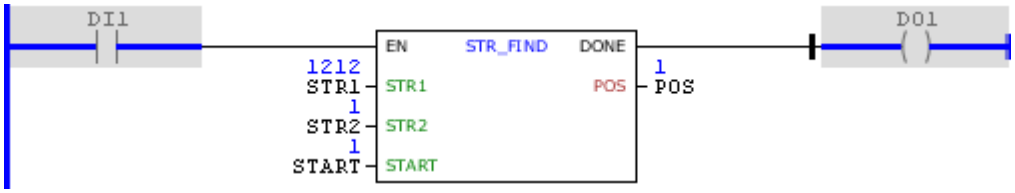
Compatibility

Device	Version
PLC300	2.10 or higher

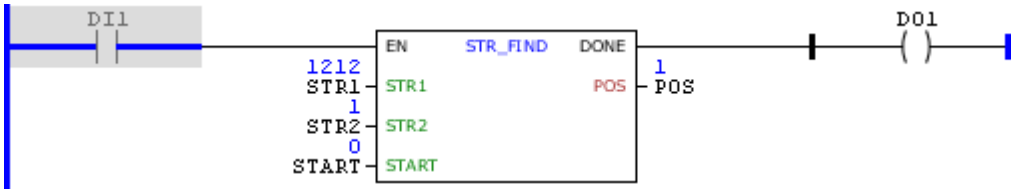
Block Flowchart


Start
EN == TRUE?
No
DONE = FALSE
Yes
START > 0 &&
START <= LENGTH(STR1)?
No
Yes
AUX = START
STR2 found in
position AUX?
Yes
POS = AUX
AUX = AUX + 1
No
AUX == LENGTH(STR)?
No
DONE = TRUE
Yes
DONE = FALSE

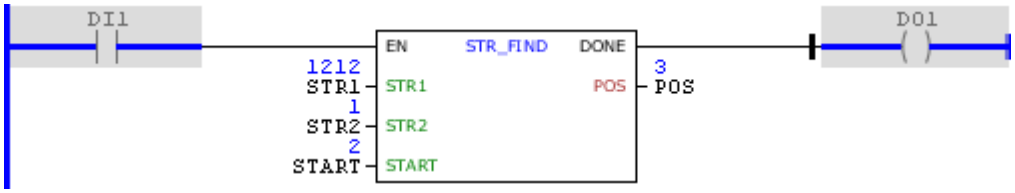
Example


DI1
EN
STR_FIND
DONE
1212
STR1
STR1
1
STR2
STR2
1
START
START
POS
1
POS
DO1

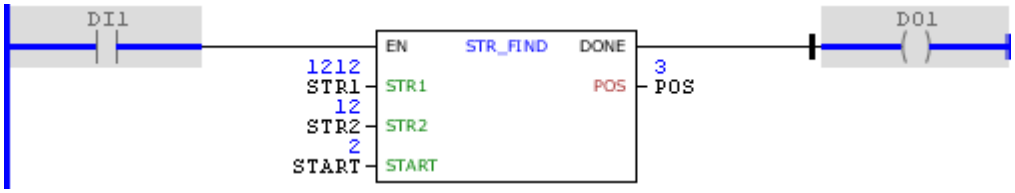
In the above example, a search for STR2 in STR1 from position 1 is done. Since STR2 is found in position 1, POS receives 1. The block ends successfully, the DONE output is activated.



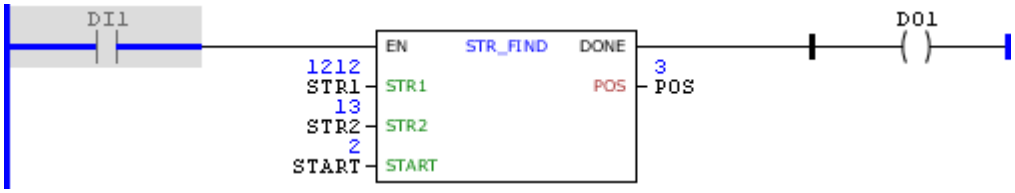
In the above example, as the value of START is invalid, the block is not completed successfully, and the DONE output is deactivated.



In the above example, a search for STR2 in STR1 from position 2 is done. Since STR2 is found in position 3, POS receives 3. The block ends successfully, the DONE output is activated.



In the above example, a search for STR2 in STR1 from position 2 is done. Since STR2 is found in position 3, POS receives 3. The block ends successfully, the DONE output is activated.

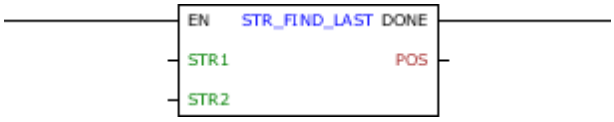


In the above example, a search for STR2 in STR1 from position 2 is done. Since STR2 is not found, POS remains unchanged and the DONE output is activated.

11.8.5.16.7 STR_FIND_LAST

Block that searches for the last occurrence of one STRING in another, returning the position of this occurrence.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	STR1	STRING	STRING where the search will be performed
	STR2	STRING	STRING to be searched
VAR_OUTPUT	DONE	BOOL	Output enabling
	POS	BYTE	Position where STR2 was found

Operation

This block remains active as long as EN is at TRUE level, updating the value of POS according to the input parameters.
A search in STR1 is performed, from its last position, by an occurrence of STR2.

If it is found, it inserts into the POS the occurrence of this position, while DONE forwards to the next Ladder block the value of EN when ending the operation.

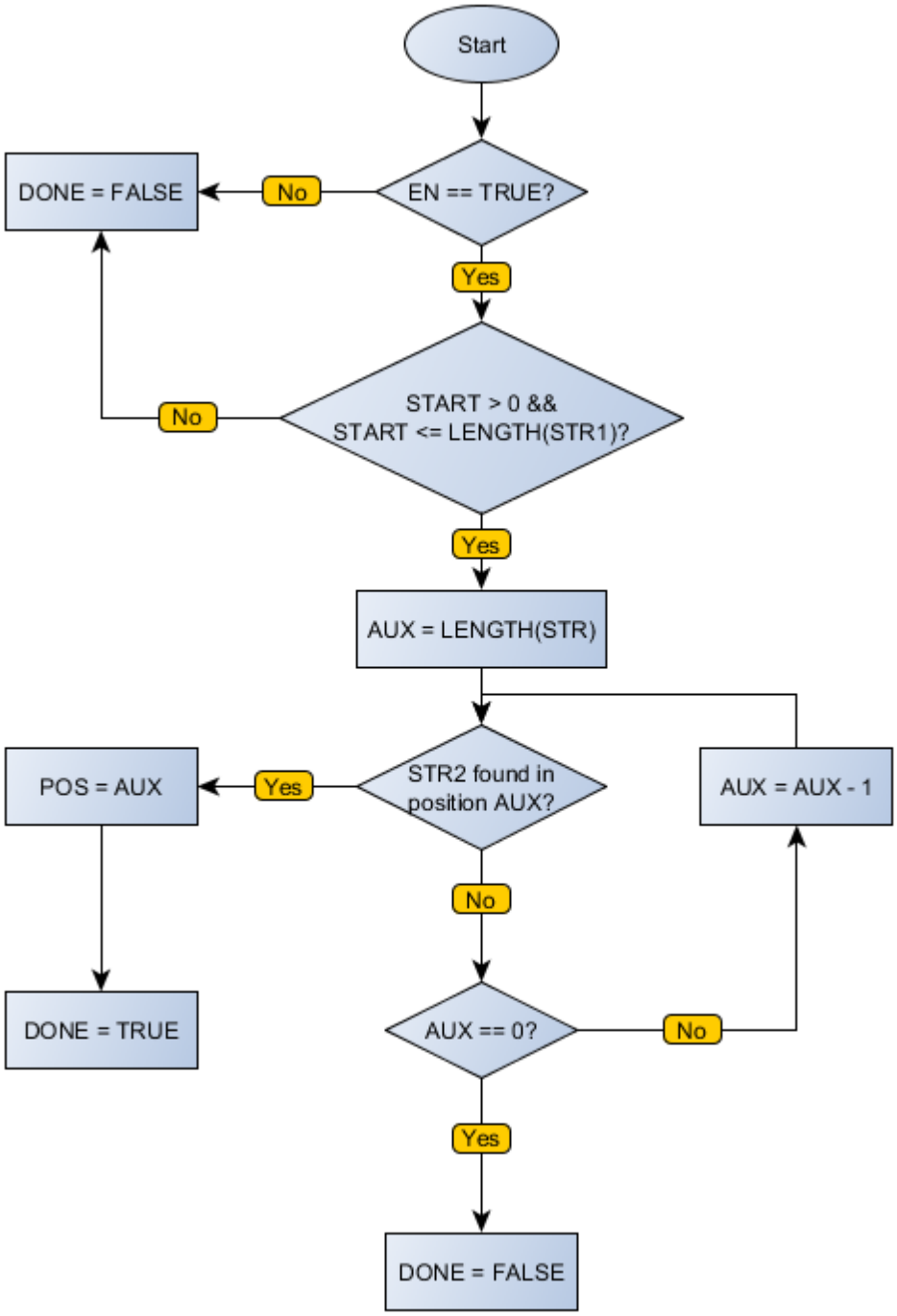
If it is not found, DONE remains FALSE and POS remains unchanged.

NOTE!
POS is treated with index "based one". That is, POS = 1 references the first position of STR.

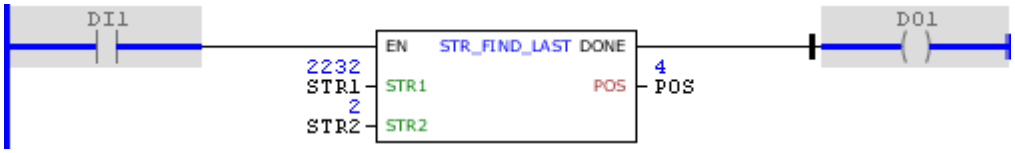
Compatibility

Device	Version
PLC300	2.10 or higher

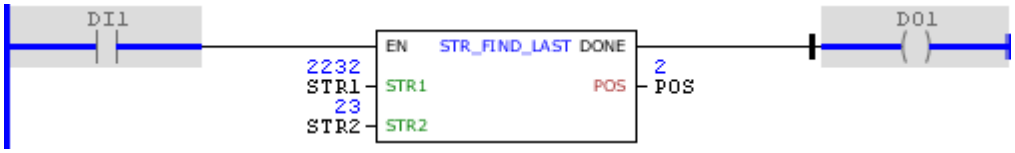
Block Flowchart


Start
EN == TRUE?
No
DONE = FALSE
Yes
START > 0 &&
START <= LENGTH(STR1)?
No
Yes
AUX = LENGTH(STR)
STR2 found in
position AUX?
Yes
POS = AUX
AUX = AUX - 1
No
DONE = TRUE
AUX == 0?
No
Yes
DONE = FALSE

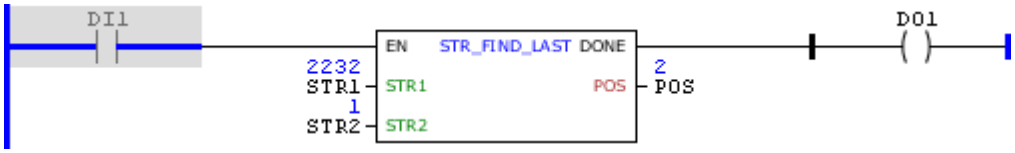
Example


DI1
EN
STR_FIND_LAST
DONE
2232
STR1
STR1
POS
4
POS
2
STR2
STR2
DO1

In the example above, a search is done by STR2 in STR1 from its final position. Since STR2 is found in position 4, POS receives 4. The block ends successfully, the DONE output is activated.



In the example above, a search is done by STR2 in STR1 from its final position. Since STR2 is found in position 2, POS receives 4. The block ends successfully, the DONE output is activated.

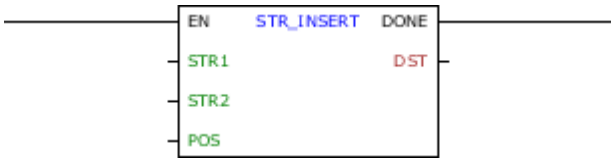


In the example above, a search is done by STR2 in STR1 from its final position. Since STR2 is not found, POS remains unchanged and the DONE output is disabled.

11.8.5.16.8 STR_INSERT

Block that inserts a STRING in certain position of another, returning the resulting STRING.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	STR1	STRING	STRING where the insertion will be performed
	STR2	STRING	STRING to be inserted
	POS	BYTE	Position where STR2 will be inserted
VAR_OUTPUT	DONE	BOOL	Output enabling
	DST	STRING	Resulting STRING

Operation

This block remains active as long as EN is at TRUE level, updating the value of DST according to the input parameters.
An insertion in STR1 from the defined position POS of the STR2 content is performed.

If POS is outside the allowable range of values (between 1 and the size of STR plus one), DONE receives FALSE and DST remains unchanged.

If successful, the DONE value forwards to the next Ladder block the EN value when the operation is completed.

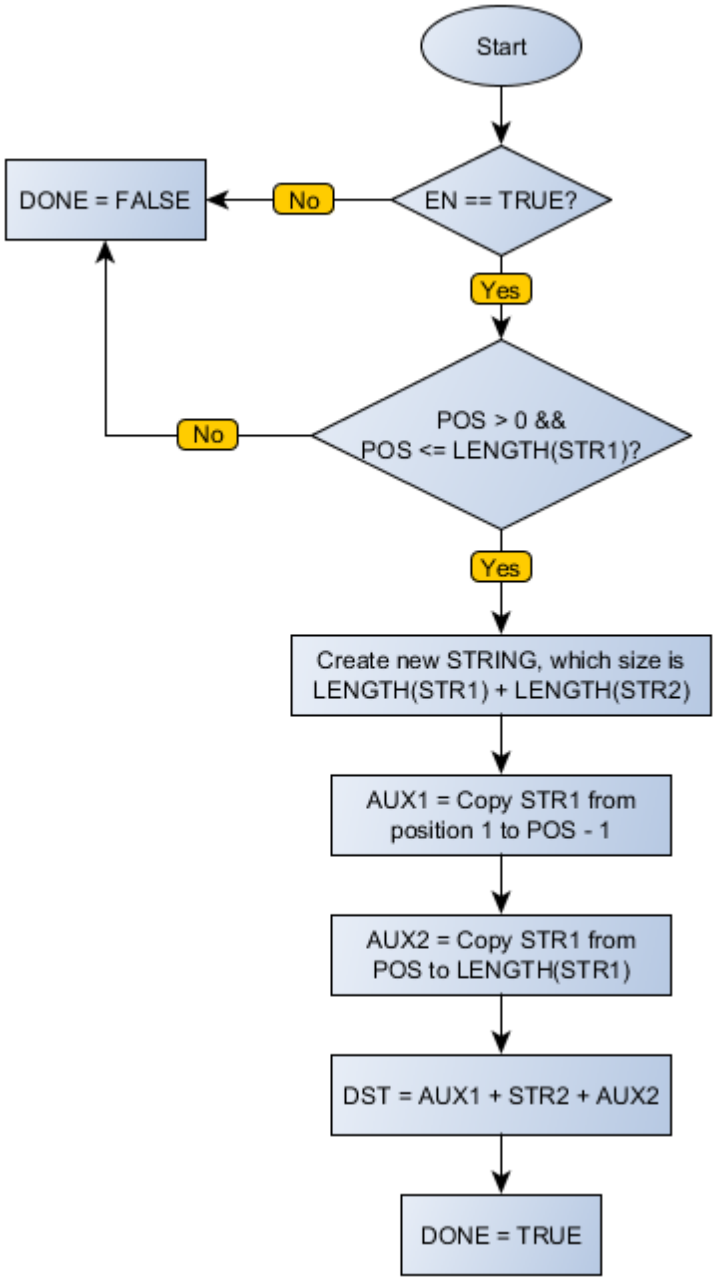
NOTE!
POS is treated with index "based one". That is, POS = 1 references the first position of STR.

NOTE!
If the size of DST is less than the sum of the number of characters STR1 and STR2, the resulting value will be truncated.

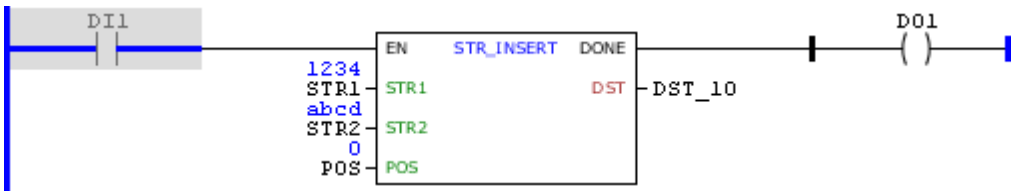
Compatibility

Device	Version
PLC300	2.10 or higher

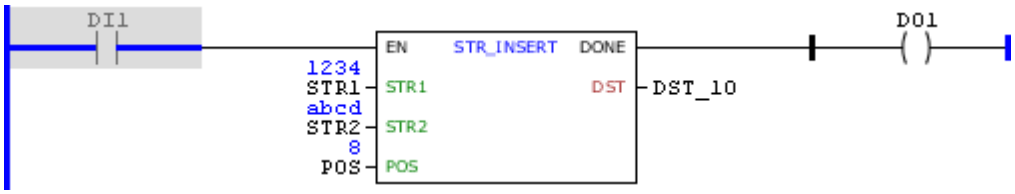
Block Flowchart



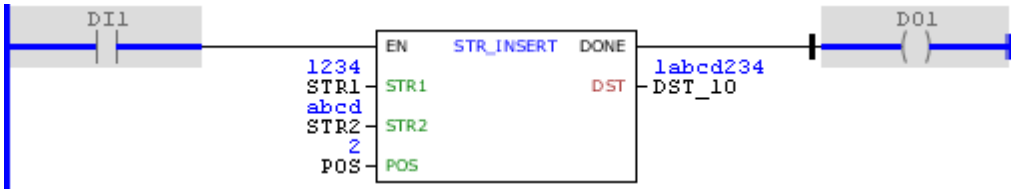
Example



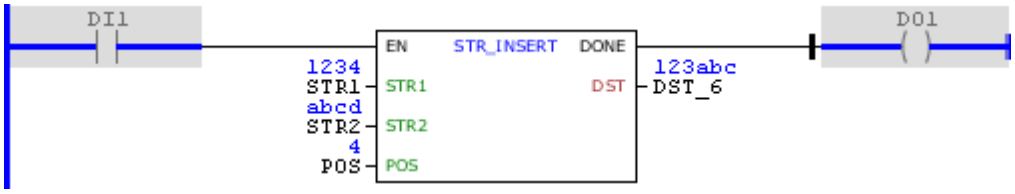
In the above example, as the value of POS is invalid, the block is not completed successfully, and the DONE output is disabled.



In the above example, as the value of POS is invalid, the block is not completed successfully, and the DONE output is disabled.



In the example above, an insertion of STR2 into STR1 from position 2 is done, and the result is sent to DST_10. When the block is ended successfully, the DONE output is activated.

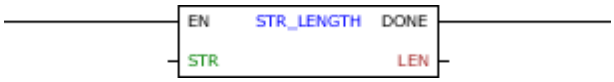


In the example above, is done of an insertion of STR2 into STR1 from position 4, and the result is sent to DST_6. Since the size of DST_6 is 6, the last two characters copied are discarded. The block ends successfully, Done output is activated.

11.8.5.16.9 STR_LENGTH

Block that calculates the size of a STRING.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	STR1	STRING	Input STRING
VAR_OUTPUT	DONE	BOOL	Output enabling
	LEN	BYTE	STRING size

Operation

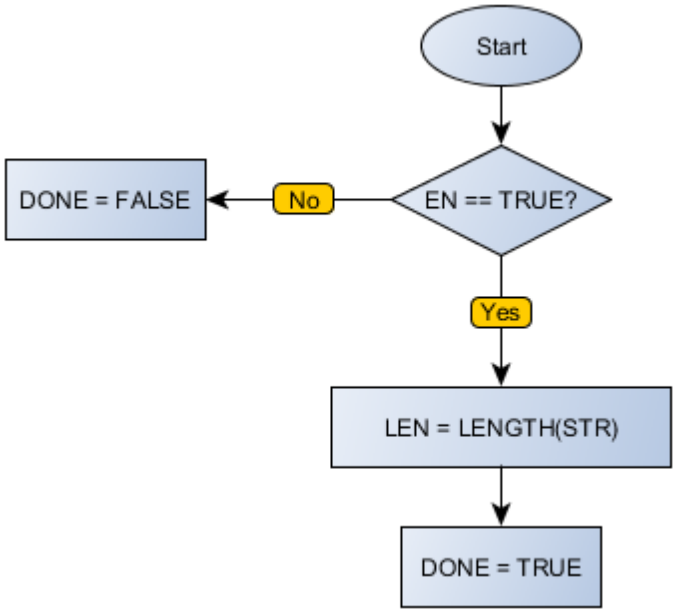
This block remains active as long as EN is at TRUE level, updating the value of LEN according to the identified size of STR.

The DONE value forwards to the next Ladder block the EN value when the operation is completed.

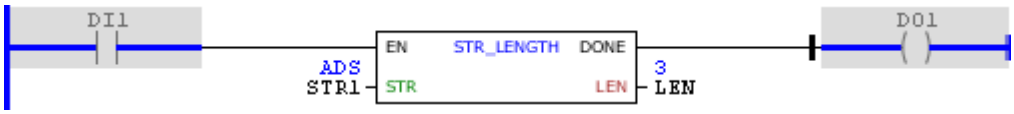
Compatibility

Device	Version
PLC300	2.10 or higher

Block Flowchart



Example



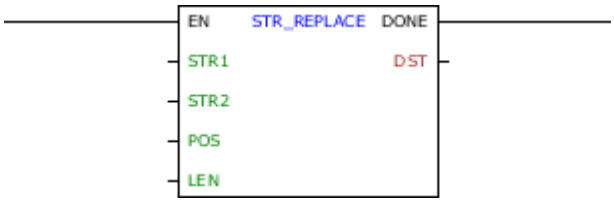
In the above example, the length of STRING in STR is calculated and the result is sent to LEN. The block ends successfully, Done output is activated.

In the above example, the length of STRING in STR is calculated and the result is sent to LEN. The block ends successfully, Done output is activated. Note that null STRINGS have length zero.

11.8.5.16.10 STR_REPLACE

Block that replaced a section of a STRING by another STRING, returning the resulting STRING.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	STR1	STRING	STRING where the replacement will take place
	STR2	STRING	STRING to be inserted
	POS	BYTE	Position where STR2 will be inserted
	LEN	BYTE	Number of characters to be replaced by STR2 in STR1
VAR_OUTPUT	DONE	BOOL	Output enabling
	DST	STRING	Resulting STRING

Operation

This block remains active as long as EN is at TRUE level, updating the value of DST according to the input parameters.
From the position defined in POS a number of characters defined by LEN are deleted. After that, in this position, the content of STR2 is inserted.

If POS is outside the allowable range of values (between 1 and the size of STR plus one), DONE receives FALSE and DST remains unchanged.

If successful, the DONE value forwards to the next Ladder block the EN value when the operation is completed.

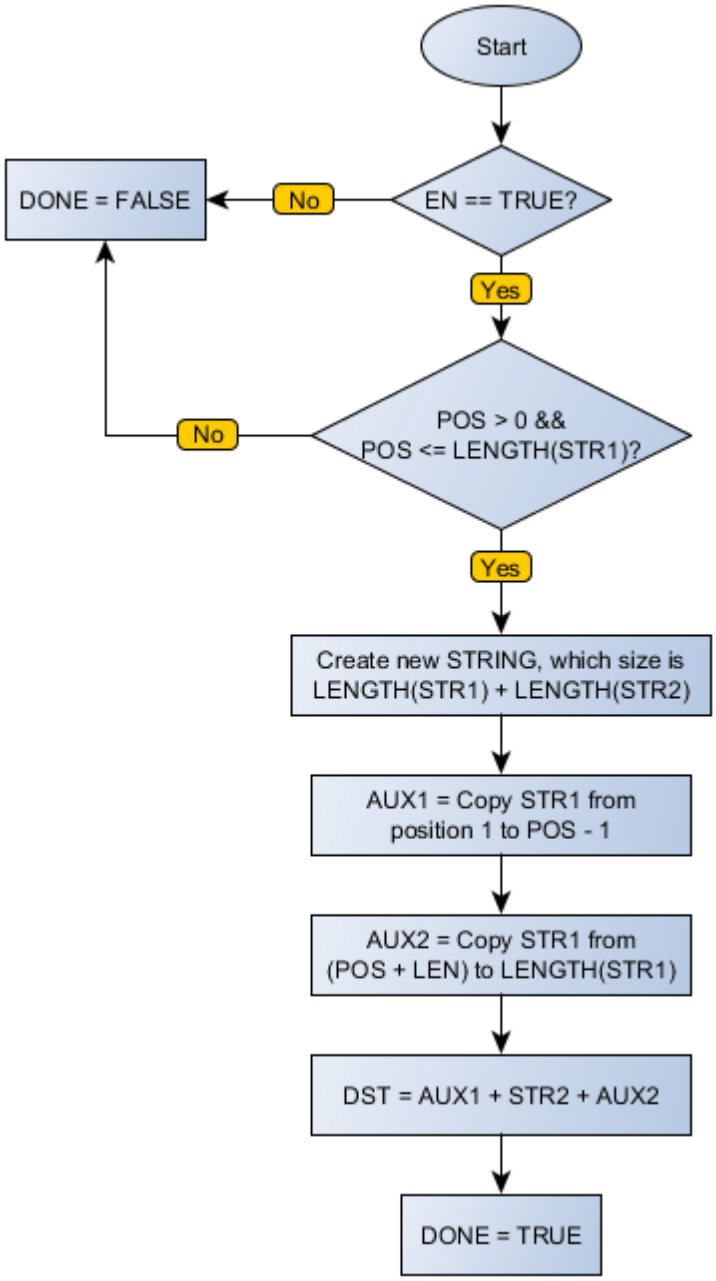
NOTE!
POS is treated with index "based one". That is, POS = 1 references the first position of STR.

NOTE!
If the size of DST is less than the sum of the number of remaining characters of STR1 and STR2, the resulting value will be truncated.

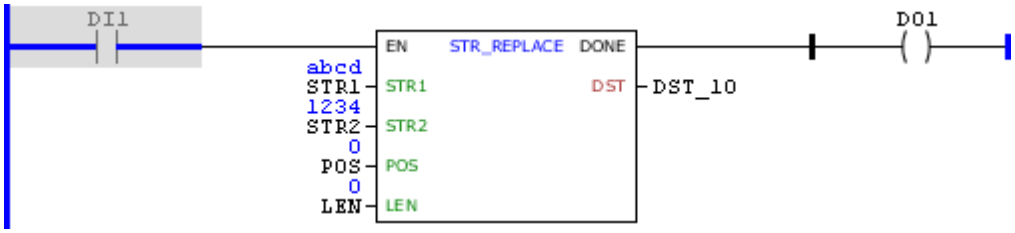
Compatibility

Device	Version
PLC300	2.10 or higher

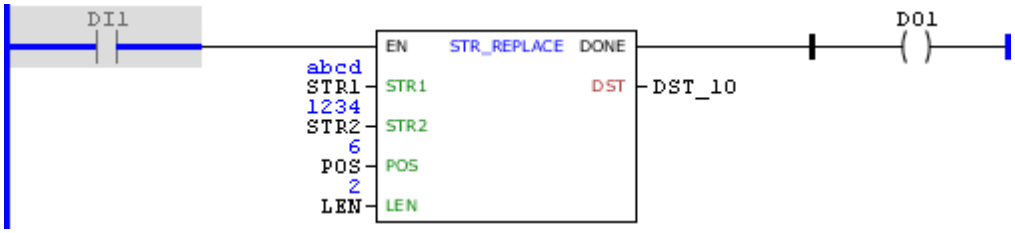
Block Flowchart



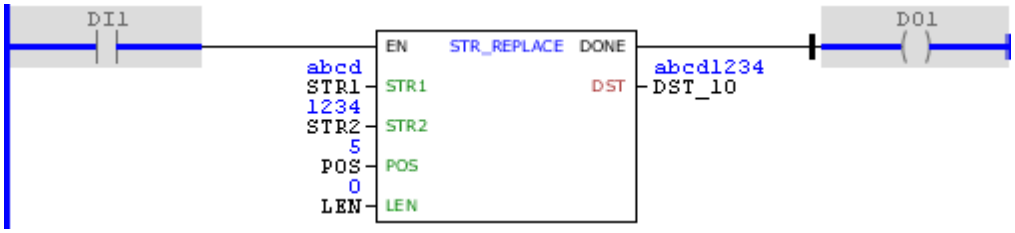
Example



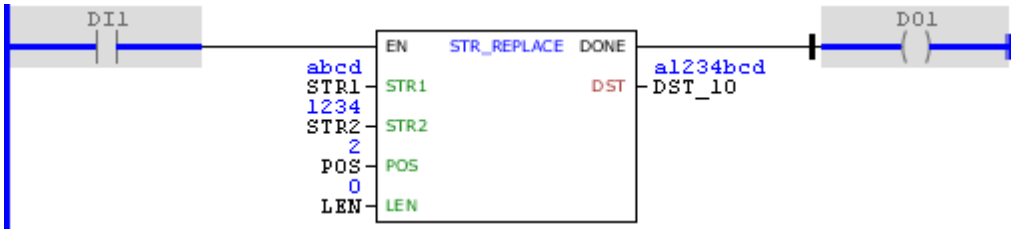
In the above example, as the value of POS is invalid, the block is not completed successfully, and the DONE output is disabled.



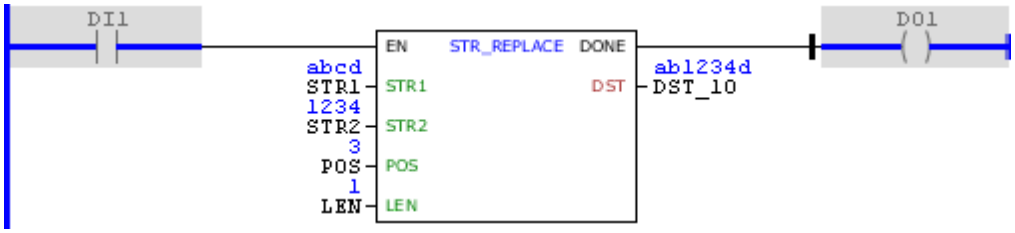
In the above example, as the value of POS is invalid, the block is not completed successfully, and the DONE output is disabled.



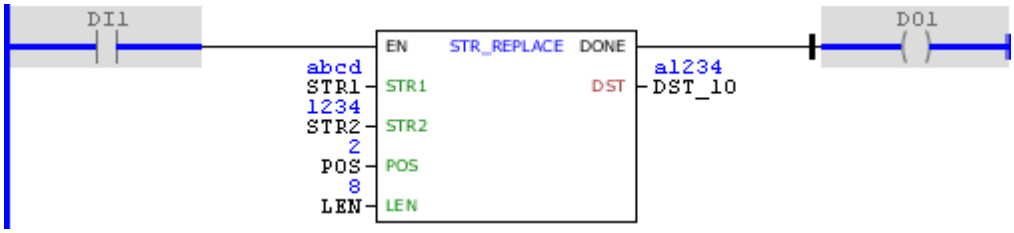
In the example above, a replacement of zero characters from the position 5 of STR1 is done by the contents of STR2, and the result is sent to DST_10. When the block is ended successfully, the DONE output is activated. Note that this block acted as a STR_INSERT.



In the example above, a replacement of zero characters from the position 2 of STR1 is done by the contents of STR2, and the result is sent to DST_10. When the block is ended successfully, the DONE output is activated. Note that this block acted as a STR_INSERT.



In the example above, a replacement of one character from the position 3 of STR1 is done by the contents of STR2, and the result is sent to DST_10. When the block is ended successfully, the DONE output is activated.



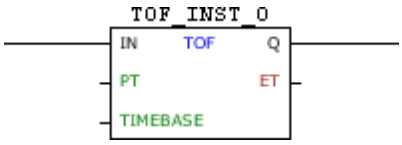
In the example above, a replacement of eight characters from the position 2 of STR1 is done by the contents of STR2, and the result is sent to DST_10. When the block is ended successfully, the DONE output is activated. Note that if POS + LEN is greater than the size of STR1, only the remaining characters from STR1 are replaced, without generating an error.

11.8.5.17 Timer

11.8.5.17.1 TOF

Timer block that, when energized, disables the output after a delay set by PT.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	IN	BOOL	Block enabling
	PT	WORD UINT DWORD UDINT	Delay of output deactivating
	TIMEBASE	WORD	Time base for PT and ET
VAR_OUTPUT	Q	BOOL	Block output
	ET	WORD UINT DWORD UDINT	Counter elapsed time
VAR	TOF_INST_0	TOF	Instance of access to block structure

NOTE!
In CFW300, the PT e ET fields can only be WORD ou UINT type.

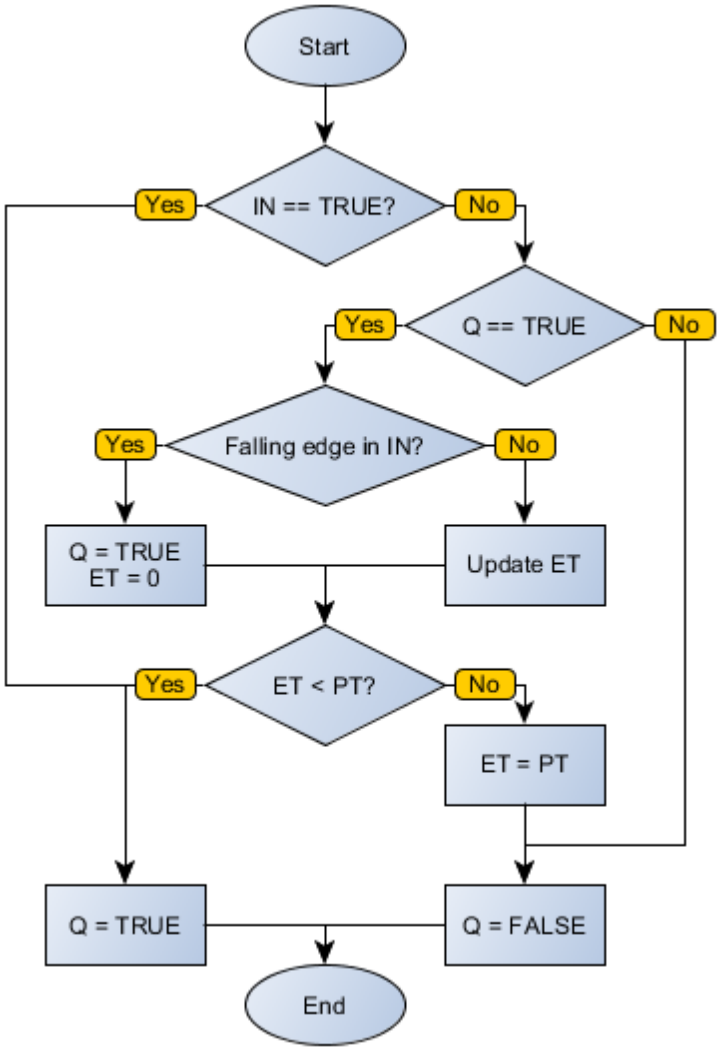
Operation

While the IN input is TRUE, the Q output is also TRUE and ET also receives the value zero.
On the negative transition edge in IN, counting is triggered and ET is incremented according to TIMEBASE. When ET equals PT, the Q output goes to state FALSE until IN revolutions to FALSE.

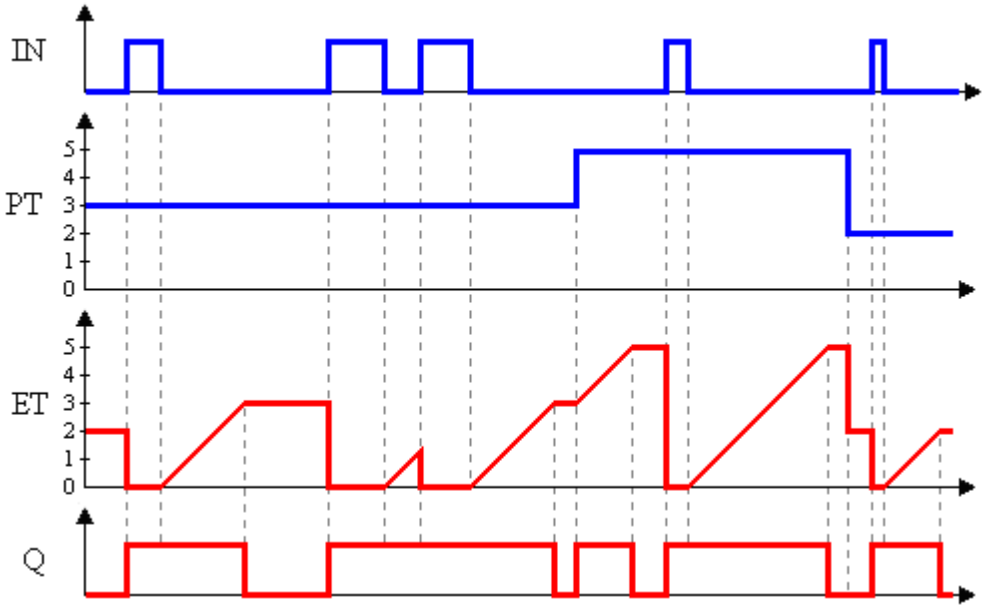
Compatibility

Device	Version
PLC300	1.50 or higher
SCA06	2.00 or higher

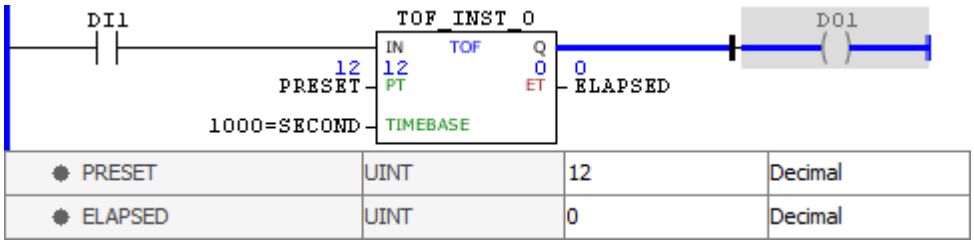
Block Flowchart



Operation Diagram



Example



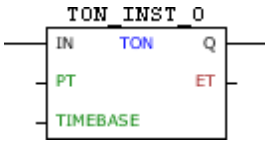
PRESET	UINT	12	Decimal
ELAPSED	UINT	0	Decimal

The above example disables the DO1 output to identify a low level in DI1 for 12 seconds, remaining disabled until DI1 again be TRUE.

11.8.5.17.2 TON

Timer block that, when energized, enables the output after a delay set by PT.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	IN	BOOL	Block enabling
	PT	WORD UINT DWORD UDINT	Delay of output drive
	TIMEBASE	WORD	Time base for PT and ET
VAR_OUTPUT	Q	BOOL	Block output
	ET	WORD UINT DWORD UDINT	Counter elapsed time
VAR	TON_INST_0	TON	Instance of access to block structure

NOTE!
In CFW300, the PT e ET fields can only be WORD ou UINT type.

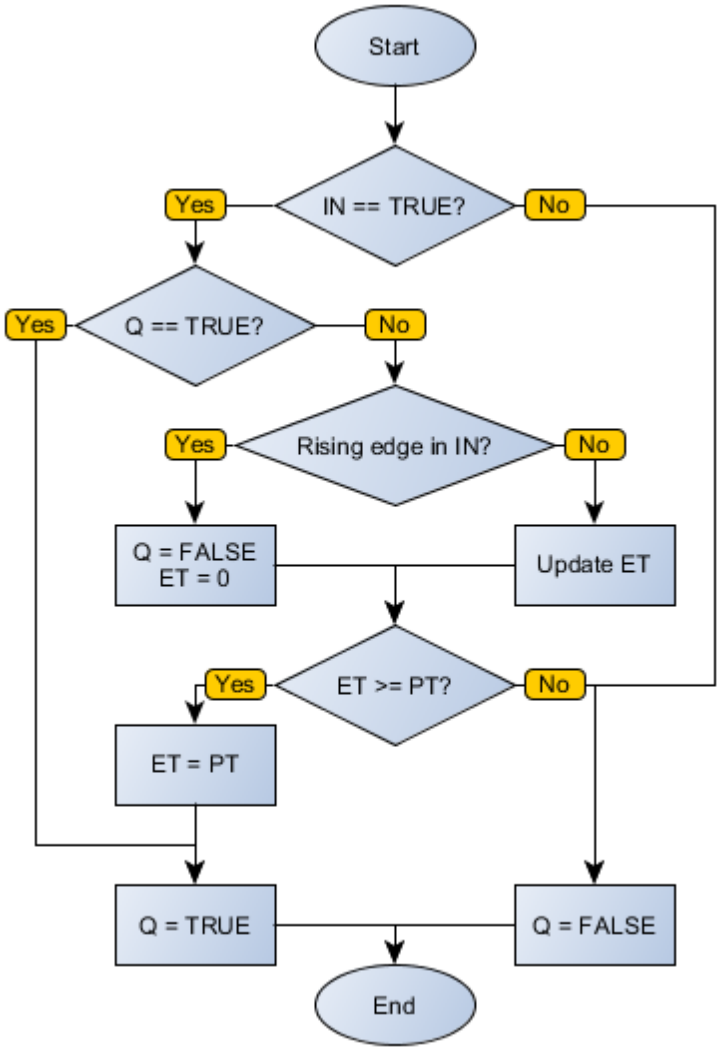
Operation

While the IN input is FALSE, the Q output is FALSE and ET also receives the value zero.
On the edge positive transition in IN, counting is triggered and ET is incremented according to TIMEBASE. When ET equals PT, the Q output goes to state TRUE until IN revolutions to FALSE.

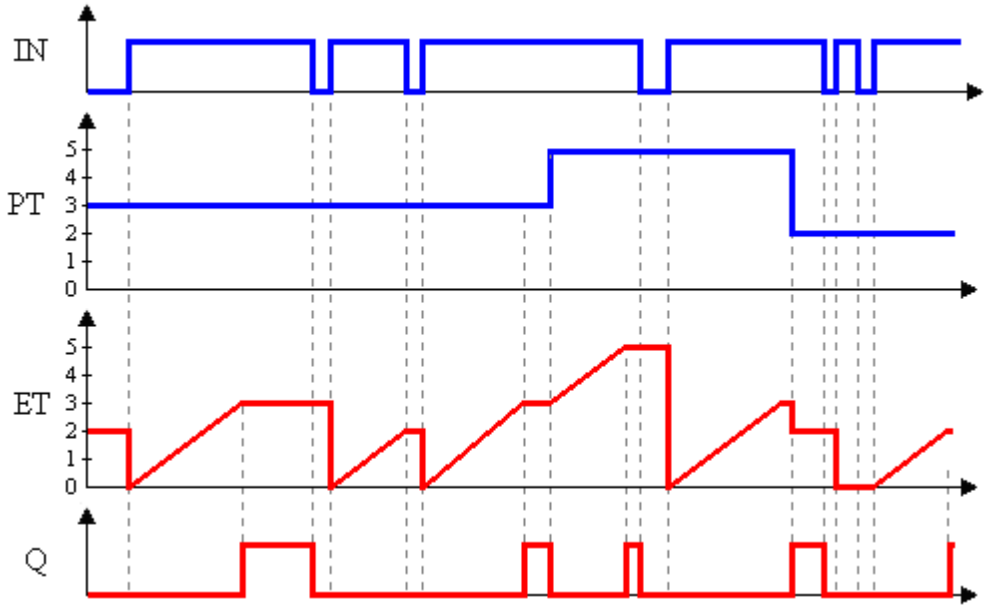
Compatibility

Device	Version
PLC300	1.50 or higher
SCA06	2.00 or higher

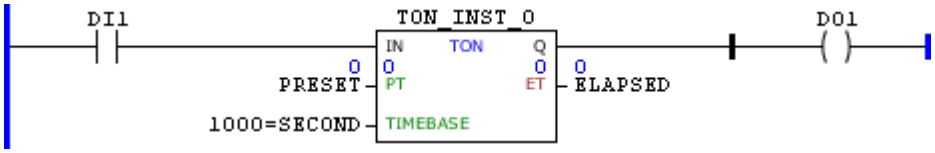
Block Flowchart



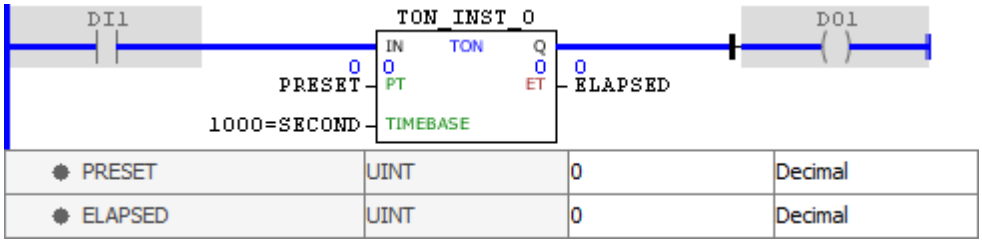
Operation Diagram



Example

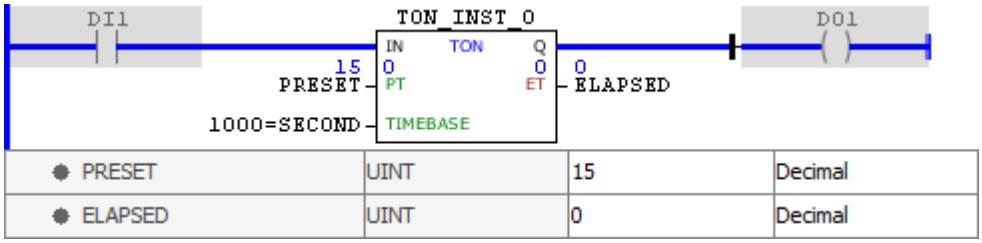


The above example shows the initial conditions of the block and of the routine variables.



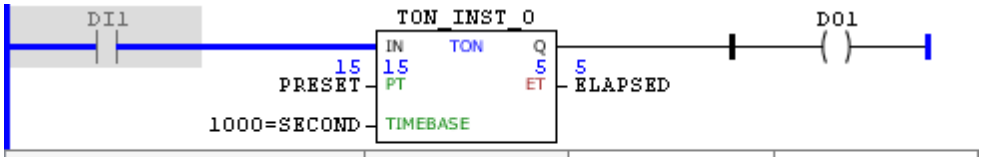
PRESET	UINT	0	Decimal
ELAPSED	UINT	0	Decimal

When activated the IN input, counting is triggered. Since ET equals PT, the Q output is enabled.



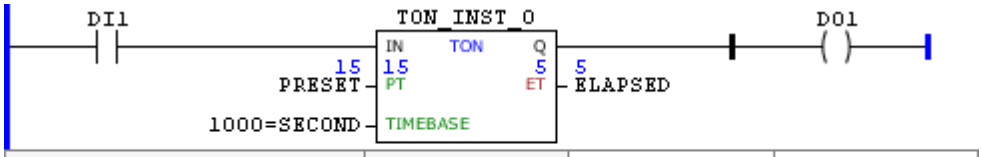
PRESET	UINT	15	Decimal
ELAPSED	UINT	0	Decimal

Note that a change in PRESET variable is not forwarded to the PT field while the IN entry remains enabled.



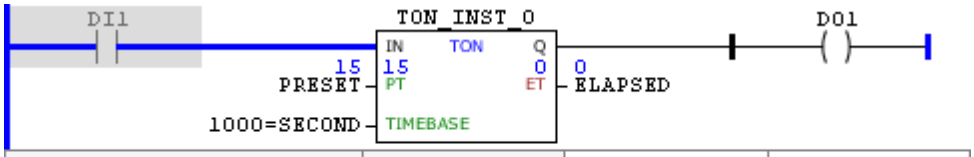
PRESET	UINT	15	Decimal
ELAPSED	UINT	5	Decimal

Disabling the IN input, the value of PT is updated and the Q output is disabled. When activating it again, counting is triggered.



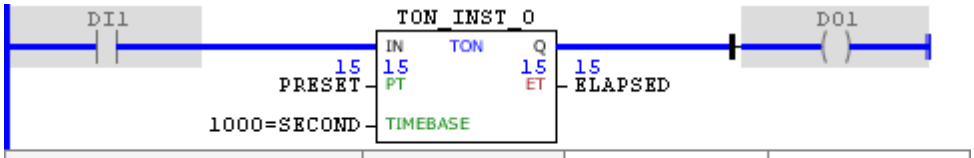
PRESET	UINT	15	Decimal
ELAPSED	UINT	5	Decimal

Disabling the IN input, the value of ET remains saved.



PRESET	UINT	15	Decimal
ELAPSED	UINT	0	Decimal

Enabling the IN input, the value of ET is reset and counting is triggered.



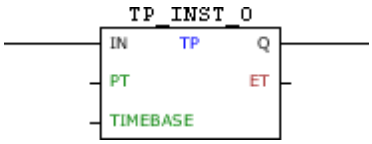
PRESET	UINT	15	Decimal
ELAPSED	UINT	15	Decimal

When ET reaches the value PT, the Q is output enabled and remains so while IN is at TRUE level.

11.8.5.17.3 TP

Timer block that, when identifies it is energized, enables the output after a delay set by PT.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	IN	BOOL	Block enabling
	PT	WORD UINT DWORD UDINT	Time w hile the output is enabled
	TIMEBASE	WORD	Time base for PT and ET
VAR_OUTPUT	Q	BOOL	Block output
	ET	WORD UINT DWORD UDINT	Counter elapsed time
VAR	TP_INST_0	TP	Instance of access to block structure

NOTE!
In CFW300, the PT e ET fields can only be WORD ou UINT type.

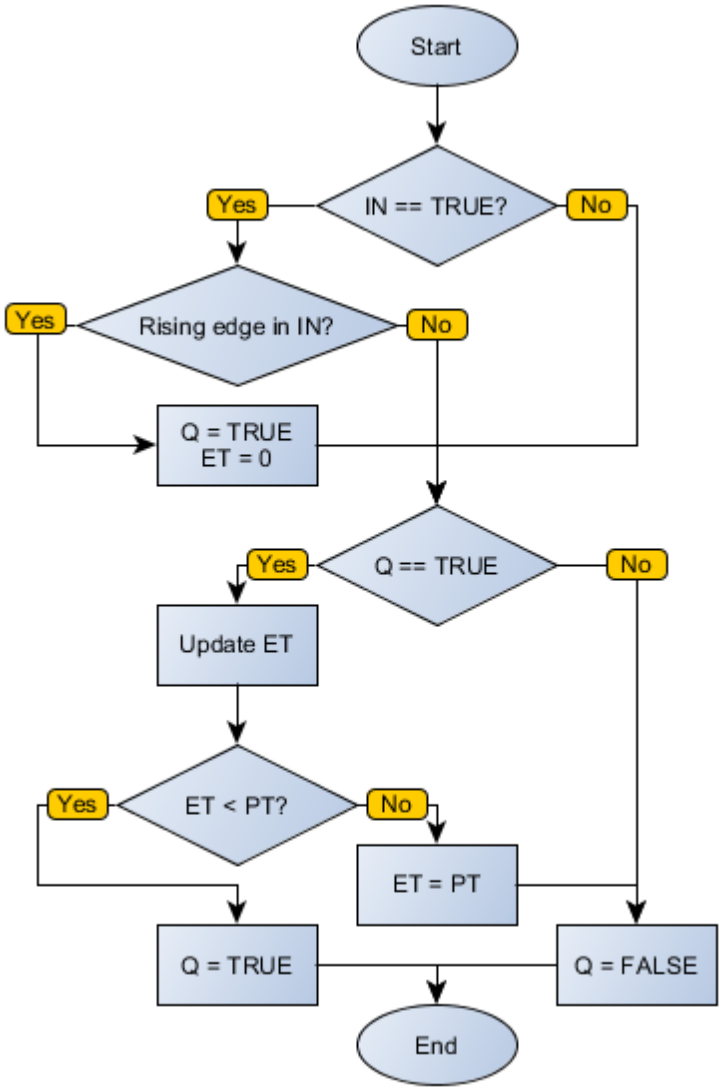
Operation

On the edge positive transition in IN, Q receives TRUE value, counting is triggered and ET is incremented according to TIMEBASE. When ET equals PT, the Q output goes to state FALSE until IN revolutions to FALSE. At that moment, if IN is at TRUE level, nothing happens. On the edge positive transition in IN, ET is automatically reset.

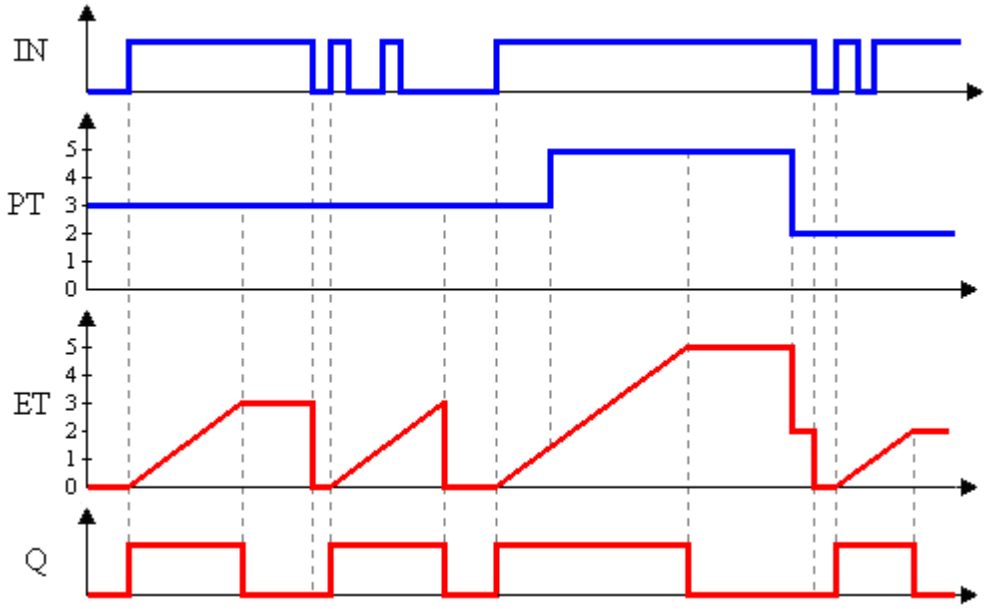
Compatibility

Device	Version
PLC300	1.50 or higher
SCA06	2.00 or higher

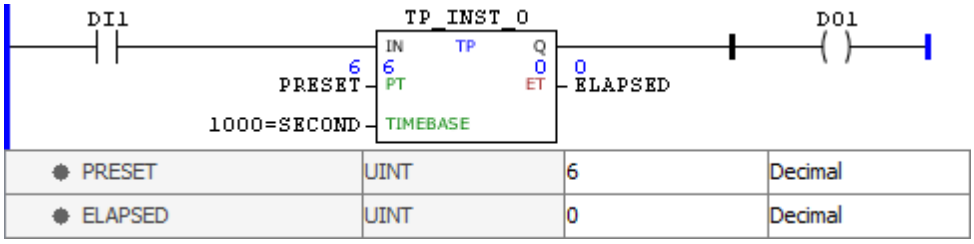
Block Flowchart



Operation Diagram



Example



PRESET	UINT	6	Decimal
ELAPSED	UINT	0	Decimal

The above example enables the DO1 output for six seconds at each DI1 positive transition.

11.8.5.18 Tasks

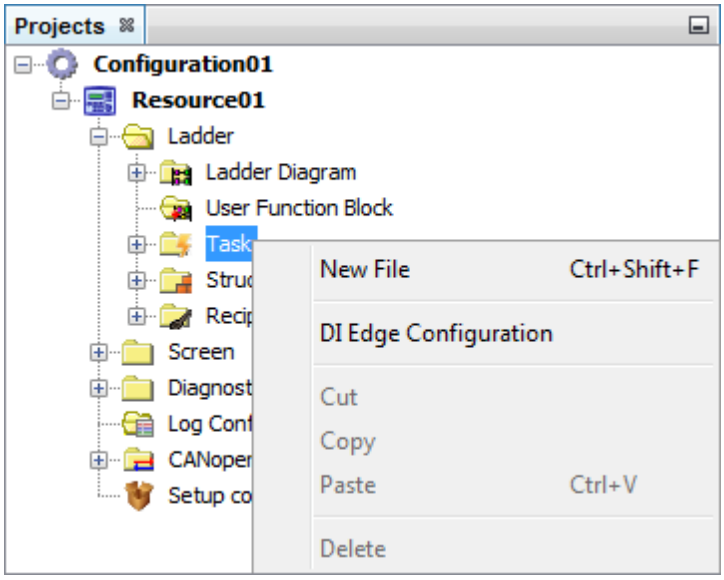
Overview

A task is a scheduling mechanism very useful in real time systems to periodically control the performance of programs, or whenever triggered by events.

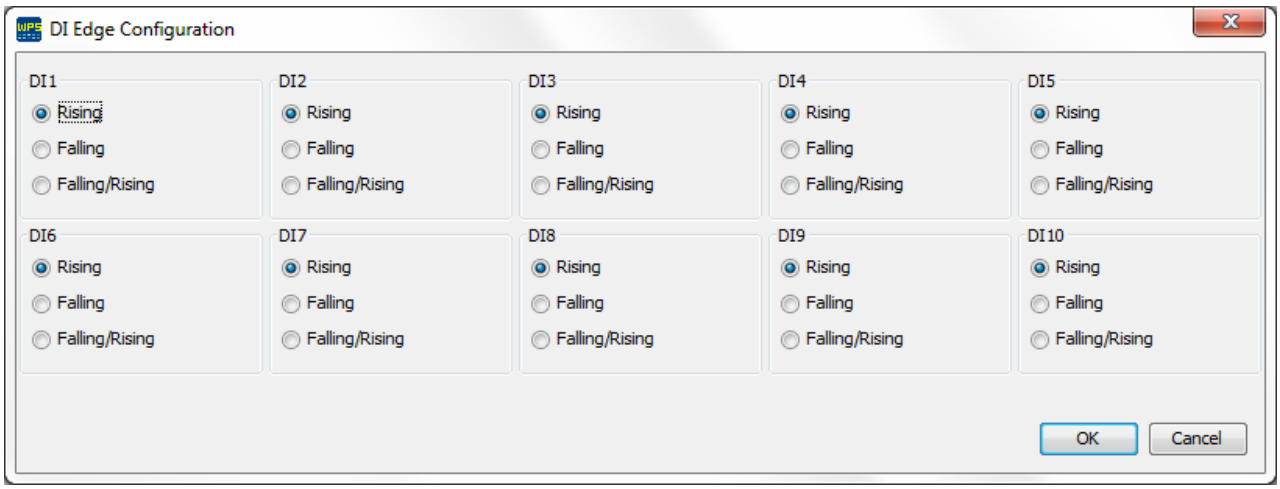
Task configuration is carried out by adding task files to the **Task** folders, inside the resource. When the task becomes enabled, programs that perform the logic of these task files are associated to them. A watchdog (see *Watchdog* section) may be configured for every task. Every task has an associated system marker that can be enabled or disabled through the program.

Inside a resource, there is a single task called **Main** that cannot be deleted, in which the main program (**Main Ladder**) is run.

In the task folder option menu, it is possible to perform the following actions:

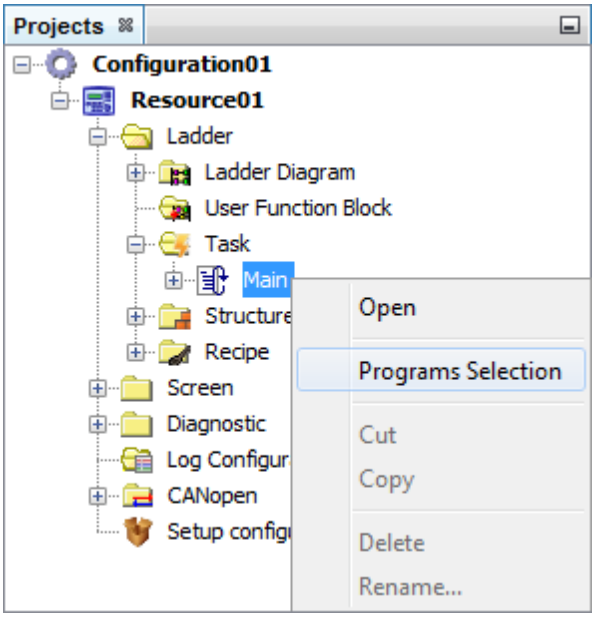


- **Add a new task**: in order to add a new task, it is necessary to select the **New file** option in the **Task** folder.
- **Configuring the transition edge of digital inputs (DIs)**: digital inputs DI9 and DI10 have the option of triggering events through a rising, falling, or rising and falling transition edge. The configuration of these transition edges is made through the DI configuration window, which is accessed by selecting the **DI Edge Configuration** option in the **Task** folder.



These DIs are used in tasks of the external event and count types.

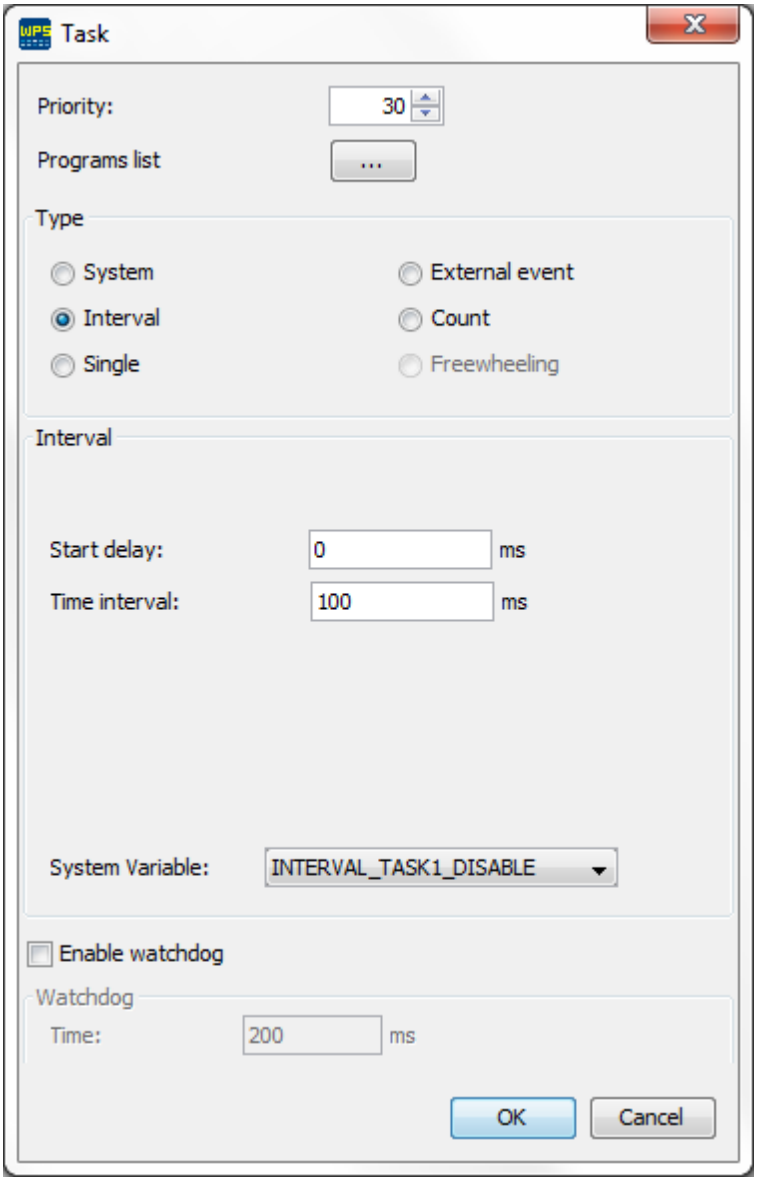
Through the task folder option menu, it is possible to perform the following actions:



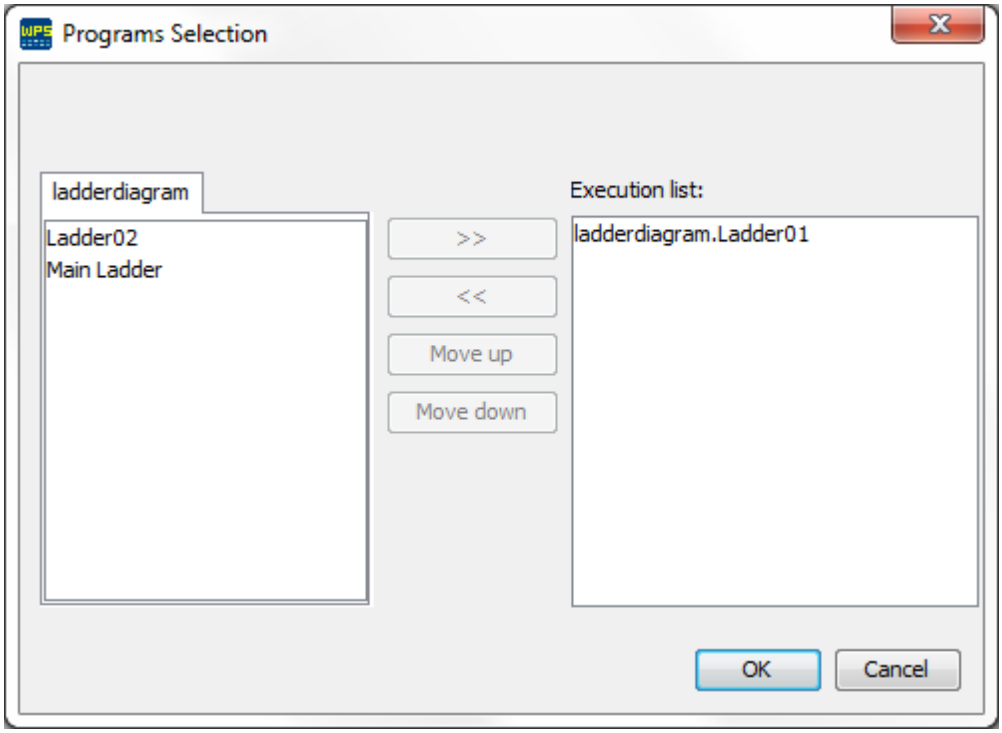
- **Configuring tasks**: in order to configure the task, it is necessary to select the **Open** option in the task file to be configured. For further information on how to configure the tasks, see the *Task configuration* section.
- **Add, remove, or order task programs**: in order to add, remove, or order programs of a certain task, it is necessary to select the **Select programs** option in the task file to be configured. For further information on how to select the programs, see *Task configuration* section in item *List of programs*.

Task Configuration

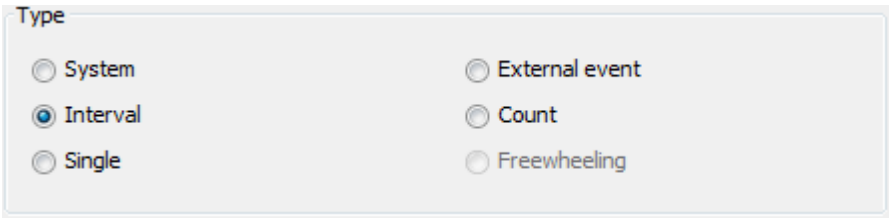
The following items are configured in the task configuration window:
- Priority,
- Sequence of programs,
- Type of task, and
- Task watchdog options.



- **Priority**: configured with a number between 0 and 30 (0 - higher priority, 30 - lower priority) which defines the priority in which the task will be performed. In case a task with a higher priority than that of the running task is activated, it will immediately start running. Otherwise, in case a task with a lower priority than that of the current one is activated, it waits for the current task to stop running before it starts running.
- **List of programs**: the selection window in the sequence of programs serves to add, remove, or change the sequence of programs associated to a certain task. All programs available in the resource are included in the left side list, and the programs selected for this task are on the right side. The order in which the programs will start running will be the same order as the one defined in this list.

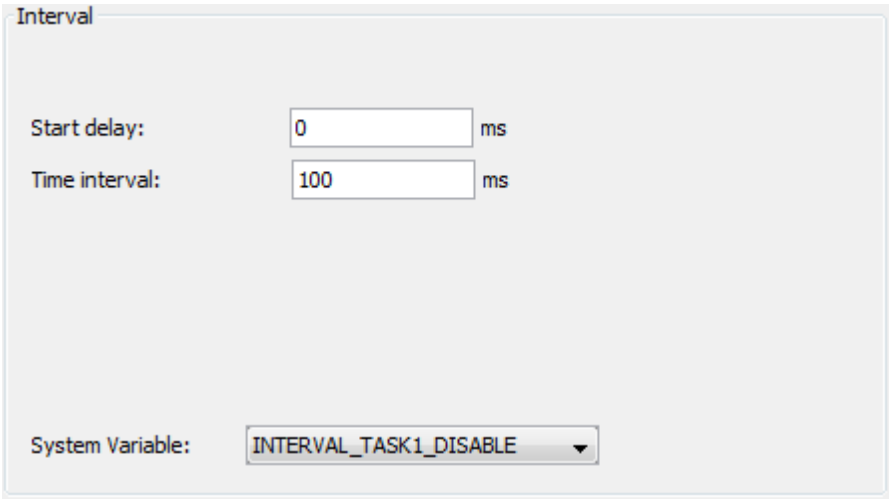


- **Types of Tasks**: The tasks are divided in: system, interval, single, external event, count, and freewheeling.



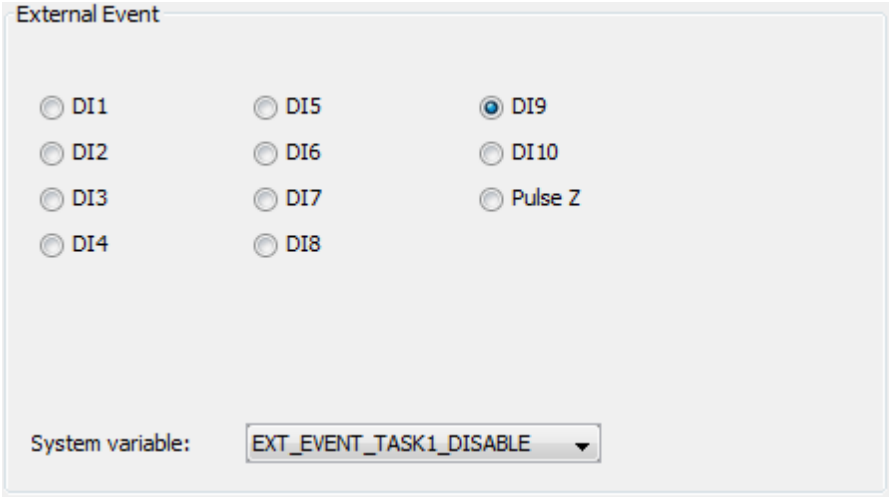
  - **System**: This task may be of the Initialization or Stop type. When the **Initialization** option is selected, the task will start running as soon as the user program runs for the first time. In case the **Stop** option is selected, programs associated to this task will start running right after the user program stops.

- **Interval**: Programs associated to this task run repeatedly within the time interval defined in the **Time interval** field, being the delay time for the first performance defined in the **Initial delay** field. This task is associated to a system variable that allows the task to be disabled through the program.



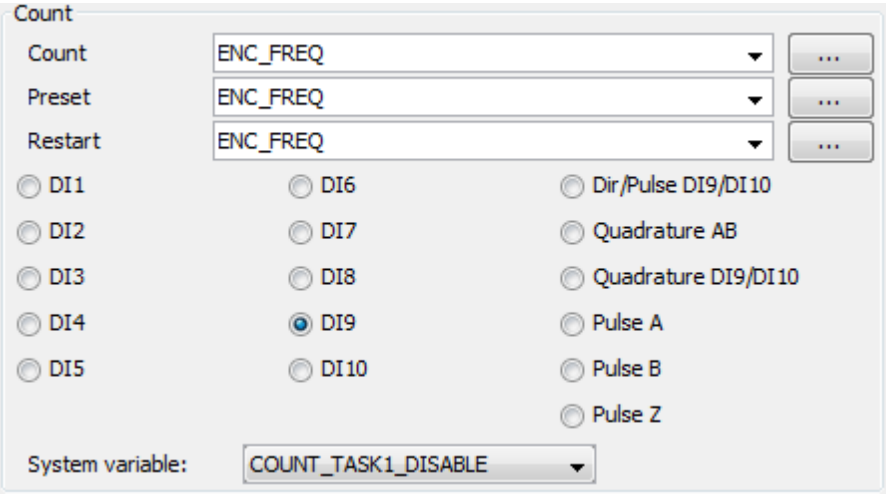
- **Single**: Programs associated to this task run one single time whenever the selected variable undergoes a positive transition, i.e. from zero to some other value. This task is associated to a system variable that allows the task to be disabled through the program.

- **External Event**: Programs associated to this task run one single time whenever event DI9, DI10, or Pulse Z is enabled. Events DI9 and DI10 have the option to select the transition that enables the task (see item *Configuring the transition edge of digital inputs* in the *Overview* section). This task is associated to a system variable that allows the task to be disabled through the program.



NOTE!
The options DI1 a DI8 are available from version 2.10.

- **Count**: The programs associated to this task are executed every time the pulse count on the selected input (DI9, DI10, Pulse A, Pulse B, Pulse Z, Quadrature AB, Quadrature DI9/DI10, Dir/Pulse DI9/DI10) is greater than the value of the variable defined in the **Predefined** field. The pulse count value is stored in the **Count** variable. The pulse count value is reset when this value exceeds the value defined in the **Restart** field. The variables configured in these fields can be of the DWORD and UDINT type for inputs DI9, DI10, Pulse A, Pulse B and Pulse Z, and DINT for inputs quadrature AB. A system variable is associated to this task which allows the task to be disabled through the program. The Dir/Pulse_DI9/DI10 option can be used as fast count up to 100 KHz.



Important:
The Quadrature_DI9/DI10 and Dir/Pulse_DI9/DI10 functions cannot be used simultaneously in tasks or in tasks and blocks;
The Quadrature_DI9/DI10 and Dir/Pulse_DI9/DI10 functions can be used in different blocks, provided that they are not used in tasks.

NOTE!
Those options, after selected, will not cause error in case the others (DI9, DI10, Pulse A...) are also selected.

Examples that do not generate compilation error:
- two blocks in Quadrature_DI9/DI10;
- two blocks one block Quadrature_DI9/DI10 and another Dir/Pulse_DI9/DI10;
- two tasks Quadrature_DI9/DI10 or two tasks Dir/Pulse_DI9/DI10;

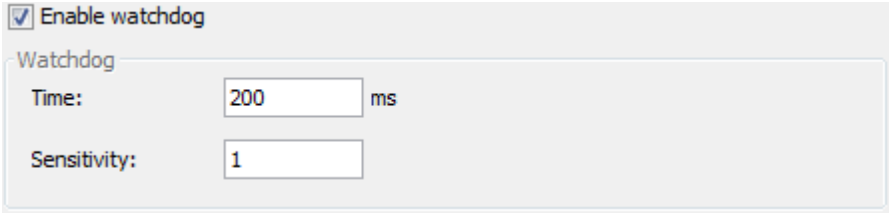
Examples that generate compilation error:
- two tasks, one Quadrature_DI9/DI10 and another Dir/Pulse_DI9/DI10;
- task Quadrature_DI9/DI10 and block Dir/Pulse_DI9/DI10;
- task Dir/Pulse_DI9/DI10 and block Quadrature_DI9/DI10.

NOTE!
The Quadrature_DI9/DI10 and Dir/Pulse_DI9/DI10 options are available from version 2.0 up.

NOTE!
The options DI1 a DI8 are available from version 2.10.

- **Freewheeling**: Programs associated to this task run in a cyclic form. When the list of programs stops running, the list of programs is reinitialized until the user program stops. The main program (Main Ladder) is associated to this task, and it is not possible to remove it or associate it with another task.

- **Watchdog**: When the watchdog option is activated, the user program stops with an error alarm in case the task running time is longer than the time defined in the **Time** field for a higher number of times than that defined in the **Sensitivity** field.



- **Time**: Maximum time to run the task without watchdog error.
- **Sensitivity**: Number of watchdog errors allowed before generating the error alarm and stopping the user program.

Compatibility

Device	Version
PLC300	1.20 or higher

11.8.5.19 Structures

Structure is a data grouping used to define a recipe or an object.

In the Ladder program, it is possible to create variables of the structure type and use them in the blocks. To access the internal members of the structure, the '.' is used followed by its respective member.

Creating a structure

1. With the right button of the mouse on the folder **Structure**, click on **New file**.

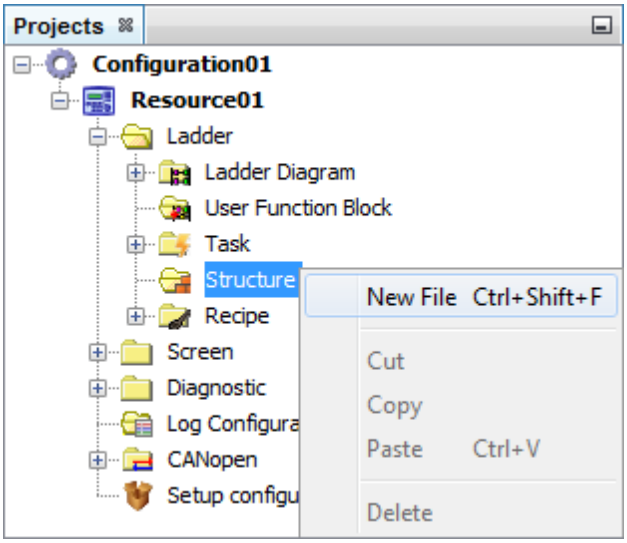


Figure 1: Creating a structure

2. Define the file name and press the **Next** button.

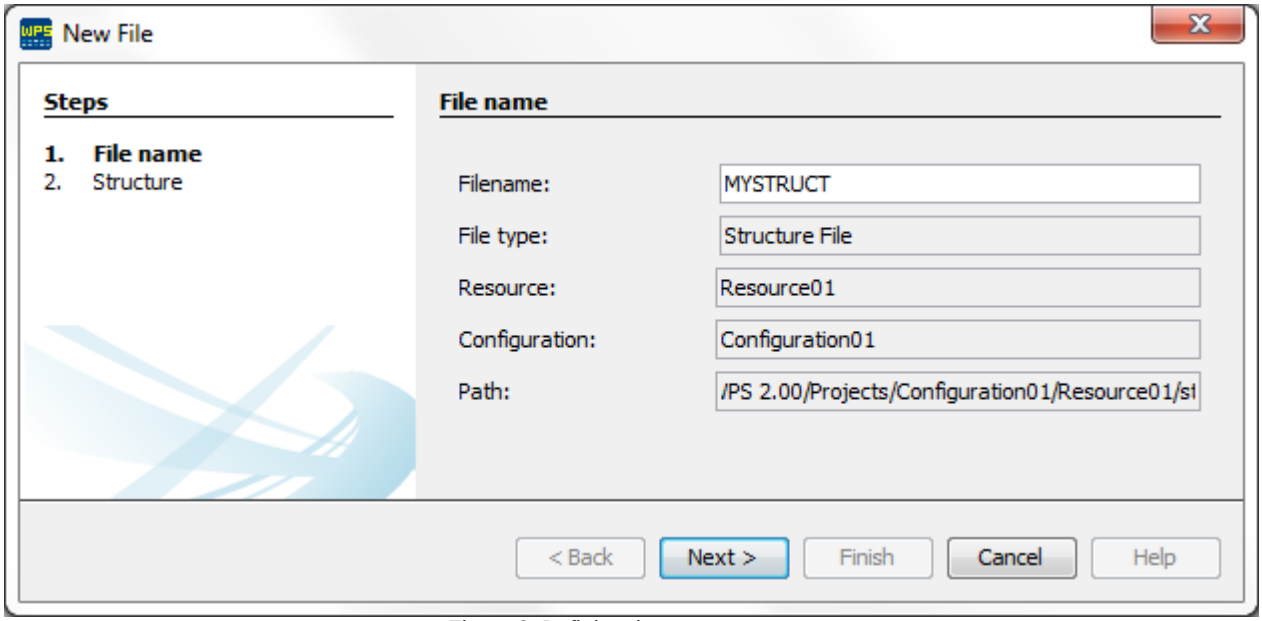


Figure 2: Defining the structure name

3. Configure the structure using the buttons presented in the figure below.

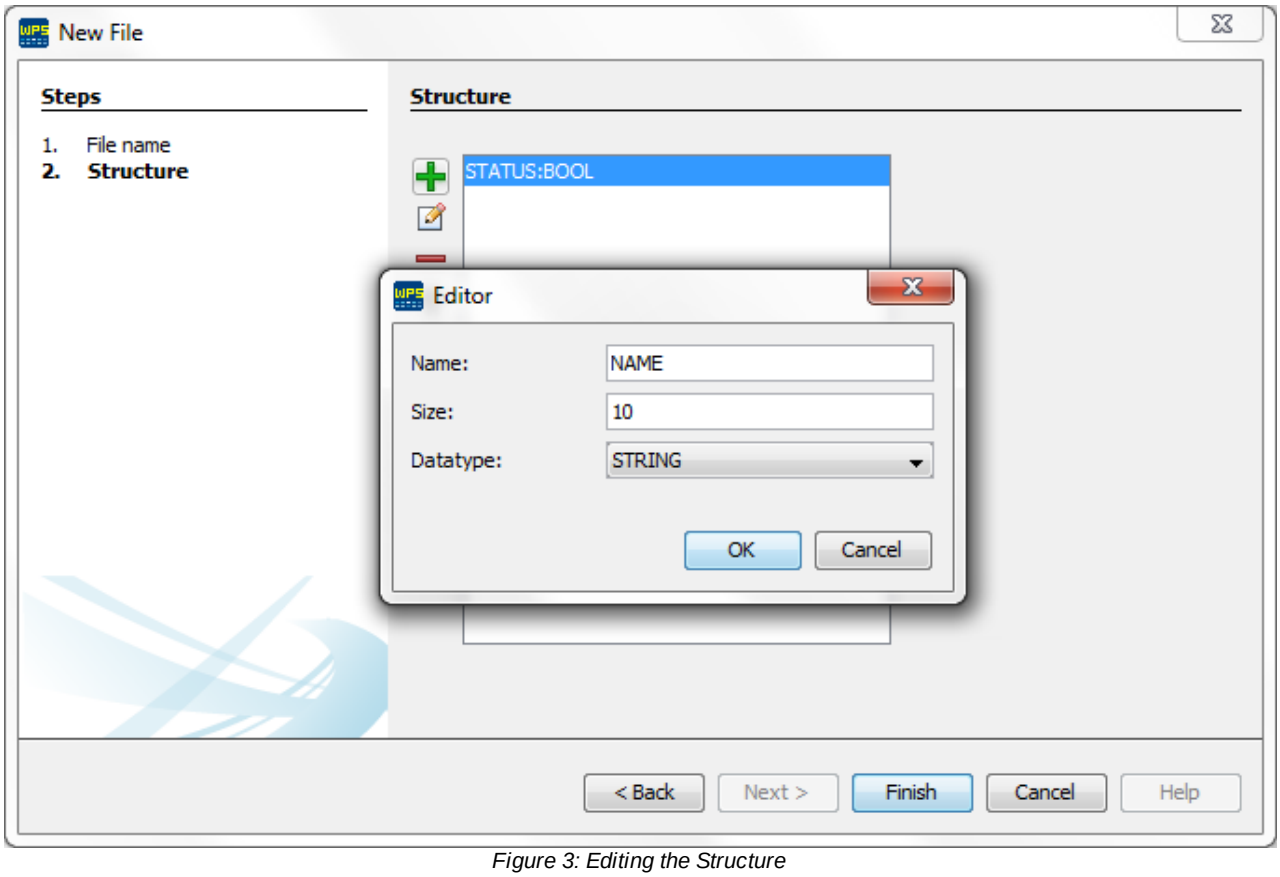


Figure 3: Editing the Structure

4. After finishing the edition of the structure, click on the button **Finish**.

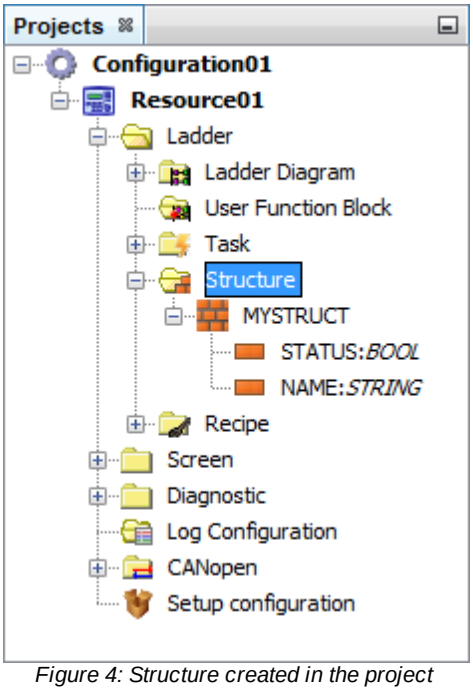


Figure 4: Structure created in the project

Editing a structure

Just double click on the desired structure, as shown in figure 4, and a window will open as shown in figure 3, allowing to insert new data, erase or move the position of the data.

11.8.5.20 Recipes

A recipe is a data set organized in the memory which define certain configurations for a process, such as: time of each step, minimum and maximum values, setpoint, number of repetitions, etc.
In order to create a recipe table, first it is necessary to define the data that will compose it through a data structure. To create a data structure, see the content *Structures*.

Creating a recipe

1. With the right button of the mouse on the folder **Recipe**, click on **New file**.

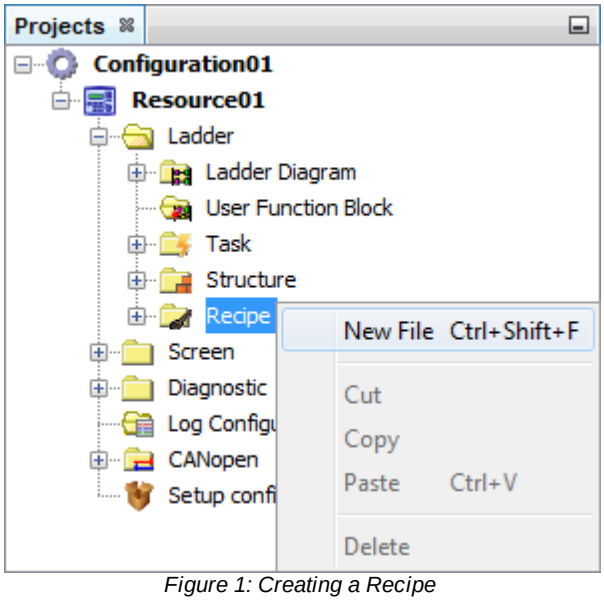


Figure 1: Creating a Recipe

2. Define the file name and press the **Next** button.

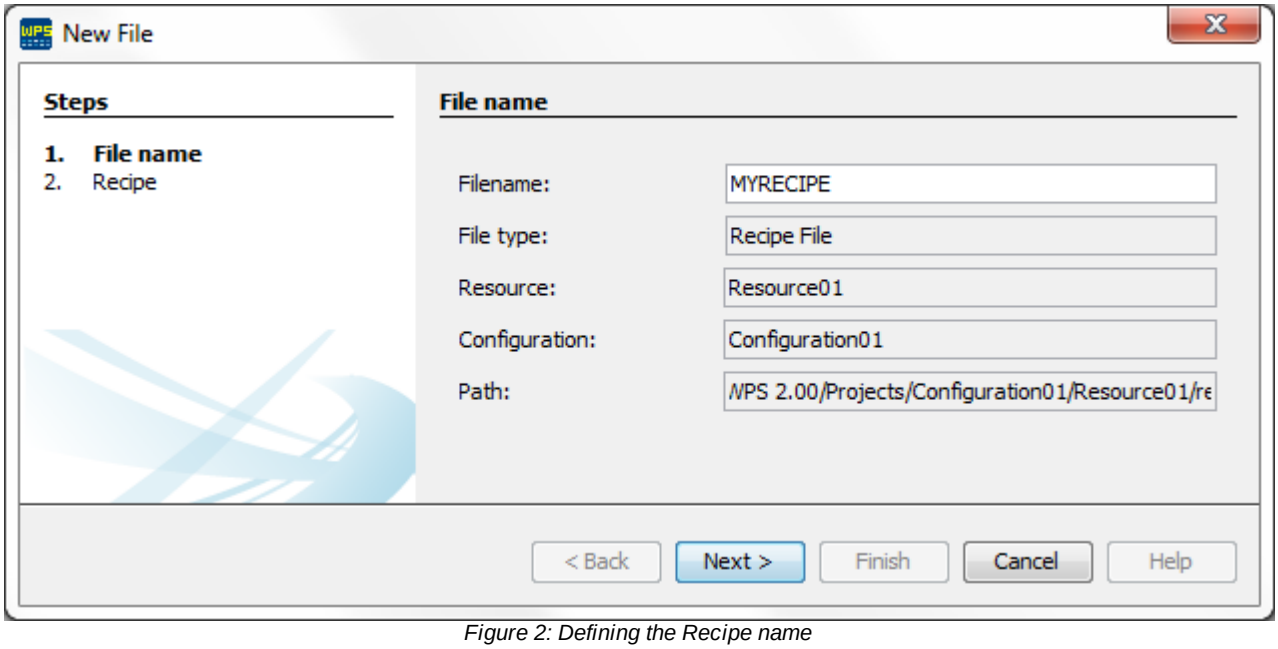


Figure 2: Defining the Recipe name

3. Configure the recipe by configuring the fields as shown in the figure below.

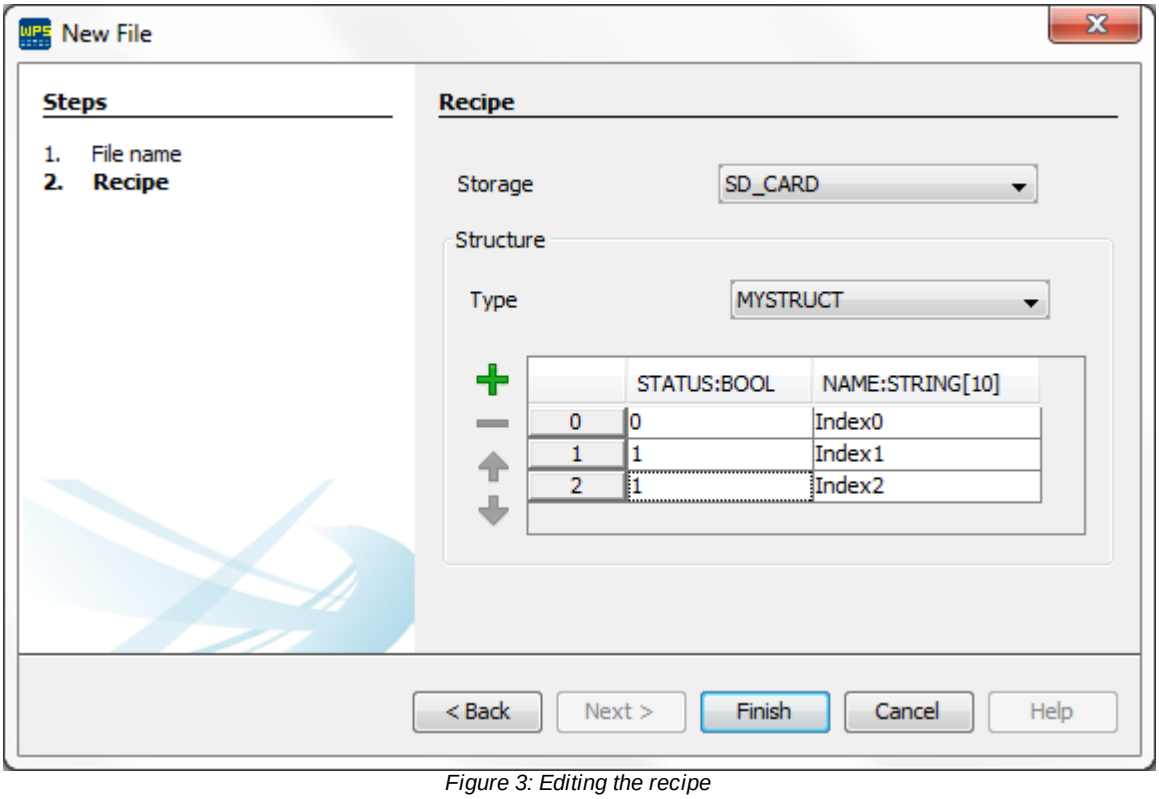


Figure 3: Editing the recipe

- Storage
  - **RAM Memory**: the recipe data are stored in the RAM memory, making its handling faster, but with storage capacity limited to the memory of the PLC300.
  - **SD card**: the recipe data are stored in files in the memory card, making its handling much slower, but with storage capacity according to the capacity of the SD card.
- Type
  - It allows to select all the structures that were defined by the user in the project.
- Data Table
  - The columns represent the structure elements. Note that besides the member name, the respective data type is also presented after the ':';
  - The lines represent each recipe.

4. After finishing the edition of the structure, click on the **Finish** button.

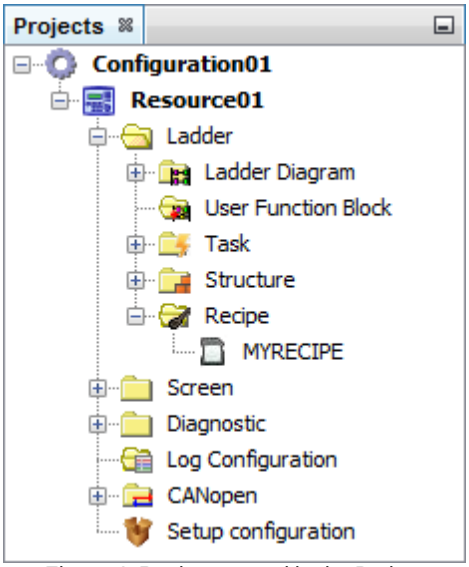


Figure 4: Recipe created in the Project

Editing a recipe

Just double click on the desired structure, as shown in figure 4, and a window will open, as shown in figure 3, allowing to insert new data, erase or move the position of the data.

Using Recipes

To use de recipe data, you must create a variable of the desired structure type:

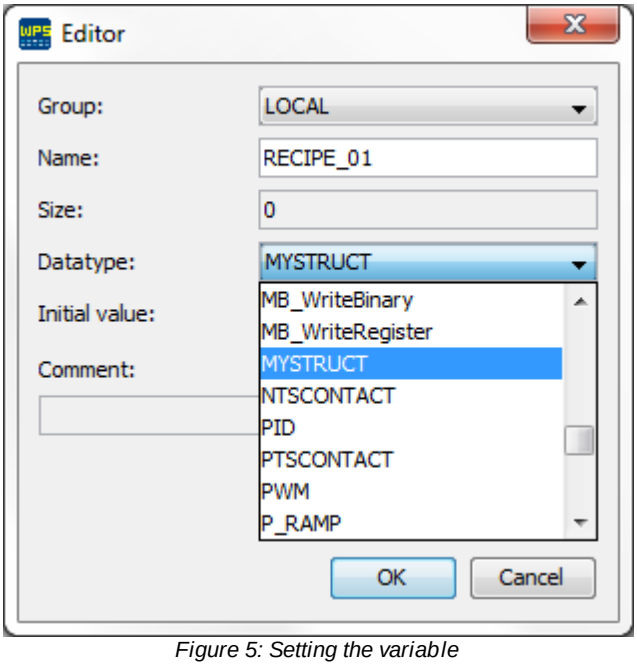


Figure 5: Setting the variable

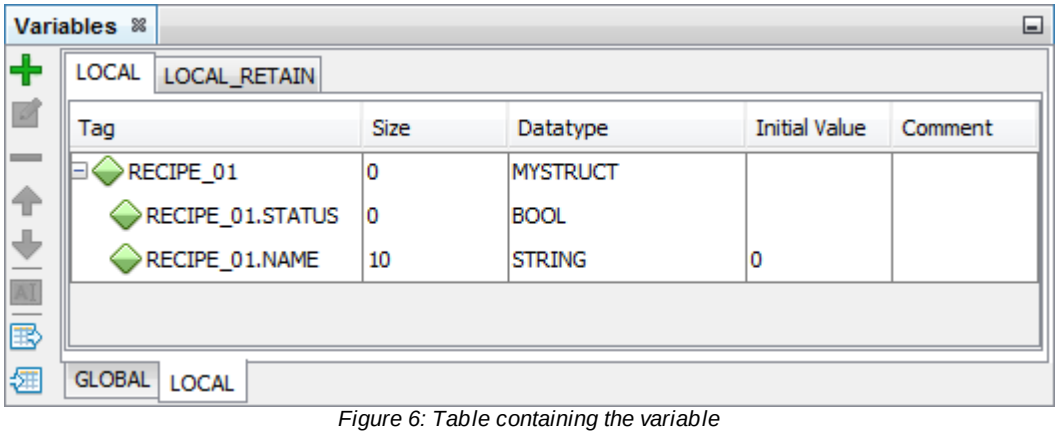


Figure 6: Table containing the variable

After creating the variable, the block ReadRecipe should be used to load the recipe data for the variable or WriteRecipe to record the data contained in the variable.

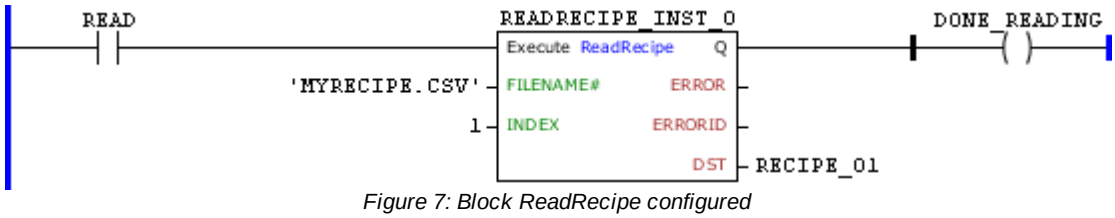


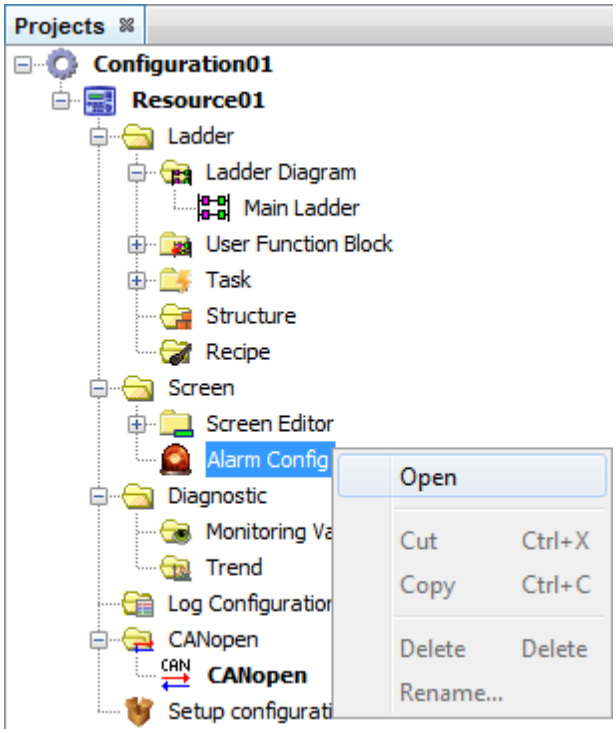
Figure 7: Block ReadRecipe configured

11.8.6 Screen

11.8.6.1 Alarms

Overview

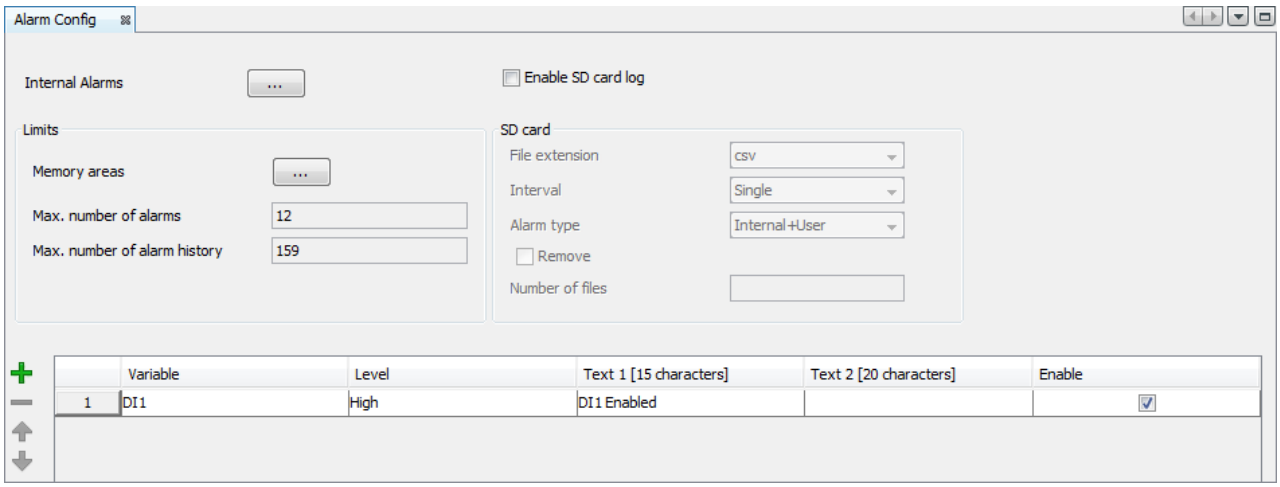
Alarms are important tools in the automation of processes, allowing the user to monitor their plant by checking critical points and signaling to the operator.
Configurable alarms in PLC300 are programmed by the user, being activated by a bit marker that can be enabled by the Ladder program.
Internal alarms are relative to some hardware components that occupy the internal memory.
The alarms are configured through the **Alarm Config** screen accessed by the project folders:



Alarm Configuration

In the alarm configuration window, it is configured:

- **Internal alarms**: Alarms generated by the device that can be enabled or disabled by the user;
- **Limits**: Maximum area taken by the user's alarms and by the alarm history;
- **SD card**: Alarm storage configurations in the SD card;
- **User's alarm table**: Table for configuration of the alarms activated through the variables of the device.

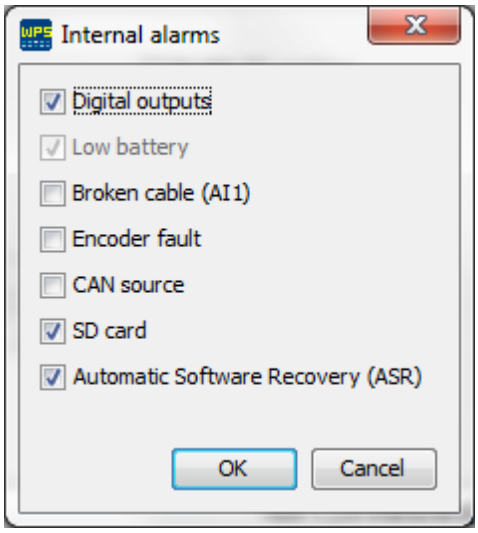


Internal alarms

The internal alarms are alarms generated by the device that can be enabled or disabled by the user. In order to

access the internal alarm configuration window it is necessary to press the button ... in the alarm window.

The PLC 300 has six internal alarms, five o which can be enabled by the user:



- **Digital Output Fault**: Indicates that some problem is occurring in some of the outputs DO1 and DO8.
- **Broken Wire**: Current below 2 mA, when the AI1 analog input is in current mode 4 to 20mA.
- **Encoder Fault**: One of the signals of the encoder is missing.
- **Supply of CAN**: Power supply missing in the CAN interface.
- **SD card**: This alarm occurs when there is a problem in the writing or reading of the SD card. The most common problems are: SD card missing, card protected against writing and formatting of the file system different from FAT32 .

Limits

In the field limits, the maximum area taken by the user's alarms and by the alarm history is configured. The button ..., in the limit field, opens the memory area configuration window. In this window you can configure the size of the memory area that the user's alarms and the user's history must take.

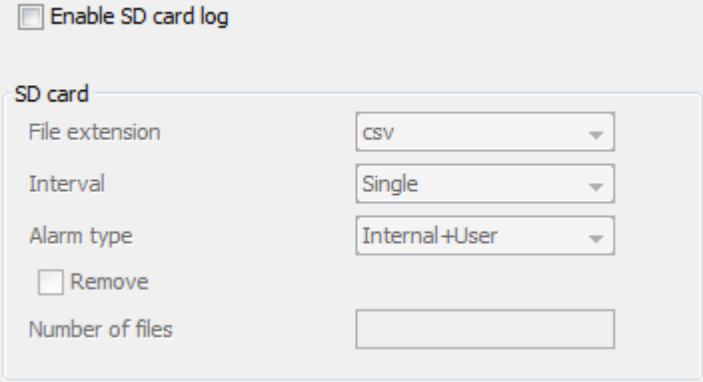
The size that the user's alarm takes in the memory is calculated through the formula:

Alarm size (bytes) = 32 + (80 x number of alarms)

The size that the alarm history takes in the memory is calculated through the formula:

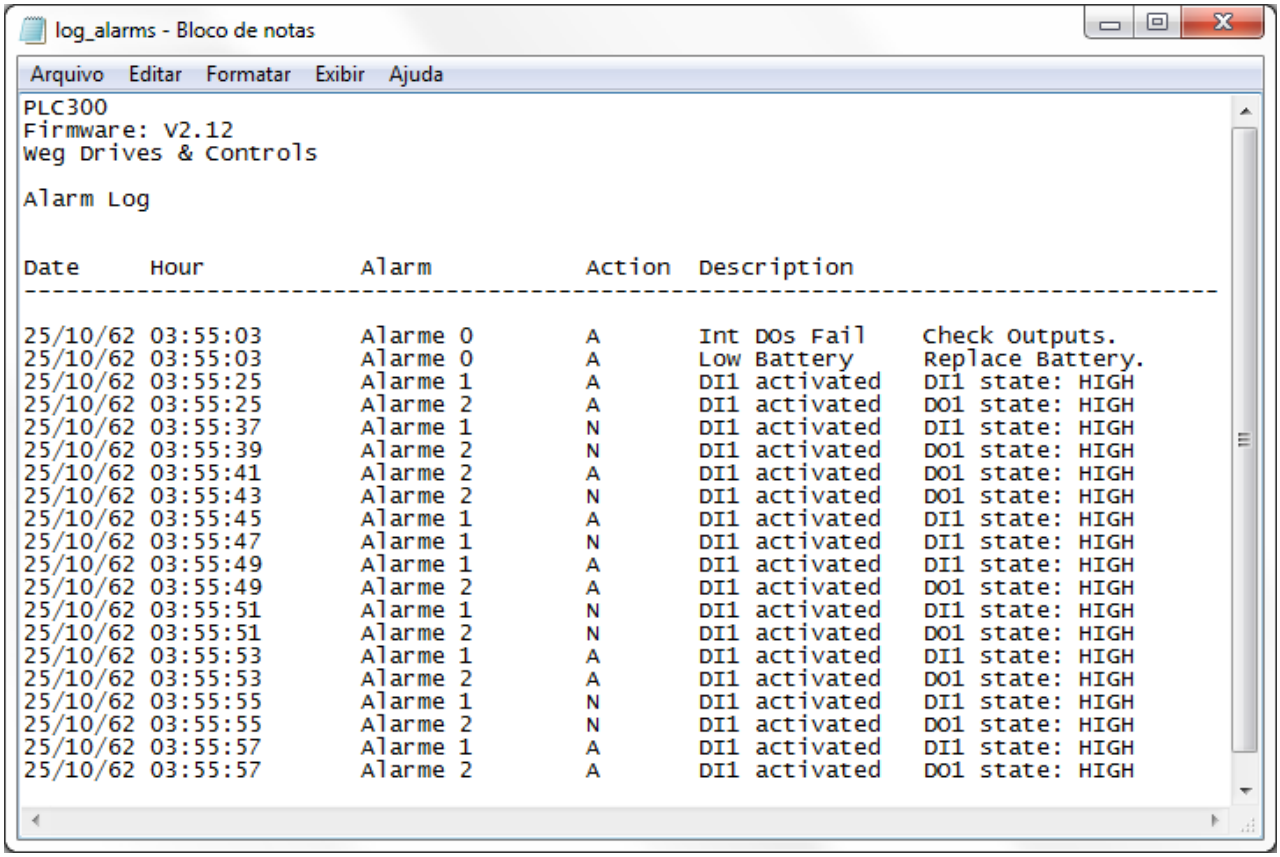
History size (bytes) = 32 x number of histories

SD Card

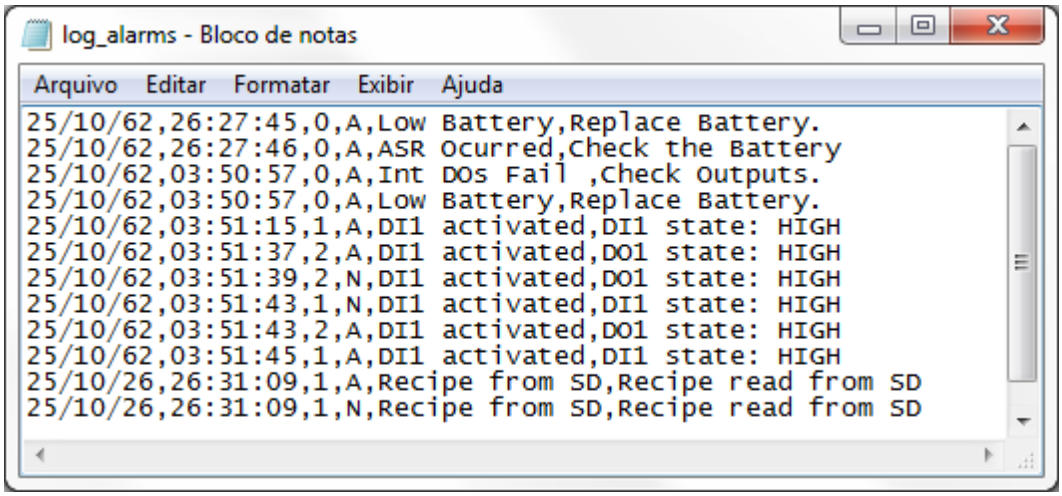


The SD card option, when enabled, configures the storage characteristics of the alarms in the SD card. The storage of alarm files has the following options:

- **File extension**: Format in which the alarms will be stored in the SD card. The options are:
  - **txt**: text with formatting of easy comprehension for the users.



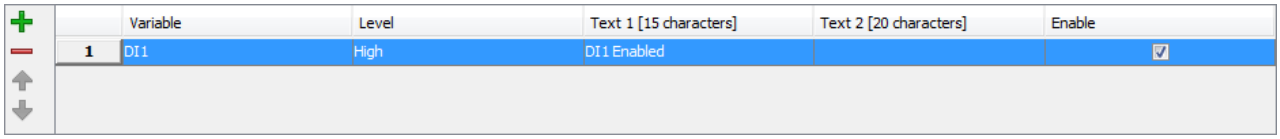
- **csv (comma separated value)**: comma separated values, generally used in spreadsheets.



The values stored are date, time, alarm (0 – internal, 1 – user's alarm), action (A – actuated and N – standardized) and description.

- **Interval**: In the interval field, it is configured the duration time of the recording of the data in a single file. The options of this field are the following:
  - **Single**: The data will be recorded in a single file.
  - **Daily**: The data are recorded in a file a day. The recording of a new file begins whenever the day on the clock of the device changes. The file is recorded with a suffix containing the day, month and year on which its recording began.
  - **Monthly**: The data are recorded in a file a month. The recording of a new file begins whenever the month on the clock of the device changes. The file is recorded with a suffix containing the month and year in which its recording began.
  - **Annual**: The data are recorded in a file a year. The recording of a new file begins whenever the year on the clock of the device changes. The file is recorded with a suffix containing the year in which its recording began.
- **Alarm type**: Alarm values that will be stored in the SD card. They can be of the **User** type to store only the alarms configured by the user or **User+Internal** to store the alarms configured by the user and the internal alarms.
- **Remove**: When this option is selected, the dialog box **Number of files** is enabled, allowing to input a whole number. This number represents the number of files that will be maintained in the SD card. Whenever a file is created daily, monthly or annually, the number of files created for this alarm is checked and then the older files are removed.

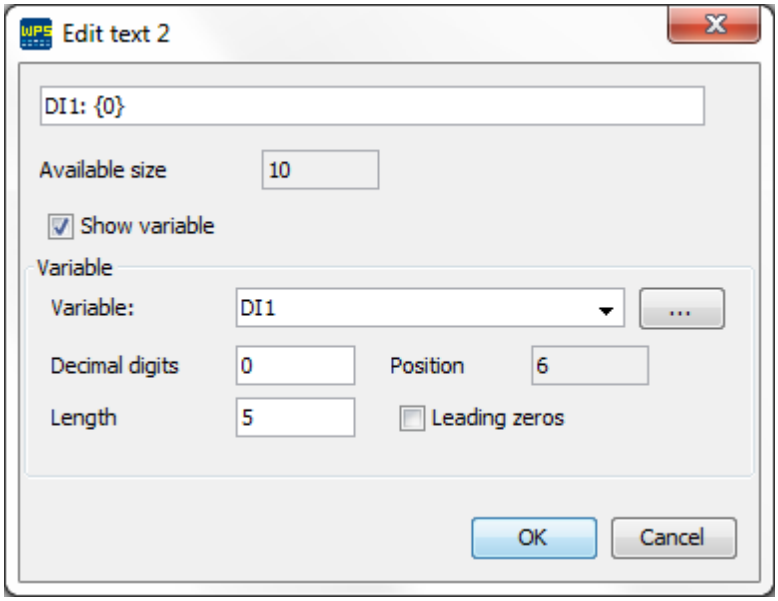
User's alarm table



In the user's alarm table, the alarms activated are configured through variables of the devices with texts edited by the user. The alarm table has four fields to be filled out:

- **Marker**: Bit marker that activates the alarm. Global variable of the Boolean type.
- **Enable**: Enable/Disable alarm option.

- **Edge**: Transition edge in which the alarm will be activated. The possible values are positive (from 0 to 1) or negative (from 1 to 0).
- **Text 1**: Text for the alarm message. This field can contain at most 15 characters. This text will be viewed in the active alarm list and alarm history.
- **Text 2**: Text for alarm message description. This field can contain at most 20 characters. This text will be viewed in the detailed description of the active alarms and alarm history.



The view and configuration fields of text 2 are the followings:

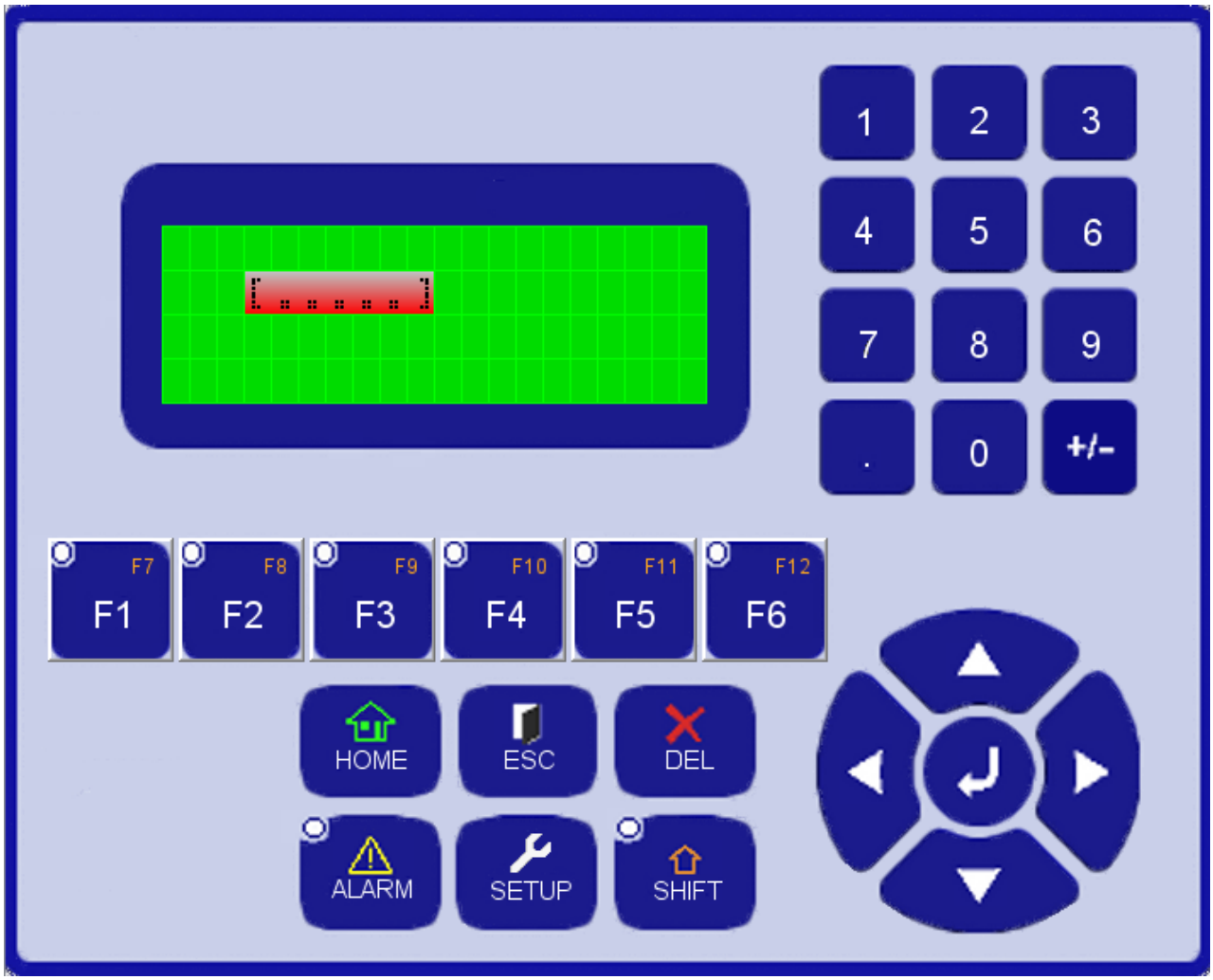
- **Available space**: Number of available characters for editing text. The maximum number of characters is twenty (if the option show variable is selected, the space taken by the variable is added to the characters taken by the text).
- **Show variable**: It enables the option to show variable in text 2 of the alarm. If the text does not have the variable location marking **{0}**, it is automatically added.
- **Variable**: Variable that will be displayed in the text, in the position and format defined in the fields: position, decimal digits, length and filling with zeros.
- **Decimal digits**: Number of decimal digits for displaying the variable.
- **Position**: Position in which the variable will be inserted. This is a read-only field and it is updated at each position change.
- **Length**: Space that will be reserved for displaying the variable. The user must take care to reserve enough space for displaying signal and decimal point, if necessary.
- **Leading zeros**: it fills the spaces that are empty between the configured length and the variable size with zeros.

11.8.6.2 Screen Editor

Overview

The screen editor allows the user configuring the screens by adding and removing components, so that values will be read and written in the program or showed on the display of the device.
The editor has function keys that allow browsing the screens or manipulating bits. A level of access can be configured for each screen, so that only authorized users will see the content.

NOTE!
The PLC300 allows editing at most 512 screens.



The components used to edit the screens are the following:

Bargraph: Shows the value of a variable graphically;

Text: Shows the text on the screen;

Numeric Input: Enters numeric values in a variable;

Numeric Output: Shows the numeric value of a variable;

Message: Shows configured texts on a table, selected by means of an index;

Text Input: Enters variables values in ASCII code;

Text Output: Shows variable values in ASCII code.

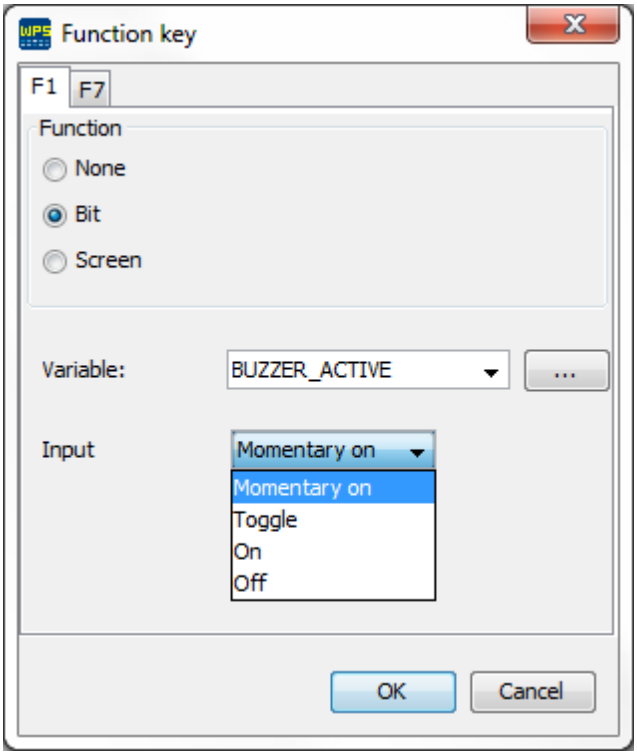
Function Keys

The browsing of screens and the manipulation of bits are done by means of the configuration of the function keys.

The options available to configure the keys are the following:
- **None**: This function key will not execute any actions;
- **Bit**: Selects one of the functions of bit manipulation;
- **Screen**: Selects one key for browsing.

Bit Manipulation

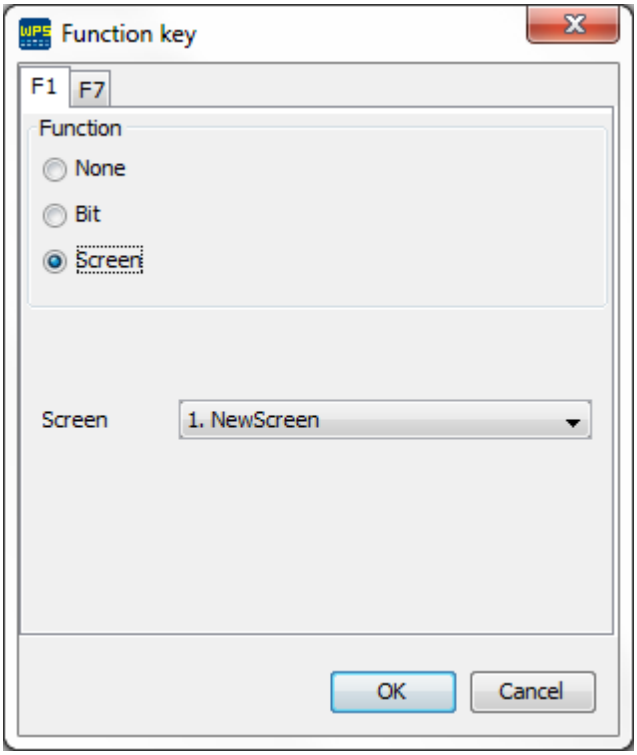
The bit manipulation function allows turning a Boolean variable on, off, inverting its status or turning it off temporarily.
The options available for selection are the following:



- **MomentaryOn**: Turns on a selected Boolean variable (writes value 1) while the key is pressed. When you release the key, the marker returns to zero.
- **Toggle**: Inverts the status of the selected Boolean variable;
- **On**: Turns on a selected Boolean variable (writes value 1);
- **Off**: Turns off a selected Boolean variable (writes value 0).

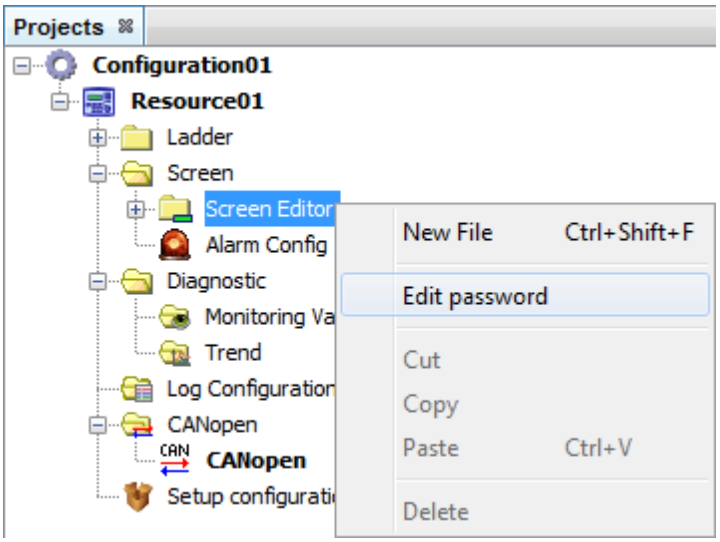
Browsing of the Screens

The browsing through the screens is done by means of the edition of the Screen option.
In the example below, screen 2 will be accessed when F1 key is pressed.

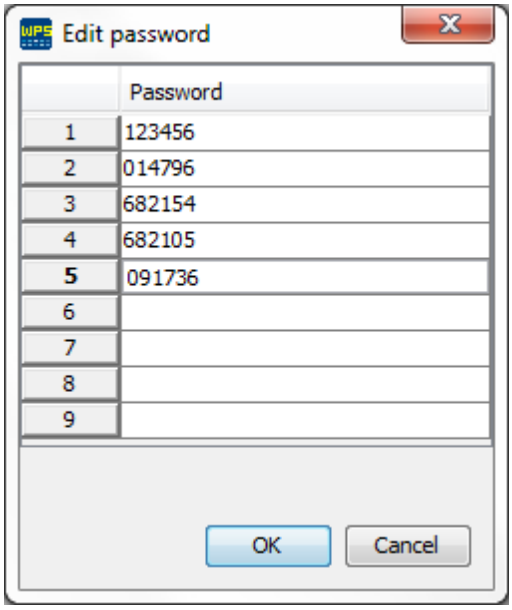


Passwords and levels of access

The PLC300 has ten password levels (0 - 9). Level zero (default value) is chosen when you wish free access to the screen. Except for the zero screen (Home), which always has zero access level, the other screens can be set with an access level. The password configuration window is opened when the options of the **Screen editor** folder are selected.



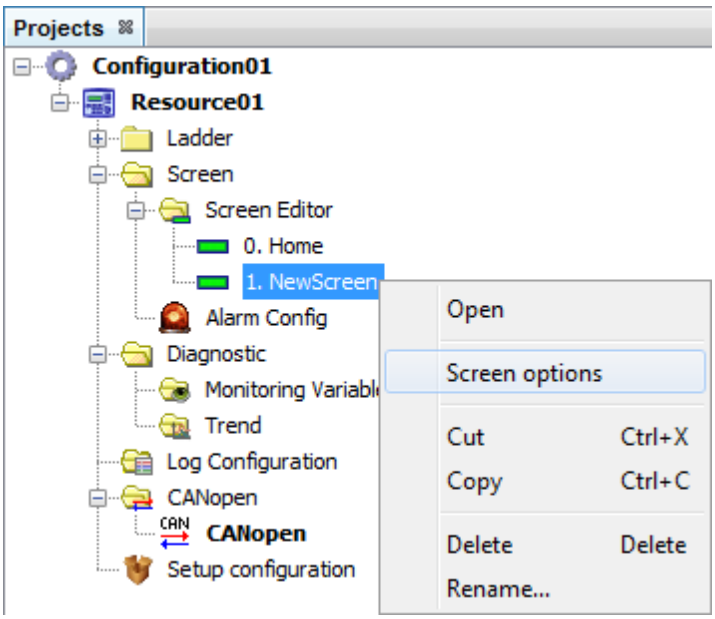
The configured passwords must have six decimal characters.



In order to configure the screen access level, it is necessary to configure this option in the screen options.

Screen options

In the screen options, you can configure the current screen number and the properties of the screen password. In order to open those options, it is necessary to select the options of the screen file in the resource.

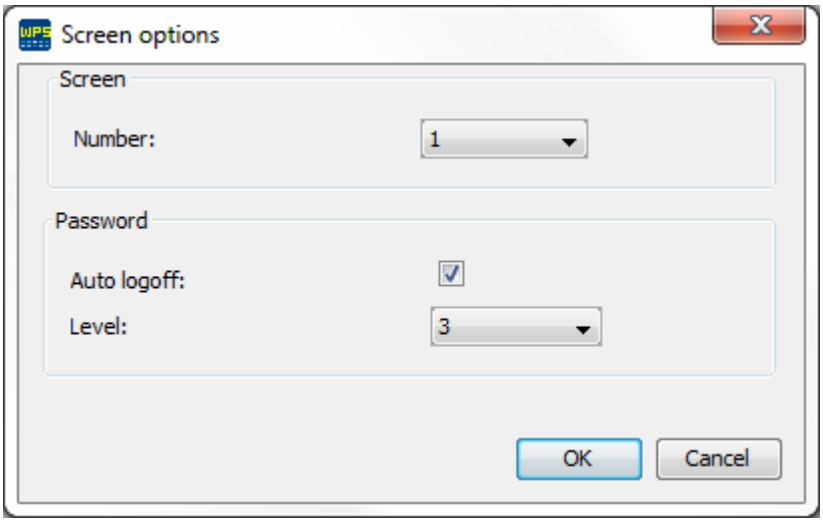


The screen options are the following:

- **Auto Logoff**: When a screen with access level is viewed, if the access level of the screen is higher than the

current level, the device access level becomes the same as the current screen level. If the auto logoff option is enabled, after the access to the screen, the device access level will not change.

- **Level**: Selects the screen access level. The higher the level of access, the greater the privilege. Up to nine access levels can be programmed to allow different kinds of users accessing certain screens. Zero access level means that the screen will not require a password in order to be accessed.

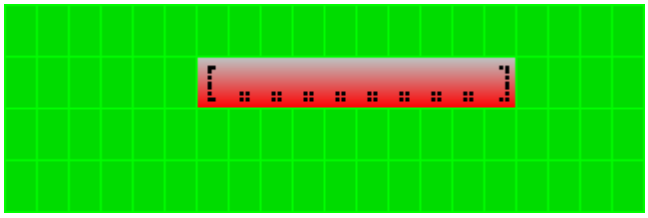


Screen edition

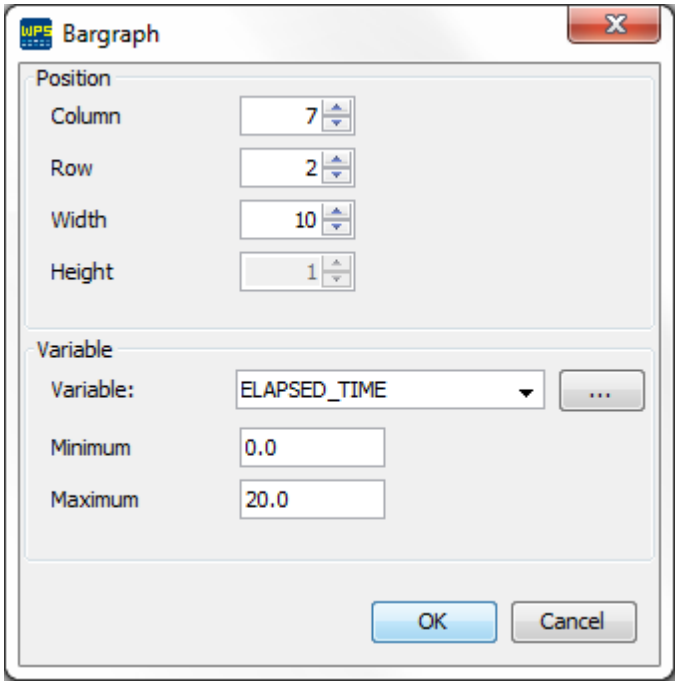
The screens are composed of components, such as Bargraph, Numeric Input, Numeric Output, Text, Message and Text Output. For the setting of the screens, the components are entered in the display by dragging the components from the pallete to the display or by right clicking the mouse on the display.
The components used to edit the screens are the following:

Bargraph

The Bargraph component shows a bargraph on the display with block-type characters proportional to the value of the selected variable.



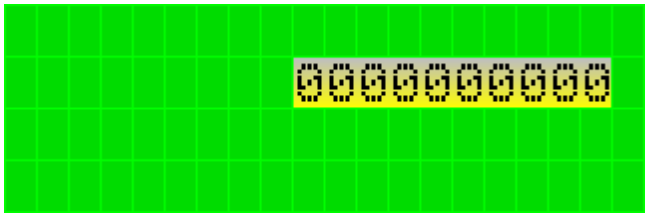
The properties of this component are:



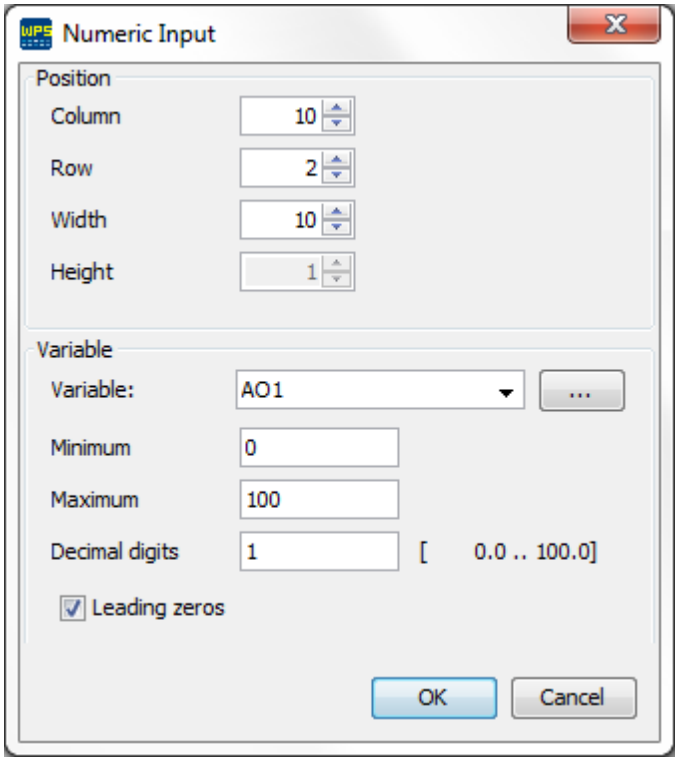
- **Variable**: Variable used to display the bargraph.
- **Maximum**: Maximum value shown by the graph. In case the value of the variable is equal to or above the maximum value, the graph is displayed with the all bars completely filled.
- **Minimum**: Minimum value shown by the graph. In case the value of the variable is equal to or below the minimum value, the graph will not show any bars filled.

Numeric Input

The Numeric Input component allows the user entering a numeric value within a certain range. The entered content is stored in a variable.



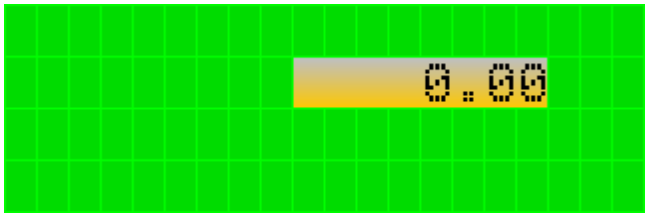
The properties of this component are:



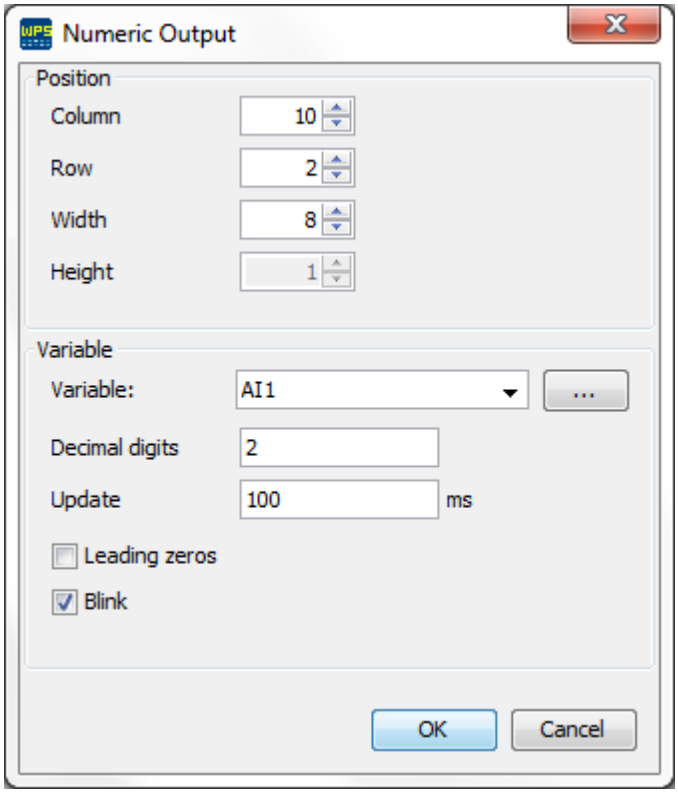
- **Variable**: Variable used to enter data in the display;
- **Maximum**: Maximum value accepted as input for this field. In case the entered value is higher than this value, the maximum value will be assigned;
- **Minimum**: Minimum value accepted as input for this field. In case the entered value is higher than this value, the minimum value will be assigned;
- **Decimal digits**: Number of decimal places in the presentation of the variable on the display;
- **Leading zeros**: Fills with zeros the spaces that are empty between the configured component length and the variable size.

Numeric Output

The Numeric Output component presents the formatted value (decimal places, leading zeros or flashing) of the variable selected on the display.



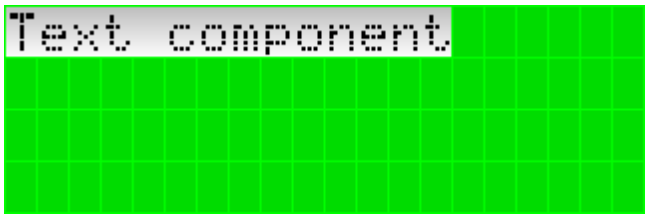
The properties of this component are:



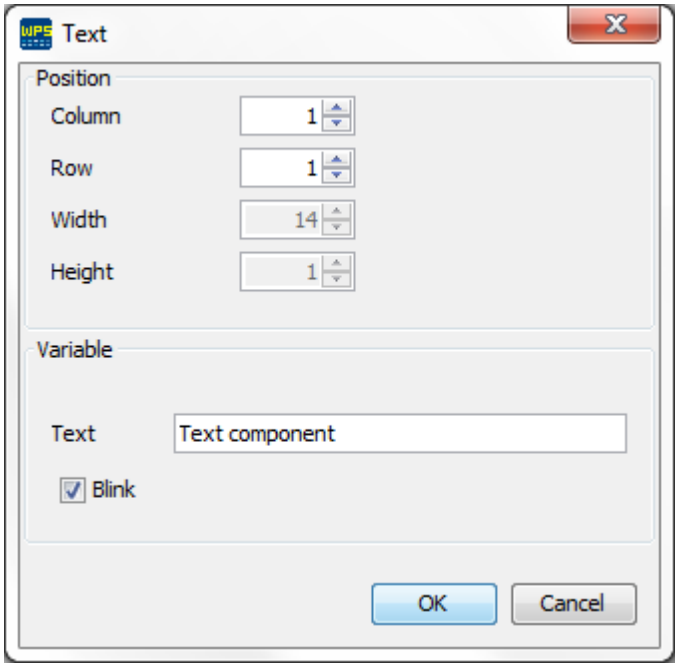
- **Variable**: Variable used to present the data on the display;
- **Decimal digits**: Number of decimal places in the presentation of the variable on the display;
- **Update**: Time (in milliseconds) used for the update of the field on the display. This value is only an approximated value, since is depends on the scan of the device;
- **Leading zeros**: Fills with zeros the spaces that are empty between the configured component length and the variable size;
- **Blink**: Enables the option of blinking text.

Text

The Text component is used to insert fixed texts in the screen.



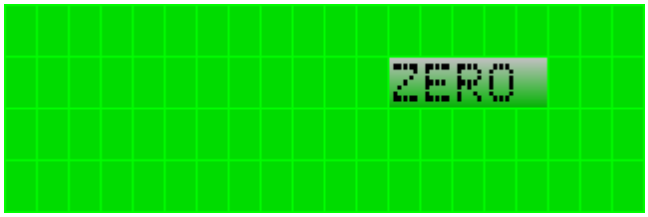
The properties of this component are:



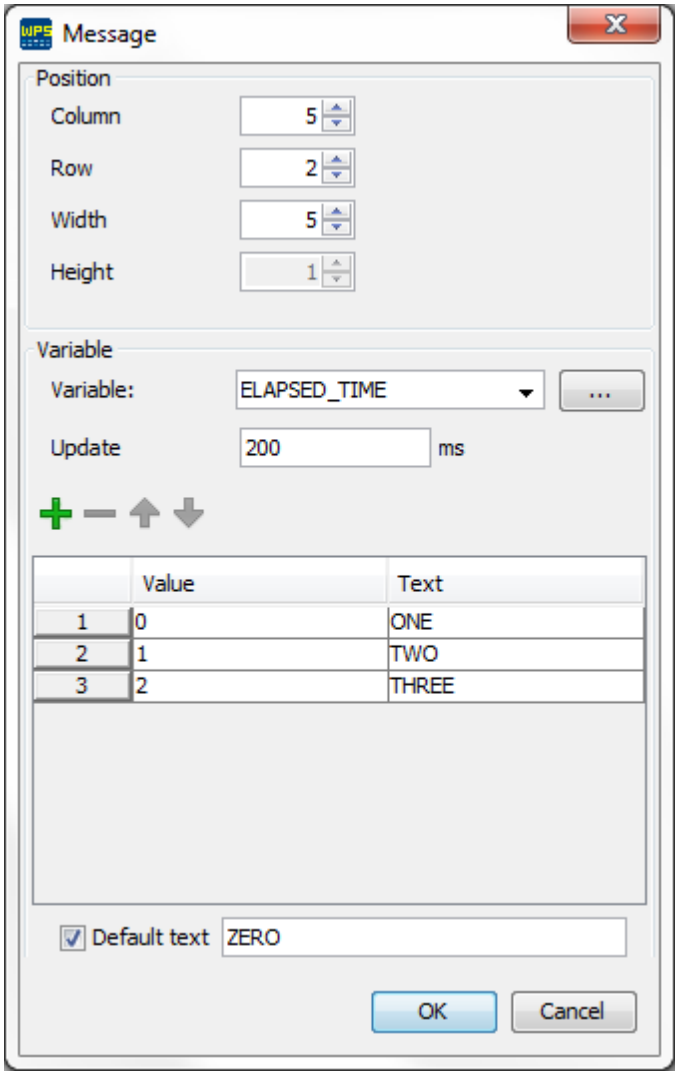
- **Text**: Text displayed on the screen;
- **Blink**: Enables the option of blinking text.

Message

The Message component shows configured texts on a table, selected by means of an index.



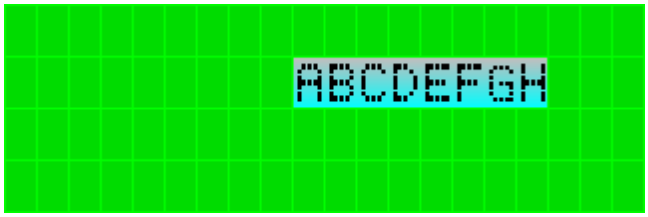
The properties of this component are:



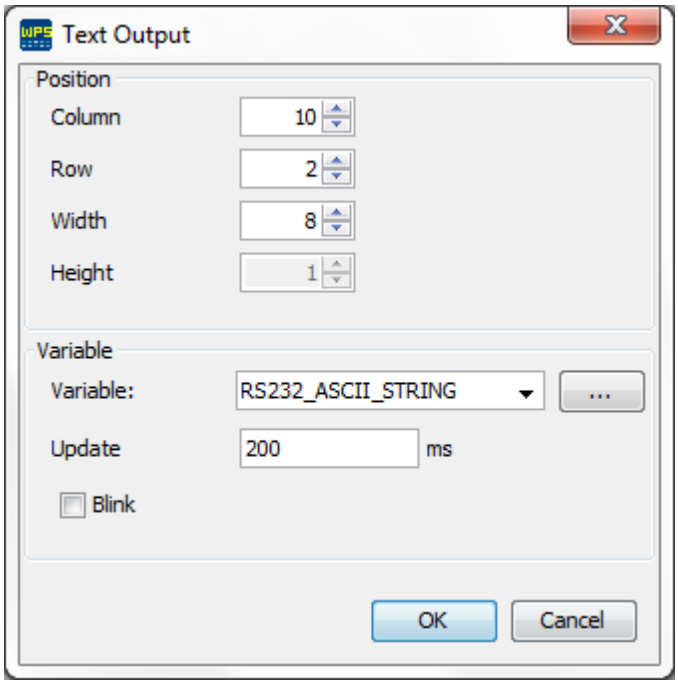
- **Variable**: Variable used as index to display the text on the screen;
- **Update**: Time (in milliseconds) used for the update of the field on the display. This value is only an approximated value, since is depends on the scan of the device;
- Table
  - **Value**: Table index;
  - **Text**: Text displayed when the variable value is equivalent to the table index;
  - **Default text**: Text displayed when the variable assumes a value not set in column value. When this option is disabled the displayed text is the variable value.

Text Output

The Text component displays configurable texts for values of a variable.



The properties of this component are:



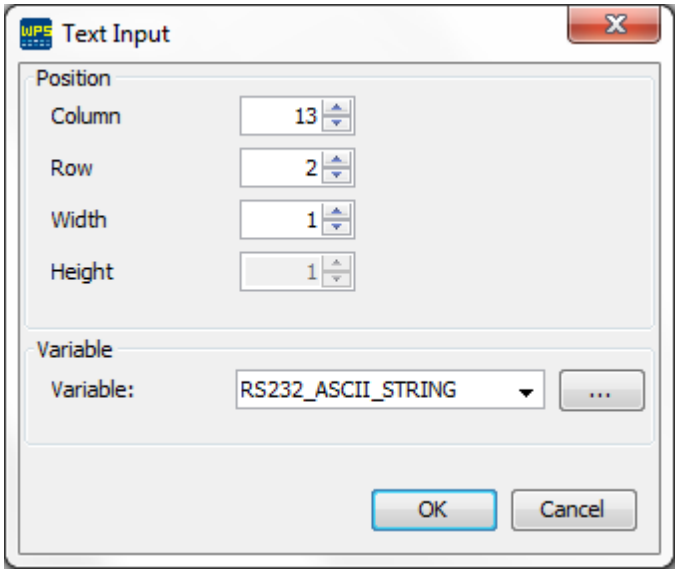
- **Variable**: STRING type variable containing the text;
- **Update**: Time (in milliseconds) used for the update of the field on the display. This value is only an approximated value, since is depends on the scan of the device;
- **Blink**: Enables the option of blinking text.

> **NOTE!**
> A BYTE variable was used instead of STRING in firmware versions below 2.10. The array size was defined by the component size.

Text Input

The Text Input component allows the user to enter text;

The properties of this component are:



- **Variable**: STRING type variable used as storage for the text entered.

NOTE!
A BYTE variable was used instead of STRING in firmware versions below 2.10. The array size was defined by the component size.

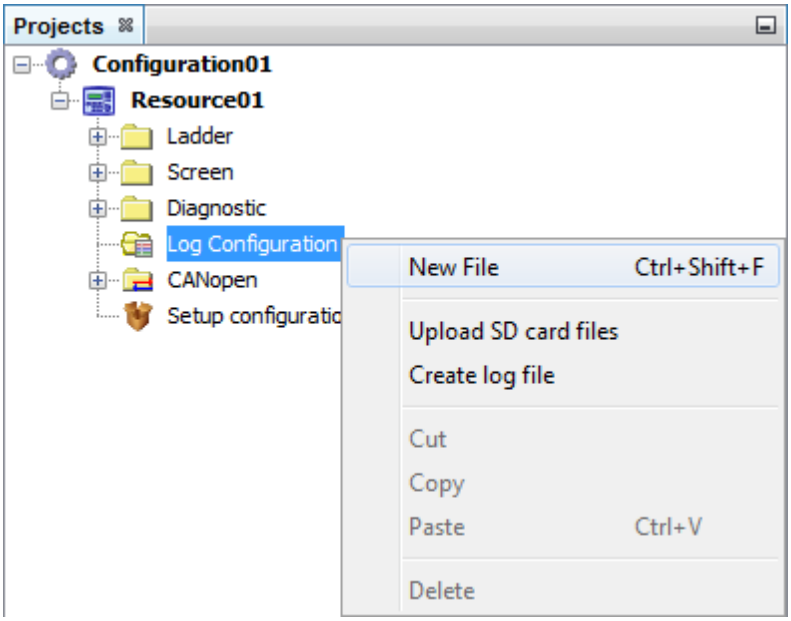
11.8.7 Event Log

Overview

The event log is a set of variable values that are stored, with date and time, in the SD card of the device in the csv (comma separated value) format. Those values are recorded after the occurrence of events that may be of the following types: time, change of state or trigger. For each event log file, the file recording interval is configured and a Boolean variable, which is responsible for enabling or disabling the log through the program, is associated to it.

A text may be associated to each log configuration file, which will be presented together with the variables as soon as the upload of the event log is performed (see *Text Field Configuration* section).

In the option menu of the event log, it is possible to carry out the following actions:

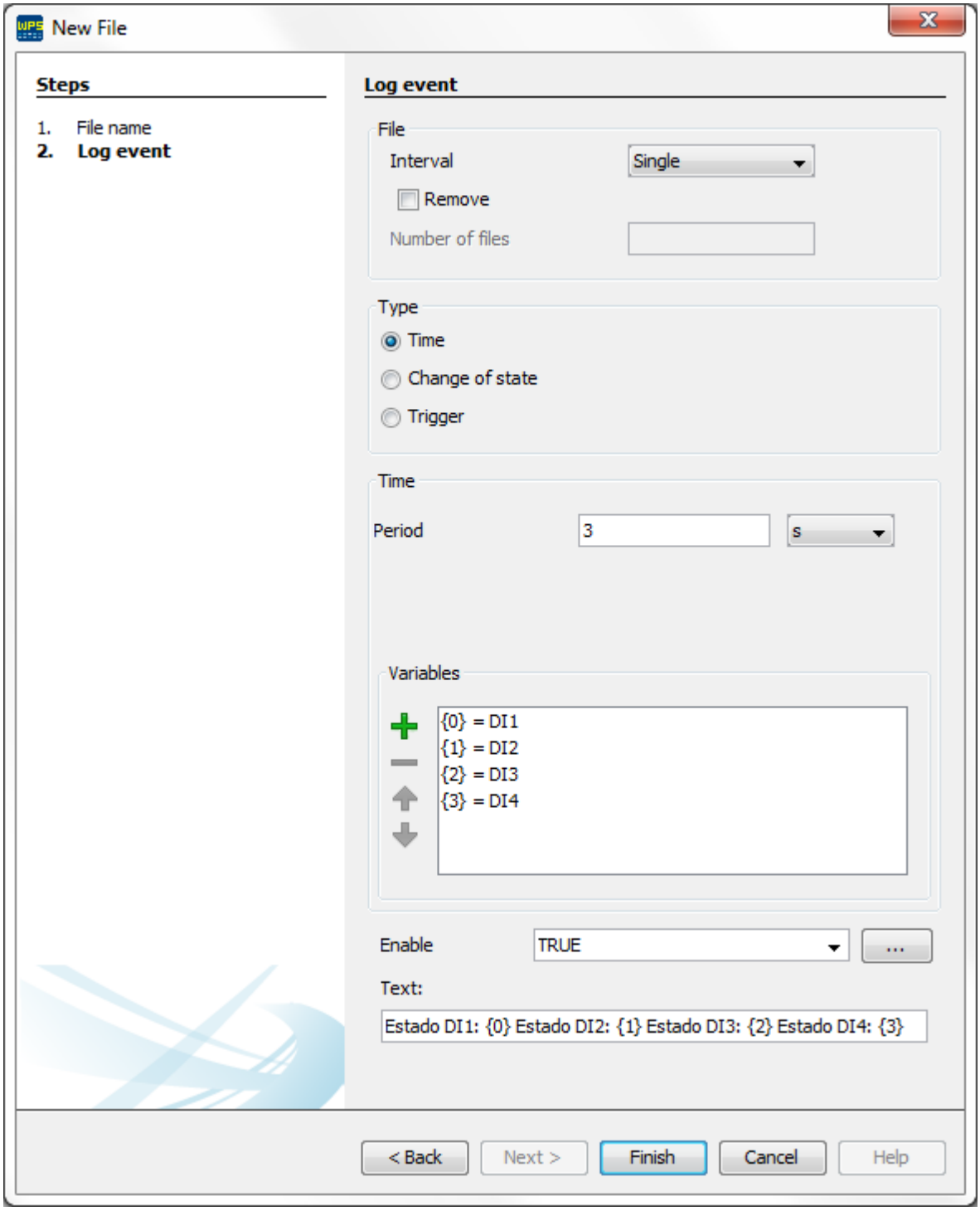


- **Add a new event log**: in order to add a new event log, it is necessary to select the **New file** option from the folder **Event log**. For further information on how to configure the event log, see *Event log configuration* section.
- **Upload SD card files**: after the download of the event log files configured for the equipment, it is possible to upload these files to view the events occurred. For more details, see *Upload SD card files* section.
- **Create log file**: This functionality allows the user to create a log file within a defined period with the log files loaded in the project. For more details see *Creation of log files* section.

Event Log Configuration

In the event log configuration window, you configure: the recording interval of the event log file, the type of event log, the Boolean variable that enables the event log and the text to be displayed in the event log.

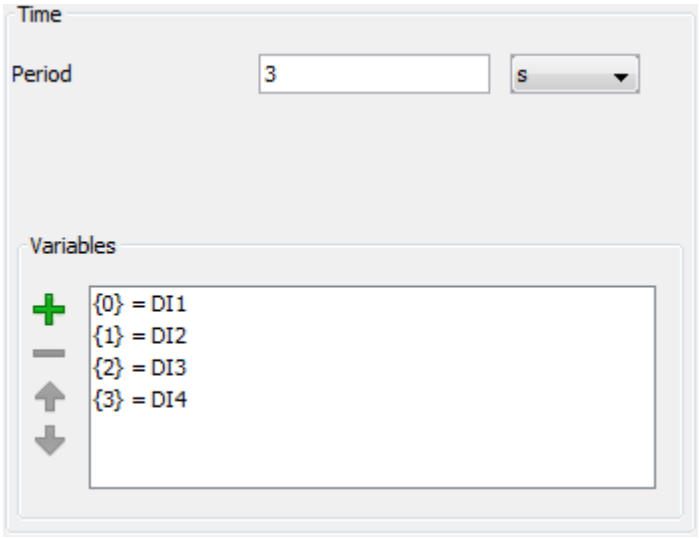
- **File interval**: In the file interval field , the duration time of the recording of the data in a single file is configured. The options of this field are the following:
  - **Single**: The data will be recorded in a single file.
  - **Daily**: The data will be recorded in a file a day. The recording of a new file begins whenever the day on the clock of the device changes. The file is recorded with a suffix containing the day, month and year on which its recording began.
  - **Monthly**: The data will be recorded in a file a month. The recording of a new file begins whenever the month on the clock of the device changes. The file is recorded with a suffix containing the month and year in which its recording began.
  - **Annual**: The data will be recorded in a file a year. The recording of a new file begins whenever the year on the clock of the device changes. The file is recorded with a suffix containing the year in which its recording began.



Event log type

The event log can be of time, change of state and trigger type.

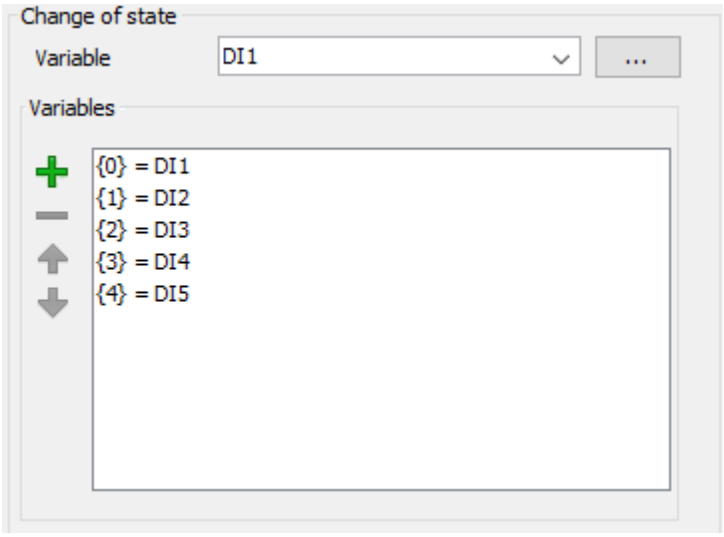
- **Time**



The event log is recorded in time intervals defined in the field **Period**. The value configured in the field **Period** must be a whole value greater than zero and smaller than 4294967295 s. The time unit can be selected from seconds, minutes or hours.

In the field **Variables** you select the variables that will be sampled in the configured period.

- **Change of State**

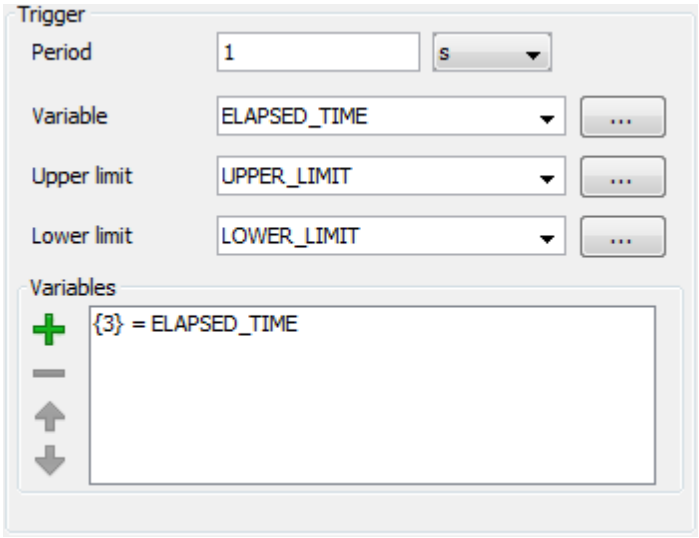


The event log is recorded after a change in the value of the selected variable. Only the value of the selected variable is recorded in the event log.

In the field **Variables** you select the variables that will be sampled after the change of state.

NOTE!
The list of variables is possible only in firmware versions higher than 3.30. For these versions, the variable that triggers the event log is not written in the event log, unless the variable is added to the list.

- **Trigger**



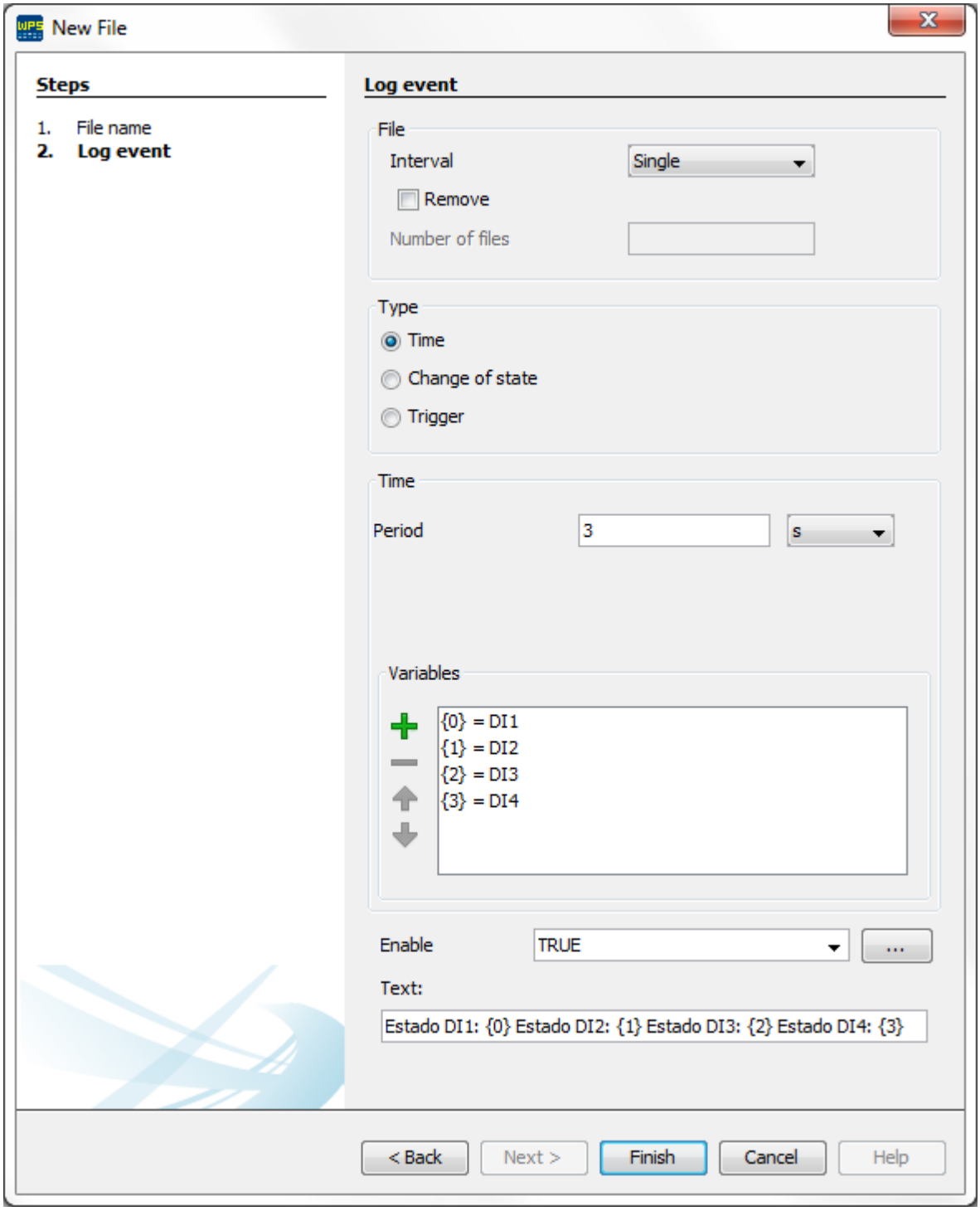
As in the **Time** event, the event log is recorded in time intervals defined in the field **Period**, but only when the value of the configured variable in the field **Trigger** is within the limits selected in the variables of the **Upper limit** and **Lower limit** fields. The value configured in the **Period** field must be a whole value greater than zero and smaller than 4294967295 s. The time unit can be selected from seconds, minutes or hours.

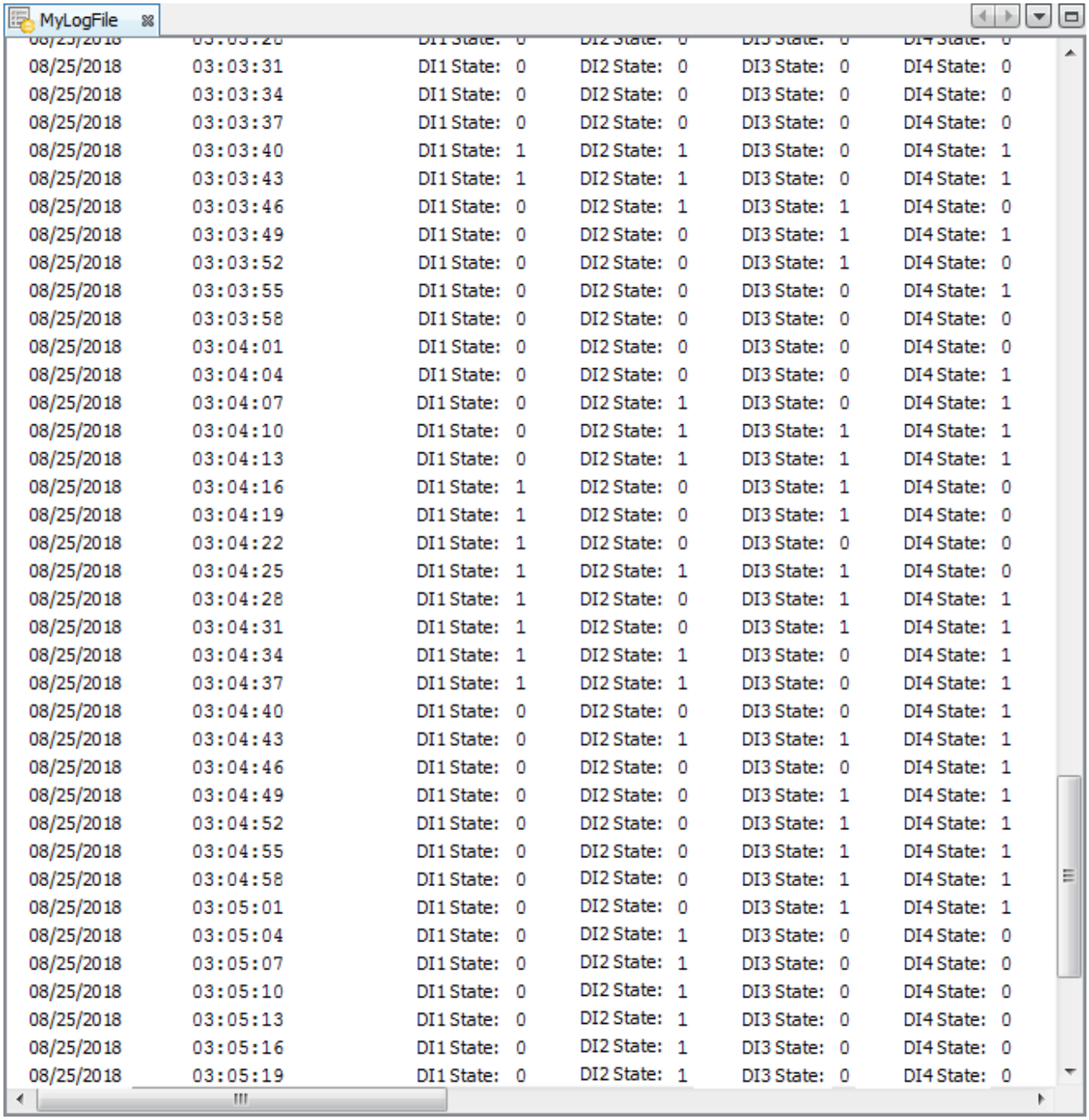
In the field **Variables** you select the variables that will be sampled in the configured period.

Text field configuration

In the **Text** field, it is possible to add a text to be shown together with the variables. This text will be added as soon as the event log file is loaded (through the upload of log files) to the WPS.
For the variable values to be inserted in any position of the text, markers will be added to the text which will be replaced by the variables.
According to the type of event log selected, the markers must be added the following way:

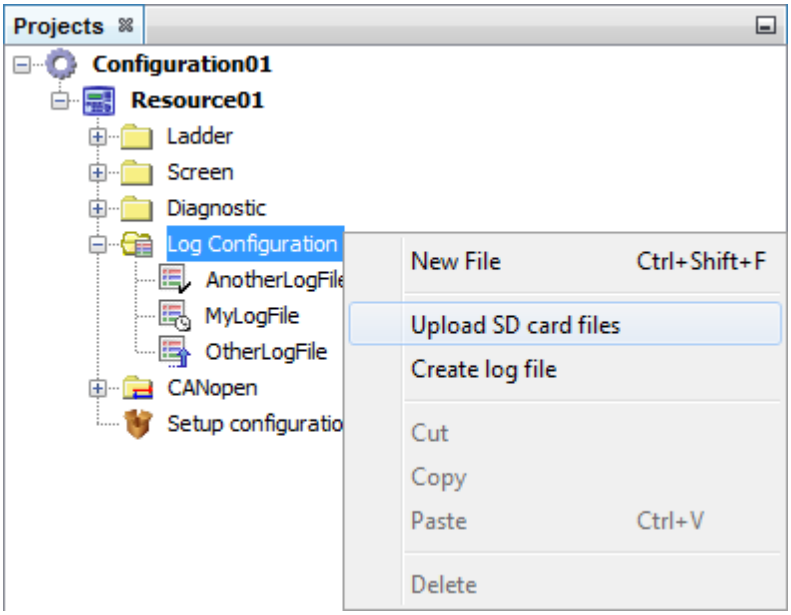
- **Time**: the first variable of the list is represented by the marker {0}, the second by the marker {1} and thus successively.
- **Change of State**: the only variable selected will be represented by the marker {0}.
- **Trigger**: the **Trigger** variable is represented by the marker {0}, the **Upper limit** by the marker {1}, the **Lower limit** by the marker {2}, the first variable of the list by the marker {3}, the second by the marker {4} and thus successively. As an example, below is the configuration of an event log of the **Trigger** type and its log file.


New File
Steps
1. File name
2. Log event
Log event
File
Interval
Single
Remove
Number of files
Type
Time
Change of state
Trigger
Time
Period
3
s
Variables
{0} = DI1
{1} = DI2
{2} = DI3
{3} = DI4
Enable
TRUE
...
Text:
Estado DI1: {0} Estado DI2: {1} Estado DI3: {2} Estado DI4: {3}
< Back
Next >
Finish
Cancel
Help

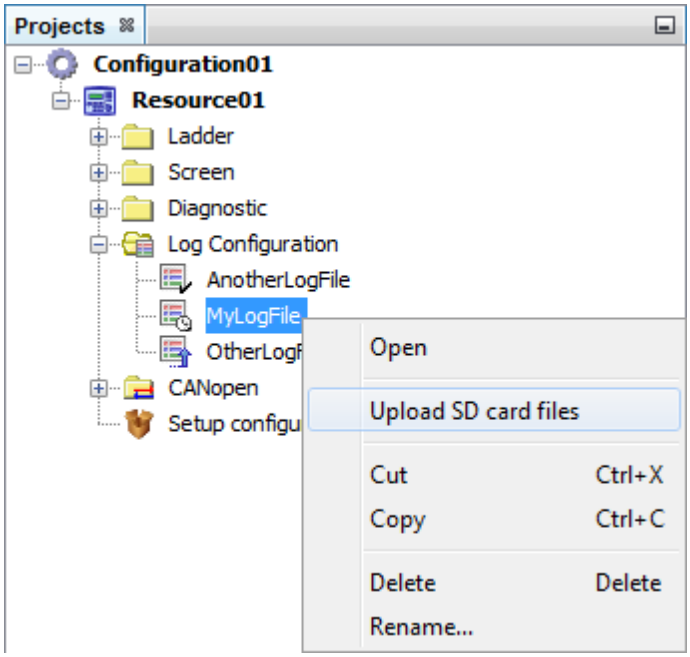


Upload of log files

There are two ways to upload the log files: upload all the project files or upload the files individually.



In order to upload all the log files configured in the project, it is necessary to select the folder **Event log** with the right button and select the option **Upload SD card files**.

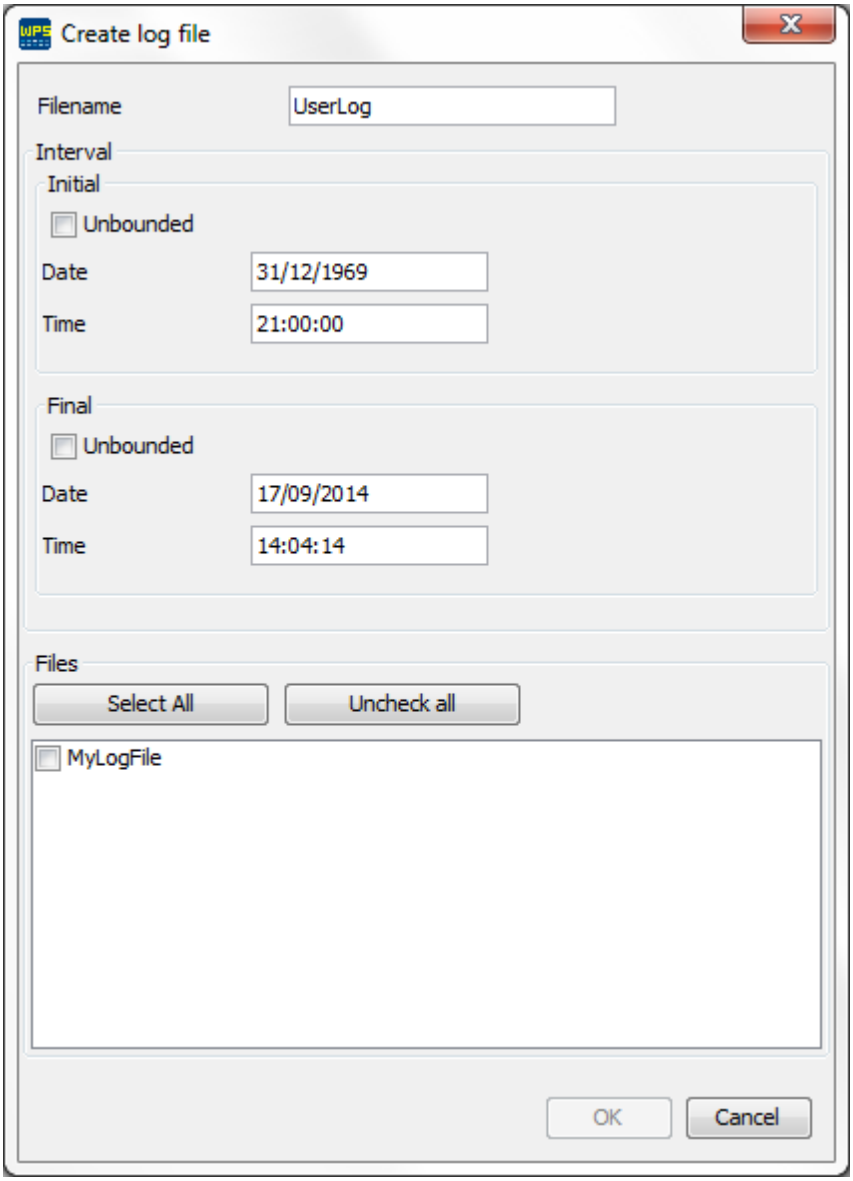


In order to upload the files associated to only one event log configuration file, it is necessary to click the right button on the event log configuration file and select the option **Upload SD card files**.

ATTENTION!
When uploading files, the previous files (with the same name) will be overwritten.

Creation of log files

Through the log file creation tool, it is possible to create new event log files using the log files previously loaded on the project. Thus, it is possible to define the initial and final log interval and which log files must be used. In order to begin the creation of a new log file, it is necessary to click the right button of the mouse on the folder **Event log** and select the option **Create log file**. The configuration options for the new file are the following:



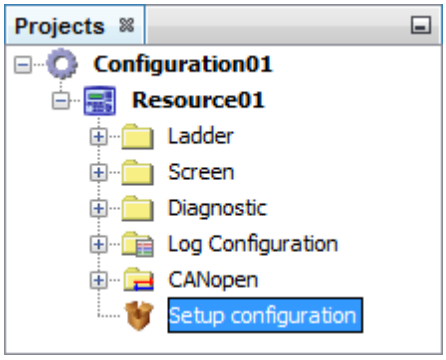
- **File name**: Name that will be used in the log file created. If the name already exists, the data will be overwritten.
- **Initial interval**: It determines the initial day and time of the file logs. If the **Unlimited** option is selected, there will not be a minimum value for the date.
- **Final interval**: It determines the final day and time of the file logs. If the option **Unlimited** is selected, there will not be a maximum value for the date.
- **Files**: It selects the log files that will be used to generate the new log file.

11.8.8 Setup

11.8.8.1 Configuration

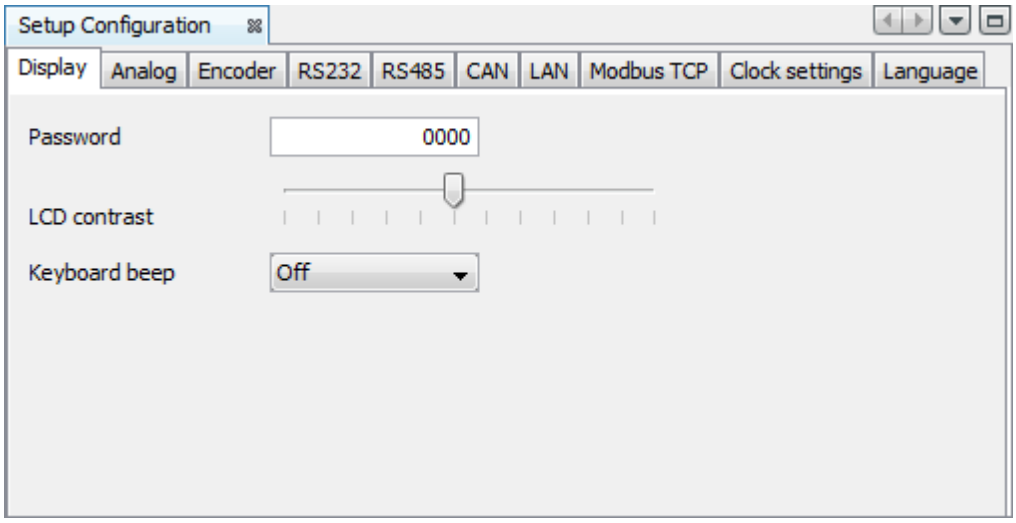
Overview

The setup configuration of the PLC300 is accessed by double clicking the shortcut available in the resource as shown in the following figure.



ATTENTION!
When creating a new resource in the WPS, the setup configuration values will be the standard values of the PLC300. We recommend to review these values according to the requirements of the application and to send these adjustments according to the following explanation.

The setup configuration window is divided into tabs and has two buttons to access the PLC300 as shown in the following figure.

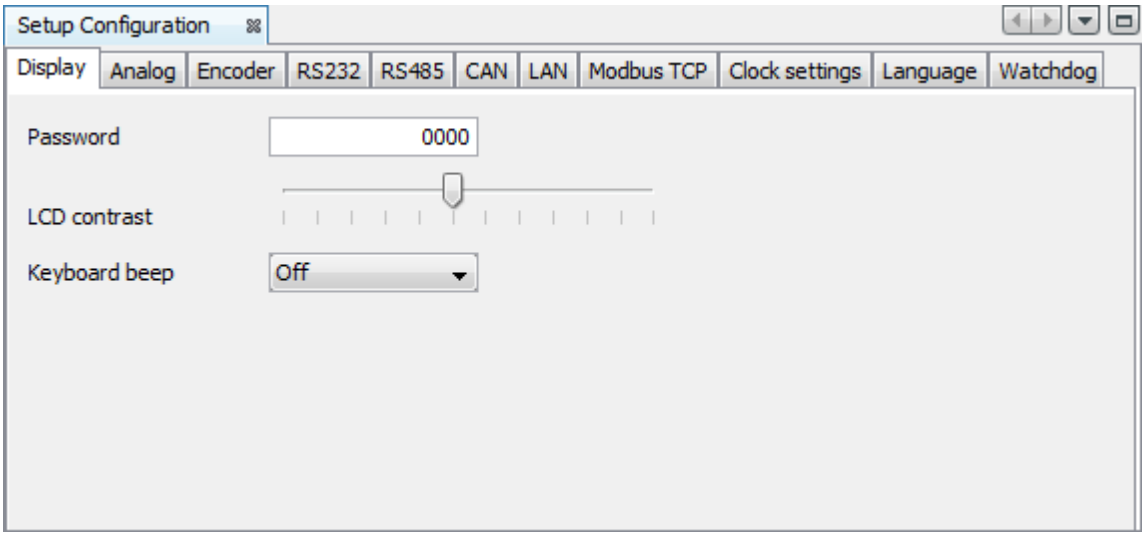


The **Write Configuration** and **Read Configuration** buttons are only active when the WPS is connected to the PLC300 through the command **Connect Device** .

- **Write Configuration**: It sends all configurations adjusted in the setup configuration screens to the equipment.
- **Read Configuration**: It receives all the configurations adjusted in the equipment and configures the screens according to the received values.

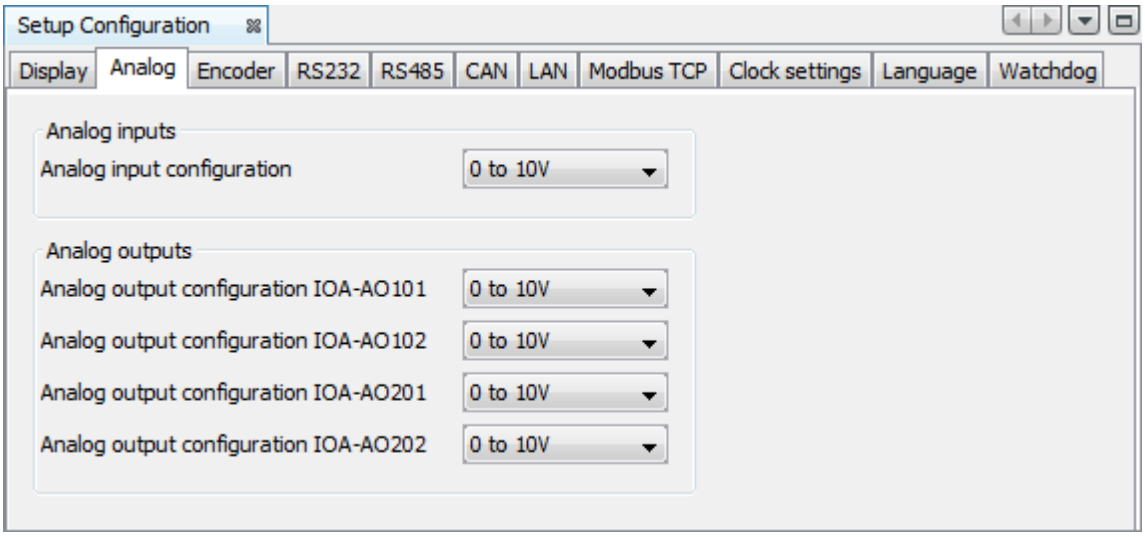
11.8.8.2 Configuration Windows

11.8.8.2.1 Display



- **Password**: It allows to change the password of the setup function of the PLC300. The standard password is '0000'. The new password must be a number with four digits.
- **LCD Contrast**: It adjusts the LCD contrast of the PLC300.
- **Keyboard Beep**: It enables the beep of the keys of the PLC300.

11.8.8.2.2 Analog

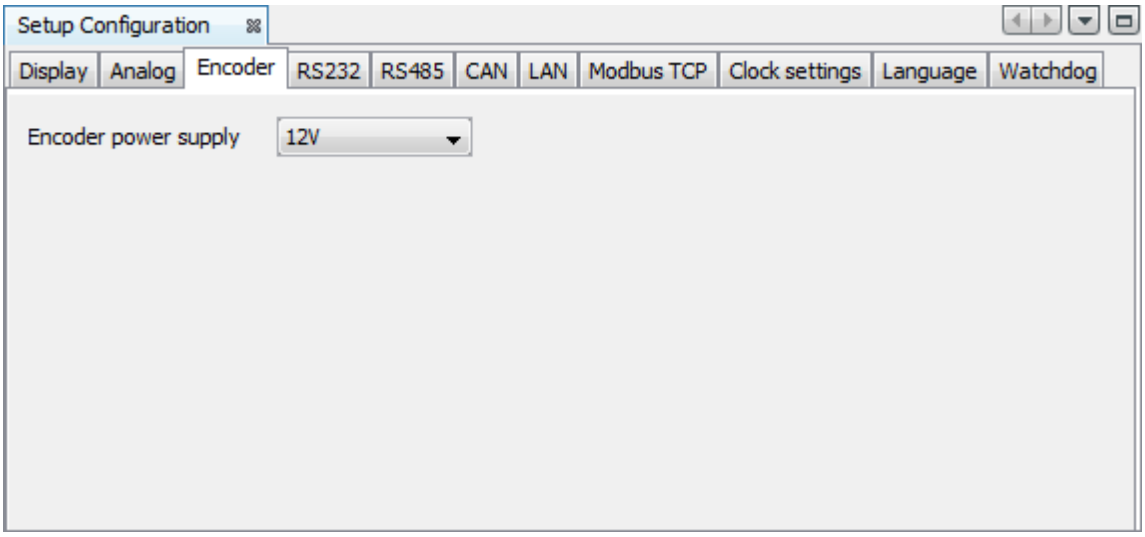


- **Analog Inputs**: It selects one of the three operation modes of the AI1 analog input of the PLC300: 'Voltage 0 to 10 V', 'Current 0 to 20 mA' or 'Current 4 to 20 mA'.

NOTE!
In the option 4 to 20 mA, the value that the Ladder sees is a proportional, standardized value, that is, 4 to 20 mA - 0 to 32767.

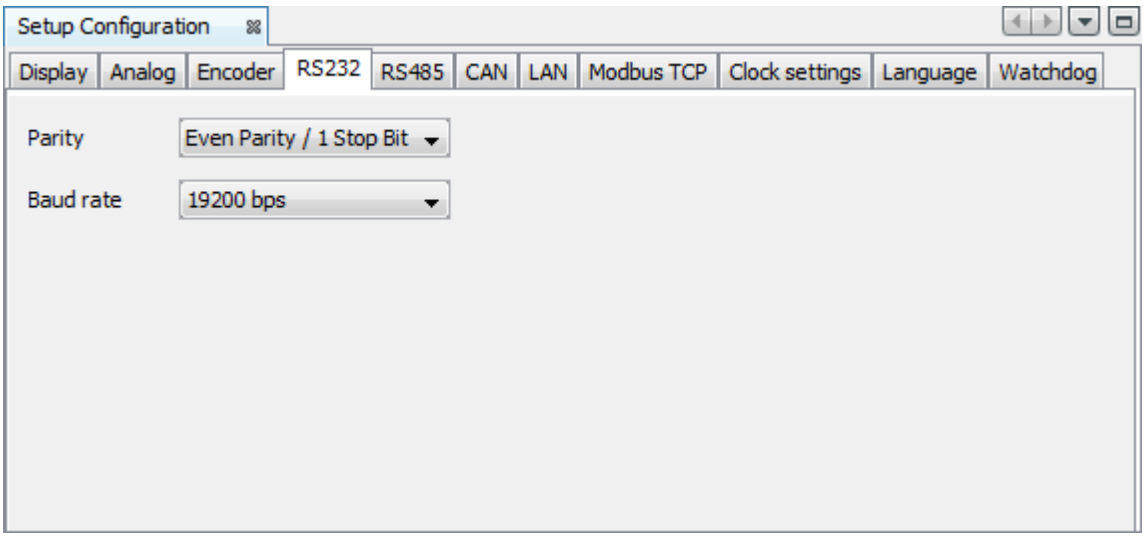
- **Analog Outputs**: It selects one of the four operation modes of the analog outputs (AOs) of the IOA accessories installed in the PLC300, seeing that the AOs 101 and 102 are the AOs 1 and 2 of the IOA card installed in the slot 1 of the PLC300, and the AOs 201 and 202 are the AOs 1 and 2 of the IOA card installed in slot 2.

11.8.8.2.3 Encoder



It selects the power supply of the encoder of the PLC300 between 5 or 12V.

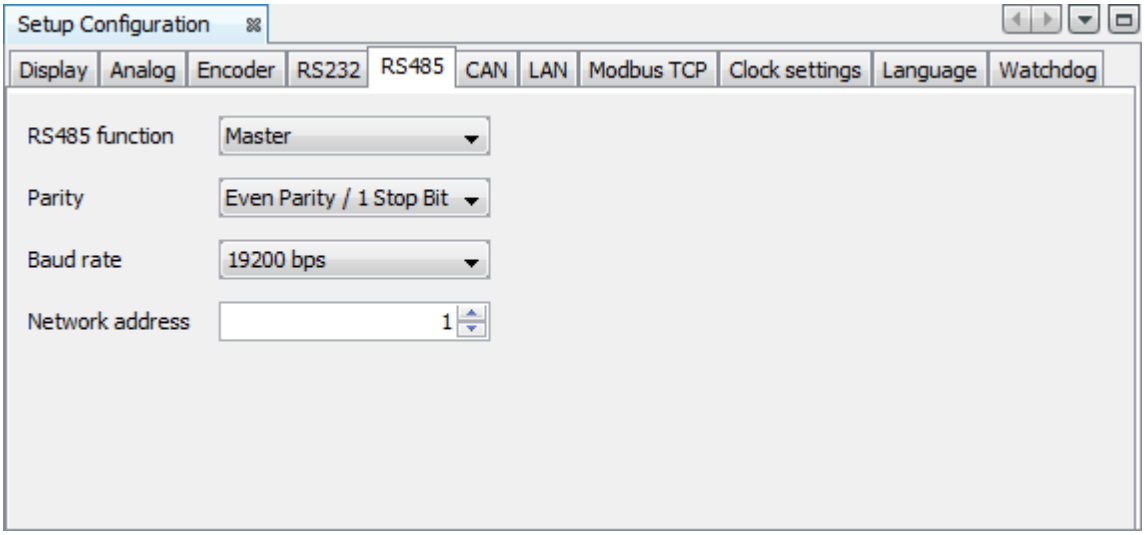
11.8.8.2.4 RS232



It is possible to configure the baud rate, parity and number of stop bits of the serial interface of the RS232 of

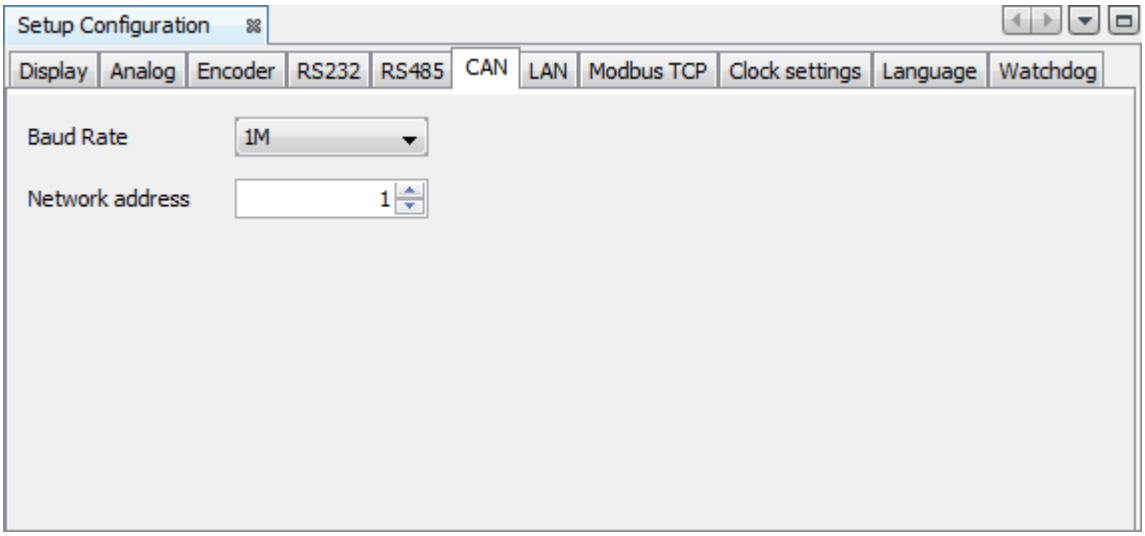
the PLC300.

11.8.8.2.5 RS485



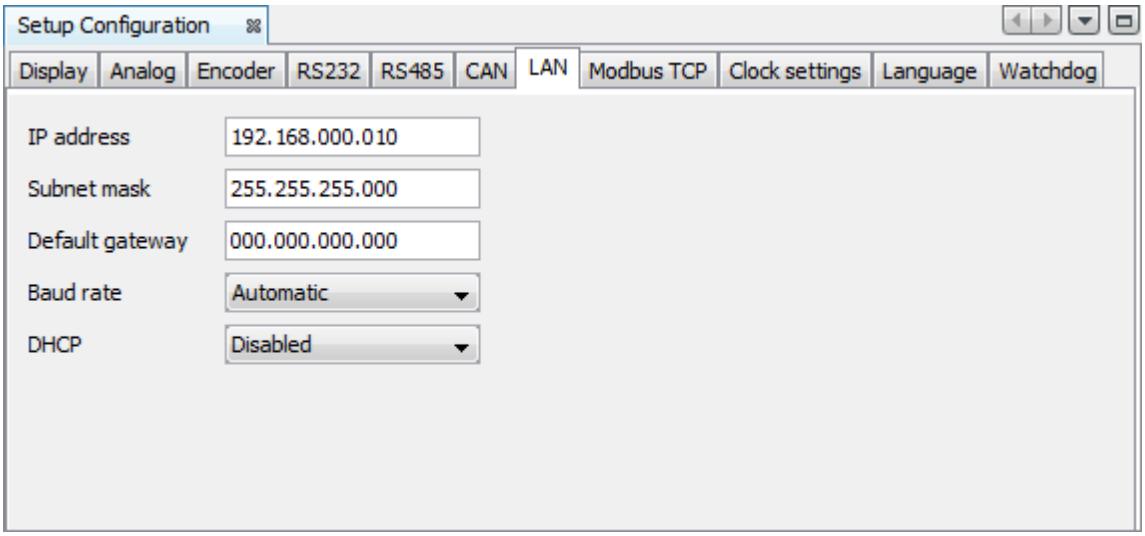
It is possible to configure the baud rate, the parity and the number of stop bits, the mode (master/slave) and the address of the PLC300 in a Modbus RTU network by means of the serial interface RS485 of the PLC300.

11.8.8.2.6 CAN



It is possible to configure the baud rate and the address of the PLC300 in a CANopen network by means of the CAN interface of the PLC300.

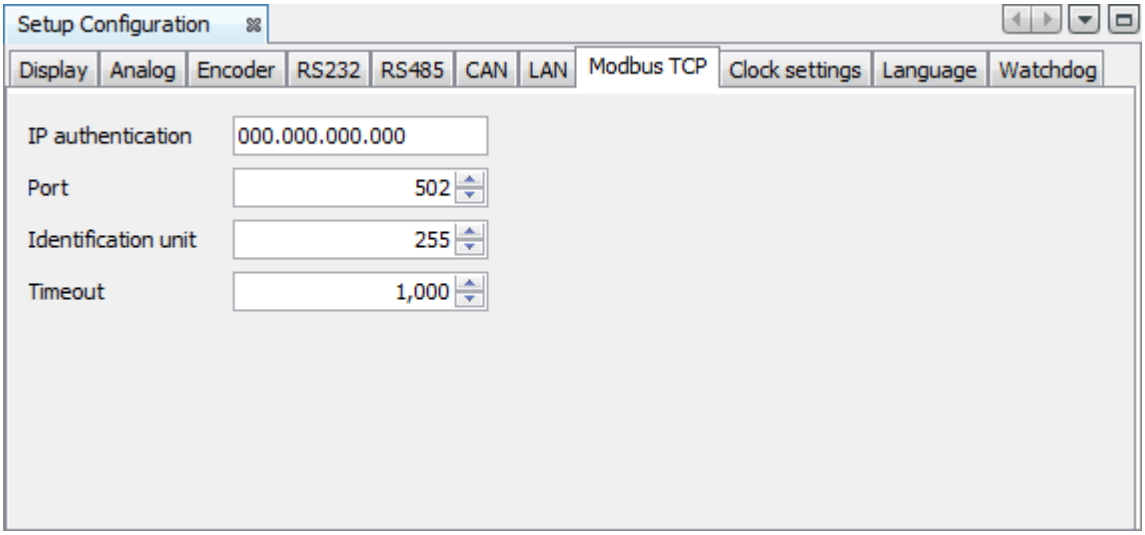
11.8.8.2.7 LAN



It is possible to configure the IP address, subnet mask, default gateway, DHCP, speed and duplex mode of the PLC300 in an Ethernet network.

- **IP Address**: Four bytes of address that identify the PLC300 in the IP network;
- **Subnet Mask**: Four bytes that identify the subnet to which the PLC300 belong in the IP network;
- **Gateway**: Four address bytes that identify the default gateway to access other subnets in the IP network;
- **DHCP**: Disabled, Enabled;
- **Speed/Duplex**: Auto, 10MBps Full Duplex, 10MBps Half Duplex, 100MBps Full Duplex, 100MBps Half Duplex.

11.8.8.2.8 Modbus TCP



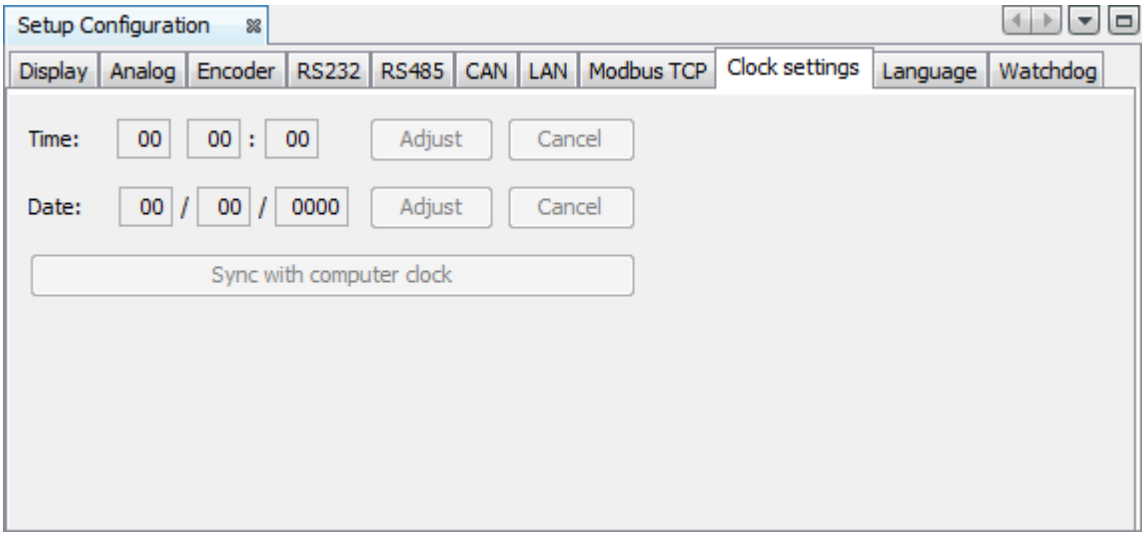
It is possible to configure the TCP, Unit ID, IP authentication and timeout of the Gateway Modbus TCP/RTU of the PLC300 in an Ethernet network by using the ModbusTCP protocol.

- **IP Authentication**: Four bytes of the address that identify the single remote IP address that can connect to

the PLC300. All the fields in zero disable the IP authentication and any remote address can connect to the PLC300;

- **TCP Port**: 0 to 65535,
- **Unit ID**: 1 to 255,
- **Gateway Timeout**: 20 to 5000 ms

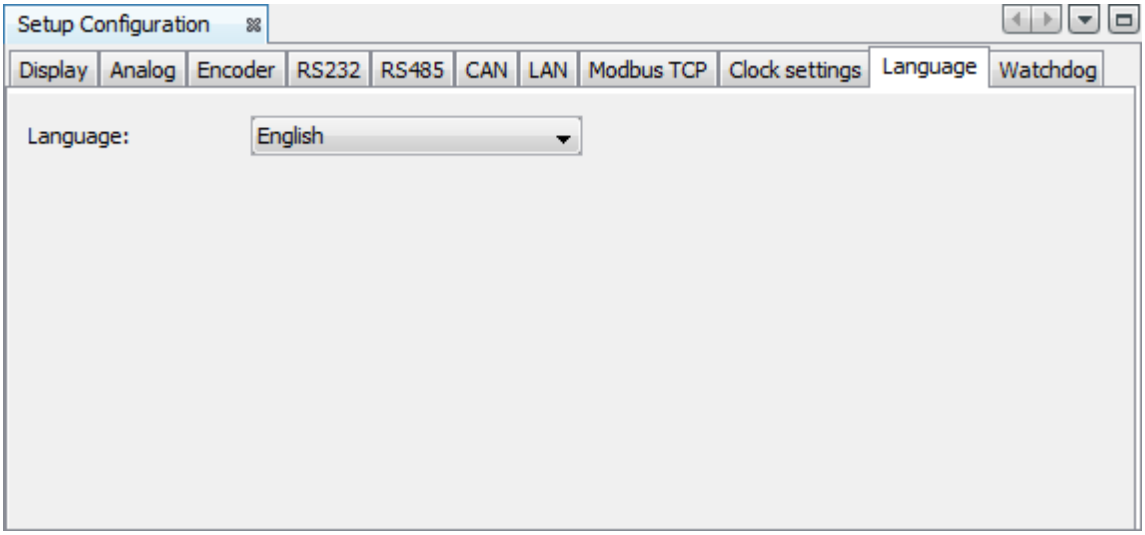
11.8.8.2.9 Clock Settings



The clock settings are only active when the WPS is connected to the PLC300 through the command **Connect Device**.

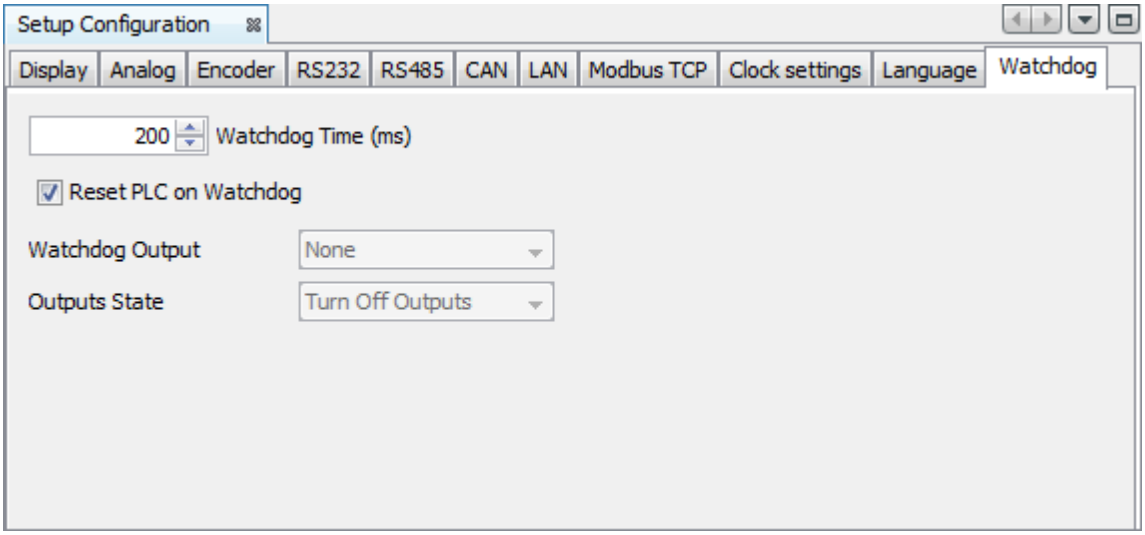
It allows to set the RTC clock of the PLC300.

11.8.8.2.10 Language



Allows to change the configured language in PLC300.

11.8.8.2.11 Watchdog



Allows configuring custom watchdog in the PLC300.

1. **Time of Watchdog (ms)**: waiting time of the watchdog. Minimum value: 200ms.
2. **Reset the PLC in case of Watchdog**: if checked, this option resets the PLC300 in case of watchdog and runs the user application again after restart. By checking this option, options 3 and 4 are disabled automatically;
3. **Watchdog Output**: allows selecting an output for the exclusive use as a flag from the watchdog.
   3.1. The selected output (DO8 or DO9) cannot be written via Ladder. It will remain at a high level after configured for this option and, in the event of watchdog, it will turn off.
4. **Output Status**: allows selecting if the digital outputs maintain their status in case of activation of the watchdog or if they will be turned off.
   4.1. This option will have no action on the selected output in option 3. For example: if in option 2 DO9 is selected, even if in option 3 'Keeping the status' is checked, at the moment of watchdog it will be switched off.

Compatibility

Device	Version
PLC300	2.30 or higher

11.8.9 Communication

11.8.9.1 Online Commands

Overview

The online commands are commands performed when the device is communicating with the application (online monitoring active).

With the online commands it is possible:

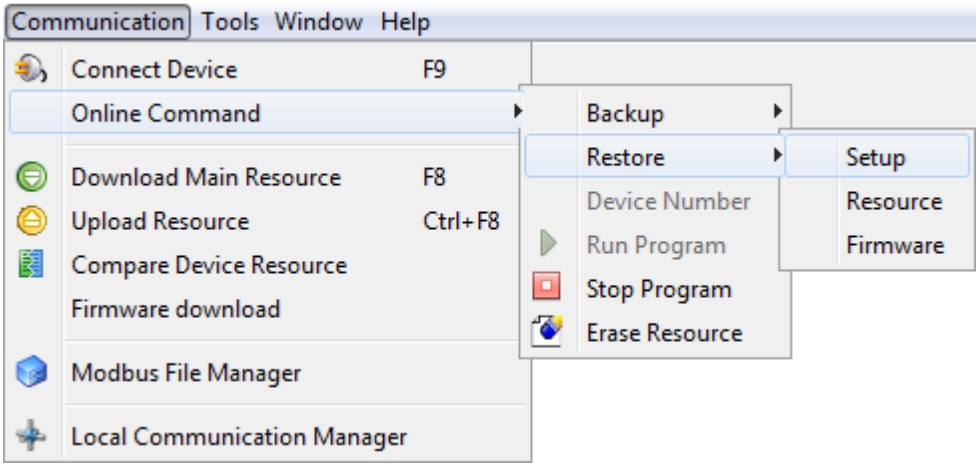
- To perform the commands for recording and loading the resource, setup and firmware files.
- To configure the device number.
- To stop and run the program and delete the resource.

Saving and loading configuration

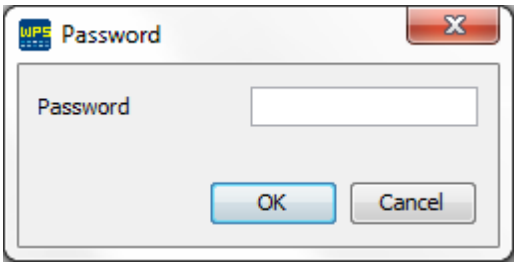
The commands for recording the resource, setup and firmware files make a copy of the device binary files to the SD card. The loading of the files make the copy of the binary files from the SD card to the device.
For the security of the files, it is possible to configure a protection password. It is also possible to configure the number of devices so that backup files will be copied to different folders.

Saving and loading command

To perform the commands for recording and loading the resource, setup and firmware, it is necessary to select the menu **Communication** > **Online commands** > choose one of the options listed **Save**/**Load** and then **Resource**/**Setup**/**Firmware**.

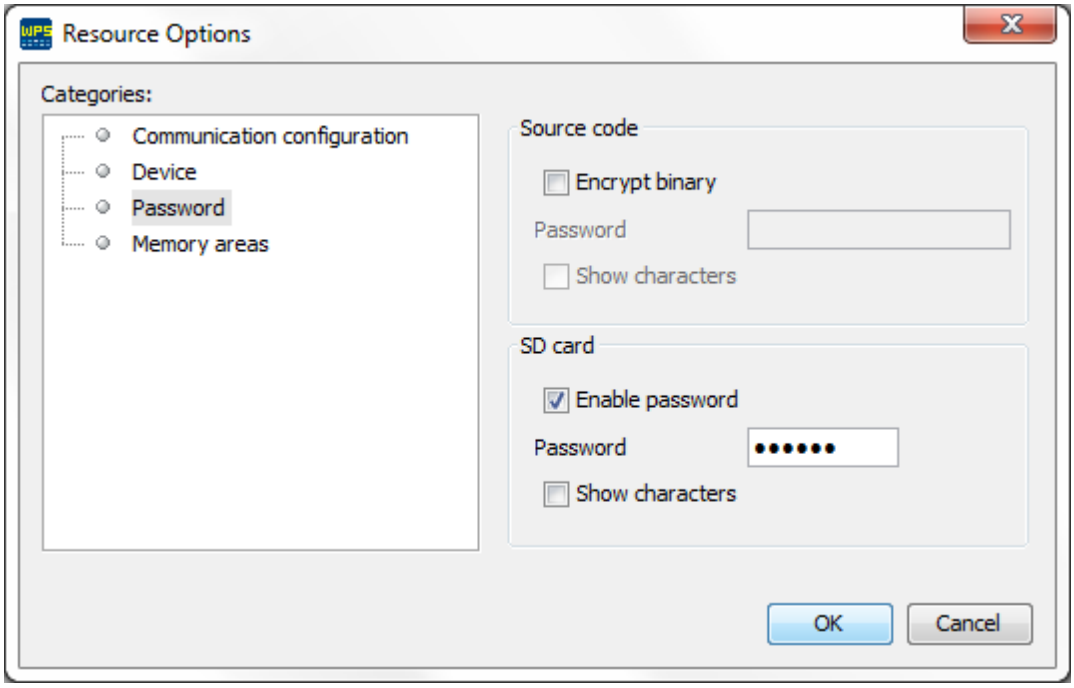


If a password has been configured for the resource, a dialog window prompting the password will show. The password has one to eight decimal digits.



Password configuration

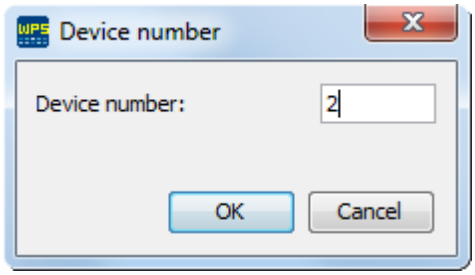
The password configuration for the online commands is done trough the configuration of the resource properties, in the password option.



After entering the password (with one to eight decimal digits), it is necessary to build the resource and send it to the device. The password will be stored on the equipment and, when one of the online commands is performed, the password is requested. The password will not be requested again until that the online monitoring is completed.

Device number

The device number is configured to record the backup of different devices on the same SD card. When selecting the functions for recording resource, setup or firmware, the files will be recorded on the following folders, according to the device number (1 in the example).



\PLC300\0001\Resource
\PLC300\0001\Setup
\PLC300\0001\Firmware

ATTENTION!
Existing files in the destination folder will be overwritten when you perform recording functions.

Run program 

It runs the user's program.
In order to perform the run program command, it is necessary to select the menu **Communication** > **Online commands** > **Run program**.

Stop program 

Stop the user's program.
In order to perform the stop program command, it is necessary to select the menu **Communication** > **Online commands** > **Stop program**.

Erase resource 

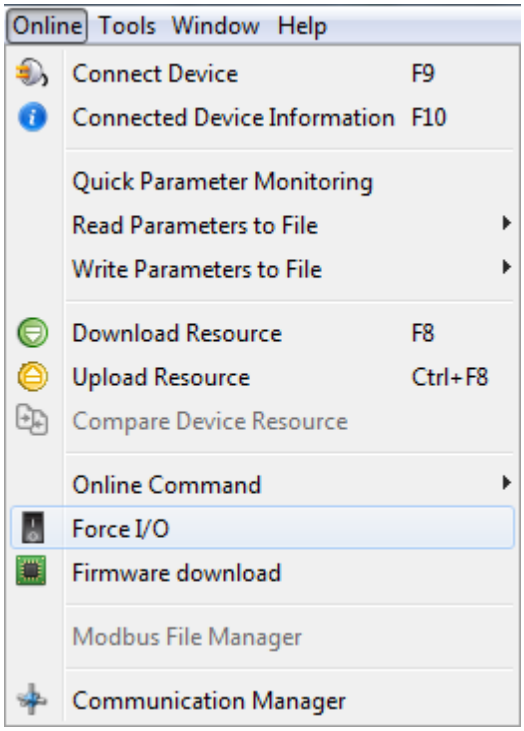
It erases the resource recorded on the device.
In order to perform the delete program command, it is necessary to select the menu **Communication** > **Online commands** > **Delete program**.

11.8.9.2 Force I/O

Overview

The force inputs and outputs window is used for the values of the digital and analog inputs to be read by the program, by values manipulated by the user, regardless their physical state. It also allows the manipulation of the physical states of the digital and analog outputs by the user independently of the values calculated by the program.
In order to force the device inputs and outputs, it is necessary that the online monitoring be active and the option **Run cyclically** be enabled. The data are sent to the device every 2 seconds.
The values can be edited with the device disconnected. The configurations are stored in the resources and recorded whenever the main resource selection is changed.
The data displayed on the force I/O window contain the values belonging to the resource (and configuration) selected as main.

The force I/O window is open trough the menu **Online** > **Force I/O**:



Toolbar

The toolbar of the force window has the options to run cyclically, upload the device force configuration, enable all and disable all:

Run cyclically: Sends the user's configurations to the device and updates the state of the inputs and outputs in a cyclic way.

Upload configuration: Allows the current configuration of the device to be read. For this option to be enabled, it is necessary that the online monitoring be active and the option run cyclically be disabled.

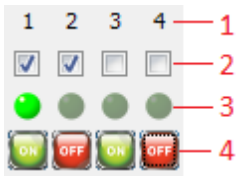
Enable all: Enables the force I/O of all of the inputs and outputs of the device.

Disable all: Disables the force I/O of all of the inputs and outputs of the device.

Input and Output commands

For each digital and analog input and output there is a selection box linked to enable the force, a status field and an edition field.

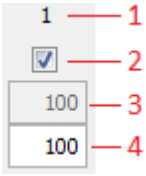
Digital:



1. Number of the digital inputs/output
2. Enable/disable Force I/O

3. Current status of the I/O: It has three statuses: 1. light green LED: activated; 2. dark green LED: deactivated; 3. gray LED: the value is not being read.
4. Enable/disable the input/output

Analog:



1. Number of the analog input/output
2. Enable/disable Force I/O
3. Current value of the input/output
4. Value of the input/output configured by the user

NOTE!
The analog signal scale has 15 bits plus 1 bit for signal, except for SSW900 which it has only 10 unsigned bits.

11.8.9.3 Download

Overview

The resource download downloads the memory area configuration files, volatile data, retentive data, programs, screens, alarms, source code (optionally) and recipes (optionally) to the device internal memory.

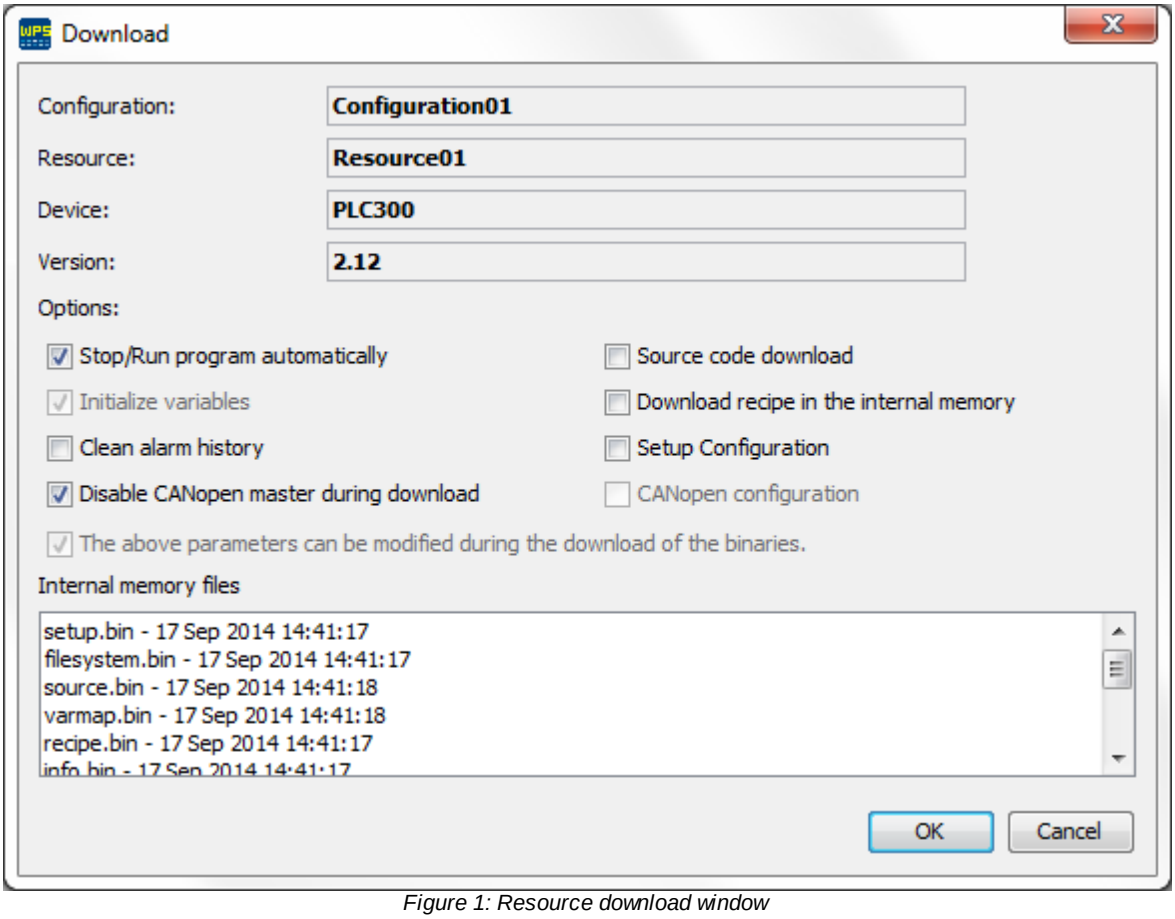


Figure 1: Resource download window

It is also possible to perform the following operations during the download of the resource:

- **Initialize variables**: Initializes the variables with the values configured in the initial values.
- **Clean alarm history**: Excludes alarm history data stored. It is recommended to clean this area whenever the memory areas or the alarm configuration are changed.
- **Source code download**: Downloads the resource source code.
- **Download recipe in internal memory**: Downloads the file containing the recipe data configured with the storage option in the RAM.
- **Disable CANopen master during download**: Disables the CANopen master during the download:
- **Stop/Run the program automatically**: Disables the warning windows informing that the program will be stopped (Figure 2) and the prompting window asking if you wish to execute the program (Figure 3).
- **Setup configuration**: Performs setup configuration data download.
- **CANopen configuration**: Performs CANopen network configuration download.

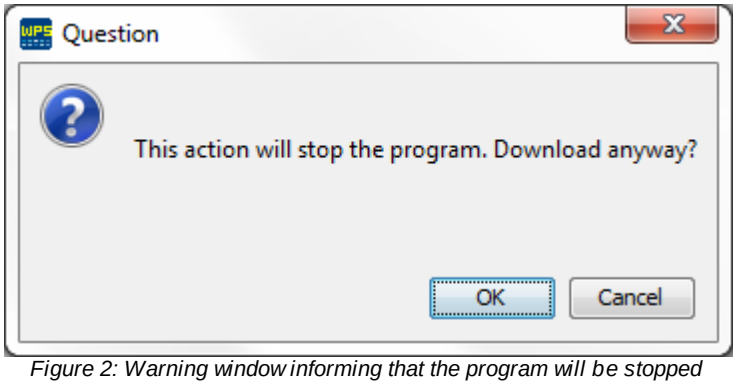


Figure 2: Warning window informing that the program will be stopped

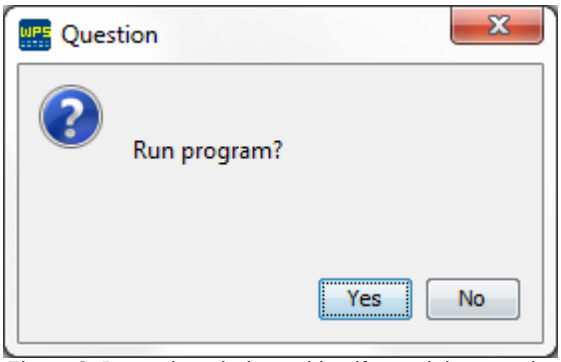


Figure 3: Prompting window asking if you wish to run the program

Download of the SD card files

If the user configured one or more recipes with the option of storage in the SD card, the confirmation window of download of files in the SD card will show (Figure 4). For the window to show, it is necessary that the SD card be connected to the equipment.

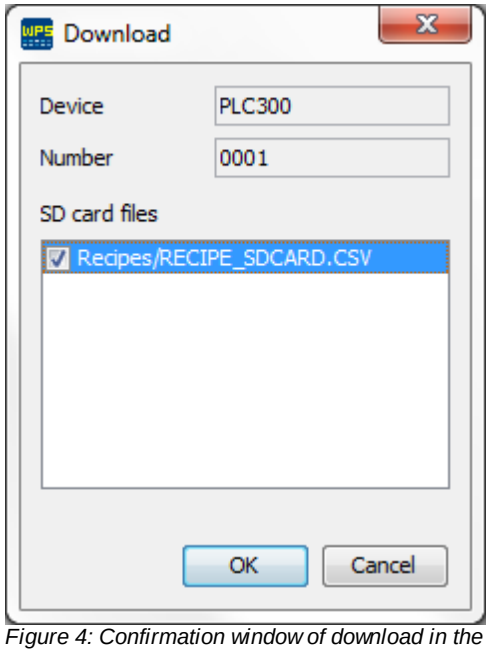


Figure 4: Confirmation window of download in the SD card

11.8.9.4 Hot Download

11.8.9.4.1 Overview

The Hot Download is a functionality that allows the loading of logic changes and configurations of a resource in a equipment without having to stop the routine of execution. The variables actual values are retained.

On the Hot Download, the logic changes and configurations must be realized off-line, in other words, with the monitoring disabled, being transfer to the equipment in one single step. This requires that every and any modification realized needs to be sent to the equipment through a command realized by the user itself, avoiding then that incomplete changes can cause a different operation than expected.

ATTENTION!
Take care during Hot Download realization: mistakes can result in damage to the people involved and to the equipment. Before realizing Hot Download, evaluate the consequences of your changes and notify the involved.

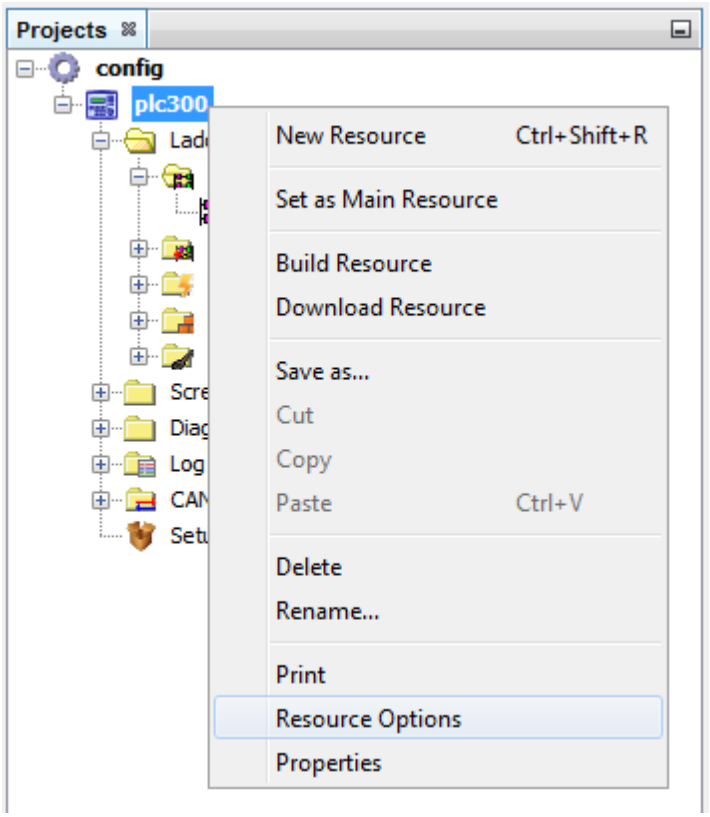
Compatibility

Equipment	Version
PLC300	2.30 or superior

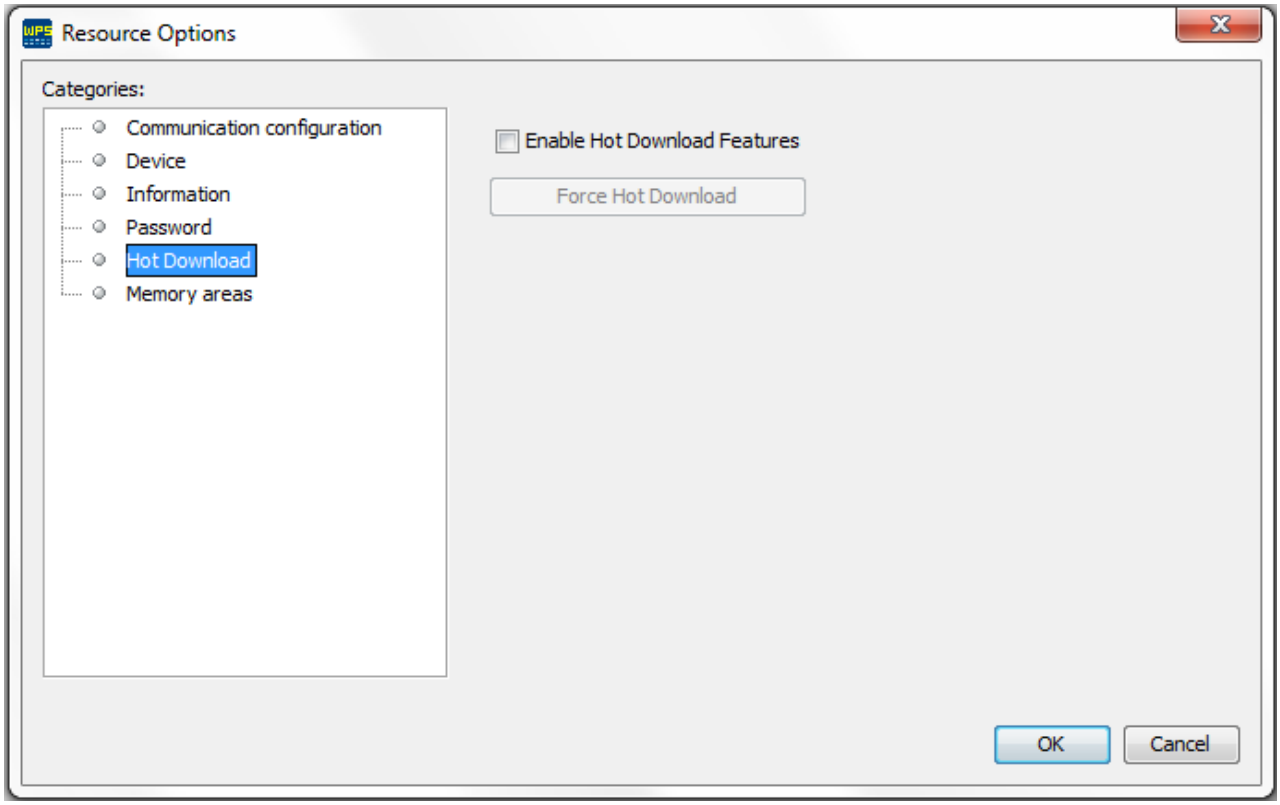
11.8.9.4.2 Enable/Disable Hot Download

The Hot Download feature comes disabled on the resource by default, to enable it just follow the steps below:

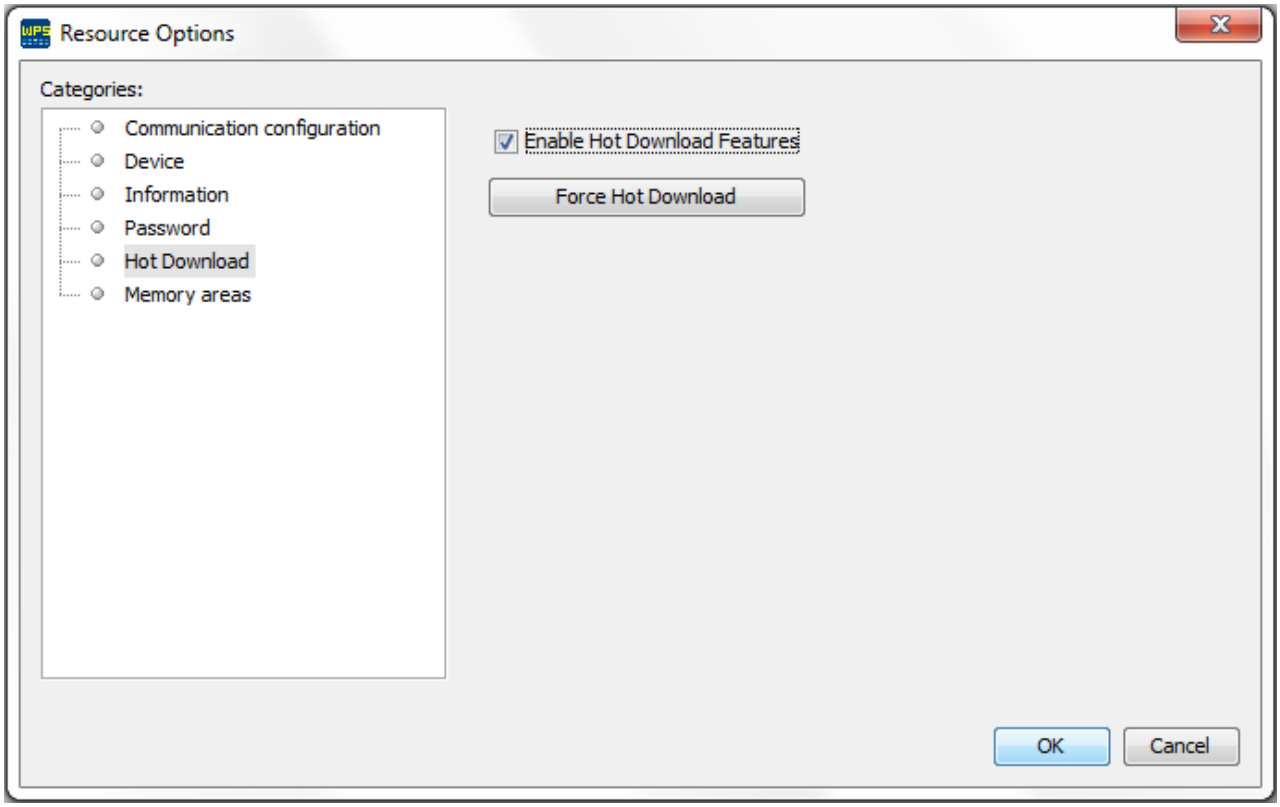
1. Open the context menu of the resource and select the option **Resource Options**.



2. Select the option **Hot Download** on the sidebar menu, it will open a window with the Hot Download configurations.



3. On the Hot Download configuration window just select the option **Enable Hot Download Features** and confirm.



4. The option **Force Hot Download** is used when it's wished that the system re-enables the Hot Download to the resource, after the user having realized one operation that disables it.

Example: The user create a local variable after the system having informed him that the creation of the same will disable Hot Download, in case the user regrets and excludes this variable he can then use the option **Force Hot Download** so the system will keep allowing Hot Download in the same way it was allowing before the creation of this variable.

11.8.9.4.3 Restrictions

For being possible to realize the Hot Download on the equipment, it's necessary that after the full resource download, some restrictions don't be violated.

Restrictions are limitations imposed to the operations that the user can execute in a resource, because such operations unfeasible the Hot Download routine.

The list below provides the restrictions to the user:

- Memory areas
  - Any memory area can't be changed;
- Programs
  - Can't be removed;
  - Can't be added to the execution list of tasks;
- Tasks
  - Can't be edited;

  - Can't be removed;
  - Can't have their execution list changed. Also the execution order of the programs can't be changed.
- Variables
  - Variables with LOCAL scope can't be created, excluded or edited.
  - Variables with GLOBAL scope without address and from the group GLOBAL_RETAIN without address can't be created, excluded or edited.
  - Variables with GLOBAL scope with address and CONSTANT can be created, excluded or edited.
  - Variables from USERFB from types VAR_IN, VAR_OUT, VAR_IN_OUT, LOCAL, LOCAL_RETAIN can't be created, excluded or edited.
- USERFB's
  - Can't be added and removed from programs.
- Structures
  - Can't be edited or removed.
- Recipes
  - Can't be transferred to the equipment.
- Screens
  - No restrictions.
  - During the loading of the changes to the equipment will be displayed a screen indicating this operation. In cases where the current screen changes, will be called the HOME screen.
- Alarm
  - No restrictions.
  - Case any alarm stored in the alarm history be removed, the alarm history become than invalid.
- Event Log
  - No restrictions.

Case any restrictions be violated, a alert message will be displayed.

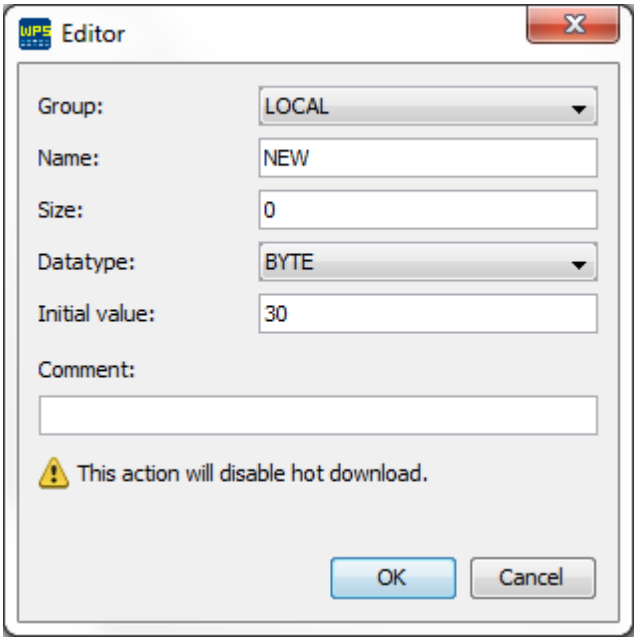


Figure 1 - Alert on the edition screen

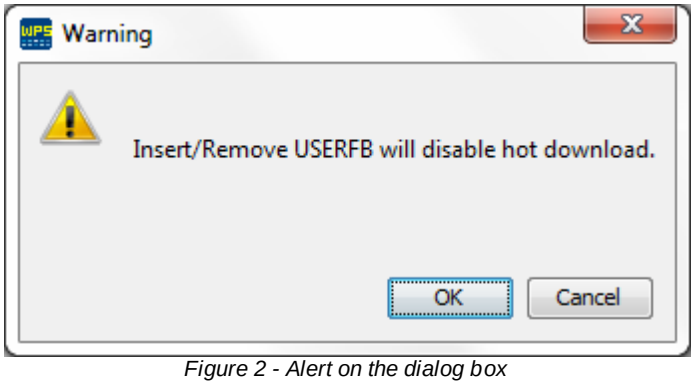


Figure 2 - Alert on the dialog box

NOTE!
The alerts displayed during the edition aren't guaranty that the Hot Download will be disabled when the user call it. Those alerts serve only as a parameter in order that the user have the consciousness that the change he is realizing will disable the Hot Download in case that its don´t be reverted.

11.8.9.4.4 Operation

The Hot Donwload function is enabled when:
1. A complete download of the compiled resource has been done previously;
2. Have sufficient resource memory space.

After compiling, the output window indicates a warning (W4002) if the Hot Download function is disabled due to lack of memory.

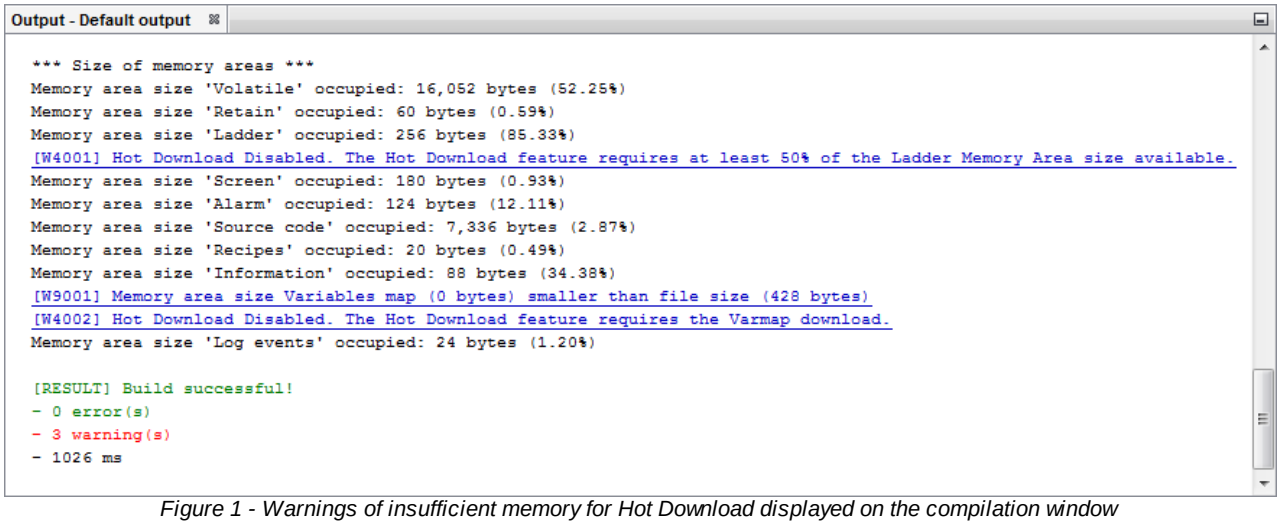


Figure 1 - Warnings of insufficient memory for Hot Download displayed on the compilation window

After realizing the desired changes on the resource, the beginning of the download routine will verify if the resource edited is the same as the resource executing on the equipment.
Case both be equal, is then displayed a dialog window for selection between Full Download or Hot Download.

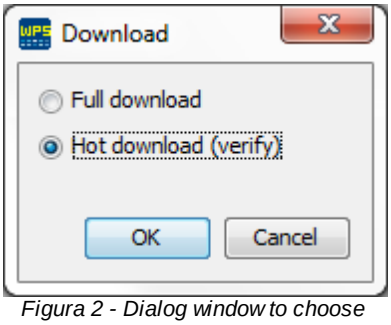


Figura 2 - Dialog window to choose between Full Download and Hot Download

Case the option Hot Download be selected, then is realized a verification to know if any restriction has been violated. Case it was, a message of error will be displayed informing the cause.
Validated the verification, is then realized a comparison of the memory areas, the files that were changed are displayed in red.

NOTE!
The variables map is presented on the gray color, since its download is obligatory in case of changes.

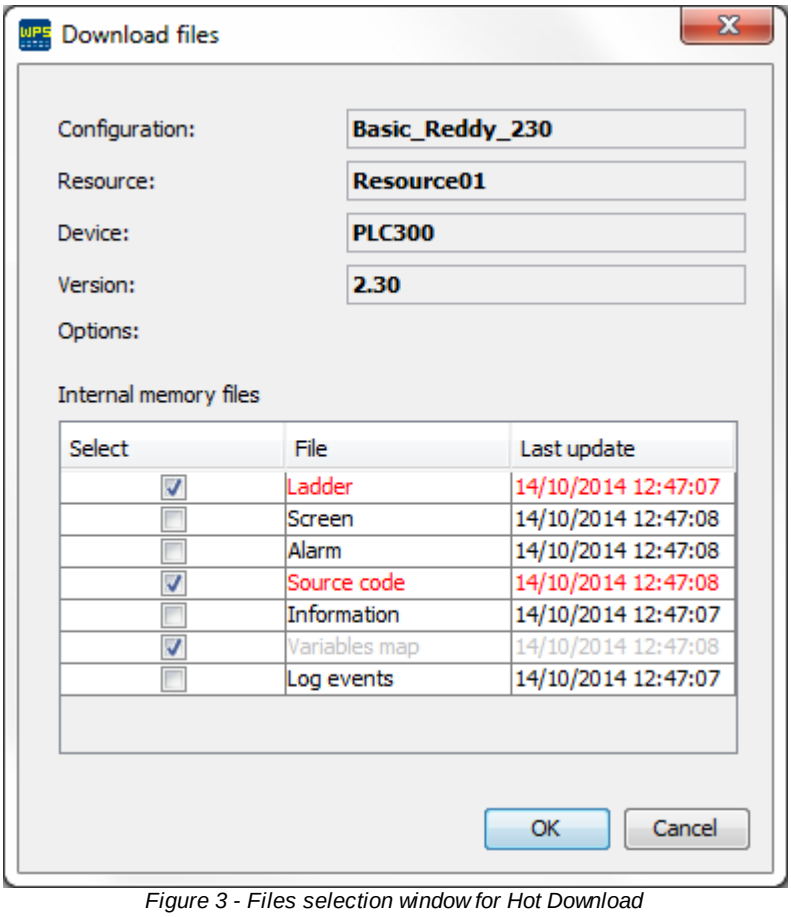


Figure 3 - Files selection window for Hot Download

Case any file be sent to the equipment without the source code, this will become invalid, thus disabling your

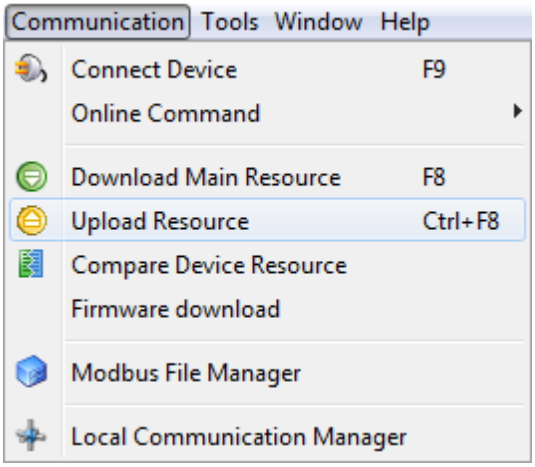
upload.
This procedure is realized because the source code don't represent anymore the files presented in the memory areas.

NOTE!
The source code becomes invalid even that it be equal to the previously saved on the equipment.

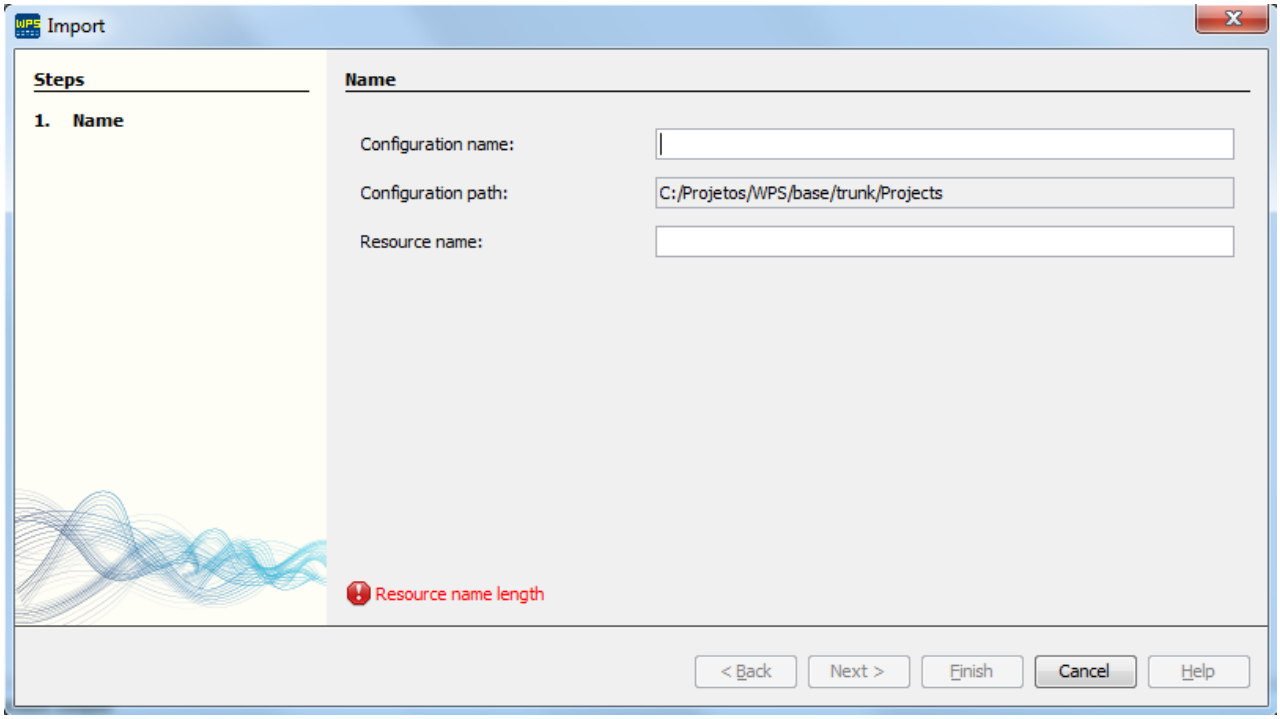
11.8.9.5 Upload

Overview

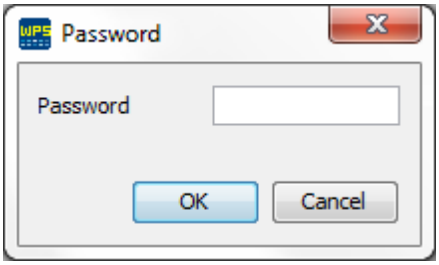
Uploads the resource source code. For this, it is necessary that the source code be previously recorded (during the download) in device memory. In order to initialize the upload, it is necessary to access the menu **Communication** > **Upload resource**.



If the upload was performed successfully, the window to select the configuration name will open.



If a password to protect the source code was configured, it will be requested:

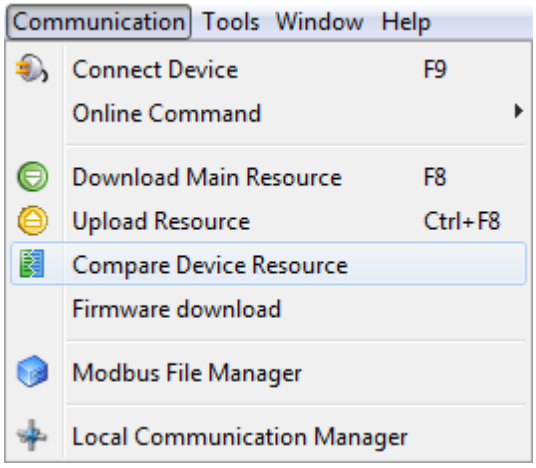


11.8.9.6 Comparison of resource and device

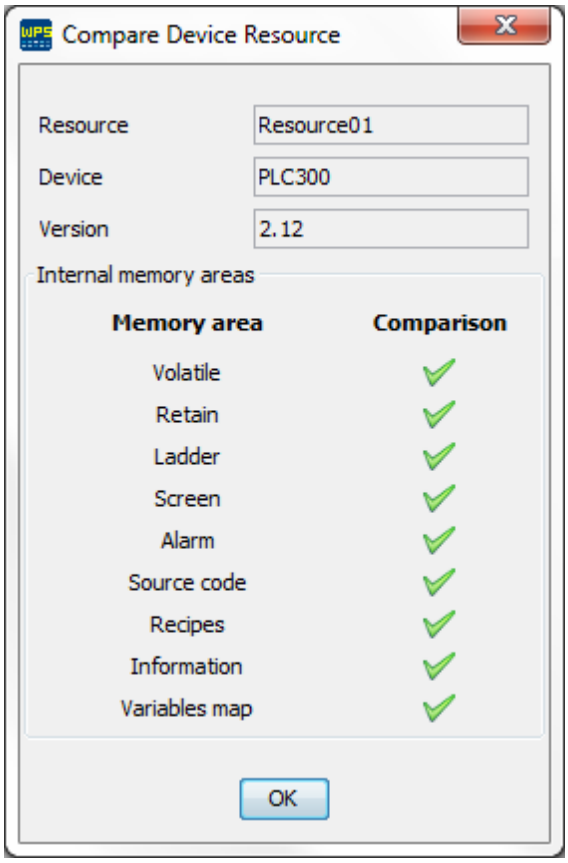
Overview

> **NOTE!**
> The comparison of resource and device works only as from firmware version 1.50.

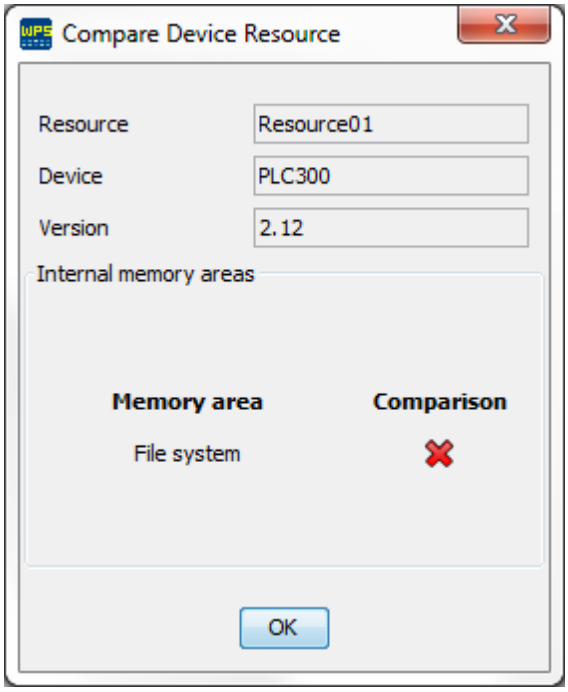
The comparison of resource and device (binary files) allows the user compare the main resource of the application with the resource that is running on the device. To begin the comparison, it is necessary to access the menu **Communication** > **Compare resource device.**



After selected the comparison option, it is checked whether the resource is compiled to read the information of size and CRC of the binary files. Then, the comparison of all memory areas is performed.



If the memory areas have changed, cannot be possible to perform the comparison and a file system comparison error is shown:

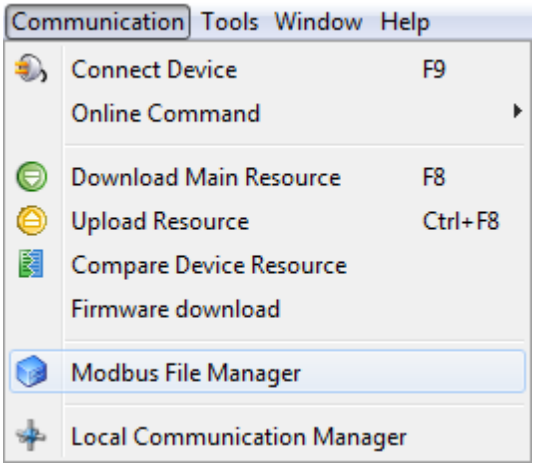


Observations:
- The information and source code files are updated every build.
- The volatile, retain, source code and recipe are download options. If the download of these files were not done, the comparison will be flagged as false.

11.8.9.7 Modbus File Manager

Overview

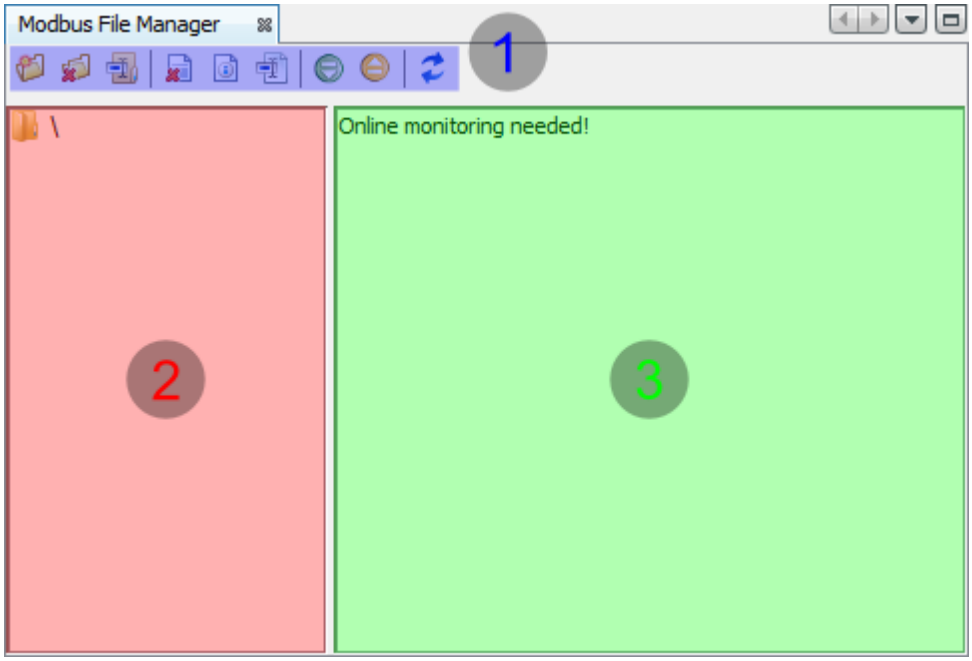
The modbus file manager of the PLC300 is accessed through the communication menu as shown in the following figure.



The window of the modbus file manager is composed of the following parts as shown in the following figure.

1. Toolbar with all the tools of the file manager.

2. Folder list of the SD Card.
3. File list of the selected folder.



It is only possible to view the files and folders of the SD Card when the WPS is connected to the PLC300 through the command Connect Device .

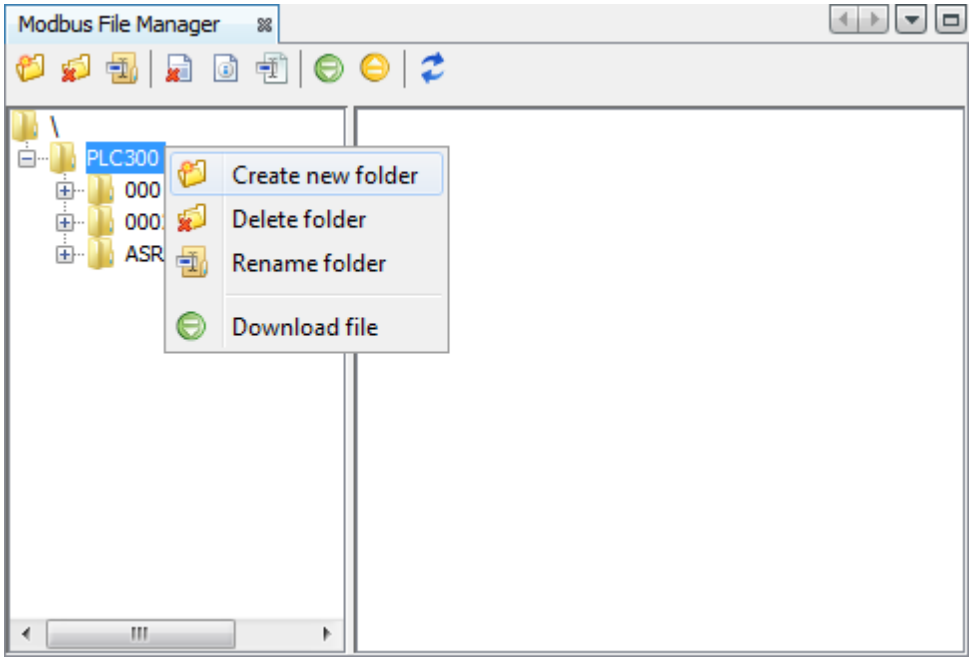
Toolbar

Creates a new folder
Erases the selected folder
Renames the selected folder
Erases the selected file
Displays the properties of the selected file (path, size and date)
Renames the selected file
Sends the file to the PLC300
Receives the selected file from the PLC300

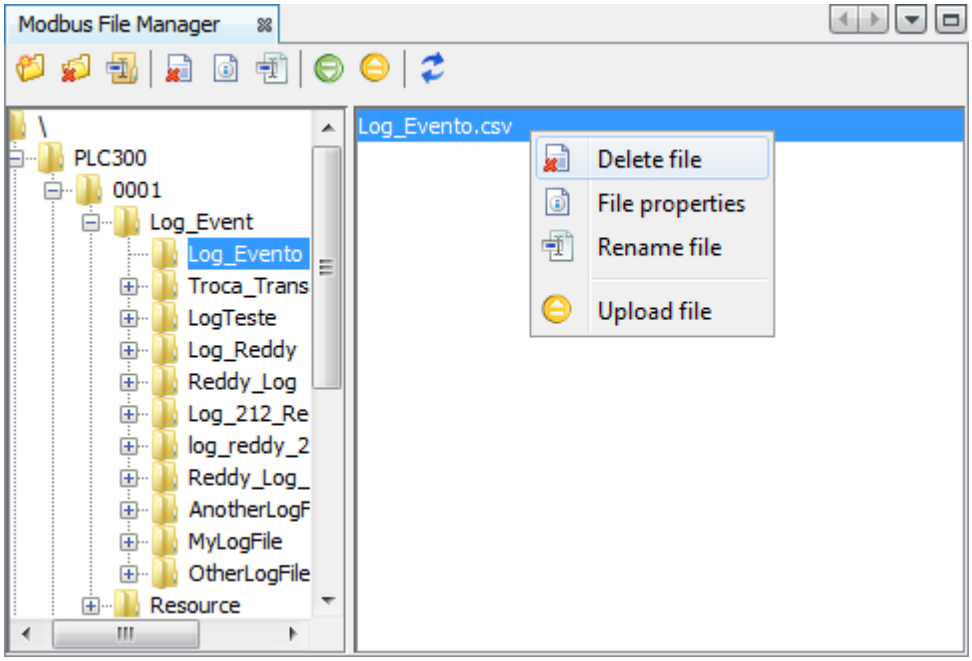
Popup menu

In order to access the popup menu, just click the right button on a folder or on a file and it will be displayed as shown in the following figure.

Popup menu for folders:



Popup menu for files:



11.8.9.8 Communication RS232

Overview

In a serial interface, the data bits are sent in sequence through a communication channel or link. Several technologies use the serial communication for data transfer, including interfaces RS232.

Configuration

Some parameters need to be configured to perform the RS232 communication; to do so, check the item *RS232*.

There are three operating modes of the RS232 communication.
- **Mode 0**: Modbus Slave;
- **Mode 2**: ASCII Protocol;
- **Mode 4**: Generic telegrams.

> **NOTE!**
> Modes 1 and 3 are reserved.

Mode 0: Modbus Slave

Refer to manual **PLC300 - Modbus RTU Communication**, available at http://www.weg.net/br.

Mode 2: ASCII Protocol

The ASCII protocol, via RS232, was developed for bar code reading.

Involved Variables:

- RS232_MODE: BYTE system marker which defines the operating mode of the RS232:
  o 2: ASCII protocol;
- RS232_RX_CLEAR: BIT marker; clears the data buffer (RS232_ ASCII_ BYTEBUFFER) and the RS232_RX_FINISHED flag;
- RS232_RX_FINISHED: BIT marker; indicates that a data package is available in the RS232_ASCII_BYTEBUFFER buffer;
- RS232_ASCII_BYTEBUFFER: 256-byte buffer which stores the received characters;
- RS232_ASCII_STRING: STRING buffer, which can be shown on the screen of the PLC300 by the 'Text Output' component.

How to use the RS232 in the ASCII mode:

1. Select the ASCII protocol by setting: RS232_MODE = 2 in the Ladder;
2. Give a pulse in the RS232_RX_CLEAR marker to clear the RS232_ASCII_ BYTEBUFFER buffer, and the RS232_RX_FINISHED flag;
3. When receiving a package, the flag: RS232_RX_FINISHED goes to TRUE;
4. The data are available for the Ladder by means of the array: RS232_ASCII_BYTEBUFFER, of 256 bytes;
5. To use with the 'Text Output' function of the HMI, use the RS232_ASCII_STRING STRING marker.

> **NOTE!**
> Even without turning on RS232_RX_CLEAR, the system can receive another reading, placing the data over the previous reading.

NOTE!
The end of the package is executed when the CR/LF characters (0x0d/0x0a) are received.

NOTE!
The CR/LF characters are not stored in the buffer.

NOTE!
The buffer is terminated with NULL character (0x00).

The configuration of the RS232 must be done according to the default configuration of the PLC300;
On the screens, the 'Text Output' component, which only accepts the STRING type, was added to show RS232_ASCII_STRING, which is the string that shows the value read by the protocol, limited to 20 characters (limit of the output function = 1 line).

Mode 4: Generic telegrams

Functionality developed to send and receive telegrams by means of serial communication RS232.

Involved command variables:

- RS232_MODE: BYTE marker; defines the operating mode of the RS232:
  - 4: telegrams via RS232.
- RS232_TIMEOUT: WORD marker; indicates the maximum waiting time (in ms) for a response, preventing new telegrams to be sent before this time has elapsed.
- RS232_END_CHARACTER: defines a character that can end a telegram, for example, ETX (03H). Upon the receipt of this character, the PLC300 considers that the bytes received up to it are the ones necessary; the others are ignored.
- RS232_ENABLE_END_CHARACTER: BIT marker; system marker that enables the use of telegram end character specified in RS232_END_CHARACTER;
  - 0: disabled;
  - 1: enabled.
- RS232_START_TX: BIT marker; the telegram is sent on the leading edge of this marker;
- RS232_TX_ADDRESS: WORD marker; indicates the initial address of the data to be transferred;
- RS232_TX_LENGTH: BYTE marker; indicates the number of bytes of the information to be sent;
- RS232_RX_ADDRESS: WORD marker; indicates the initial address of the received data;
- RS232_MAX_RX_BUFFER_LENGTH: BYTE marker; indicates the maximum number of bytes that can be received;

Involved status variables:

- RS232_TX_TELEGRAM_COUNTER: WORD marker; counts the number of sent telegrams;
- RS232_RX_TELEGRAM_FINISHED: BIT marker; indicates that a data package is available in the memory from the address indicated in RS232_RX_ADDRESS.
- RS232_TX_FINISHED: BIT marker; indicates that a data package was completely sent.
- RS232_TIMEOUT_INDICATOR: BIT marker; indicates if the time specified in RS232_TIMEOUT has elapsed.
  - 0: there was no overflow;
  - 1: there was time overflow.
- RS232_RX_TELEGRAM_COUNTER (status): WORD marker; counts the number of received telegrams;
- RS232_RX_BYTE_COUNTER (status): WORD marker; counts the number of received bytes;

How to use the RS232 to send and receive telegrams:

1. Select **Send and receive telegrams via RS232** by setting RS232_MODE = 4 in the Ladder;
2. Configure the initial address of the telegrams to be sent and received in, respectively, RS232_TX_ADDRESS and RS232_RX_ADDRESS;
3. Specify the size (in bytes) of the telegram to be sent by means of marker RS232_TX_LENGTH;
4. Specify the maximum size (in bytes) of the telegram that can be received by means of RS232_RX_LENGTH marker;
5. Specify the timeout in RS232_TIMEOUT;
6. If applicable, configure the telegram end characters of the markers respectively in: RS232_ENABLE END_CHARACTER and RS232_END_CHARACTER;
7. Reset the RS232_RX_TELEGRAM_FINISHED flag to be ready to receive a telegram;
8. Give a pulse in RS232_START_TX to send a telegram.

NOTE!
It will be considered the end of receipt of telegram, the receipt of the special character when configured in RS232_END_CHARACTER with the RS232_ENABLE_END_CHARACTER flag enabled or, also, when the timeout is equal to twice the transmission time of a byte.

Compatibility

Device	Version
PLC300	Mode 0: 1.00 or higher
	Mode 2: 1.11 or higher
	Mode 4: 1.50 or higher

11.8.9.9 Communication RS485

Overview

In a serial interface, the data bits are sent in sequence through a communication channel or link. Several technologies use the serial communication for data transfer, including interfaces RS485.

Configuration

Some parameters need to be configured to perform the RS485 communication; to do so, check the item *RS485*.

There are three operating modes of the RS485 communication.
- **Mode 0**: Modbus Slave;
- **Mode 1**: Modbus Master;
- **Mode 4**: Generic telegrams.

NOTE!
Modes 2 and 3 are reserved.

Mode 0: Modbus Slave

Refer to manual **PLC300 - Modbus RTU Communication**, available at http://www.weg.net/br.

Mode 1: Modbus Master

Refer to manual **PLC300 - Modbus RTU Communication**, available at http://www.weg.net/br.

Mode 4: Generic telegrams

Functionality developed to send and receive telegrams by means of serial communication RS485.

Involved command variables:

- RS485_MODE: BYTE marker; defines the operating mode of the RS485:
  - 4: telegrams via RS485.
- RS485_TIMEOUT: WORD marker; indicates the maximum waiting time (in ms) for a response, preventing new telegrams to be sent before this time has elapsed.
- RS485_END_CHARACTER: defines a character that can end a telegram, for example, ETX (03H). Upon the receipt of this character, the PLC300 considers that the bytes received up to it are the ones necessary; the others are ignored.
- RS485_ENABLE_END_CHARACTER: BIT marker; system marker that enables the use of telegram end character specified in RS485_END_CHARACTER;
  - 0: disabled;
  - 1: enabled.
- RS485_START_TX: BIT marker; the telegram is sent on the leading edge of this marker;
- RS485_TX_ADDRESS: WORD marker; indicates the initial address of the data to be transferred;
- RS485_TX_LENGTH: BYTE marker; indicates the number of bytes of the information to be sent;
- RS485_RX_ADDRESS: WORD marker; indicates the initial address of the received data;
- RS485_MAX_RX_BUFFER_LENGTH: BYTE marker; indicates the maximum number of bytes that can be received;

Involved status variables:

- RS485_TX_TELEGRAM_COUNTER: WORD marker; counts the number of sent telegrams;
- RS485_RX_TELEGRAM_FINISHED: BIT marker; indicates that a data package is available in the memory from the address indicated in RS485_RX_ADDRESS.
- RS485_TX_FINISHED: BIT marker; indicates that a data package was completely sent.
- RS485_TIMEOUT_INDICATOR: BIT marker; indicates if the time specified in RS485_TIMEOUT has elapsed.
  - 0: there was no overflow;
  - 1: there was time overflow.
- RS485_RX_TELEGRAM_COUNTER (status): WORD marker; counts the number of received telegrams;
- RS485_RX_BYTE_COUNTER (status): WORD marker; counts the number of received bytes;

How to use the RS485 to send and receive telegrams:

1. Select **Send and receive telegrams via RS485** by setting RS485_MODE = 4 in the Ladder;
2. Configure the initial address of the telegrams to be sent and received in, respectively, RS485_TX_ADDRESS and RS485_RX_ADDRESS;
3. Specify the size (in bytes) of the telegram to be sent by means of marker RS485_TX_LENGTH;
4. Specify the maximum size (in bytes) of the telegram that can be received by means of RS485_RX_LENGTH

marker;
5. Specify the timeout in RS485_TIMEOUT;
6. If applicable, configure the telegram end characters of the markers respectively in: RS485_ENABLE END_CHARACTER and RS485_END_CHARACTER;
7. Reset the RS485_RX_TELEGRAM_FINISHED flag to be ready to receive a telegram;
8. Give a pulse in RS485_START_TX to send a telegram.

NOTE!
It will be considered the end of receipt of telegram, the receipt of the special character when configured in RS485_END_CHARACTER with the RS485_ENABLE_END_CHARACTER flag enabled or, also, when the timeout is equal to twice the transmission time of a byte.

Compatibility

Device	Version
PLC300	Mode 0: 1.00 or higher
	Mode 1: 1.00 or higher
	Mode 4: 1.50 or higher

11.9 PSRW

11.9.1 Description

PSRW is a configurable safety relay which can be configured using the WPS graphic interface, it has 4 (four) double channel safety inputs and 2 (two) OSSDs (Safety double channel safety outputs).

PSRW is capable of monitoring the following safety sensors and components:
- Safety Light curtain;
- Two hands control;
- Emergency stops;
- Magnetic sensors;
- Mechanical switches;
- Safety sensors.

Refer to the user's manual of the PSRW for further details about the product.

11.1(SCA06

11.10.1Description

The SCA06 servo drive is a high-performance product which allows controlling the speed, torque and position of three-phase, alternate-current servomotors. The main characteristic of this product is the high performance and high precision of movement control of the servomotor due to the operation in closed loop by means of the position feedback given by a sensor inside the servomotor.

The SCA06 features independent control supply and power supply, allowing, for instance, that the product communication networks keep on working normally even if the power circuit must be turned off for some

reason.

The use of braking resistors provides greatly reduced braking times, optimizing the processes that require high performance. Several special functions are available, such as programming in Ladder with positioning blocks which provides extreme flexibility and integration to the drive.

The SCA06 can be used in different applications with many options of cables, both for simple applications and complex applications like handling, environments with oil, etc.

NOTE!

SCA06 versions below V2.00 do not have the Ladder tool available in WPS.
You can use the WLP application if this feature is required.

11.10.2System Markers

The following variables contained in the **GLOBAL_SYSTEM** group of the variables table, have the fixed tag. The tag of system markers were divided into groups and subgroups, where:

Groups:
- SCA: reading and writing variables of the SCA06 servo drive;
- CO: reading and writing variables of the CANopen network.

Subgroups:
- STS: reading variable (status);
- CMD: writing variable (command).

Reading System Markers (Status)

Address	Bit	Modbus	Tag	Description
Ladder				
%SB6000	0	0	FREQ_2HZ	Oscillator with frequency of 2 Hz
%SB6000	1	1	PULSE_1SCAN	Pulse during the first scan cycle
%SB6000	2	2	FALSE	Always in 0
%SB6000	3	3	TRUE	Always in 1
%SW6002	--	3001	ELAPSED_SCAN_CYCLES	Elapsed scan cycles
Real Axis				
%SW6004	--	3002	SCA_STS_REAL_AXIS_STATUS	Real axis status (see note)
%SD6008	--	3004	SCA_STS_REAL_AXIS_VELOCITY	Real axis velocity
%SL6024	--	3012	SCA_STS_REAL_AXIS_POSITION	Real axis position
Virtual Axis				

%SW6006	--	3003	SCA_STS_VIRTUAL_AXIS_STATUS	Virtual axis status (see note)
%SD6012	--	3006	SCA_STS_VIRTUAL_AXIS_VELOCITY	Virtual axis velocity
%SL6032	--	3016	SCA_STS_VIRTUAL_AXIS_POSITION	Virtual axis position
Current				
%SD6016	--	3008	SCA_STS_MOTOR_CURRENT	Motor current
Position in the DI´s transition				
%SD6040	--	3020	SCA_STS_DI1_POSITION_STORED	Position stored in the DI1 transition
%SD6048	--	3024	SCA_STS_DI2_POSITION_STORED	Position stored in the DI2 transition
%SD6056	--	3028	SCA_STS_DI3_POSITION_STORED	Position stored in the DI3 transition
Counters				
%SD6064	--	3032	SCA_STS_BUILT_IN_COUNTER	Built-in counter value
%SD6068	--	3034	SCA_STS_BUILT_IN_COUNTER_DI3	Built-in counter stored in the DI3 transition
%SD6072	--	3036	SCA_STS_ENC1_COUNTER	Encoder 1 counter value
%SD6076	--	3038	SCA_STS_ENC2_COUNTER	Encoder 2 counter value
%SD6080	--	3040	SCA_STS_ENC_COUNTER_Z1	Encoder counter stored in the Z1 transition as defined in P00511
%SD6084	--	3042	SCA_STS_ENC_COUNTER_Z2	Encoder counter stored in the Z2 transition as defined in P00521
CANopen				
%SB6100	0	800	CO_STS_MASTER_CONTACTED	The CANopen master contacted all the slaves
%SB6100	1	801	CO_STS_MASTER_CONFIG_OK	The CANopen master dow nloaded the configurations of the slaves
%SB6100	2	802	CO_STS_MASTER_ERROR_CTRL_OK	Error control protocol (node guarding/heartbeat) initiated w ith the slaves
%SB6100	3	803	CO_STS_MASTER_INIT_FINISHED	The CANopen master initialized all the slaves
%SB6100	4	804	CO_STS_MASTER_INIT_ERROR	A slave presented an initialization error
%SB6100	5	805	CO_STS_MASTER_ERROR_CTRL	The CANopen master detected a fault in a slave through the error detection protocol
%SB6100	6	806	CO_STS_MASTER_EMCY	A slave reported EMCY
%SB6101	0	808	CO_STS_MASTER_NMT_TOGGLE	NMT command toggle bit feedback
%SB6101	5	813	CO_STS_MASTER_BUS_OFF	The CANopen master is in bus off
%SB6101	6	814	CO_STS_MASTER_POWER_OFF	The CANopen master has no pow er supply at the CAN interface
%SB6101	7	815	CO_STS_MASTER_COMM_DISABLED	Disabled CANopen master communication
%SB6102	0	816	CO_STS_SLAVE1_CONTACTED	The CANopen master successfully contacted the slave in the indicated address
%SB6102	1	817	CO_STS_SLAVE1_CONFIG_OK	The CANopen master successfully configured the slave
%SB6102	2	818	CO_STS_SLAVE1_ERROR_CTRL_OK	Error control protocol (node guarding/heartbeat) initiated w ith the slave

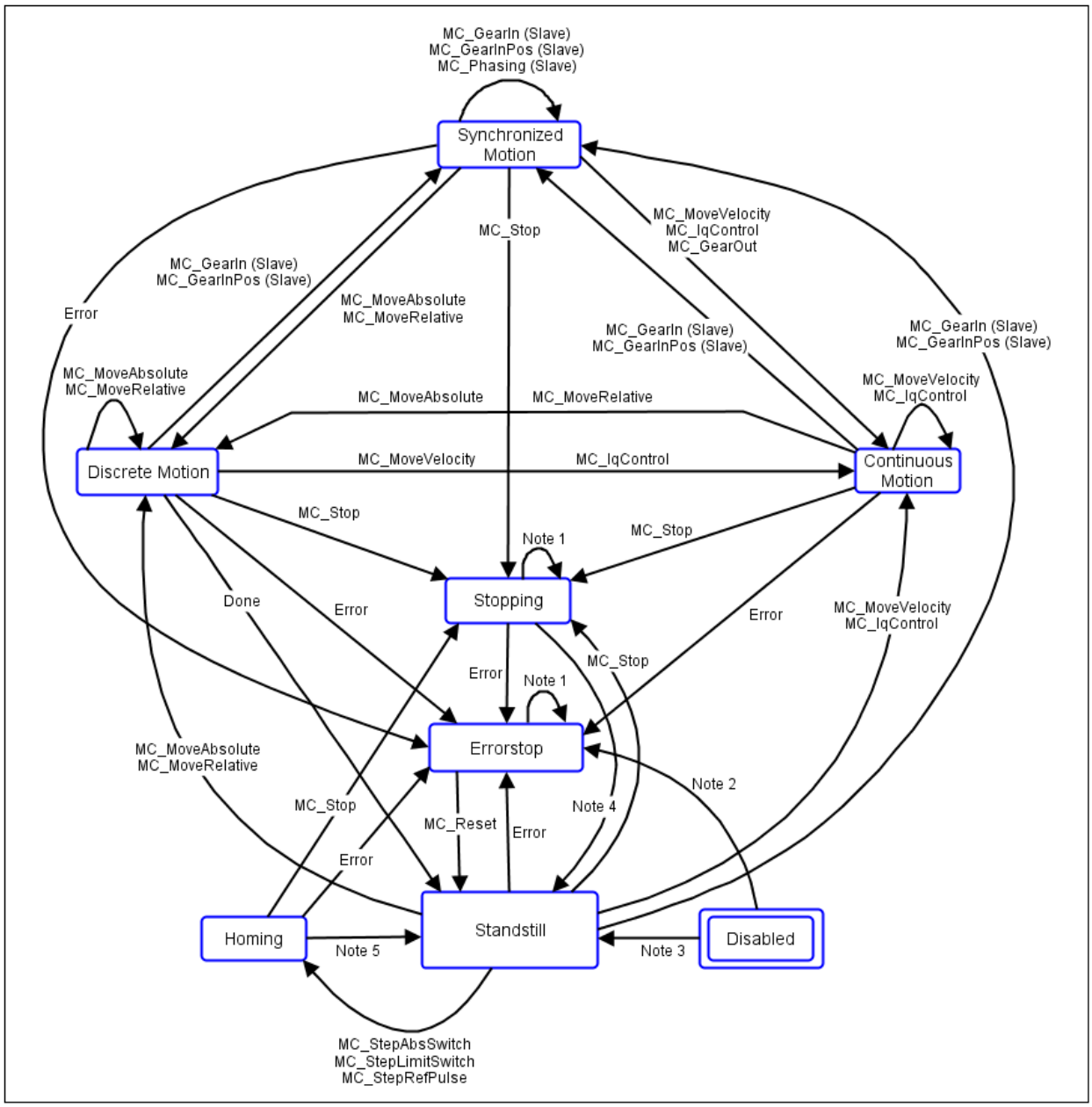
%SB6102	3	819	CO_STS_SLAVE1_INIT_FINISHED	Concluded slave initialization
%SB6102	4	820	CO_STS_SLAVE1_INIT_ERROR	Initialization error in the indicated address slave
%SB6102	5	821	CO_STS_SLAVE1_ERROR_CTRL_FAIL	Fault detected in some slave from the CANopen master error detection protocol
%SB6102	6	822	CO_STS_SLAVE1_EMCY	The slave in the indicated address reported EMCY error
%SB6104	0	832	CO_STS_SLAVE2_CONTACTED	The CANopen master successfully contacted the slave in the indicated address
%SB6104	1	833	CO_STS_SLAVE2_CONFIG_OK	The CANopen master successfully configured the slave
%SB6104	2	834	CO_STS_SLAVE2_ERROR_CTRL_OK	Error control protocol (node guarding/heartbeat) initiated with the slave
%SB6104	3	835	CO_STS_SLAVE2_INIT_FINISHED	Concluded slave initialization
%SB6104	4	836	CO_STS_SLAVE2_INIT_ERROR	Initialization error in the indicated address slave
%SB6104	5	837	CO_STS_SLAVE2_ERROR_CTRL_FAIL	Fault detected in some slave from the CANopen master error detection protocol
%SB6104	6	838	CO_STS_SLAVE2_EMCY	The slave in the indicated address reported EMCY error
...	...	...	...	...
%SB6354	0	2832	CO_STS_SLAVE127_CONTACTED	The CANopen master successfully contacted the slave in the indicated address
%SB6354	1	2833	CO_STS_SLAVE127_CONFIG_OK	The CANopen master successfully configured the slave
%SB6354	2	2834	CO_STS_SLAVE127_ERROR_CTRL_OK	Error control protocol (node guarding/heartbeat) initiated with the slave
%SB6354	3	8235	CO_STS_SLAVE127_INIT_FINISHED	Concluded slave initialization
%SB6354	4	2836	CO_STS_SLAVE127_INIT_ERROR	Initialization error in the indicated address slave
%SB6354	5	2837	CO_STS_SLAVE127_ERROR_CTRL_FAIL	Fault detected in some slave from the CANopen master error detection protocol
%SB6354	6	2838	CO_STS_SLAVE127_EMCY	The slave in the indicated address reported EMCY error
%SW6360	--	3180	CO_SDO_ERROR_NODE_ID	SDO error: address of the slave with the last detected SDO error
%SW6362	--	3181	CO_SDO_ERROR_OBJECT_INDEX	SDO error: object index
%SW6364	--	3182	CO_SDO_ERROR_OBJECT_SUBINDEX	SDO error: object sub-index
%SW6366	--	3183	CO_SDO_ERROR_FUNCTION	SDO error: function (writing/reading)
%SD6368	--	3184	CO_SDO_ERROR_VALUE	SDO error: value
%SD6372	--	3186	CO_SDO_ERROR_CODE	SDO error: error code
%SB6380	--	3190	CO_EMCY_SLAVE_ID	Last reported EMCY: slave address
%SB6382	--	3191	CO_EMCY_DATA	Last reported EMCY: object data

Writing / Reading System Markers (Command)

Address	Bit	Modbus	Tag	Description
CANopen				
%CB6000	--	3000	CO_CMD_NMT_COMMAND	NMT command transmission by the CANopen master: command code
%CB6001	0	8	CO_CMD_NMT_TOGGLE	NMT command transmission by the CANopen master: toggle bit
%CB6001	7	15	CO_CMD_DISABLE	Disables the CANopen communication
%CB6002	--	3001	CO_CMD_NMT_SLAVE_ADDR	NMT command transmission by the CANopen master: slave address

NOTE!
Below description of the real axis and virtual status:

0. Disabled.
1. Errorstop.
2. Standstill.
3. Stopping.
4. Homing.
5. Continuous Motion.
6. Discrete Motion.
7. Synchronized Motion.

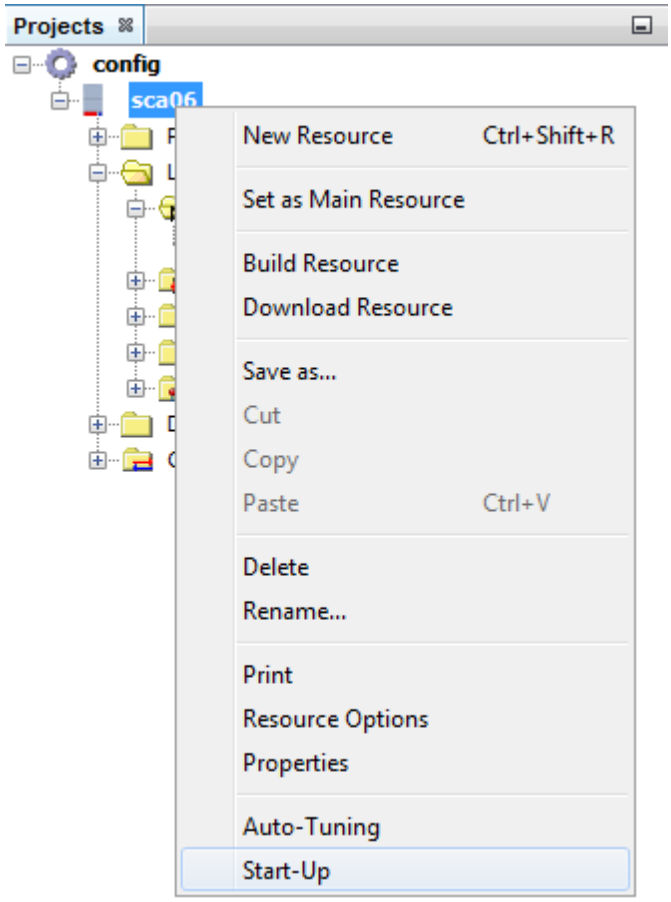


Note 1: When the drive is in "Stopping" on "Errorstop" every block can be called, but only MC_Reset block is executed;
Note 2: Attempt to enable the drive, but the drive is in fault;
Note 3: Enabling the drive and the drive is not in fault;
Note 4: MC_Stop.Done is true and MC_Stop.Execute is false;
Note 5: MC_StepDirect, MC_StepRefPulse or MC_FinishHoming.

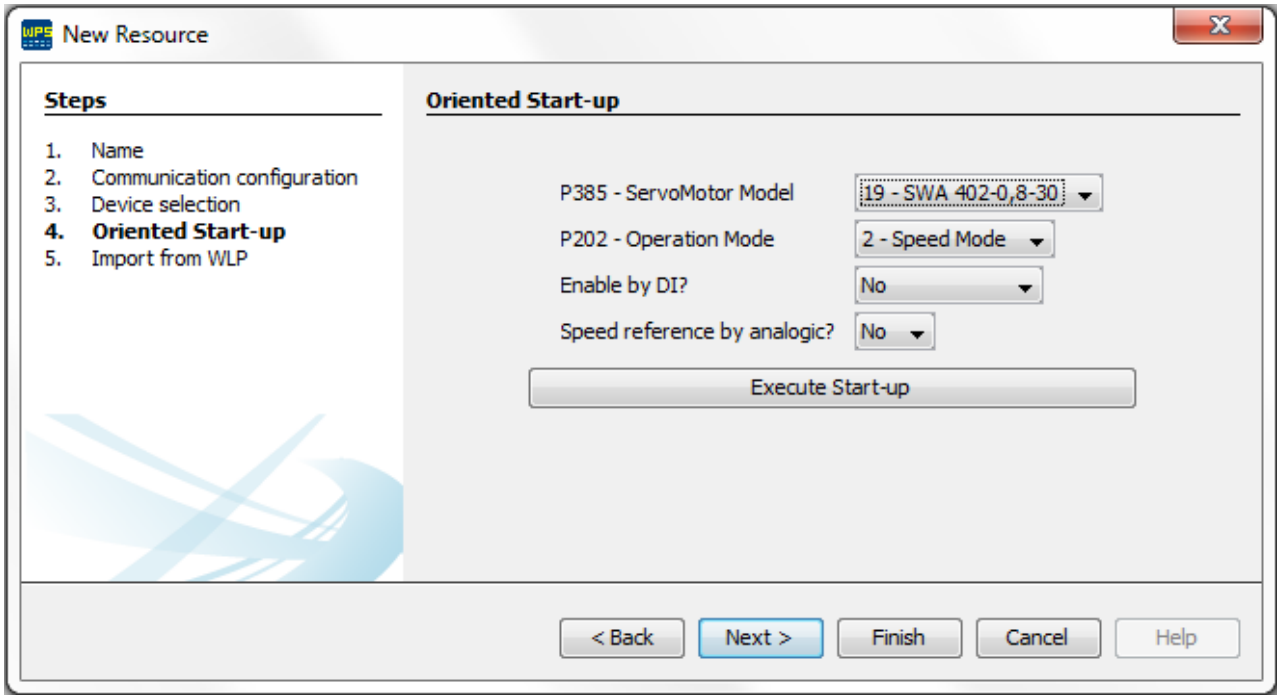
11.10.3Oriented Start-Up

The function Oriented Start-Up is utilized to realize the minimal required configuration to put SCA-06 into operation.

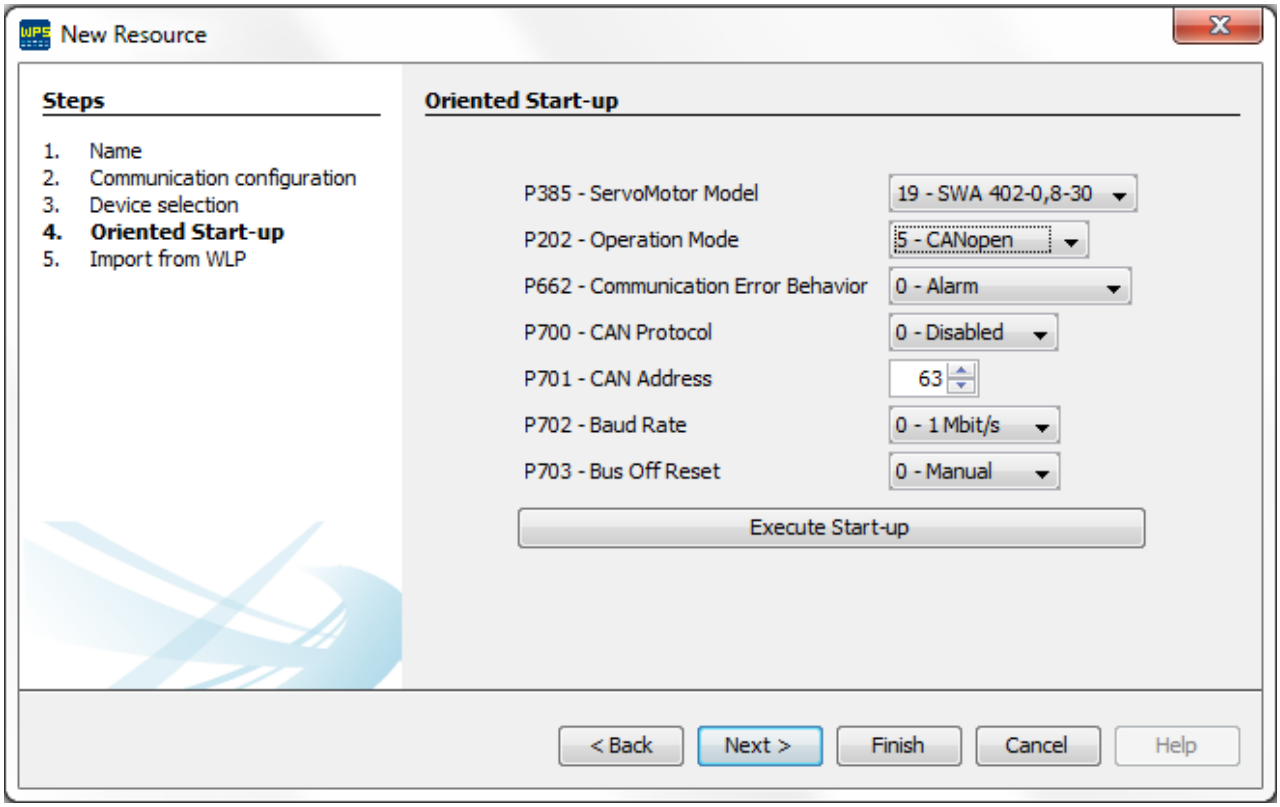
The Oriented Start-Up can be executed during the resource creation, or throught the context menu off the resource by selecting the option **Oriented Start-Up**.



1. On the Start-Up screen the main options that require configuration are the parameters P385 (Servomotor Model) and P202 (Operation Mode).

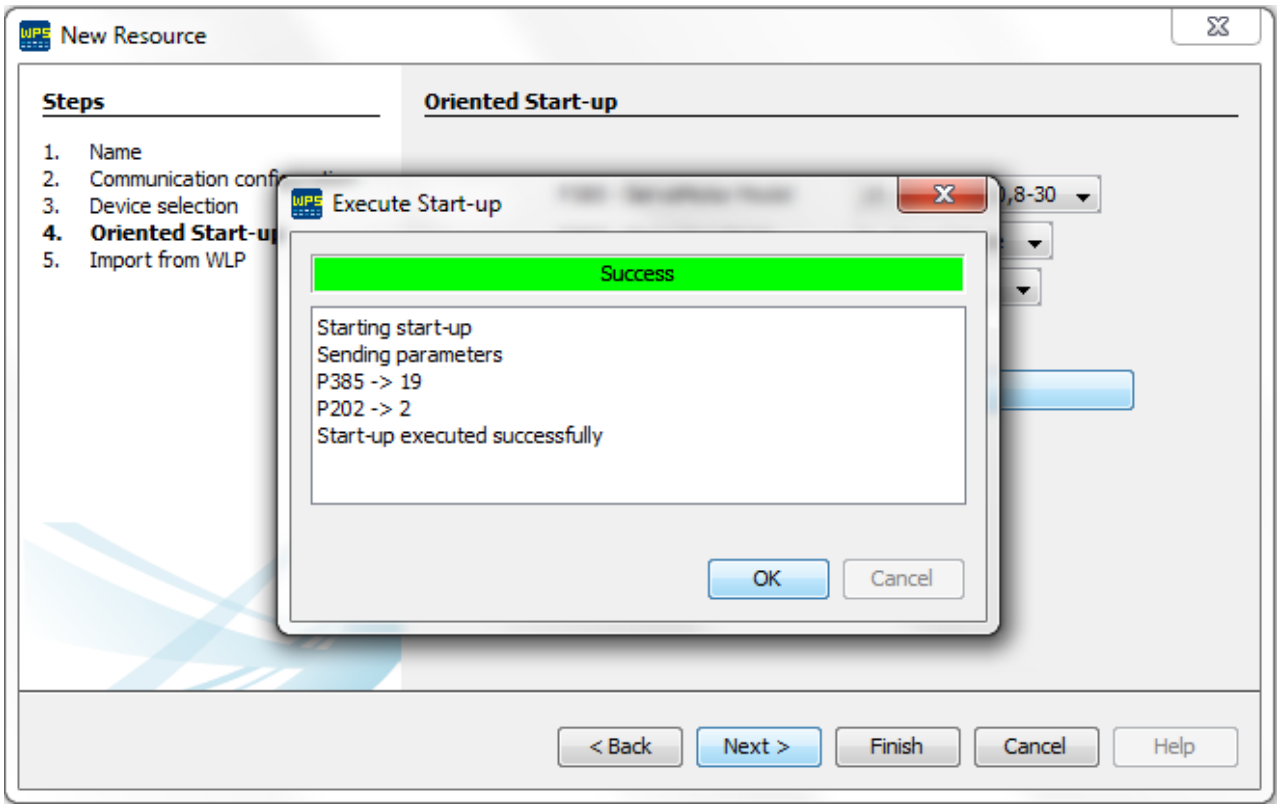


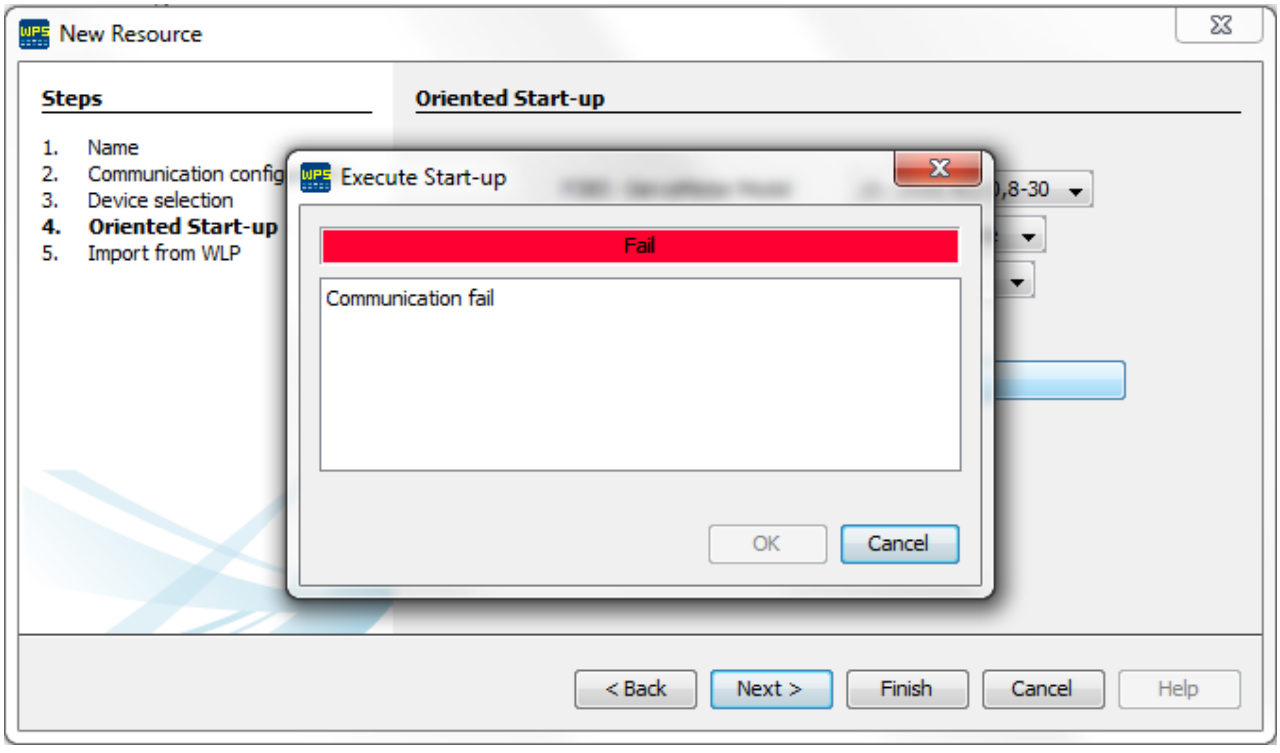
2. According to the operation mode selected the configurator will be adjusted enabling or disabling other options, below is a example of the options enabled when the operation mode is **5 - CANopen**.



3. After the definition of the operation mode and other options, is only necessary to click on **Execute Start-Up** , if it is executed properly a message informing success will be displayed otherwise a message informing

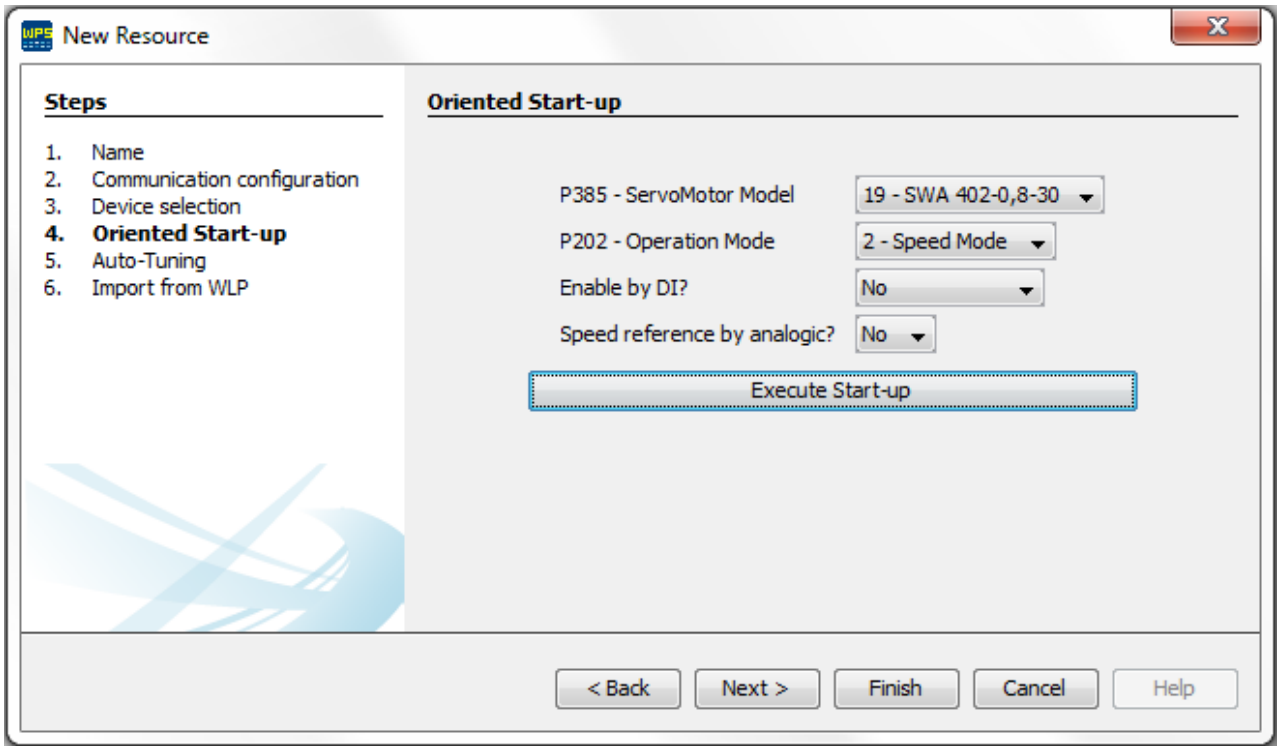
fail will be displayed.





4. After the successfull execution of the Start-Up during the resource creation, the system will enable a step

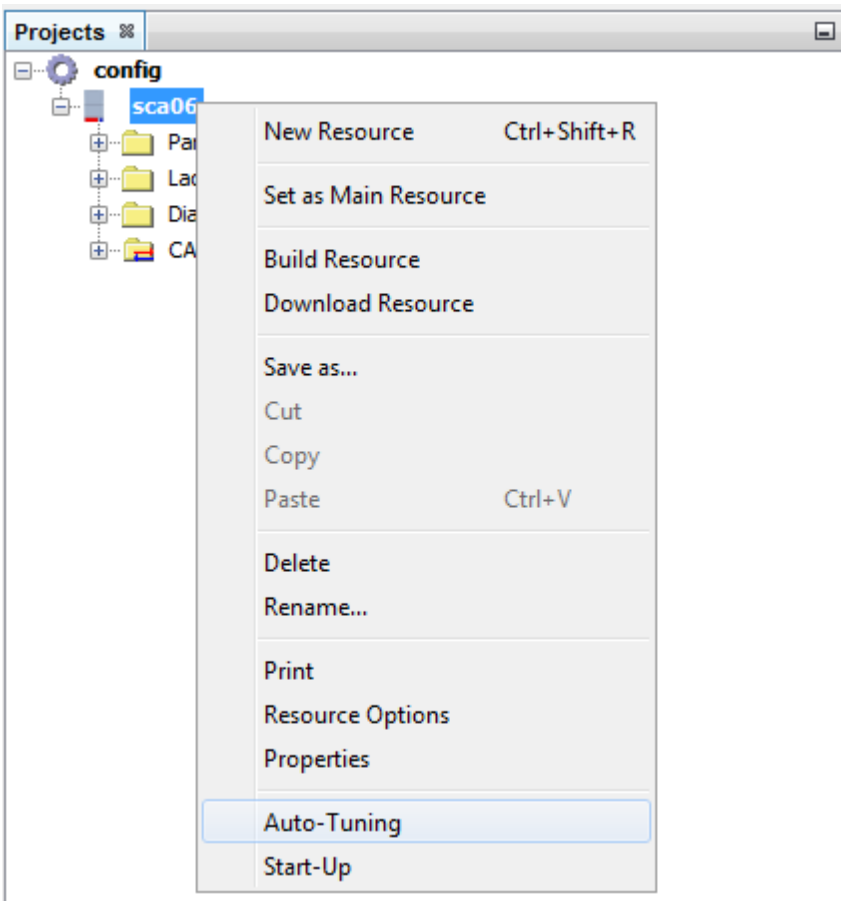
named Auto-Tuning in the wizard, this step is up to the user to perform or not.



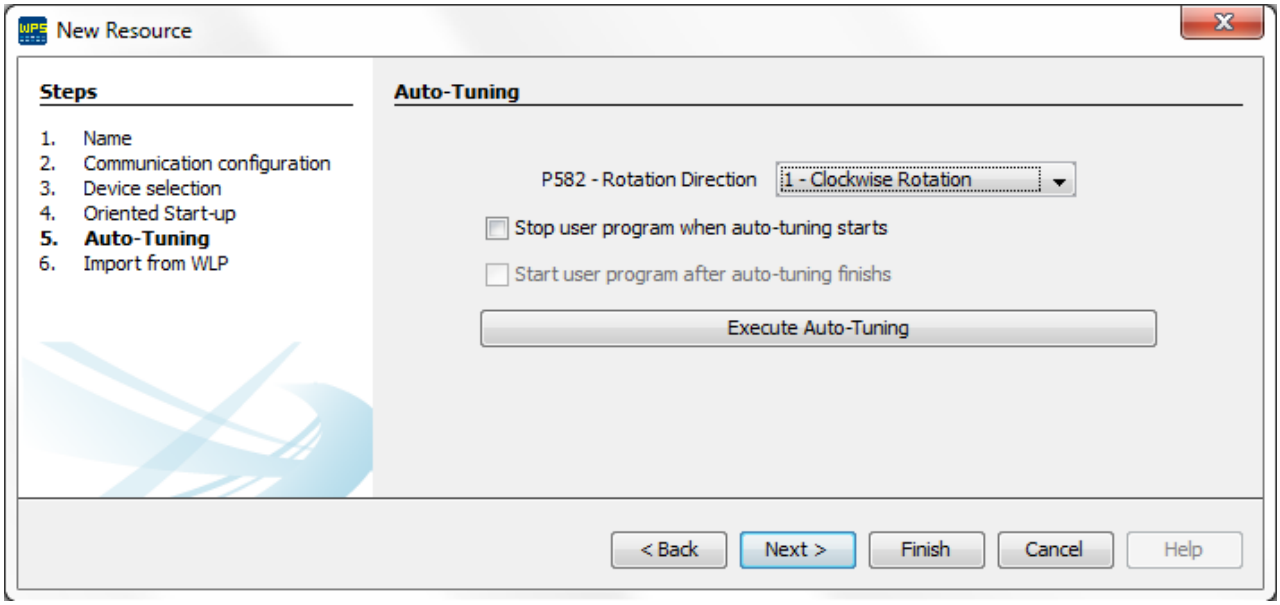
11.10.4Auto-Tuning

The function Auto-Tuning is utilized to realize automatic adjusts on SCA-06 to obtain a better performance of the equipment.

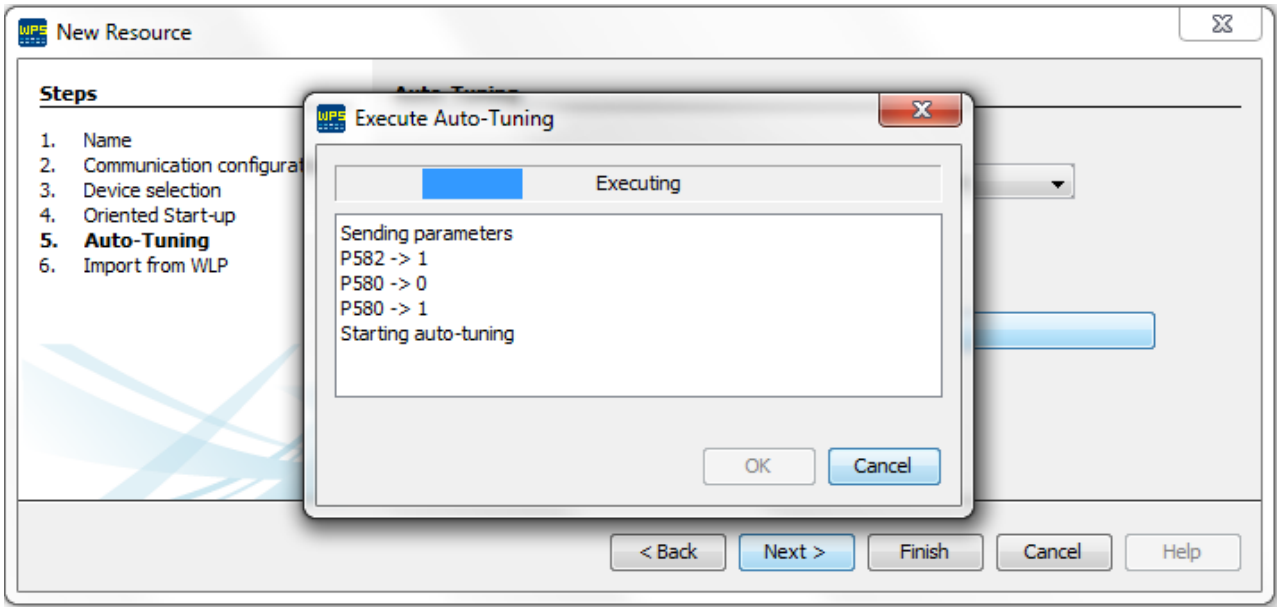
The Auto-Tuning can be executed during the resource creation, or throught the resource context menu by selecting the option **Auto-Tuning**.

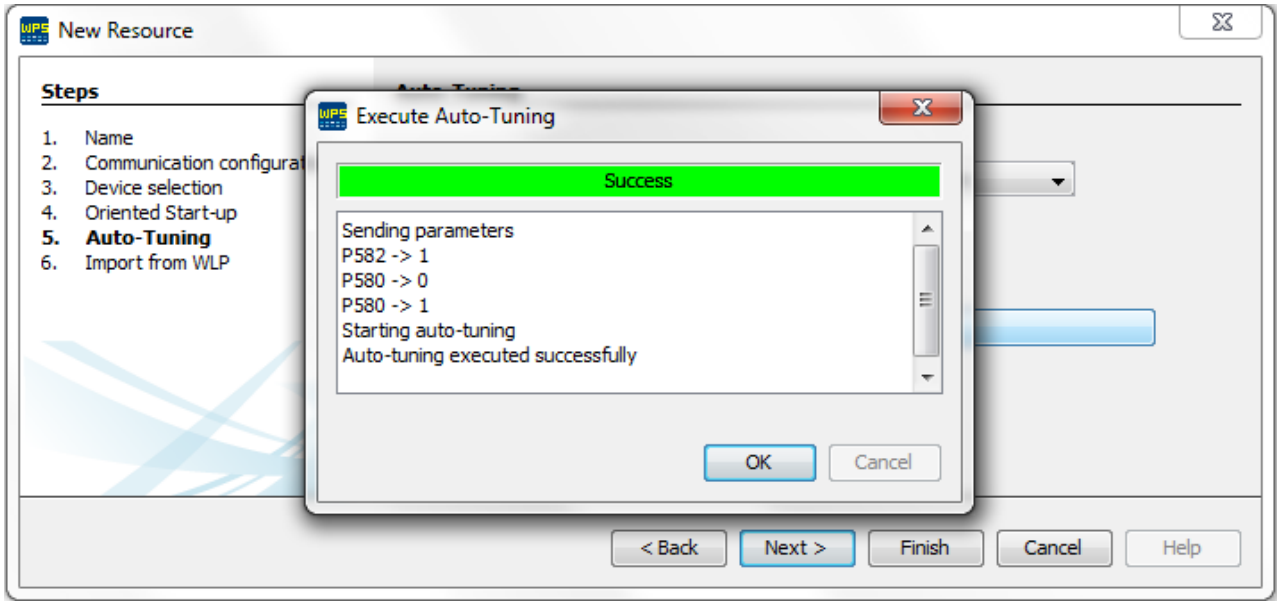


1. On the Auto-Tuning screen the options that require configuration are P582 (Rotation Direction) and if the user program should be stopped before execution and started again after the Auto-Tuning conclusion.



2. After choosing the options is only necessary to click on **Execute Auto-Tuning** to start the process that takes less than a minute. if it is executed properly a message informing success will be displayed otherwise a message informing fail will be displayed.

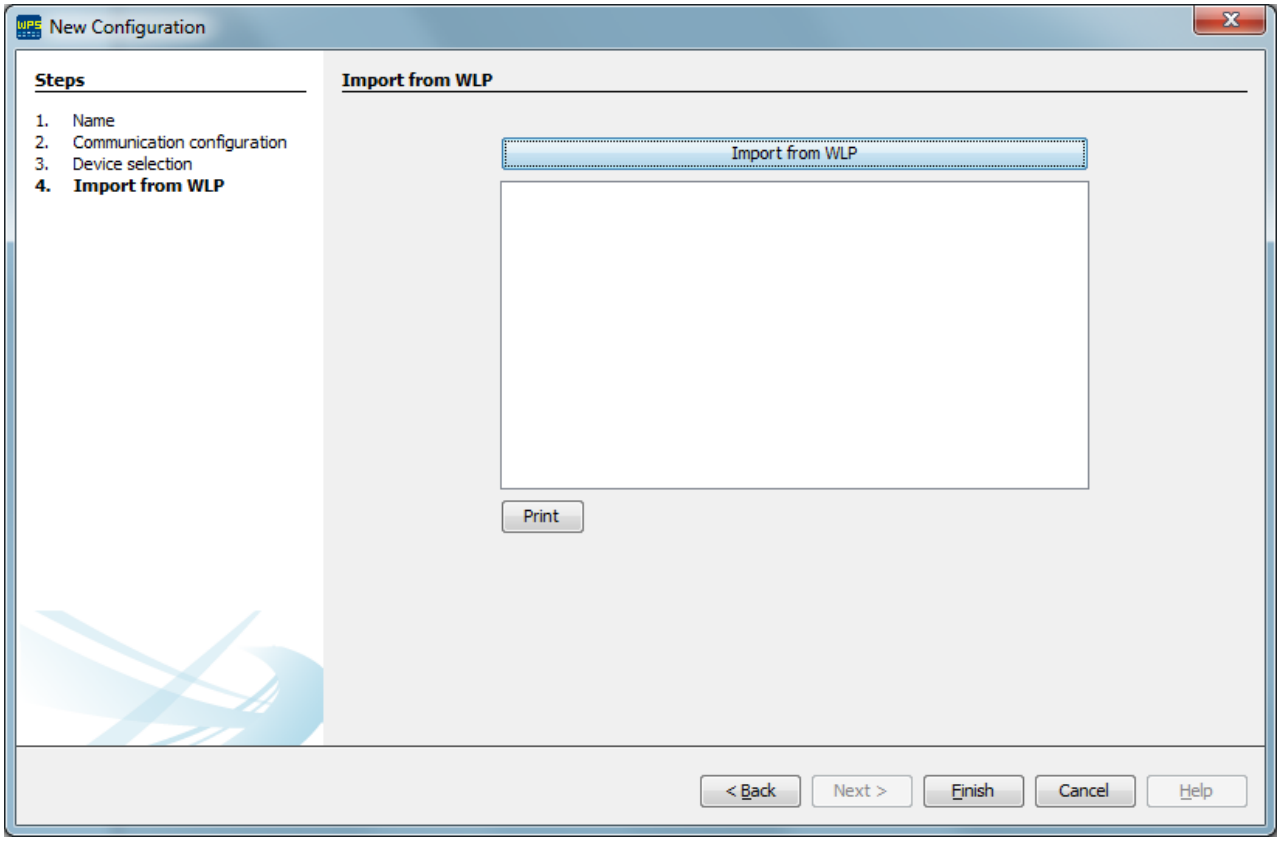




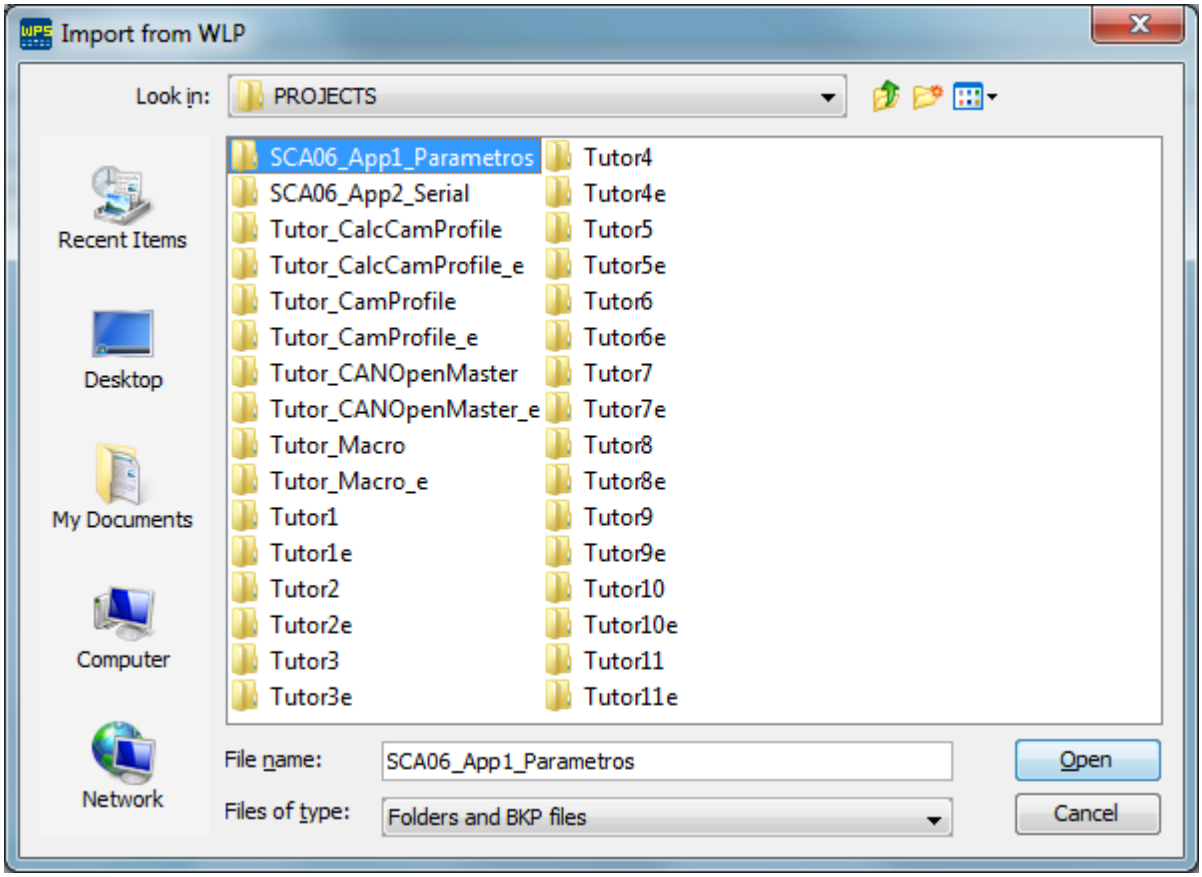
11.10.5Import from WLP

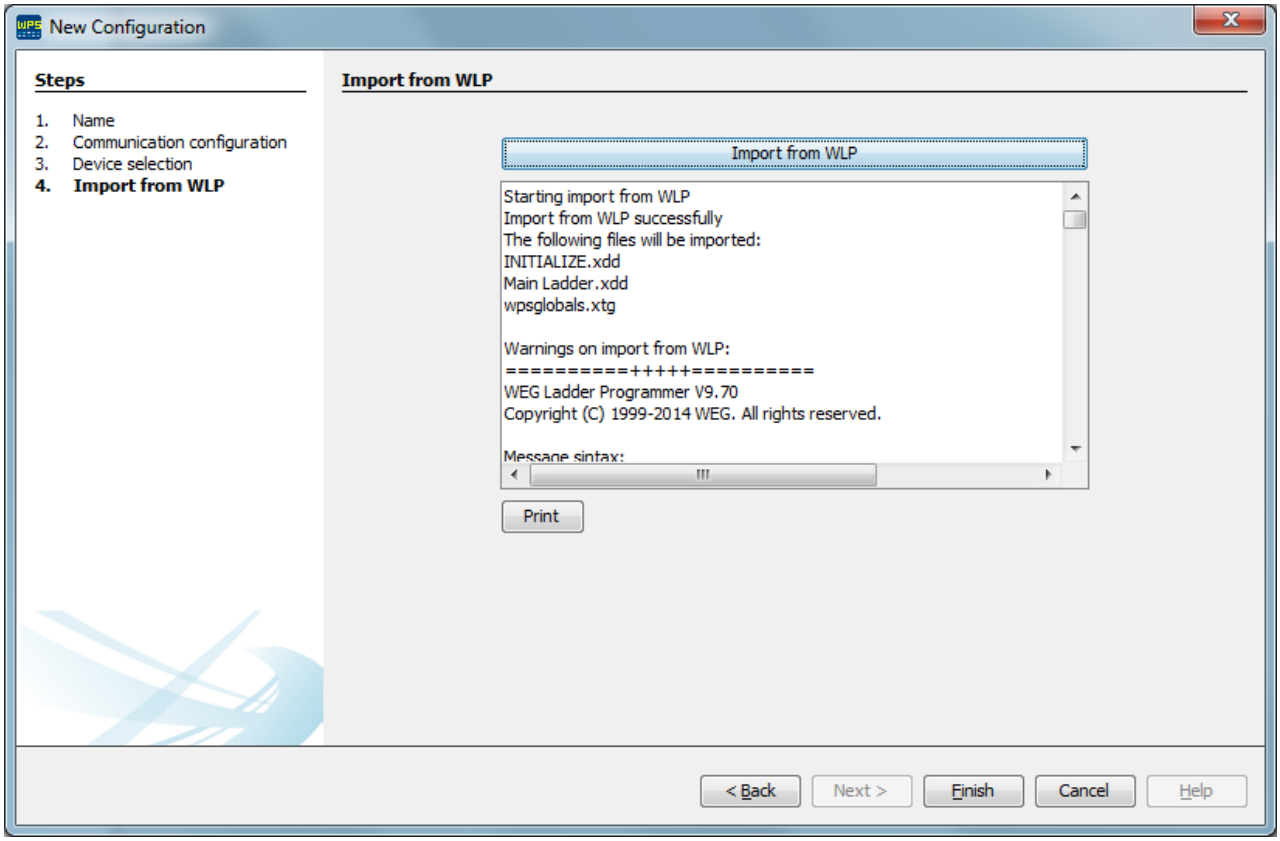
The function import from WLP is utilized to import Ladder developed on WLP software to equipment (device).

The import from WLP can be executed during the resource creation.



1. To execute the import WLP function click the Import from WLP button and select the WLP project folder or the WLP BKP file.


Import from WLP
Look in:
PROJECTS
SCA06_App1_Parametros
SCA06_App2_Serial
Tutor_CalcCamProfile
Tutor_CalcCamProfile_e
Tutor_CamProfile
Tutor_CamProfile_e
Tutor_CANOpenMaster
Tutor_CANOpenMaster_e
Tutor_Macro
Tutor_Macro_e
Tutor1
Tutor1e
Tutor2
Tutor2e
Tutor3
Tutor3e
Tutor4
Tutor4e
Tutor5
Tutor5e
Tutor6
Tutor6e
Tutor7
Tutor7e
Tutor8
Tutor8e
Tutor9
Tutor9e
Tutor10
Tutor10e
Tutor11
Tutor11e
Recent Items
Desktop
My Documents
Computer
Network
File name:
SCA06_App1_Parametros
Files of type:
Folders and BKP files
Open
Cancel



2. After import from WLP completed successfully click the Finish button to copy the imported files to new resource.

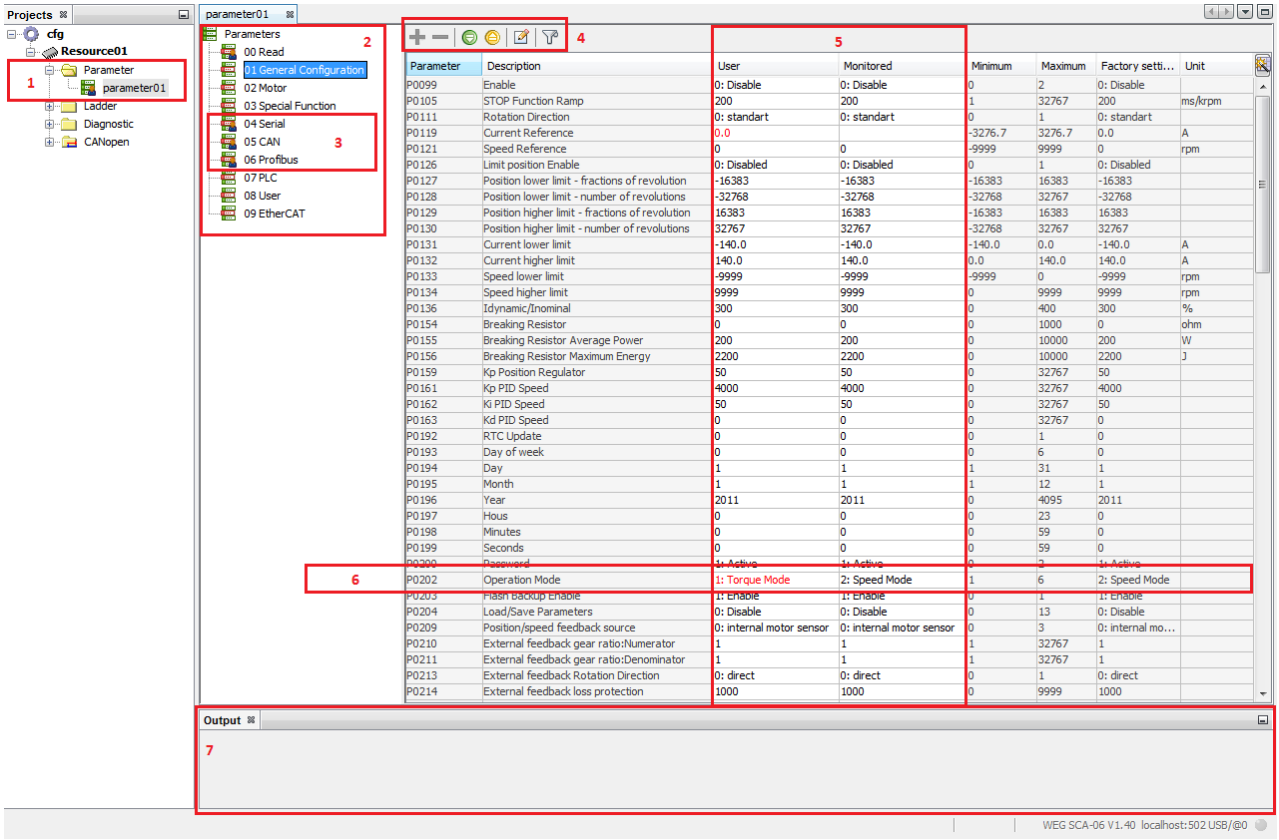
11.10.6 Parameters

11.10.6.1 Overview

The parameter configuration screen is used to configure and monitor all the parameters of the equipment, including the user parameters.

NOTE!
The reading and writing of such parameters is done on this screen; only the user parameter configuration must be sent the first time or whenever modified by means of the resource download routine.

Below is an overview of the parameter configuration screen.



1. **Parameter files**. In this part are all the parameter configuration files created by the user. Notice that when the file features a person figure on the table, it means this parameter table contains hidden parameters/ group of parameters.
2. **Group of parameters**. This tree shows all the group of parameters. Notice that the same parameter can be in more than one group, and when its value is modified, it will be modified in all the groups to which it belongs.
3. **Modified group of parameters**. Group of parameters which contain the figure of a person on the table means they have hidden parameters.
4. **Commands**. The commands are described below in the order they appear:
   4.1. **Unhide parameter**: In case some parameter has been hidden, this button allows making it visible again.
   4.2. **Hide parameter**: Just select one or more parameters on the table and trigger this command to hide them.
   4.3. **Save table**: It saves the values of the parameters shown on the equipment screen; the sent values are the ones in the User column. The flow is User -> Monitored (equipment)
   4.4. **Read table**: It reads the parameters of the equipment shown in the Monitored column and saves them in the parameter file in the User column. The flow is Monitored (equipment) -> User
   4.5. **User parameters**: It opens a screen to edit the user parameters.
   4.6. **Filter**: It opens a parameter filter option, and it can filter by parameter number or description.
   4.7. **User Parameters and Monitored Parameters**. These two columns show the off-line and on-line parameters, so to speak. The **User** column shows the values contained in the file located on the computer and the **Monitored** column shows the values that are effectively saved on the equipment. Whenever you use the **Save Parameter** option, the sent values will be from the **User** column to the **Monitored** column, that is, File -> Equipment. In case of reading, the flow is the opposite, from the **Monitored** column to the **User** column, that is, Equipment -> File. In case you wish to change the values directly on the equipment without changing it in the file, just click on the monitored column,

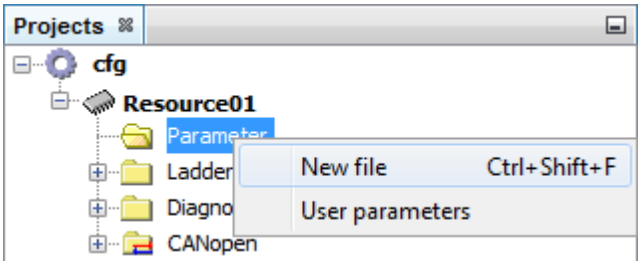
change the values and the modification will occur on-line.

5. **Modified parameters**: Whenever a parameter value in the **User** column is different from the **Monitored** column, it will be shown in red.
6. **Output**. This screen shows error information in case they occur during the writing or reading of the parameters.

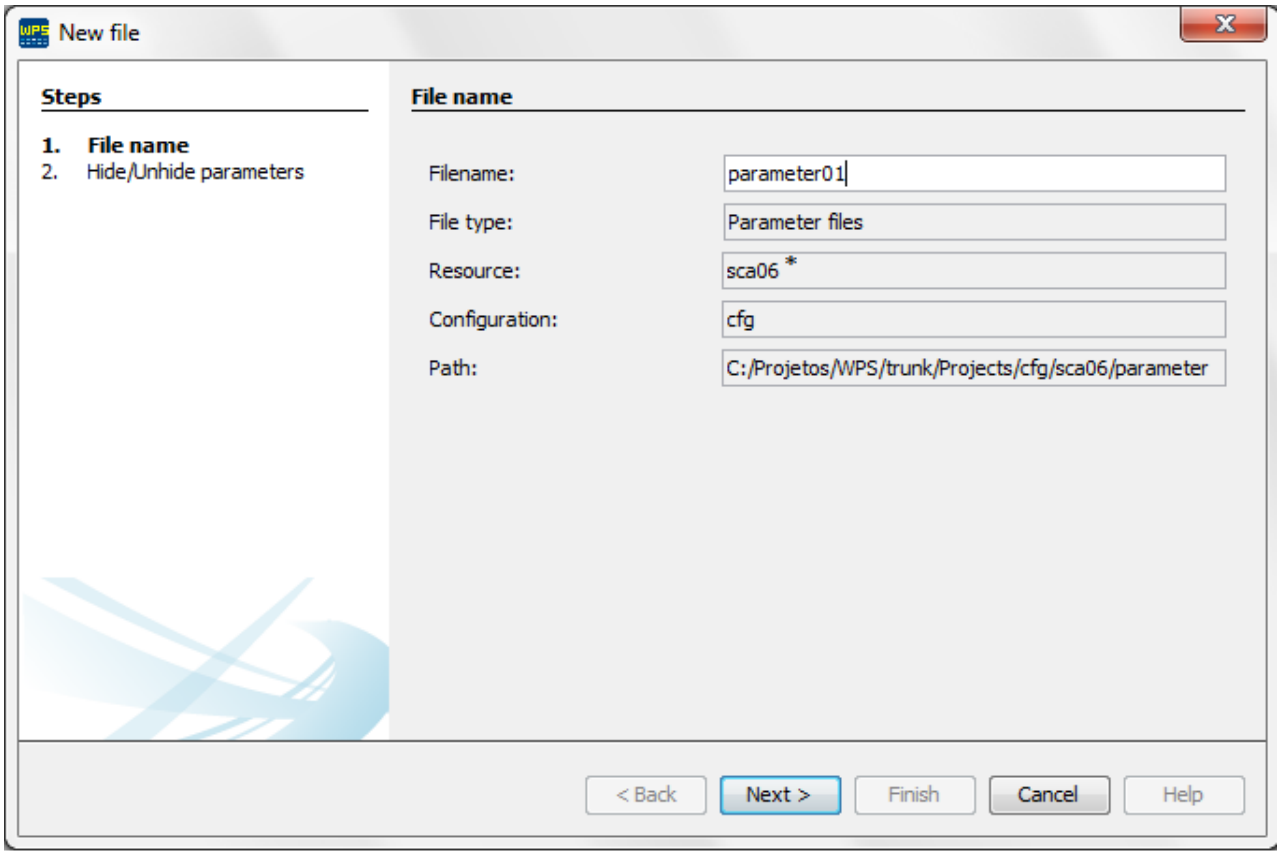
11.10.6.2 Configuration

Below is the list of the required steps to create a parameter file.

1. Create a new parameter file.

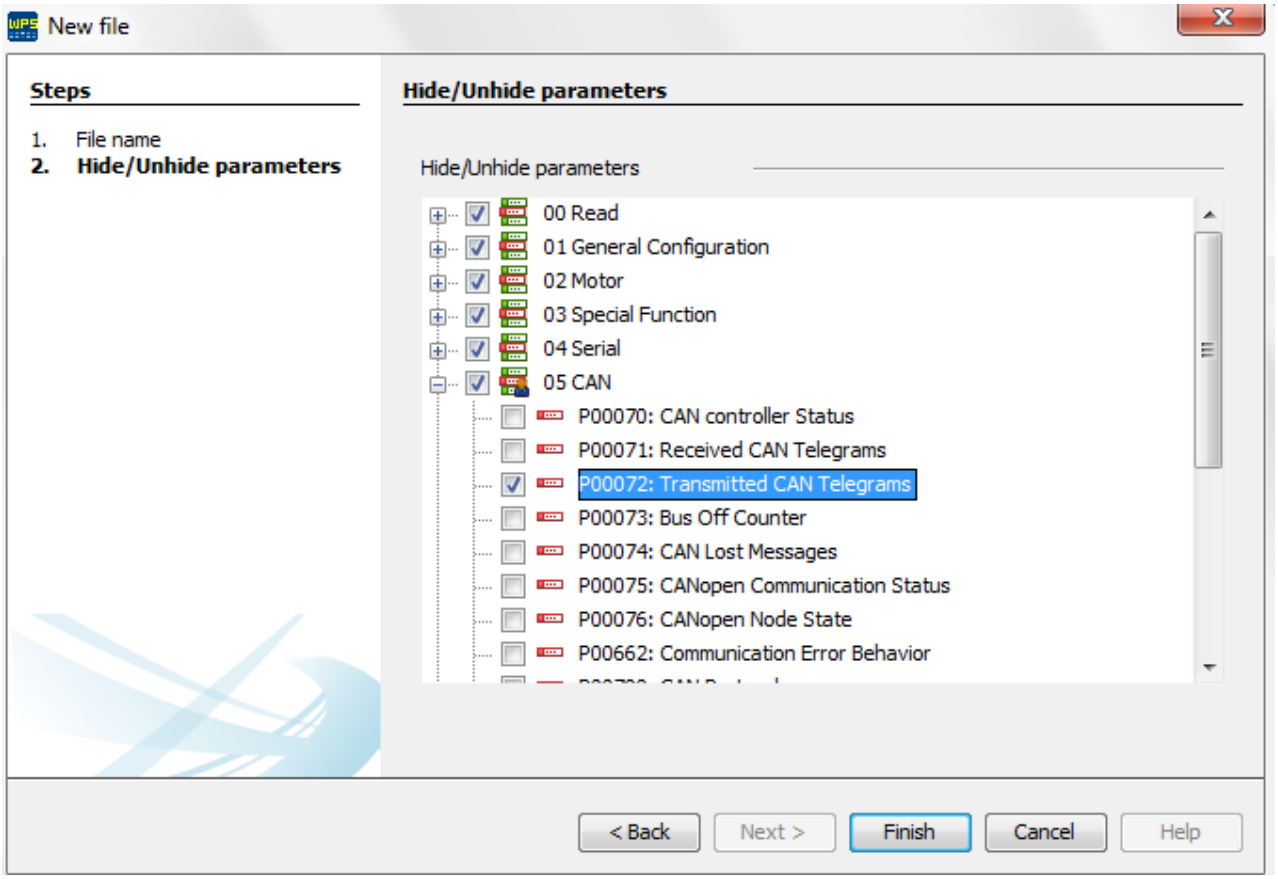


2. Define a name for the parameter file

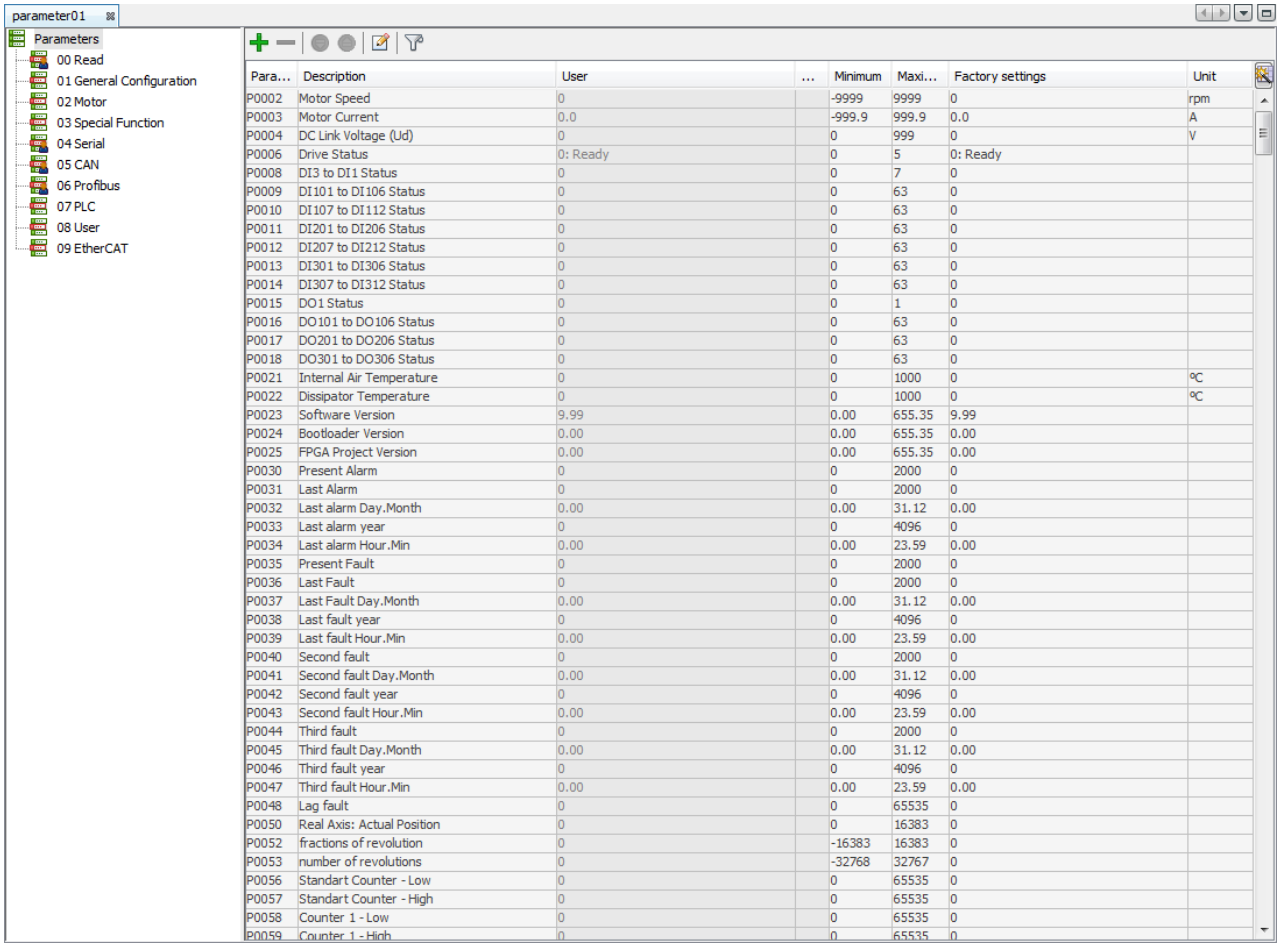


*** Resource: Resource01, SCA06, CFW300, etc.**

3. Configure which parameters you wish to view in your parameter table



4. After performing the steps above, the parameter file will be created and the equipment can be parameterized.

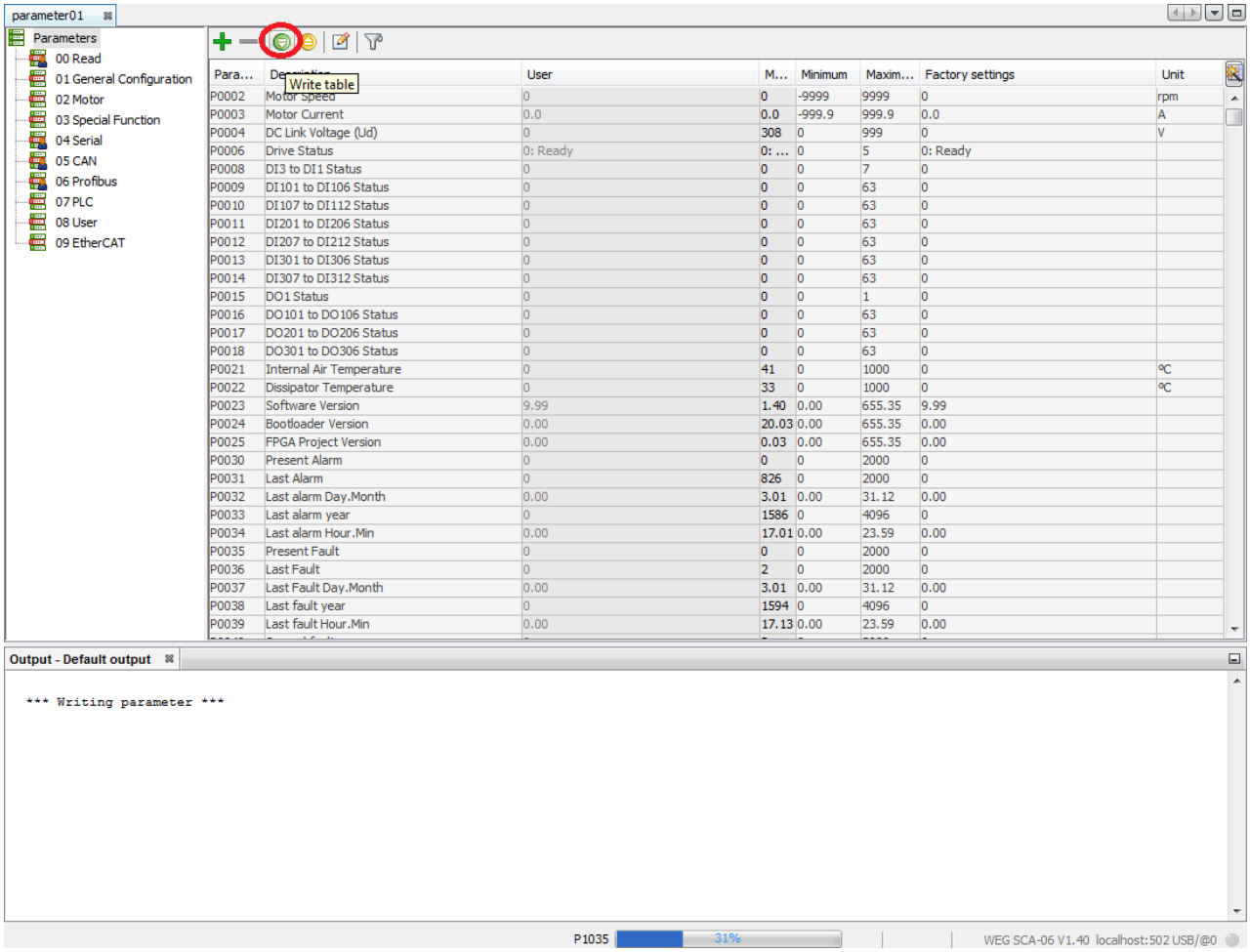


Para...	Description	User	...	Minimum	Maxi...	Factory settings	Unit
P0002	Motor Speed	0		-9999	9999	0	rpm
P0003	Motor Current	0.0		-999.9	999.9	0.0	A
P0004	DC Link Voltage (Ud)	0		0	999	0	V
P0006	Drive Status	0: Ready		0	5	0: Ready	
P0008	DI3 to DI1 Status	0		0	7	0	
P0009	DI101 to DI106 Status	0		0	63	0	
P0010	DI107 to DI112 Status	0		0	63	0	
P0011	DI201 to DI206 Status	0		0	63	0	
P0012	DI207 to DI212 Status	0		0	63	0	
P0013	DI301 to DI306 Status	0		0	63	0	
P0014	DI307 to DI312 Status	0		0	63	0	
P0015	DO1 Status	0		0	1	0	
P0016	DO101 to DO106 Status	0		0	63	0	
P0017	DO201 to DO206 Status	0		0	63	0	
P0018	DO301 to DO306 Status	0		0	63	0	
P0021	Internal Air Temperature	0		0	1000	0	ºC
P0022	Dissipator Temperature	0		0	1000	0	ºC
P0023	Software Version	9.99		0.00	655.35	9.99	
P0024	Bootloader Version	0.00		0.00	655.35	0.00	
P0025	FPGA Project Version	0.00		0.00	655.35	0.00	
P0030	Present Alarm	0		0	2000	0	
P0031	Last Alarm	0		0	2000	0	
P0032	Last alarm Day.Month	0.00		0.00	31.12	0.00	
P0033	Last alarm year	0		0	4096	0	
P0034	Last alarm Hour.Min	0.00		0.00	23.59	0.00	
P0035	Present Fault	0		0	2000	0	
P0036	Last Fault	0		0	2000	0	
P0037	Last Fault Day.Month	0.00		0.00	31.12	0.00	
P0038	Last fault year	0		0	4096	0	
P0039	Last fault Hour.Min	0.00		0.00	23.59	0.00	
P0040	Second fault	0		0	2000	0	
P0041	Second fault Day.Month	0.00		0.00	31.12	0.00	
P0042	Second fault year	0		0	4096	0	
P0043	Second fault Hour.Min	0.00		0.00	23.59	0.00	
P0044	Third fault	0		0	2000	0	
P0045	Third fault Day.Month	0.00		0.00	31.12	0.00	
P0046	Third fault year	0		0	4096	0	
P0047	Third fault Hour.Min	0.00		0.00	23.59	0.00	
P0048	Lag fault	0		0	65535	0	
P0050	Real Axis: Actual Position	0		0	16383	0	
P0052	fractions of revolution	0		-16383	16383	0	
P0053	number of revolutions	0		-32768	32767	0	
P0056	Standart Counter - Low	0		0	65535	0	
P0057	Standart Counter - High	0		0	65535	0	
P0058	Counter 1 - Low	0		0	65535	0	
P0059	Counter 1 - High	0		0	65535	0	

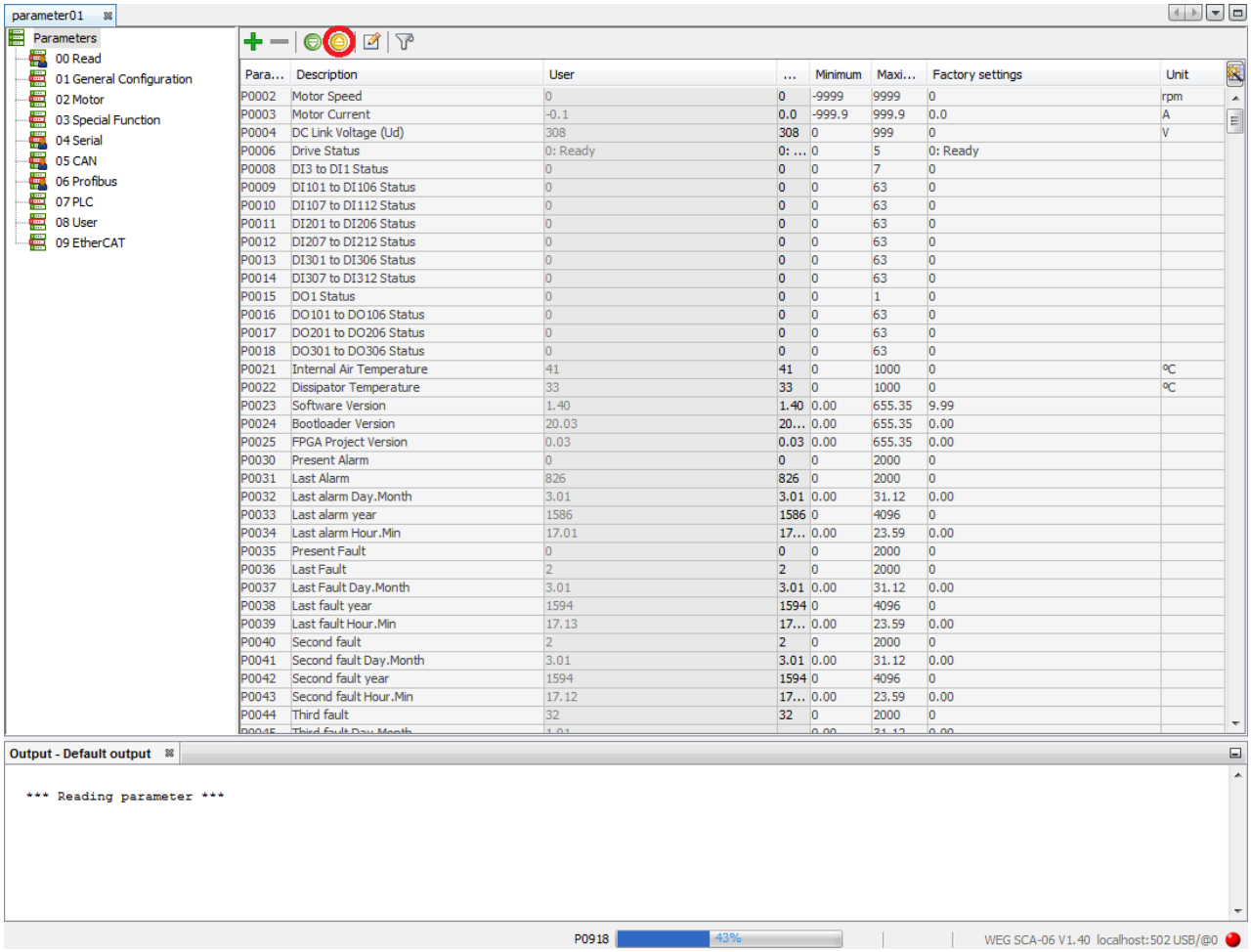
11.10.6.3 Read and Write of Parameters

There are 3 (three) ways to do the reading and writing of the parameters: by means of table, selection and group.

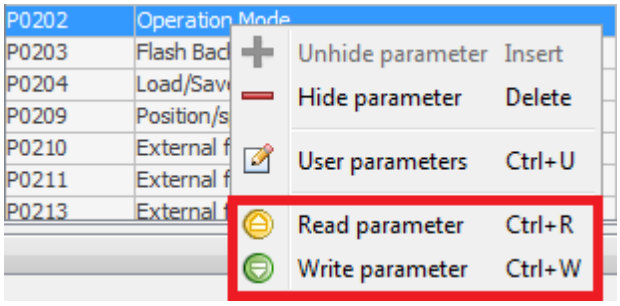
1. **Table writing**. The table writing command will send all visible parameters on the equipment screen. If and error occurs during the sending of some specific parameter, a message will be shown on the output window informing the error. It is important to notice that only visible parameters will be sent; therefore, it is necessary attention to which node of the group of parameters tree you are viewing. Example: If you wish to write all of them without filtering per group, just select the tree root.



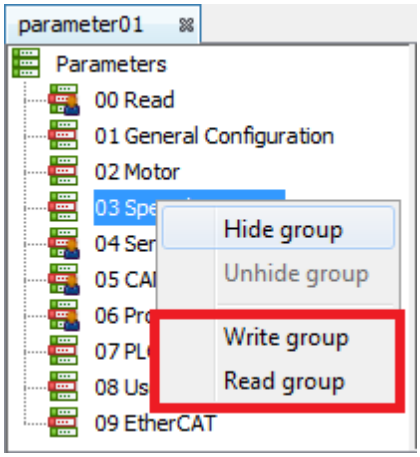
2. **Table reading**. The table reading command will read all the parameters of the equipment. If a error occurs during the reading of some specific parameter, a message will be shown on the output window informing the error. It is important to notice that only visible parameters will be read; therefore, it is necessary attention to which node of the group of parameters tree you are viewing. Example: If you wish to read all of them without filtering per group, just select the tree root.



3. **Reading/writing of specific parameters**. In order to read/write one or more specific parameters, just select them on the table, right click and choose the desired option: read or write parameter.



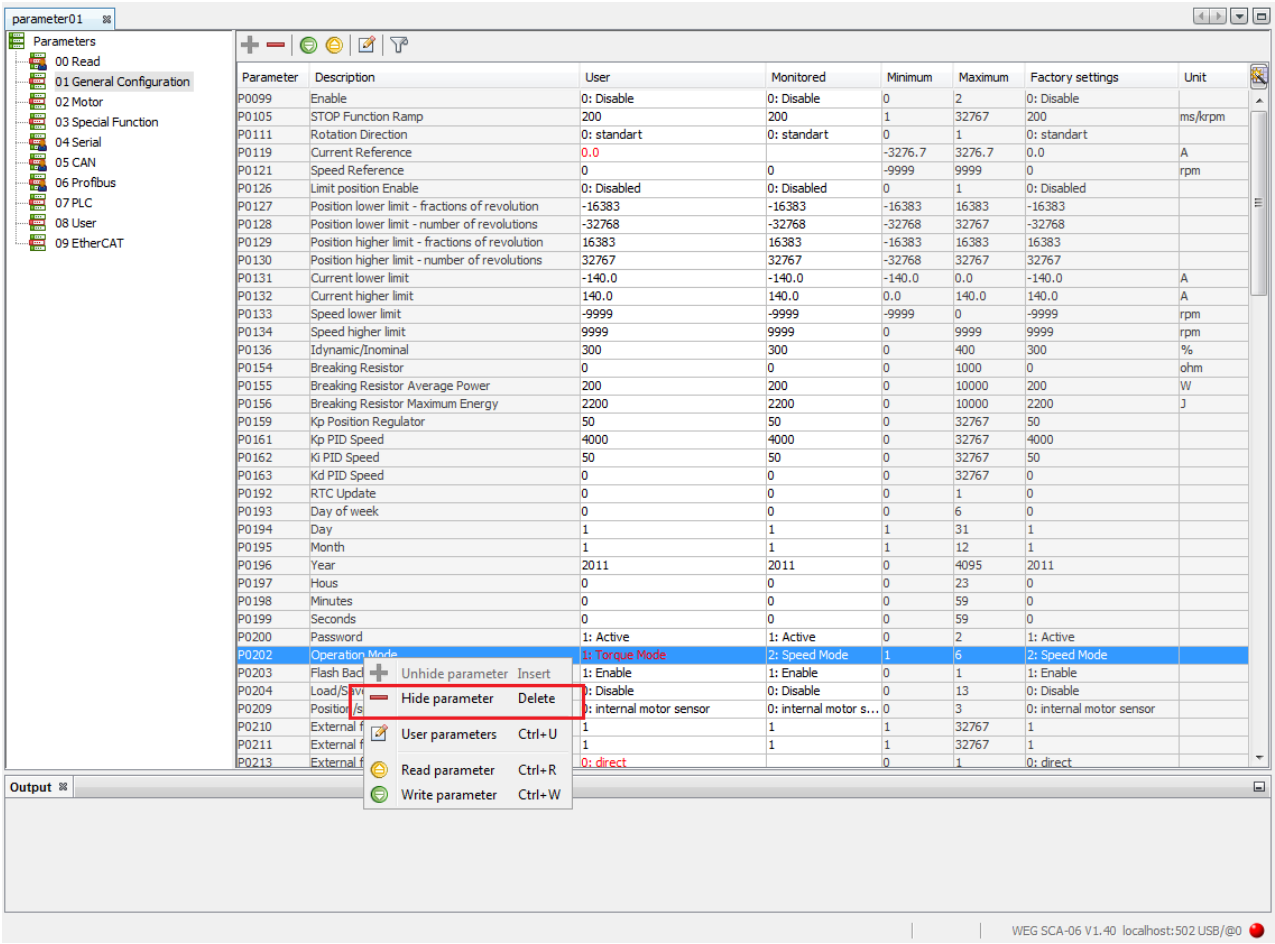
4. **Reading/writing of group of parameters**. In order to read/write only one group of parameters, just select it on the group tree, right click and choose the desired option: read or write group.



11.10.6.4 Hide/Unhide Parameters and Group of Parameters

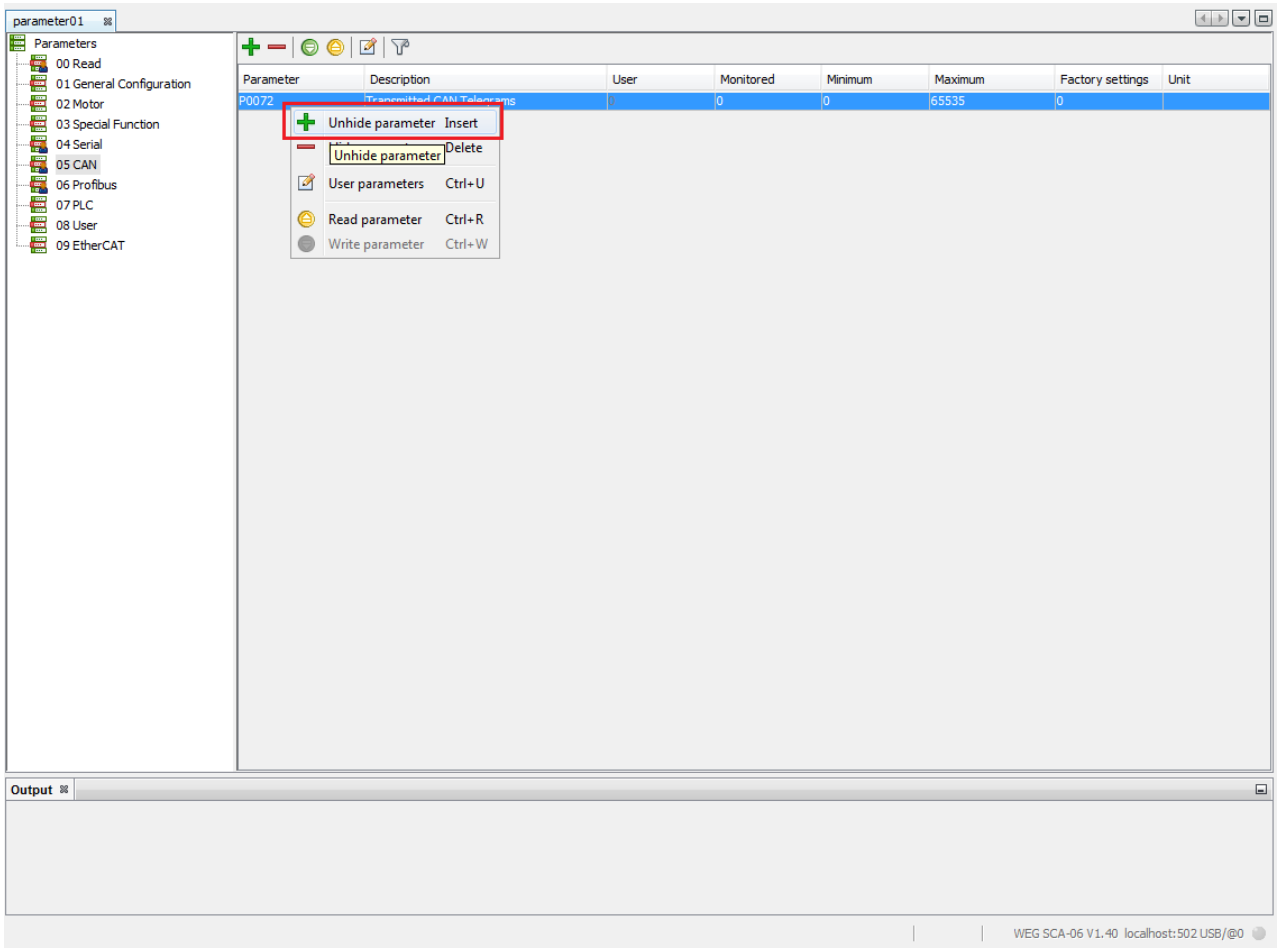
The parameter can be hidden/unhidden in two ways: individually or in group.

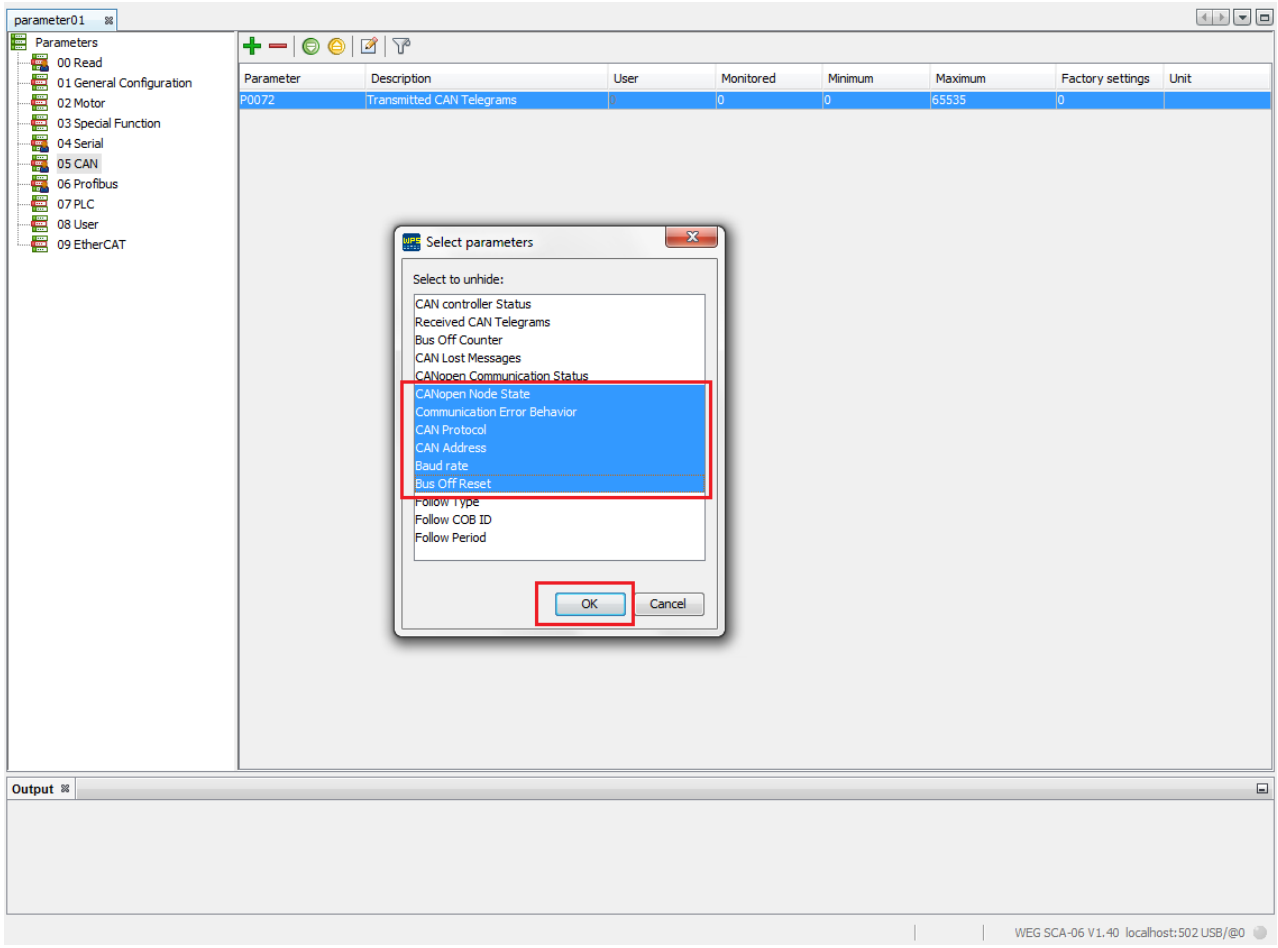
1. **Hide parameters**. In order to hide a parameter individually, just right click on the desired parameters and select the **Hide Parameter** option. You can also press the **Delete** key.

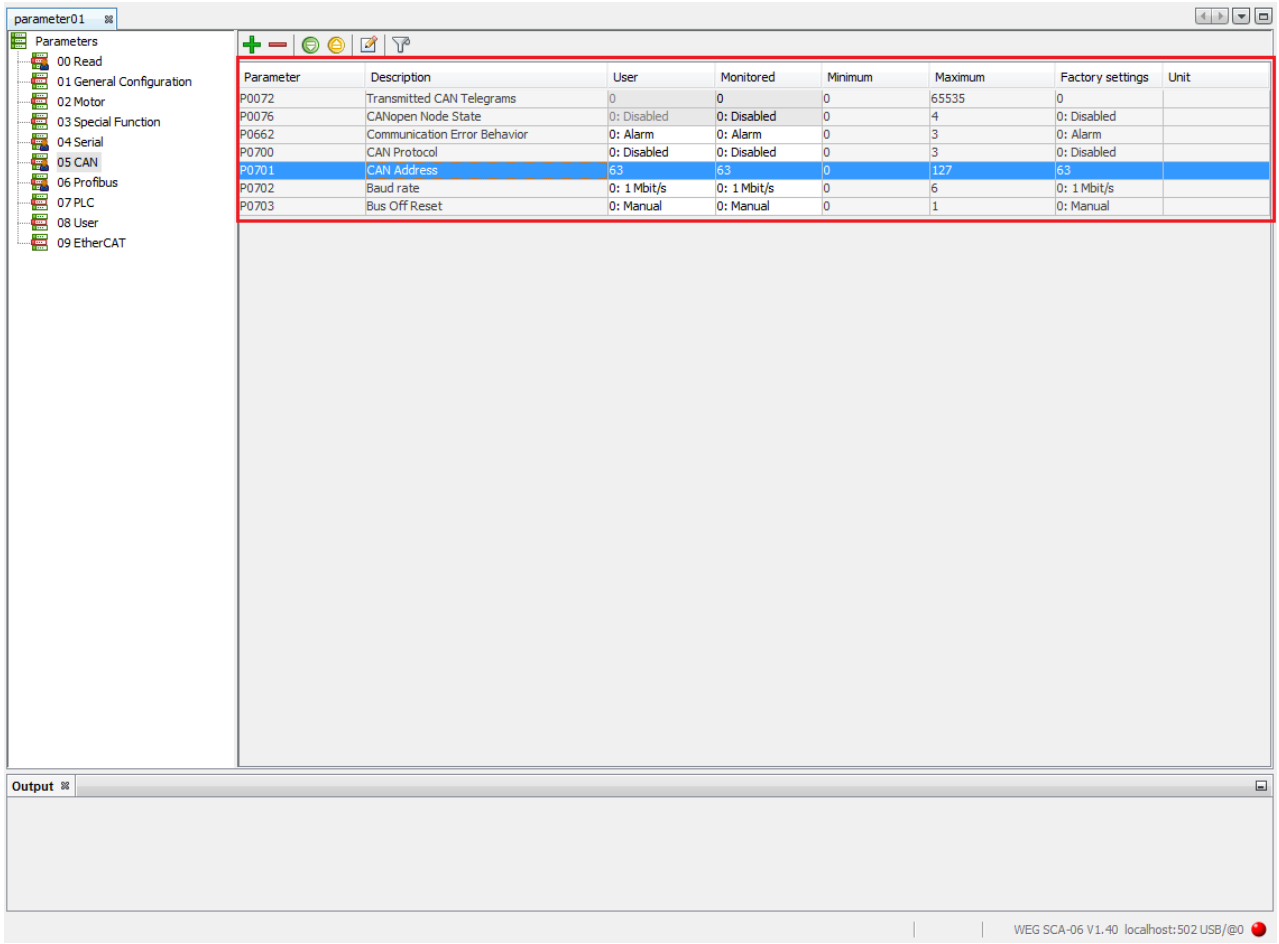


2. **Unhide Parameters**. In order to show hidden parameters, right click and choose the **Unhide Parameters**

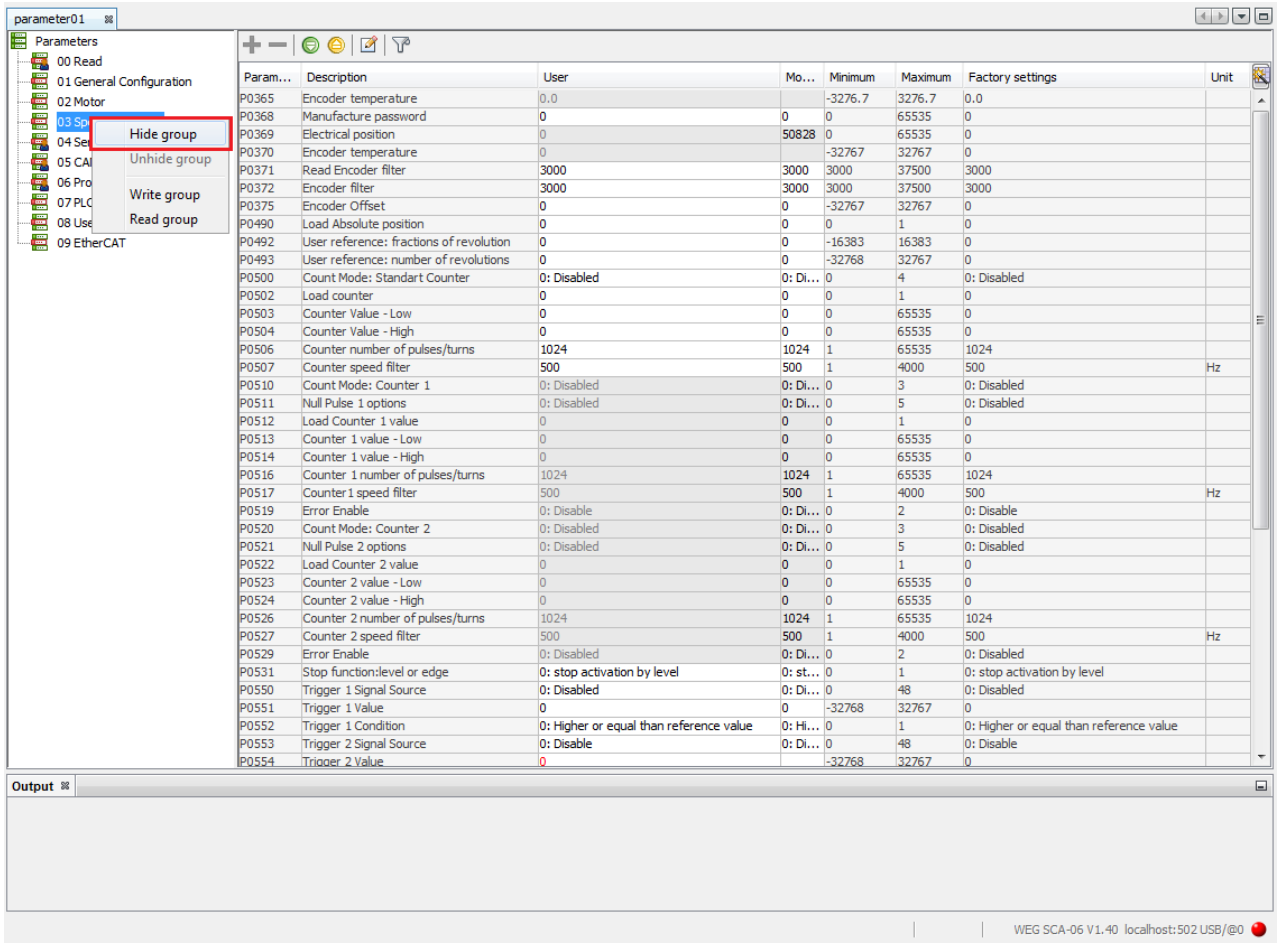
or press the **Insert** key. Then, a window will open and show the hidden parameters. Now, you just have to select the desired parameters and confirm.
Note: The parameters shown on this new window are only those which belong to the current filter according to the selection on the parameter group tree. In the figures below, the CAN group is selected; that means that only the hidden parameters of this group will be shown.

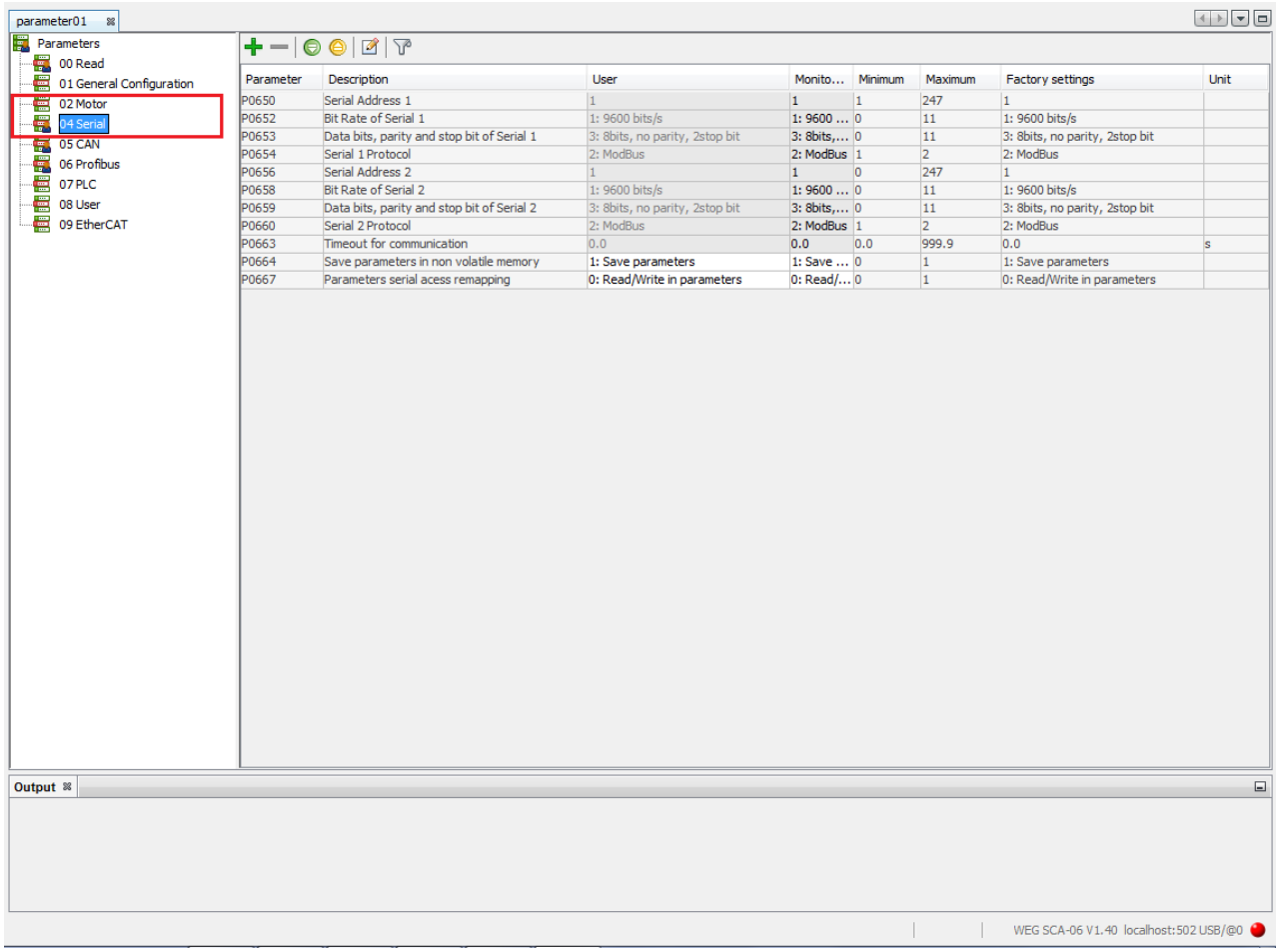



parameter01
Parameters
00 Read
01 General Configuration
02 Motor
03 Special Function
04 Serial
05 CAN
06 Profibus
07 PLC
08 User
09 EtherCAT
Parameter
Description
User
Monitored
Minimum
Maximum
Factory settings
Unit
P0072
Transmitted CAN Telegrams
0
0
65535
0
Select parameters
Select to unhide:
CAN controller Status
Received CAN Telegrams
Bus Off Counter
CAN Lost Messages
CANopen Communication Status
CANopen Node State
Communication Error Behavior
CAN Protocol
CAN Address
Baud rate
Bus Off Reset
Follow Type
Follow COB ID
Follow Period
OK
Cancel
Output
WEG SCA-06 V1.40 localhost:502 USB/@0

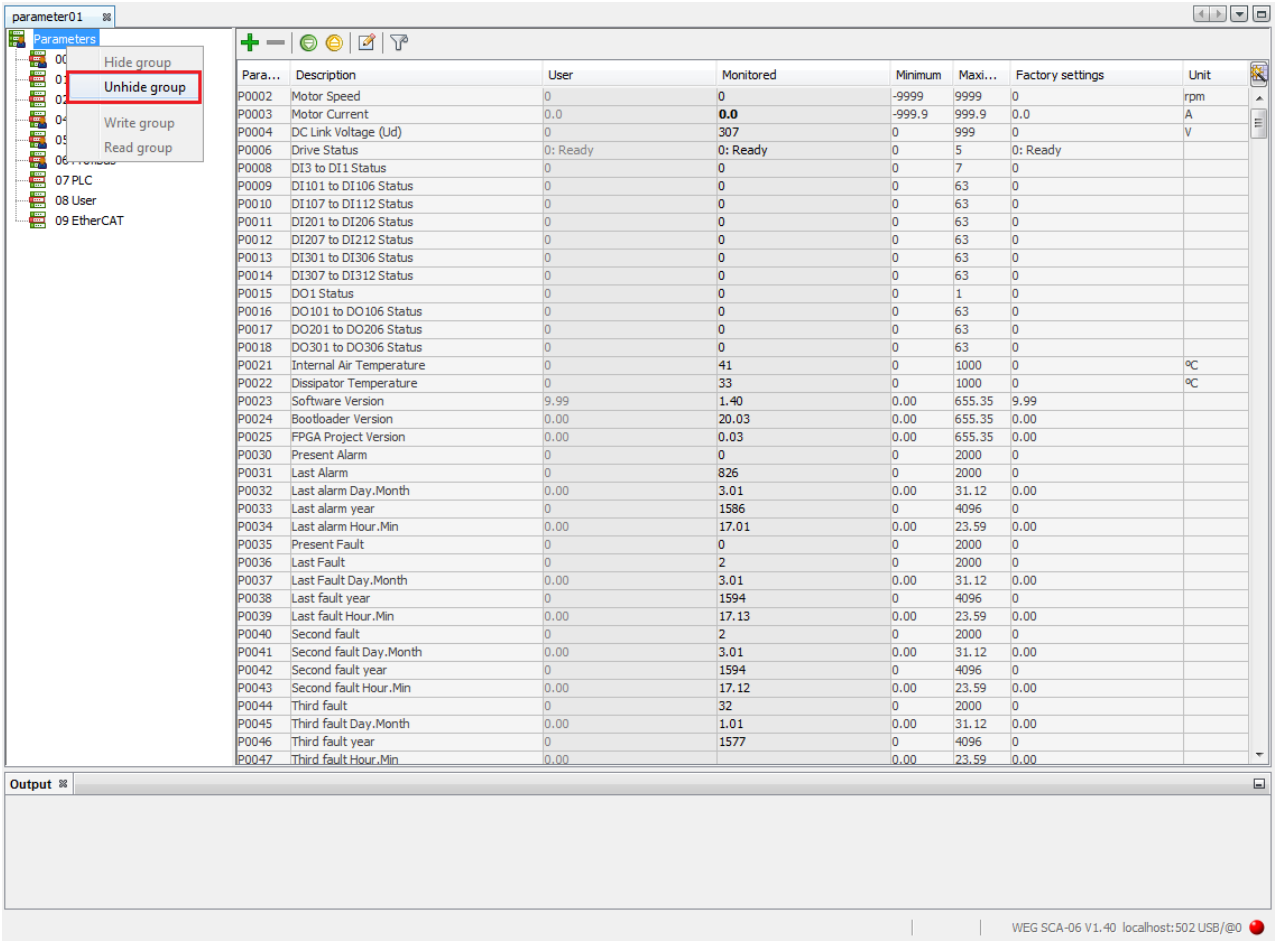


3. **Hide Group of Parameters**. In order to hide a group of parameters, just select the group on the tree and use the **Hide Group** option.

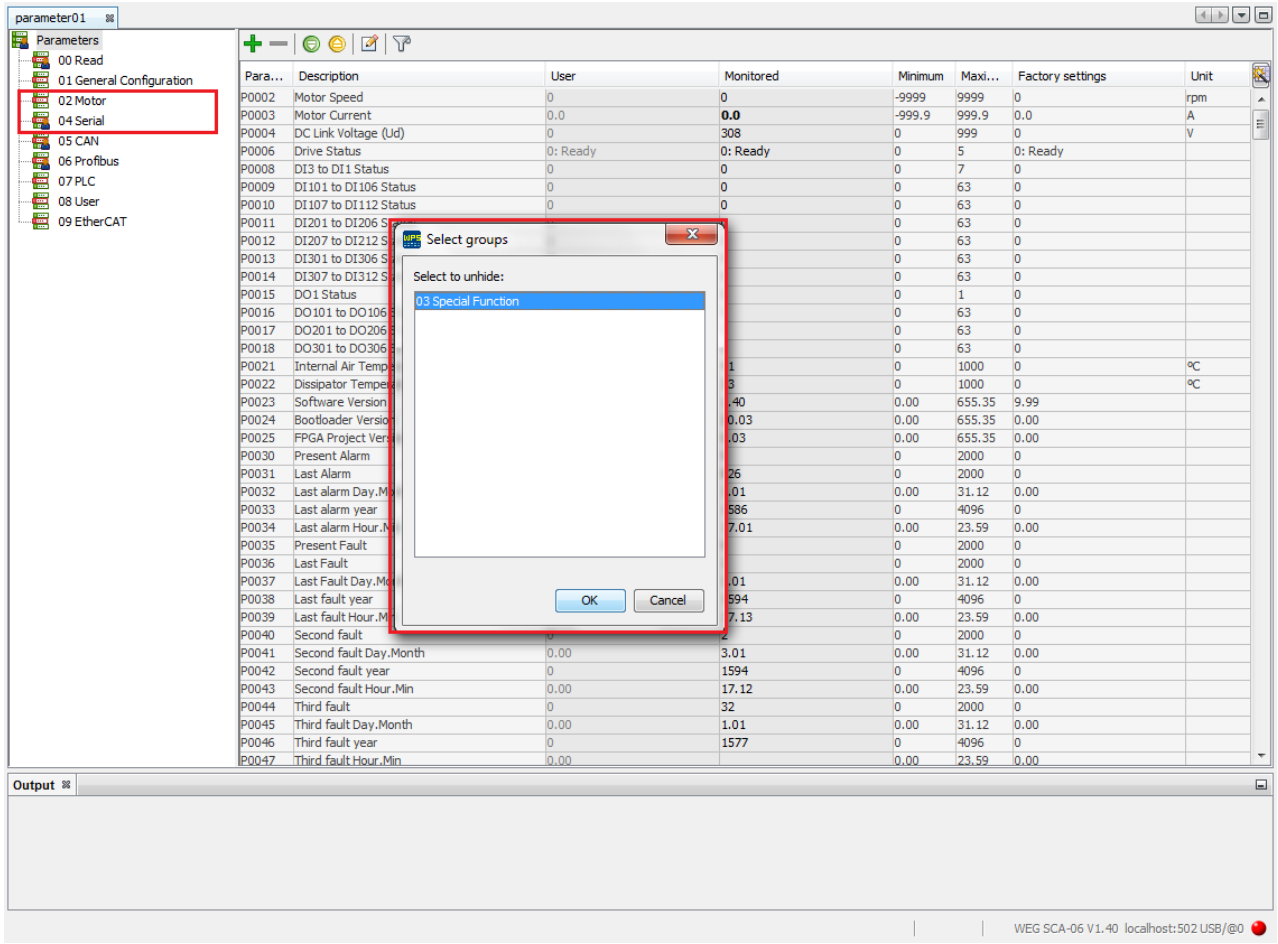

parameter01
Parameters
00 Read
01 General Configuration
02 Motor
03 Sp
04 Se
05 CA
06 Pro
07 PLC
08 Use
09 EtherCAT
Hide group
Unhide group
Write group
Read group
Param...
Description
User
Mo...
Minimum
Maximum
Factory settings
Unit
P0365 Encoder temperature 0.0 -3276.7 3276.7 0.0
P0368 Manufacture password 0 0 0 65535 0
P0369 Electrical position 0 50828 0 65535 0
P0370 Encoder temperature 0 -32767 32767 0
P0371 Read Encoder filter 3000 3000 3000 37500 3000
P0372 Encoder filter 3000 3000 3000 37500 3000
P0375 Encoder Offset 0 0 -32767 32767 0
P0490 Load Absolute position 0 0 0 1 0
P0492 User reference: fractions of revolution 0 0 -16383 16383 0
P0493 User reference: number of revolutions 0 0 -32768 32767 0
P0500 Count Mode: Standart Counter 0: Disabled 0: Di... 0 4 0: Disabled
P0502 Load counter 0 0 0 1 0
P0503 Counter Value - Low 0 0 0 65535 0
P0504 Counter Value - High 0 0 0 65535 0
P0506 Counter number of pulses/turns 1024 1024 1 65535 1024
P0507 Counter speed filter 500 500 1 4000 500 Hz
P0510 Count Mode: Counter 1 0: Disabled 0: Di... 0 3 0: Disabled
P0511 Null Pulse 1 options 0: Disabled 0: Di... 0 5 0: Disabled
P0512 Load Counter 1 value 0 0 0 1 0
P0513 Counter 1 value - Low 0 0 0 65535 0
P0514 Counter 1 value - High 0 0 0 65535 0
P0516 Counter 1 number of pulses/turns 1024 1024 1 65535 1024
P0517 Counter1 speed filter 500 500 1 4000 500 Hz
P0519 Error Enable 0: Disable 0: Di... 0 2 0: Disable
P0520 Count Mode: Counter 2 0: Disabled 0: Di... 0 3 0: Disabled
P0521 Null Pulse 2 options 0: Disabled 0: Di... 0 5 0: Disabled
P0522 Load Counter 2 value 0 0 0 1 0
P0523 Counter 2 value - Low 0 0 0 65535 0
P0524 Counter 2 value - High 0 0 0 65535 0
P0526 Counter 2 number of pulses/turns 1024 1024 1 65535 1024
P0527 Counter 2 speed filter 500 500 1 4000 500 Hz
P0529 Error Enable 0: Disabled 0: Di... 0 2 0: Disabled
P0531 Stop function:level or edge 0: stop activation by level 0: st... 0 1 0: stop activation by level
P0550 Trigger 1 Signal Source 0: Disabled 0: Di... 0 48 0: Disabled
P0551 Trigger 1 Value 0 0 -32768 32767 0
P0552 Trigger 1 Condition 0: Higher or equal than reference value 0: Hi... 0 1 0: Higher or equal than reference value
P0553 Trigger 2 Signal Source 0: Disable 0: Di... 0 48 0: Disable
P0554 Trigger 2 Value 0 -32768 32767 0
Output
WEG SCA-06 V1.40 localhost:502 USB/@0



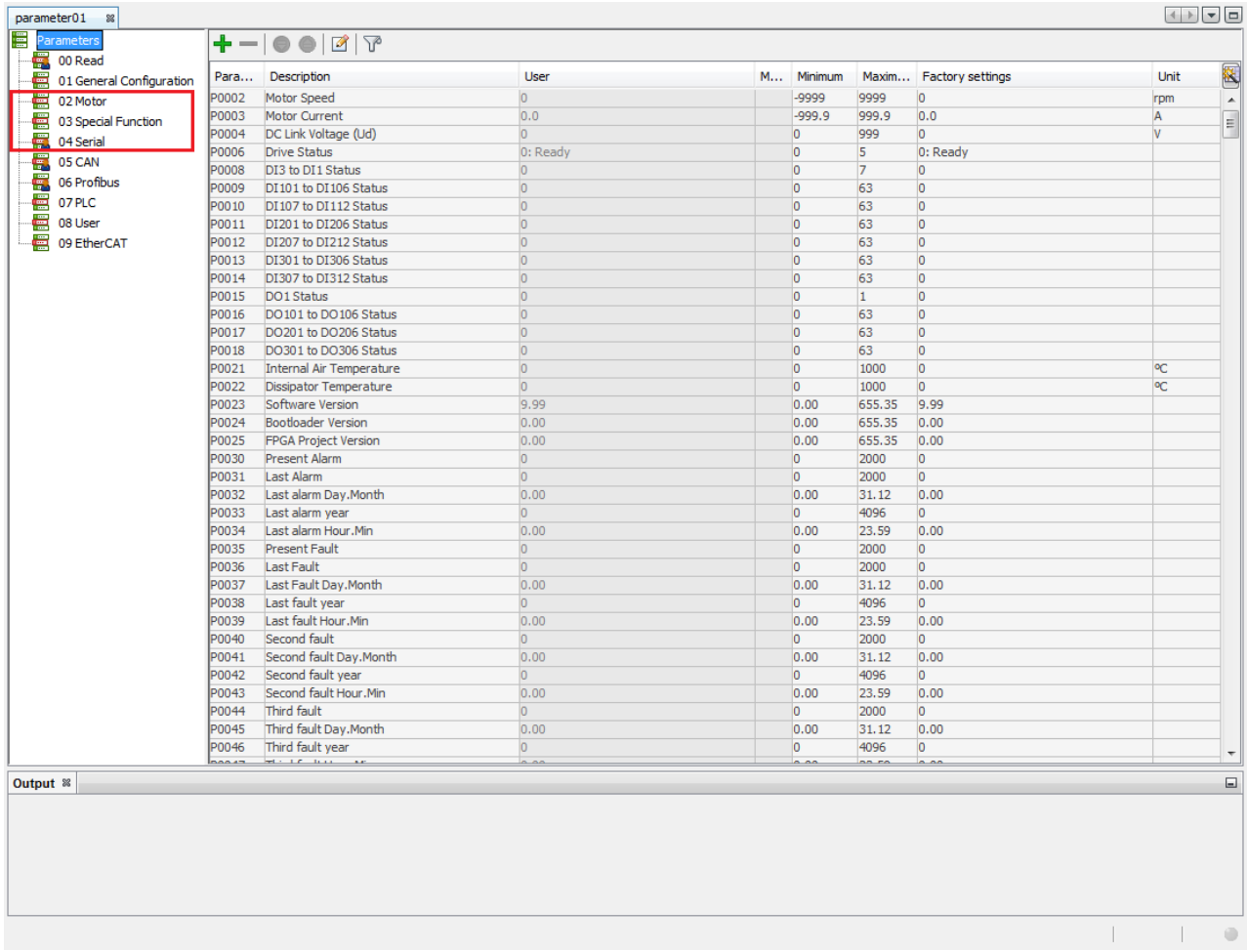
4. **Unhide Group of Parameters**. In order to show a hidden group of parameters, just select the root of the group tree and select the **Unhide Group** option. A window will open showing the groups that are hidden; then just select the group you wish to unhide.



Para...	Description	User	Monitored	Minimum	Maxi...	Factory settings	Unit
P0002	Motor Speed	0	0	-9999	9999	0	rpm
P0003	Motor Current	0.0	**0.0**	-999.9	999.9	0.0	A
P0004	DC Link Voltage (Ud)	0	307	0	999	0	V
P0006	Drive Status	0: Ready	0: Ready	0	5	0: Ready	
P0008	DI3 to DI1 Status	0	0	0	7	0	
P0009	DI101 to DI106 Status	0	0	0	63	0	
P0010	DI107 to DI112 Status	0	0	0	63	0	
P0011	DI201 to DI206 Status	0	0	0	63	0	
P0012	DI207 to DI212 Status	0	0	0	63	0	
P0013	DI301 to DI306 Status	0	0	0	63	0	
P0014	DI307 to DI312 Status	0	0	0	63	0	
P0015	DO1 Status	0	0	0	1	0	
P0016	DO101 to DO106 Status	0	0	0	63	0	
P0017	DO201 to DO206 Status	0	0	0	63	0	
P0018	DO301 to DO306 Status	0	0	0	63	0	
P0021	Internal Air Temperature	0	41	0	1000	0	°C
P0022	Dissipator Temperature	0	33	0	1000	0	°C
P0023	Software Version	9.99	1.40	0.00	655.35	9.99	
P0024	Bootloader Version	0.00	20.03	0.00	655.35	0.00	
P0025	FPGA Project Version	0.00	0.03	0.00	655.35	0.00	
P0030	Present Alarm	0	0	0	2000	0	
P0031	Last Alarm	0	826	0	2000	0	
P0032	Last alarm Day.Month	0.00	3.01	0.00	31.12	0.00	
P0033	Last alarm year	0	1586	0	4096	0	
P0034	Last alarm Hour.Min	0.00	17.01	0.00	23.59	0.00	
P0035	Present Fault	0	0	0	2000	0	
P0036	Last Fault	0	2	0	2000	0	
P0037	Last Fault Day.Month	0.00	3.01	0.00	31.12	0.00	
P0038	Last fault year	0	1594	0	4096	0	
P0039	Last fault Hour.Min	0.00	17.13	0.00	23.59	0.00	
P0040	Second fault	0	2	0	2000	0	
P0041	Second fault Day.Month	0.00	3.01	0.00	31.12	0.00	
P0042	Second fault year	0	1594	0	4096	0	
P0043	Second fault Hour.Min	0.00	17.12	0.00	23.59	0.00	
P0044	Third fault	0	32	0	2000	0	
P0045	Third fault Day.Month	0.00	1.01	0.00	31.12	0.00	
P0046	Third fault year	0	1577	0	4096	0	
P0047	Third fault Hour.Min	0.00		0.00	23.59	0.00	

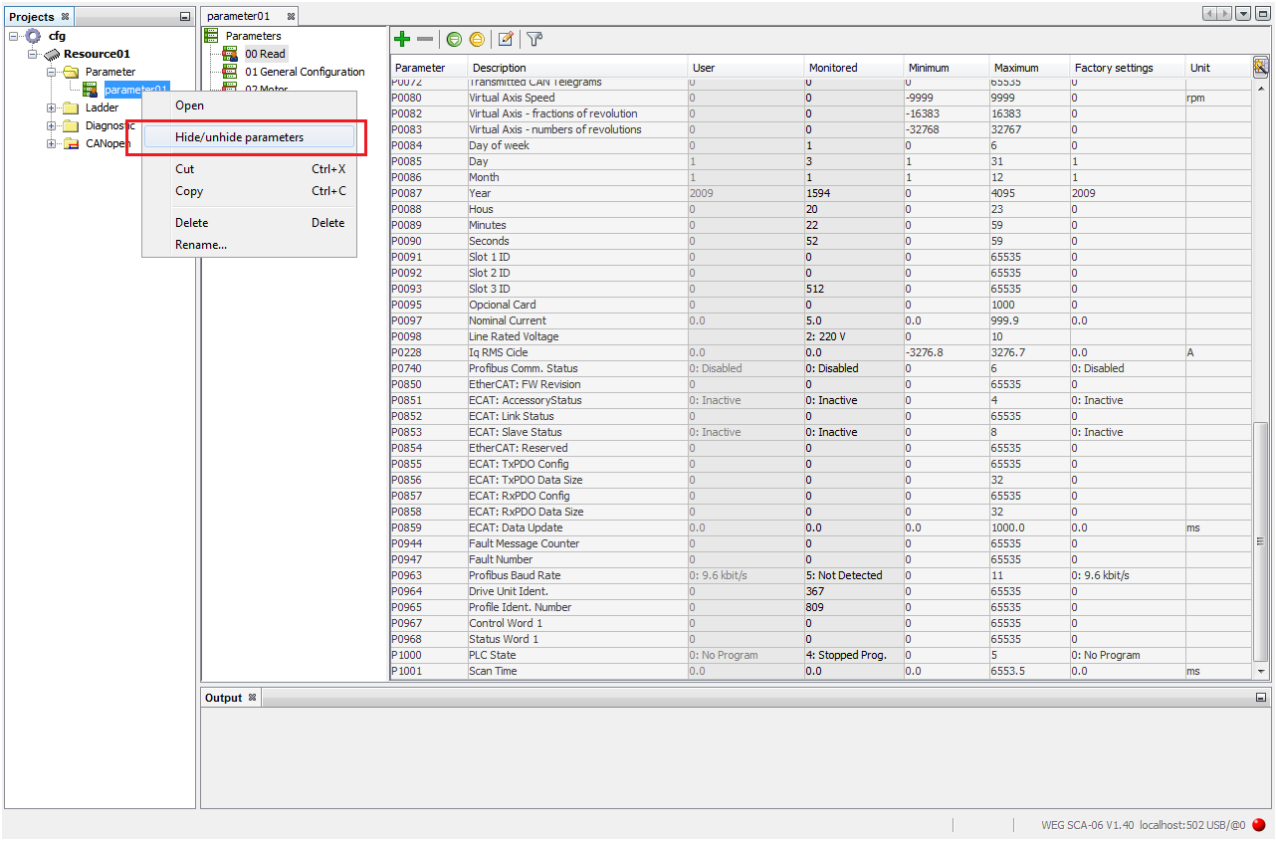


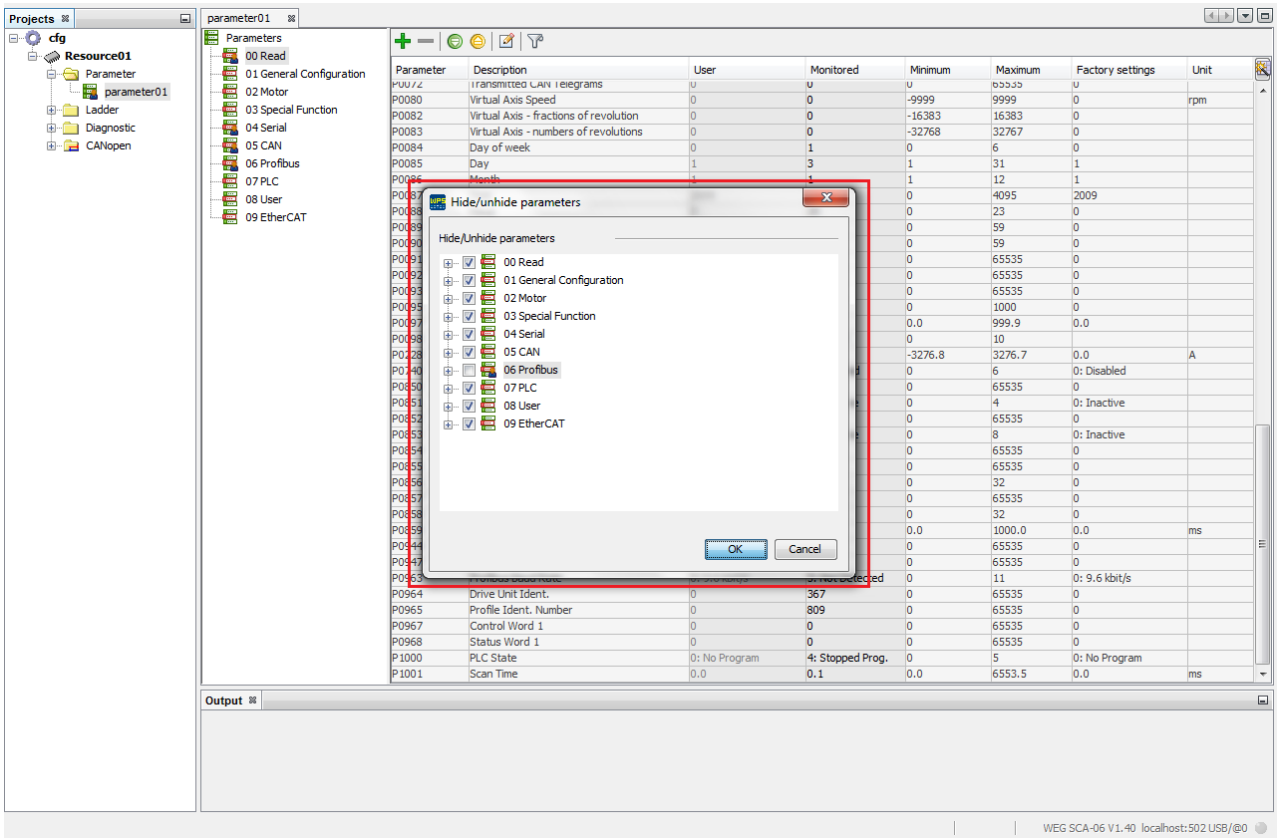
Para...	Description	User	Monitored	Minimum	Maxi...	Factory settings	Unit
P0002	Motor Speed	0	0	-9999	9999	0	rpm
P0003	Motor Current	0.0	**0.0**	-999.9	999.9	0.0	A
P0004	DC Link Voltage (Ud)	0	308	0	999	0	V
P0006	Drive Status	0: Ready	0: Ready	0	5	0: Ready	
P0008	DI3 to DI1 Status	0	0	0	7	0	
P0009	DI101 to DI106 Status	0	0	0	63	0	
P0010	DI107 to DI112 Status	0	0	0	63	0	
P0011	DI201 to DI206 St			0	63	0	
P0012	DI207 to DI212 St			0	63	0	
P0013	DI301 to DI306 St			0	63	0	
P0014	DI307 to DI312 St			0	63	0	
P0015	DO1 Status			0	1	0	
P0016	DO101 to DO106 S			0	63	0	
P0017	DO201 to DO206 S			0	63	0	
P0018	DO301 to DO306 S			0	63	0	
P0021	Internal Air Temp		1	0	1000	0	°C
P0022	Dissipator Temper		3	0	1000	0	°C
P0023	Software Version		.40	0.00	655.35	9.99	
P0024	Bootloader Versio		0.03	0.00	655.35	0.00	
P0025	FPGA Project Ver		.03	0.00	655.35	0.00	
P0030	Present Alarm			0	2000	0	
P0031	Last Alarm		26	0	2000	0	
P0032	Last alarm Day.M		.01	0.00	31.12	0.00	
P0033	Last alarm year		586	0	4096	0	
P0034	Last alarm Hour.M		7.01	0.00	23.59	0.00	
P0035	Present Fault			0	2000	0	
P0036	Last Fault			0	2000	0	
P0037	Last Fault Day.Mo		.01	0.00	31.12	0.00	
P0038	Last fault year		594	0	4096	0	
P0039	Last fault Hour.M		7.13	0.00	23.59	0.00	
P0040	Second fault		2	0	2000	0	
P0041	Second fault Day.Month	0.00	3.01	0.00	31.12	0.00	
P0042	Second fault year	0	1594	0	4096	0	
P0043	Second fault Hour.Min	0.00	17.12	0.00	23.59	0.00	
P0044	Third fault	0	32	0	2000	0	
P0045	Third fault Day.Month	0.00	1.01	0.00	31.12	0.00	
P0046	Third fault year	0	1577	0	4096	0	
P0047	Third fault Hour.Min	0.00		0.00	23.59	0.00	



Para...	Description	User	M...	Minimum	Maxim...	Factory settings	Unit
P0002	Motor Speed	0		-9999	9999	0	rpm
P0003	Motor Current	0.0		-999.9	999.9	0.0	A
P0004	DC Link Voltage (Ud)	0		0	999	0	V
P0006	Drive Status	0: Ready		0	5	0: Ready	
P0008	DI3 to DI1 Status	0		0	7	0	
P0009	DI101 to DI106 Status	0		0	63	0	
P0010	DI107 to DI112 Status	0		0	63	0	
P0011	DI201 to DI206 Status	0		0	63	0	
P0012	DI207 to DI212 Status	0		0	63	0	
P0013	DI301 to DI306 Status	0		0	63	0	
P0014	DI307 to DI312 Status	0		0	63	0	
P0015	DO1 Status	0		0	1	0	
P0016	DO101 to DO106 Status	0		0	63	0	
P0017	DO201 to DO206 Status	0		0	63	0	
P0018	DO301 to DO306 Status	0		0	63	0	
P0021	Internal Air Temperature	0		0	1000	0	°C
P0022	Dissipator Temperature	0		0	1000	0	°C
P0023	Software Version	9.99		0.00	655.35	9.99	
P0024	Bootloader Version	0.00		0.00	655.35	0.00	
P0025	FPGA Project Version	0.00		0.00	655.35	0.00	
P0030	Present Alarm	0		0	2000	0	
P0031	Last Alarm	0		0	2000	0	
P0032	Last alarm Day.Month	0.00		0.00	31.12	0.00	
P0033	Last alarm year	0		0	4096	0	
P0034	Last alarm Hour.Min	0.00		0.00	23.59	0.00	
P0035	Present Fault	0		0	2000	0	
P0036	Last Fault	0		0	2000	0	
P0037	Last Fault Day.Month	0.00		0.00	31.12	0.00	
P0038	Last fault year	0		0	4096	0	
P0039	Last fault Hour.Min	0.00		0.00	23.59	0.00	
P0040	Second fault	0		0	2000	0	
P0041	Second fault Day.Month	0.00		0.00	31.12	0.00	
P0042	Second fault year	0		0	4096	0	
P0043	Second fault Hour.Min	0.00		0.00	23.59	0.00	
P0044	Third fault	0		0	2000	0	
P0045	Third fault Day.Month	0.00		0.00	31.12	0.00	
P0046	Third fault year	0		0	4096	0	

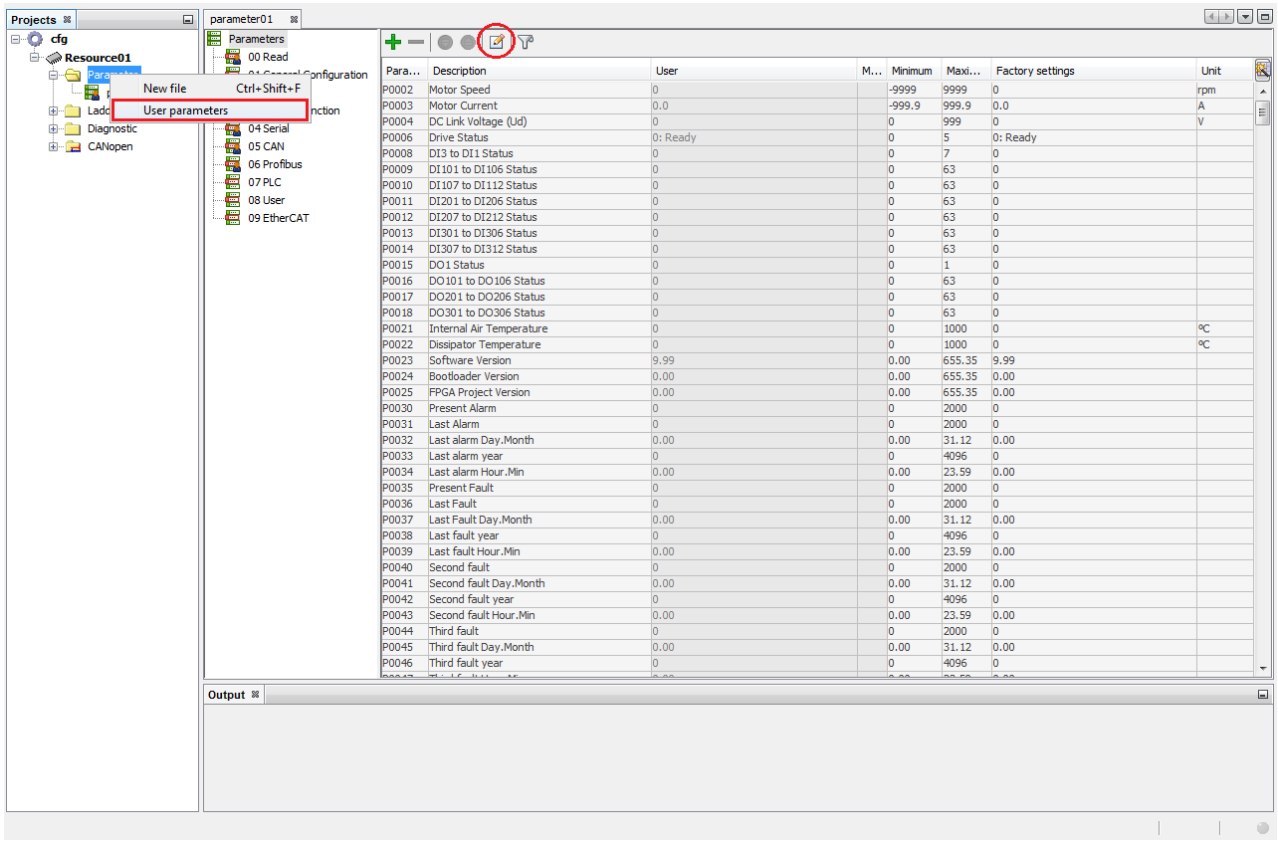
5. **Hide and Show Parameters and Groups of Parameters**. By means of this option, you have full control of the parameters and groups of parameters. It is possible to hide and unhide individual parameters, multiple parameters, individual groups and multiple groups in the same action.


Projects
cfg
Resource01
Parameter
parameter01
Ladder
Diagnostic
CANopen
Open
Hide/unhide parameters
Cut Ctrl+X
Copy Ctrl+C
Delete Delete
Rename...
parameter01
Parameters
00 Read
01 General Configuration
Parameter
Description
User
Monitored
Minimum
Maximum
Factory settings
Unit
P0080 Virtual Axis Speed
P0082 Virtual Axis - fractions of revolution
P0083 Virtual Axis - numbers of revolutions
P0084 Day of week
P0085 Day
P0086 Month
P0087 Year
P0088 Hous
P0089 Minutes
P0090 Seconds
P0091 Slot 1 ID
P0092 Slot 2 ID
P0093 Slot 3 ID
P0095 Opcional Card
P0097 Nominal Current
P0098 Line Rated Voltage
P0228 Iq RMS Cicle
P0740 Profibus Comm. Status
P0850 EtherCAT: FW Revision
P0851 ECAT: AccessoryStatus
P0852 ECAT: Link Status
P0853 ECAT: Slave Status
P0854 EtherCAT: Reserved
P0855 ECAT: TxPDO Config
P0856 ECAT: TxPDO Data Size
P0857 ECAT: RxPDO Config
P0858 ECAT: RxPDO Data Size
P0859 ECAT: Data Update
P0944 Fault Message Counter
P0947 Fault Number
P0963 Profibus Baud Rate
P0964 Drive Unit Ident.
P0965 Profile Ident. Number
P0967 Control Word 1
P0968 Status Word 1
P1000 PLC State
P1001 Scan Time
Output
WEG SCA-06 V1.40 localhost:502 USB/@0


Projects
cfg
Resource01
Parameter
parameter01
Ladder
Diagnostic
CANopen
parameter01
Parameters
00 Read
01 General Configuration
02 Motor
03 Special Function
04 Serial
05 CAN
06 Profibus
07 PLC
08 User
09 EtherCAT
Hide/unhide parameters
Hide/Unhide parameters
OK
Cancel
Output
WEG SCA-06 V1.40 localhost:502 USB/@0

11.10.6.5 User Parameters

In order to open the configuration screen of the user parameters, just click on the **User Parameters** option on the Parameter node of the project tree or click on the icon indicated on the tool bar of the parameter file.



Configuration Table.

On the user parameter configuration table, it is possible to define several attributes to the parameters, such as description, minimum and maximum values, unit, digits, data type, etc.

NOTE!
These settings will be automatically displayed in the parameter table. However, to be sent to the device, you need to *download* the resource.

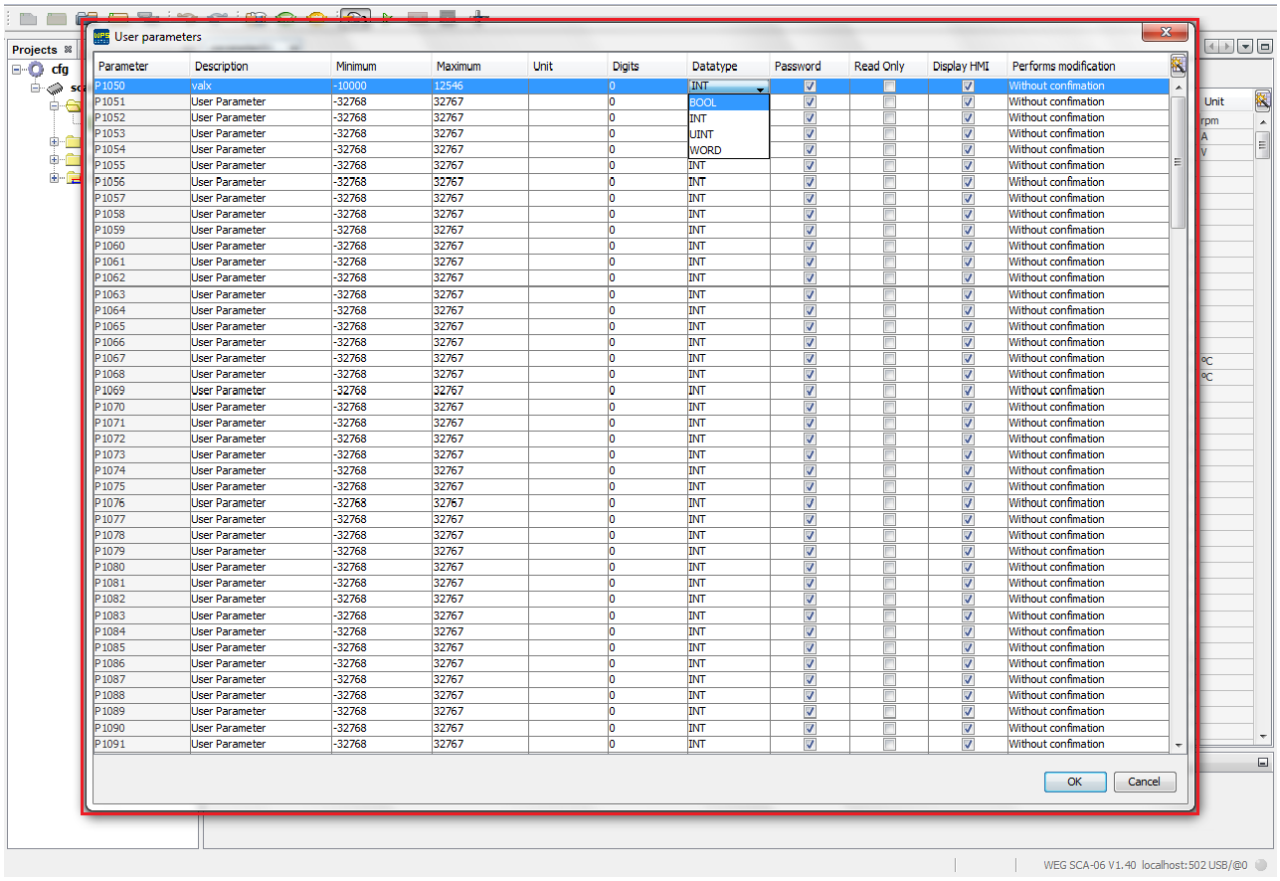


Table fields:

- Parameter: User parameter identification.
- Description: Description of the user parameter in the parameter table. On devices that have text-based HMIs, the description is sent to the machine and displayed on the HMI.
- Minimum: Minimum input value for parameter.
- Maximum: Maximum input value for parameter.
- Unit: Unit displayed on the device's HMI.
- Default: Value loaded when restore factory default is selected.
- Retentive: Retain value after rebooting devices.
- Hexadecimal: Displays the value in hexadecimal.
- Digits: Number of decimal digits for displaying value.
- Datatype: Parameter datatype used by the ladder application.
- Password: Enables password request by changing parameter value.

- Read only: It does not allow the writing of values in the parameter by the communication network or the HMI. Writing is done only by the ladder application.

- Display HMI: Displays the parameter in the HMI.

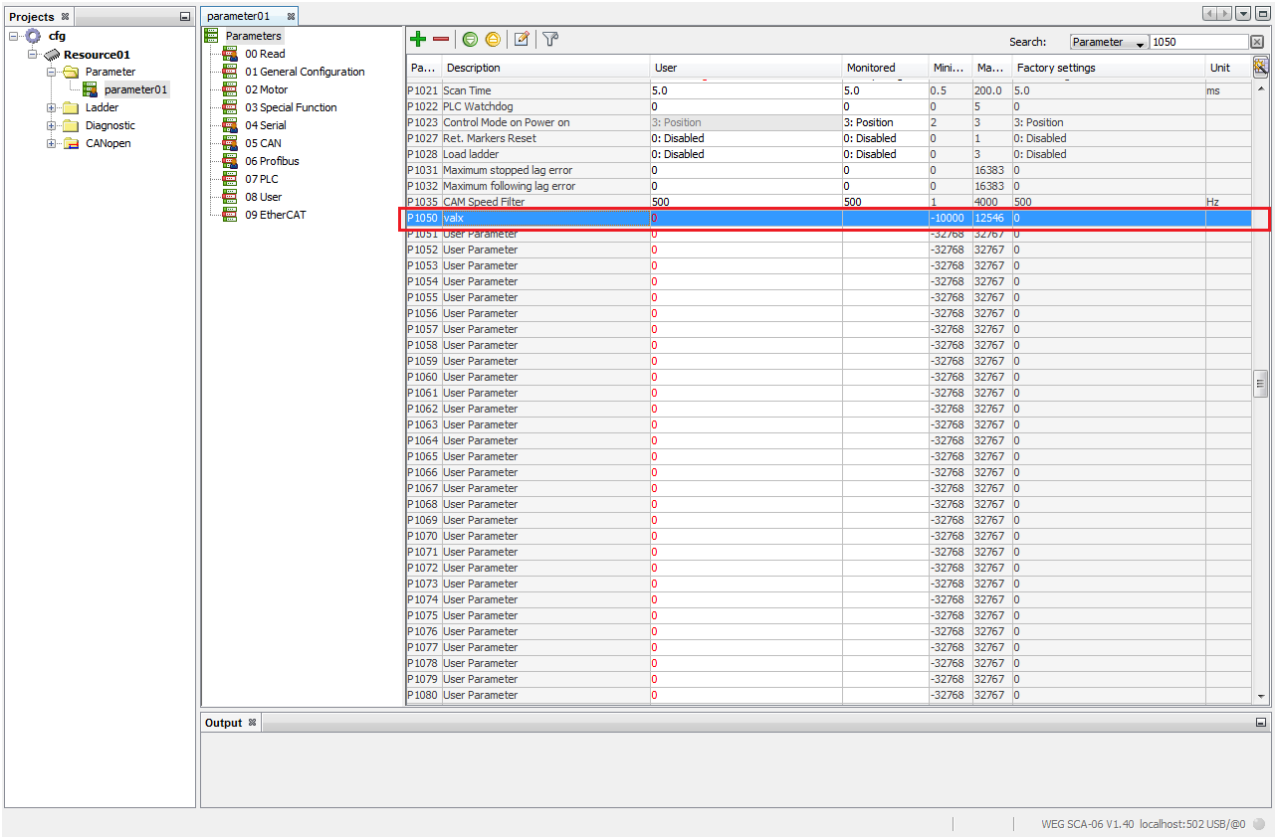
- Performs modification: Confirmation options when changing the parameter:
o No confirmation: Does not prompt for confirmation when changing parameter.
o With confirmation and engine stopped: Request confirmation and allow change only with engine stopped.
o With confirmation: Prompt for confirmation when changing parameter.

- Stopped motor: Perform change only with motor stopped.

- Help: On devices that have text-based HMI, you can edit a help text for the parameter.

View the user parameter

In the parameter table, the user parameters will be shown as they are configured on the configuration screen.



Pa...	Description	User	Monitored	Mini...	Ma...	Factory settings	Unit
P1021	Scan Time	5.0	5.0	0.5	200.0	5.0	ms
P1022	PLC Watchdog	0	0	0	5	0	
P1023	Control Mode on Power on	3: Position	3: Position	2	3	3: Position	
P1027	Ret. Markers Reset	0: Disabled	0: Disabled	0	1	0: Disabled	
P1028	Load ladder	0: Disabled	0: Disabled	0	3	0: Disabled	
P1031	Maximum stopped lag error	0	0	0	16383	0	
P1032	Maximum following lag error	0	0	0	16383	0	
P1035	CAM Speed Filter	500	500	1	4000	500	Hz
P1050	valx	0		-10000	12546	0	
P1051	User Parameter	0		-32768	32767	0	
P1052	User Parameter	0		-32768	32767	0	
P1053	User Parameter	0		-32768	32767	0	
P1054	User Parameter	0		-32768	32767	0	
P1055	User Parameter	0		-32768	32767	0	
P1056	User Parameter	0		-32768	32767	0	
P1057	User Parameter	0		-32768	32767	0	
P1058	User Parameter	0		-32768	32767	0	
P1059	User Parameter	0		-32768	32767	0	
P1060	User Parameter	0		-32768	32767	0	
P1061	User Parameter	0		-32768	32767	0	
P1062	User Parameter	0		-32768	32767	0	
P1063	User Parameter	0		-32768	32767	0	
P1064	User Parameter	0		-32768	32767	0	
P1065	User Parameter	0		-32768	32767	0	
P1066	User Parameter	0		-32768	32767	0	
P1067	User Parameter	0		-32768	32767	0	
P1068	User Parameter	0		-32768	32767	0	
P1069	User Parameter	0		-32768	32767	0	
P1070	User Parameter	0		-32768	32767	0	
P1071	User Parameter	0		-32768	32767	0	
P1072	User Parameter	0		-32768	32767	0	
P1073	User Parameter	0		-32768	32767	0	
P1074	User Parameter	0		-32768	32767	0	
P1075	User Parameter	0		-32768	32767	0	
P1076	User Parameter	0		-32768	32767	0	
P1077	User Parameter	0		-32768	32767	0	
P1078	User Parameter	0		-32768	32767	0	
P1079	User Parameter	0		-32768	32767	0	
P1080	User Parameter	0		-32768	32767	0	

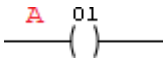
11.10.7Ladder

11.10.7.1 Coil

11.10.7.1.1 DIRECTCOIL

Logical block used to assign direct values of the output variables.

Ladder Representation



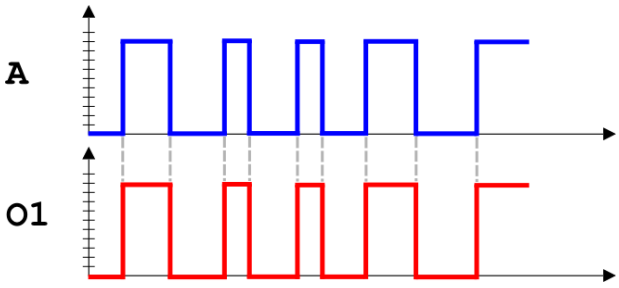
Block Structure

Variable Type	Name	Data Type	Description
VAR_OUTPUT	O1	BOOL	Block log output

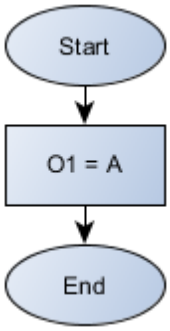
Operation

The block transfers the value of A for the memory address corresponding to O1.

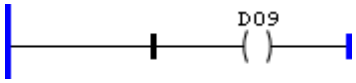
Diagram



Block Flowchart



Example

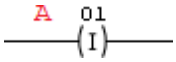


The above example keeps the digital output DO9 permanently connected, because the value of A in this case is the value of the left bus which is always considered high logic level (TRUE).

11.10.7.1.2 IMMEDIATECOIL

Logical block used for assigning values to standard digital outputs instantly.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_OUTPUT	O1	BOOL	Block log output

Operation

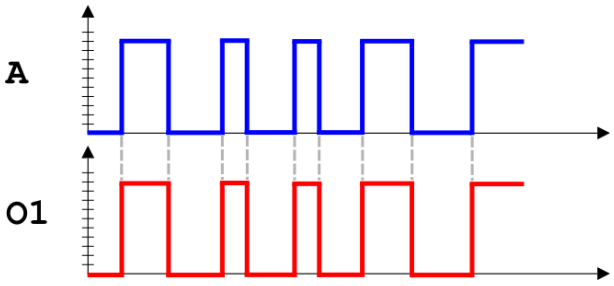
The block transfers the value of A for the digital output corresponding to O1.
Unlike the direct coil, this block does not wait until the end of the scan cycle so that the output value is updated; this is done at the same time the block is activated.

NOTE!
This block only works with standard digital outputs of the product.

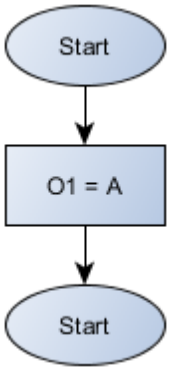
Compatibility

Device	Version
PLC300	1.20 or higher
SCA06	2.00 or higher

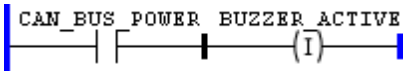
Diagram



Block Flowchart



Example

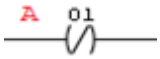


The above example immediately activates the internal buzzer when it detects that the power of the CANopen bus was stopped and remains on until the power is restored.

11.10.7.1.3 INVERTEDCOIL

Logical block used for assigning values denied to output variables.

Ladder Representation



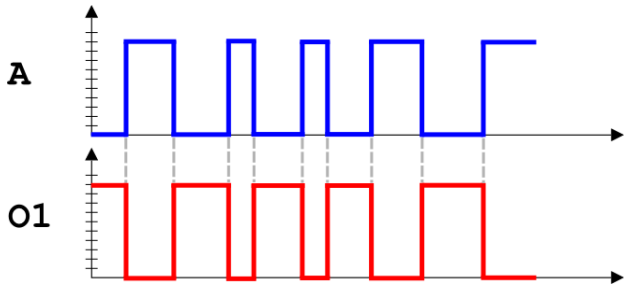
Block Structure

Variable Type	Name	Data Type	Description
VAR_OUTPUT	O1	BOOL	Block log output

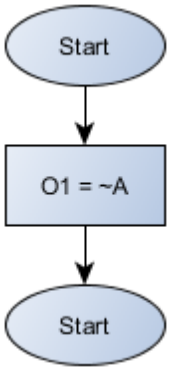
Operation

The block transfers the denied value of A for the memory address corresponding to O1.

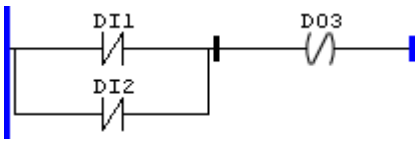
Diagram



Block Flowchart



Example

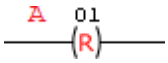


The above example disables the digital output DO3 when some of the digital inputs DI1 and DI2 are with FALSE value. When both inputs are with a TRUE value, DO3 activates.

11.10.7.1.4 RESETCOIL

Logical block used for indefinite disabling of output variables.

Ladder Representation



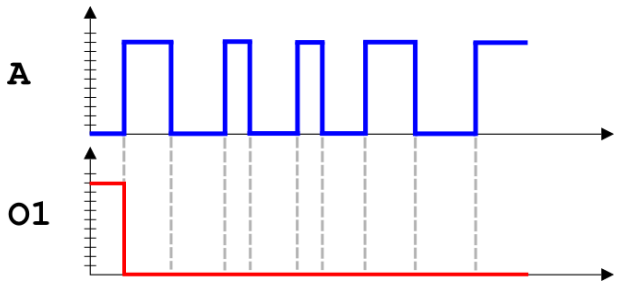
Block Structure

Variable Type	Name	Data Type	Description
VAR_OUTPUT	O1	BOOL	Block log output

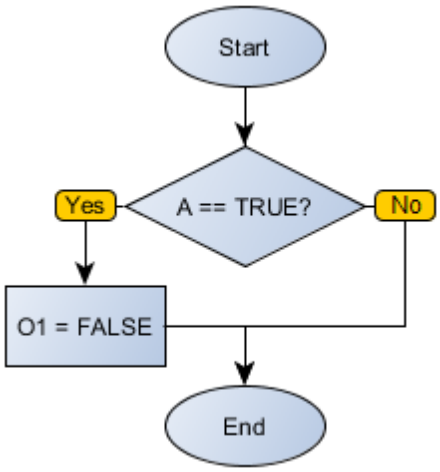
Operation

When identifying a TRUE value in A, this block transfers a FALSE value to the memory address corresponding to O1.
When identifying a FALSE value in A, this block performs no operation.

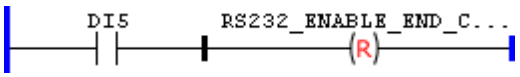
Diagram



Block Flowchart



Example

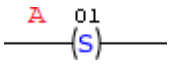


The example above activates permanently the system control marker that enables end-of-message character in RS232 communication to identify a TRUE level at the digital input DI5.

11.10.7.1.5 SETCOIL

Logical block used for indefinite enabling of output variables.

Ladder Representation



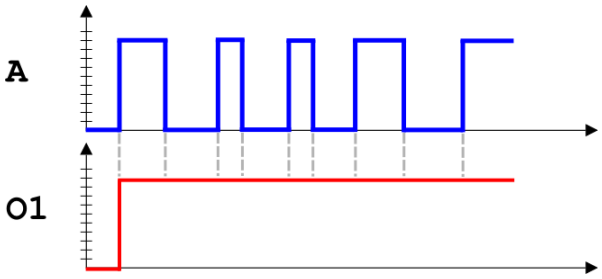
Block Structure

Variable Type	Name	Data Type	Description
VAR_OUTPUT	O1	BOOL	Block log output

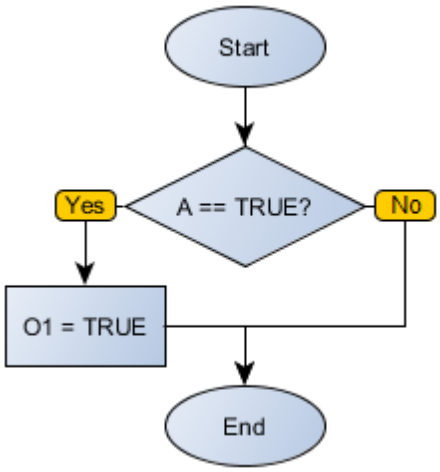
Operation

When identifying a TRUE value in A, this block transfers the value of A for the memory address corresponding to O1.
When identifying a FALSE value in A, this block performs no operation.

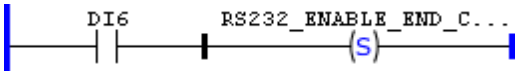
Diagram



Block Flowchart



Example

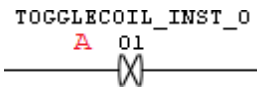


The example above activates permanently the system control marker that enables end-of-message character in RS232 communication to identify a TRUE level at the digital input DI6.

11.10.7.1.6 TOGGLECOIL

Logical block used for output variables alternance.

Ladder Representation



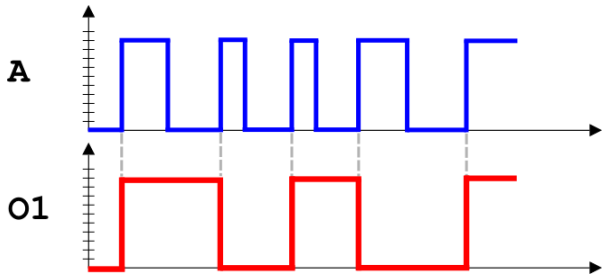
Block Structure

Variable Type	Name	Data Type	Description
VAR_OUTPUT	O1	BOOL	Block log output
VAR	TOGGLECOIL_INST_0	TOGGLECOIL	Instance of access to block structure

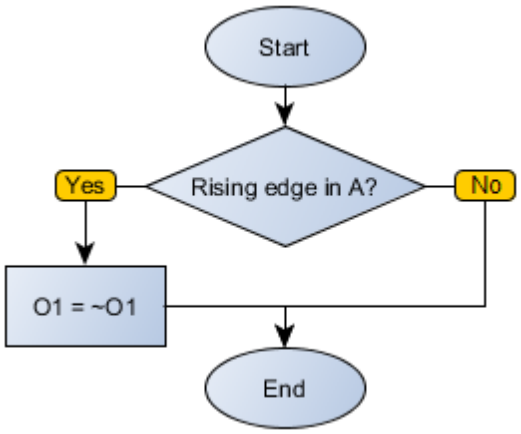
Operation

When identifying a transition from FALSE to TRUE (leading edge) on A, the block reverses the status of O1.

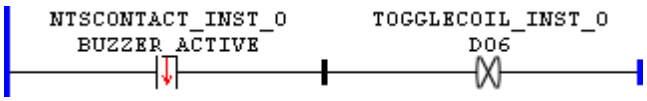
Diagram



Block Flowchart



Example



The above example inverts the state of the digital output DO6 to each disabling the internal buzzer.

11.10.7.2 Communication Network

11.10.7.2.1 CANopen

11.10.7.2.1.1 CANopen Overview

Operation in the CANopen Network - Master Mode

Besides the operation as a slave, the PLC300 programmable controller also allows operation as a master for the CANopen network. PLC300 characteristics and functions as a master for the CANopen network will be described as follows.

Enabling Function CANopen Master

By default, PLC300 programmable controller is programmed to operate as a slave for the CANopen network. Programming the equipment as a master for the network must be done by using the WSCAN software that also allows the configuration of the whole CANopen network. A detailed description of the WSCAN software windows and functions must be obtained in the menu "Help" in the software itself.

After the master configuration has been created, it is necessary to download the configurations by using one of the product's programming interfaces - refer to the user manual for further information. Once programmed as a master for the network, in case it is necessary to delete said configurations, the function to delete the user program - available in the Setup menu - also deletes the CANopen master configurations.

NOTE!
CANopen network is a flexible network that allows several different ways of configuring and operating. Nonetheless, such a flexibility requires the user to have a good knowledge of the communication functions and objects used to configure the network, as well as knowledge of the WSCAN programming software.

CANopen Master Characteristics

PLC300 programmable controller allows the control of a group of up to 63 slaves, using the following communication tasks and resources:

- Network management task (NMT)
- 63 transmission PDOs
- 63 reception PDOs
- 63 Heartbeat Consumers
- Heartbeat Producer
- SDO Client
- SYNC producer/consumer
- 512 bytes of input network markers
- 512 bytes of output network markers

Physical characteristics - installation, connector, cable, etc. - are the same for PLC300 operating both as a master and a slave. Address configurations and communication rate are also necessary for the operation as a master, but these configurations are programmed by the WSCAN software according to the properties defined for the master in the software itself.

NOTE!
Input network markers are used to map data in RPDOs, while output network markers are used to map data in TPDOs. They can be accessed in Byte (%IB or %QB), Word (%IW or %QW), or Double Word (%ID or %QD). Nonetheless, their function is not pre-defined and depends on the Ladder application developed for the PLC300 controller.

Master Operation

Once programmed to operate as a master, the PLC300 programmable controller will perform the following steps in order to perform the sequential initialization for each one of the slaves.

1. By sending the communication reset command to the whole network, so that the slaves initialize with values known for the communication objects.
2. Equipment identification in the network, through the reading via SDO of the 1000h/00h object - Object Identification.
3. Writing via SDO of all objects programmed for the slave, which usually include the configuration and mapping of TPDOs and RPDOs, node guarding, heartbeat, besides the manufacturer's specific objects, in case they are programmed.
4. Error control task initialized - node guarding or heartbeat - in case they are programmed.
5. Sending of the slave to the operating mode.

If one of these steps fails, communication error with the slave will be indicated. Depending on the configurations, the initialization of slaves will be aborted, and the master will initialize the following slave, returning to the slave presenting error, after trying to initialize all other slaves in the network.

Similarly, if during the operation of a slave, an error in the error control task is identified, depending on the configurations made for the master, the slave will be automatically reset and the initialization procedure will be performed over.

NOTE!
The communication state and the state of each slave can be observed in input system markers.

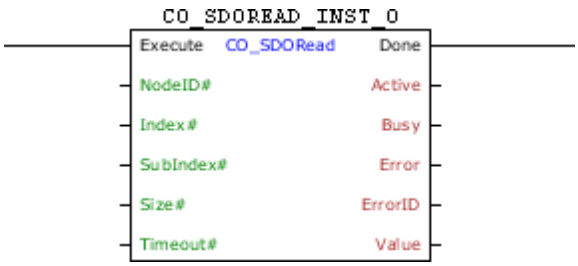
Blocks for the CANopen Master

Besides the communication and configuration objects made in the WSCAN software, blocks for the monitoring and sending of commands, which can be used during the creation of the application in Ladder for the PLC300 programmable controller, are also available. It is not necessary to use said blocks during the equipment operation, but its use provides more flexibility and facilitates the diagnosis of communication problems during the PLC300 programmable controller operation.

11.10.7.2.1.2 CO_SDORead

Block that performs a reading of data via SDO from a remote slave in CANopen network.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	Execute	BOOL	Block enabling
	NodeID#	BYTE	Slave address
	Index#	WORD	Index of the object to be accessed in slave
	SubIndex#	BYTE	Sub-index of the object
	Size#	BYTE	Size of data accessed, in bytes
	Timeout#	WORD	Maximum w aiting time for arrival of data, from the beginning of the request [ms]
VAR_OUTPUT	Done	BOOL	Output enabling
	Active	BOOL	Aw aiting response flag
	Busy	BOOL	Flag of the SDO client is busy w ith another request
	Error	BOOL	Error in the execution flag
	ErrorID	BYTE USINT	Identifier of the occurred error
	Value	BYTE USINT	Variable that stores the received data
VAR	CO_SDOREAD_INST_0	CO_SDOREAD	Instance of access to block structure

Operation

When this block detects a leading edge on Execute it checks whether the SDO client in the specified NodeID # address is free to send data (Busy variable at FALSE level). If so, it sends the reading request to the object of Size# size located in Index# and SubIndex# and sets the Active output, resetting it when receiving the response from the slave. The received data is stored in the Value variable. If the slave is not free, the block waits Busy go to FALSE level to resubmit the request.

NOTE!
If Execute goes to FALSE level and Busy is still at TRUE level, the request is canceled.

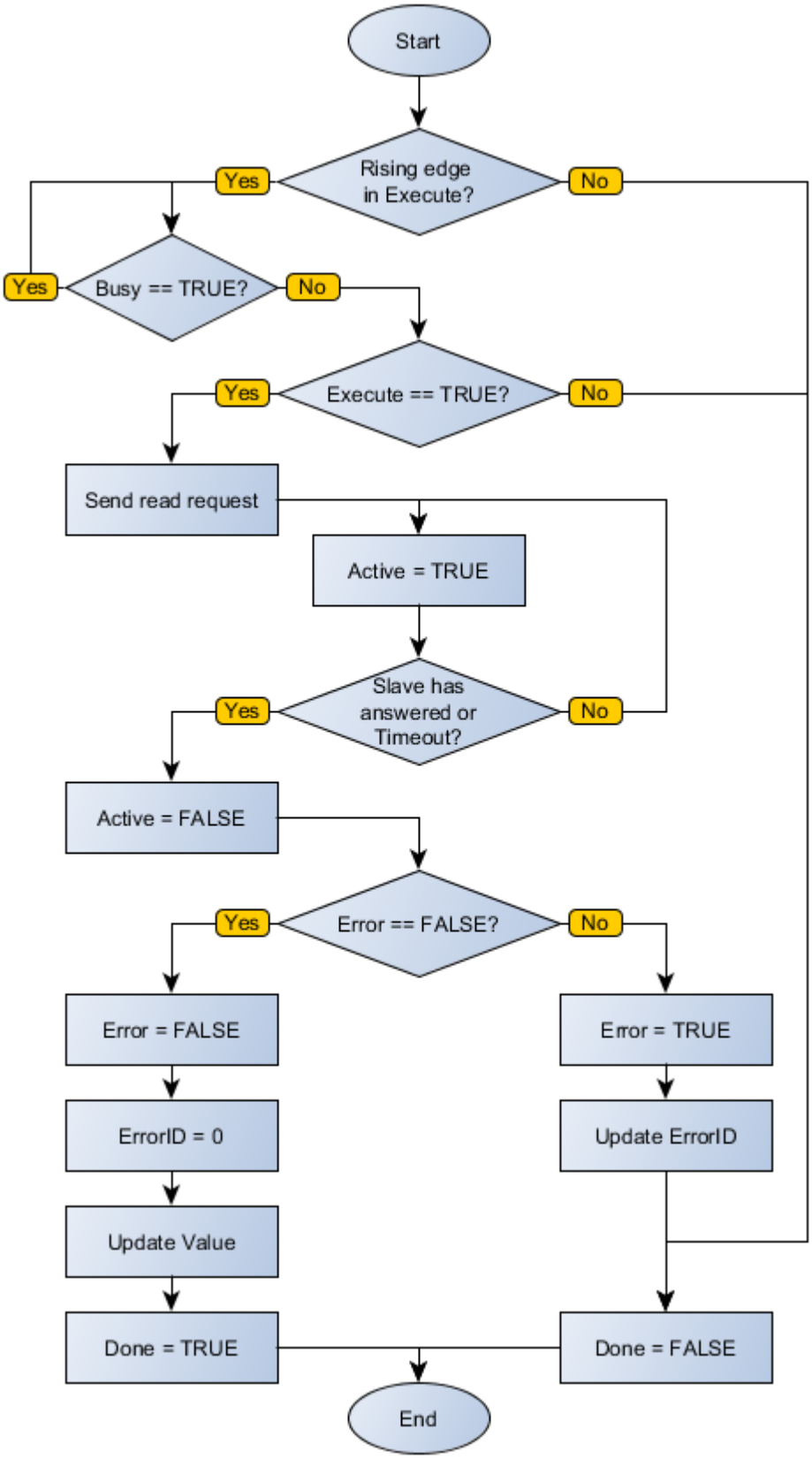
NOTE!
Value is an array of size equal to Size#. It is important to check this compatibility not to generate errors in the block.

When Execute has FALSE value, Done remains FALSE. The Done output is only activated when the block finishes executing successfully, remaining at TRUE level until Execute receives FALSE.

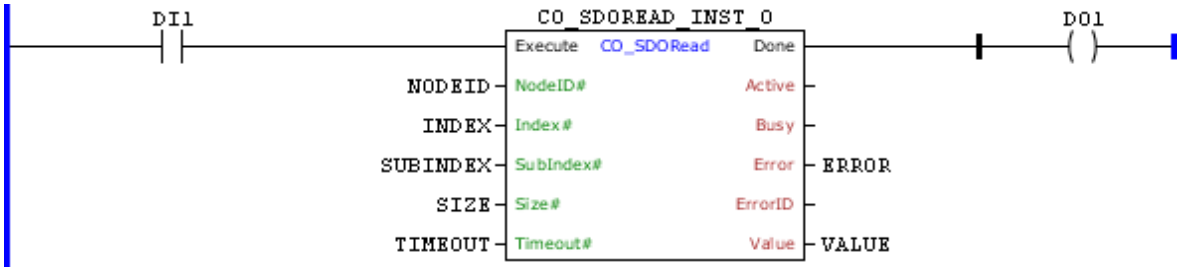
If there is any error in the execution, the Error output is enabled and ErrorID displays an error code according to the table below.

Code	Description
0	Executed successfully
1	Card cannot execute the function
2	Timeout in slave response
3	Slave returned error

Block Flowchart


Start
Rising edge in Execute?
Yes
No
Busy == TRUE?
Yes
No
Execute == TRUE?
Yes
No
Send read request
Active = TRUE
Slave has answered or Timeout?
Yes
No
Active = FALSE
Error == FALSE?
Yes
No
Error = FALSE
Error = TRUE
ErrorID = 0
Update ErrorID
Update Value
Done = TRUE
Done = FALSE
End

Example

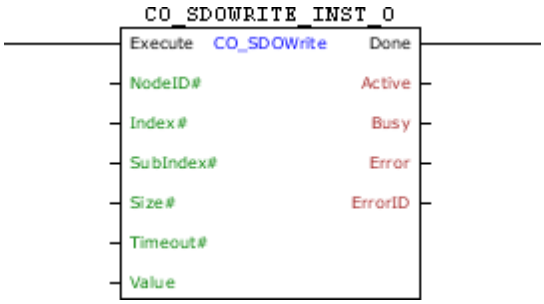


The example above requests reading of the data size SIZE, located in INDEX - SUBINDEX, of the NODEID device. This data is forwarded to VALUE. The block ends successfully, Done output is activated.

11.10.7.2.1.3 CO_SDOWrite

Block that performs a writing of data via SDO from a remote slave in CANopen network.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	Execute	BOOL	Block enabling
	NodeID#	BYTE	Slave address
	Index#	WORD	Index of the object to be accessed in slave
	SubIndex#	BYTE	Sub-index of the object
	Size#	BYTE	Size of data accessed, in bytes
	Timeout#	WORD	Maximum waiting time for arrival of data, from the beginning of the request [ms]
	Value	BYTE USINT	Variable that has the data to be written
VAR_OUTPUT	Done	BOOL	Output enabling
	Active	BOOL	Awaiting response flag
	Busy	BOOL	Flag of the SDO client is busy with another request
	Error	BOOL	Error in the execution flag
	ErrorID	BYTE USINT	Identifier of the occurred error
VAR	CO_SDOWRITE_INST_0	CO_SDOWRITE	Instance of access to block structure

Operation

When this block detects a leading edge on Execute it checks whether the SDO client in the specified NodeID # address is free to send data (Busy variable at FALSE level). If so, it sends the writing request to the object of Size# size located in Index# and SubIndex# and sets the Active output, resetting it when receiving the response from the slave. If the slave is not free, the block waits Busy go to FALSE level to resubmit the request.

NOTE!
If Execute goes to FALSE level and Busy is still at TRUE level, the request is canceled.

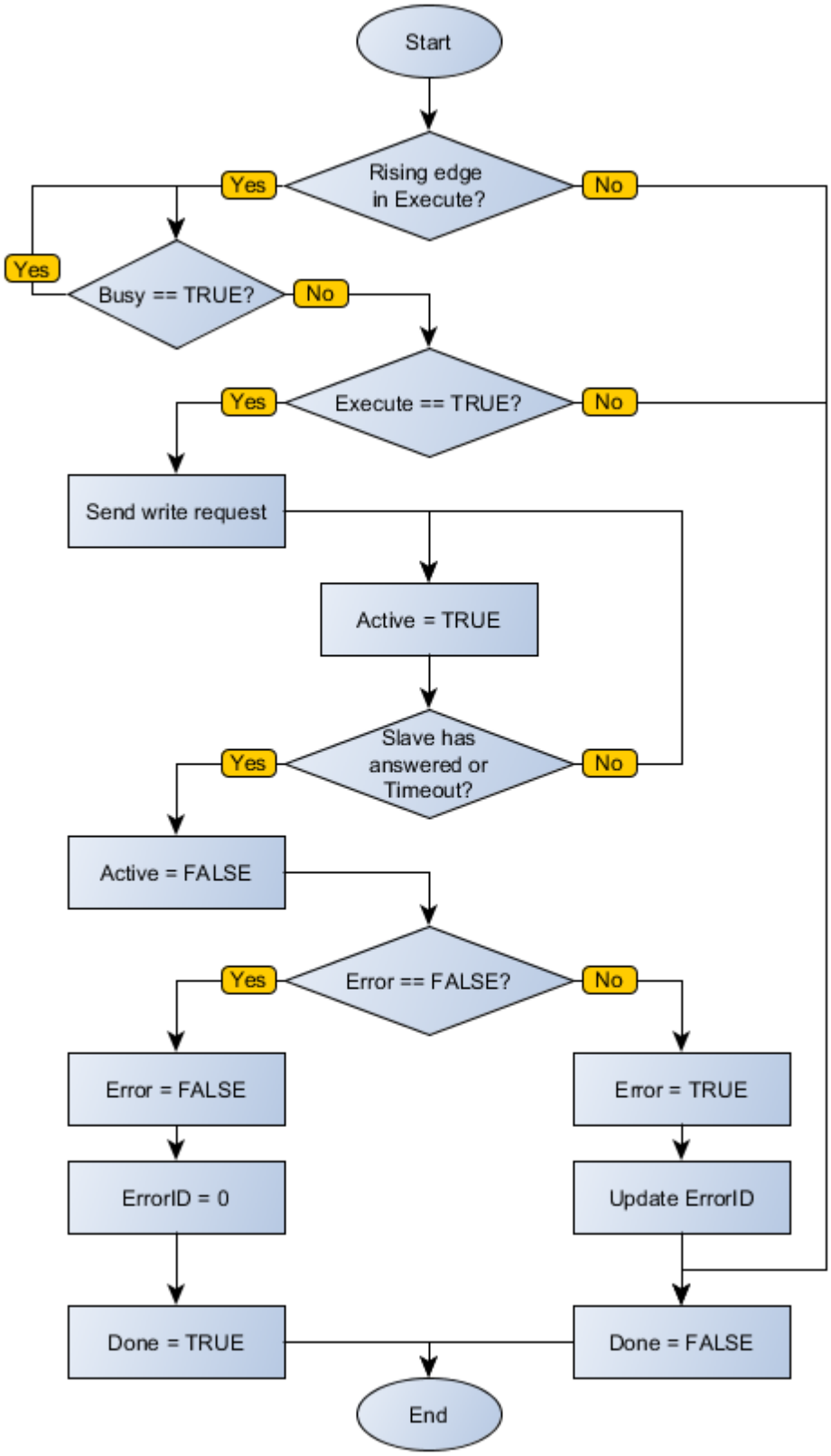
NOTE!
Value is an array of size equal to Size#. It is important to check this compatibility not to generate errors in the block.

When Execute has FALSE value, Done remains FALSE. The Done output is only activated when the block finishes executing successfully, remaining at TRUE level until Execute receives FALSE.

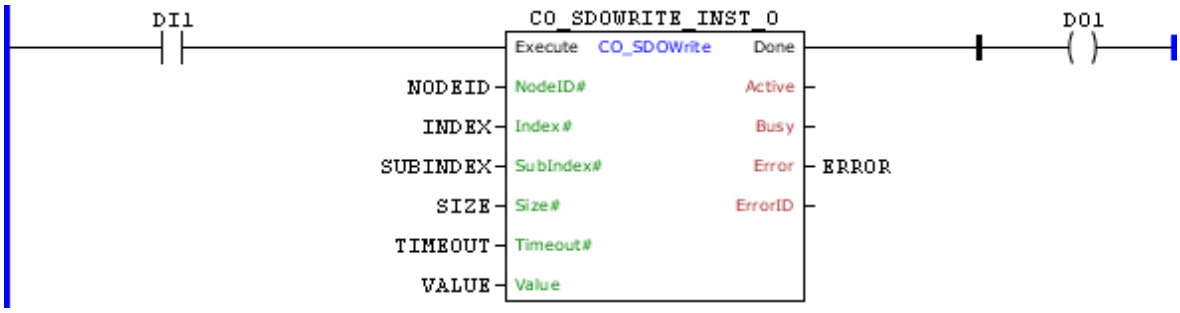
If there is any error in the execution, the Error output is enabled and ErrorID displays an error code according to the table below.

Code	Description
0	Executed successfully
1	Card cannot execute the function
2	Timeout in slave response
3	Slave returned error

Block Flowchart


Start
Rising edge in Execute?
Yes
No
Yes
Busy == TRUE?
No
Yes
Execute == TRUE?
No
Send write request
Active = TRUE
Slave has answered or Timeout?
Yes
No
Active = FALSE
Error == FALSE?
Yes
No
Error = FALSE
Error = TRUE
ErrorID = 0
Update ErrorID
Done = TRUE
Done = FALSE
End

Example



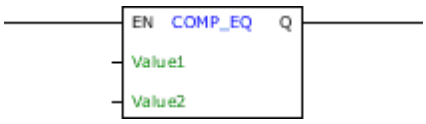
The example above requests writing of the data size VALUE, located in INDEX - SUBINDEX, of the NODEID device. The block ends successfully, Done output is activated.

11.10.7.3 Compare

11.10.7.3.1 COMPEQ

Block that compares the values of Value1 and Value2, enabling the output Q if both are equal.

Ladder Representation



Block Structure

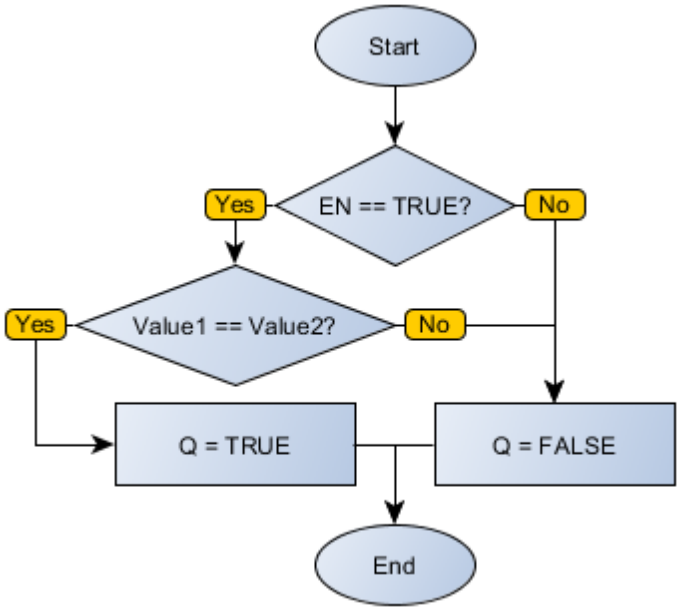
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	First value of comparison
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Second value of comparison
VAR_OUTPUT	Q	BOOL	Indicator of equality

Operation

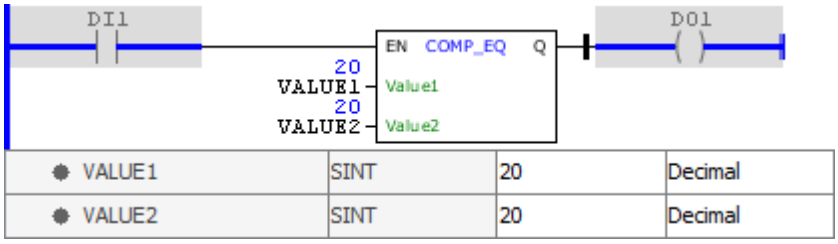
When this block has a TRUE value in EN, it sends to the output Q the TRUE value if Value1 and Value2 are the same. Otherwise, Q receives FALSE.

When EN has FALSE value, Q remains in FALSE.

Block Flowchart

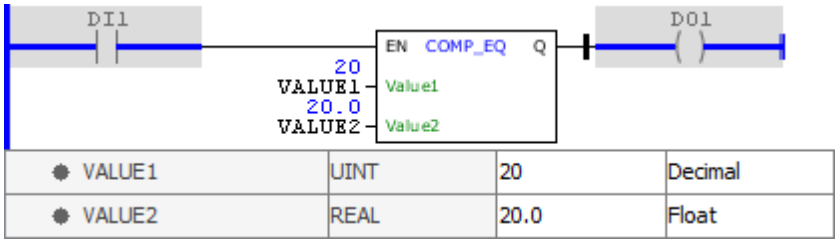


Example



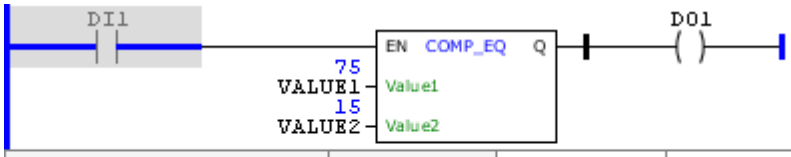
VALUE1	SINT	20	Decimal
VALUE2	SINT	20	Decimal

The example above checks equality between VALUE1 and VALUE2. Since both variables have the same value, the Q output is activated.



VALUE1	UINT	20	Decimal
VALUE2	REAL	20.0	Float

The example above checks equality between VALUE1 and VALUE2. Since both variables have the same value, the Q output is activated. Notice that the types of the input variables can be different without causing execution problems.



VALUE1	UINT	75	Decimal
VALUE2	UINT	15	Decimal

The example above checks equality between VALUE1 and VALUE2. Since both variables have different values, the Q output is disabled.

11.10.7.3.2 COMPGE

Block that compares the values of Value1 and Value2, enabling the output Q if Value1 is higher than or equal to Value2.

Ladder Representation

Block Structure

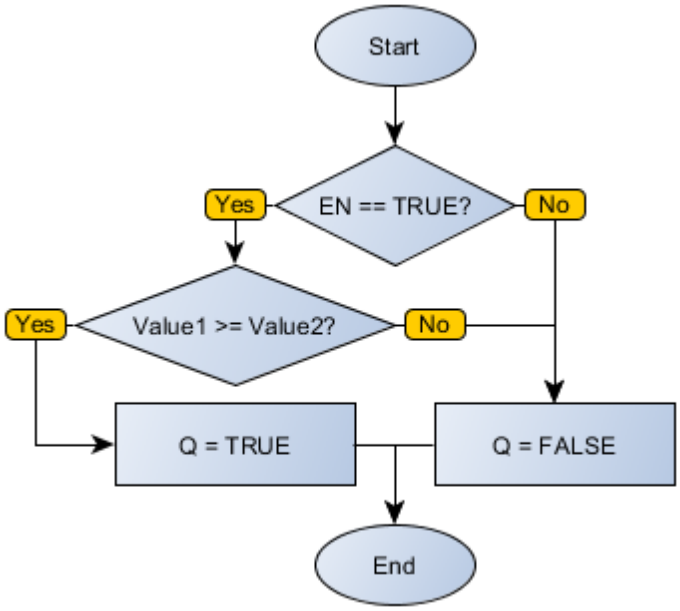
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	First value of comparison
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Second value of comparison
VAR_OUTPUT	Q	BOOL	Indicator of equality or majority of Value1

Operation

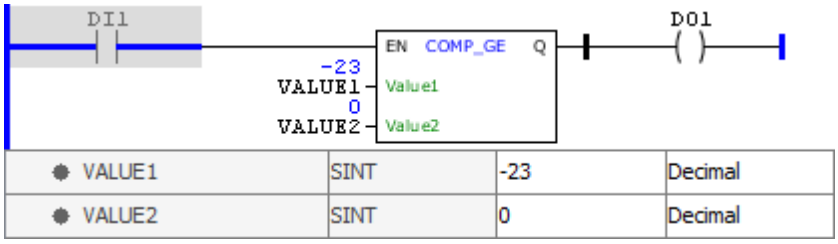
When this block has a TRUE value in EN it sends the Q output to the TRUE value if Value1 is higher than or equal to Value2. Otherwise, Q receives FALSE.

When EN has FALSE value, Q remains in FALSE.

Block Flowchart

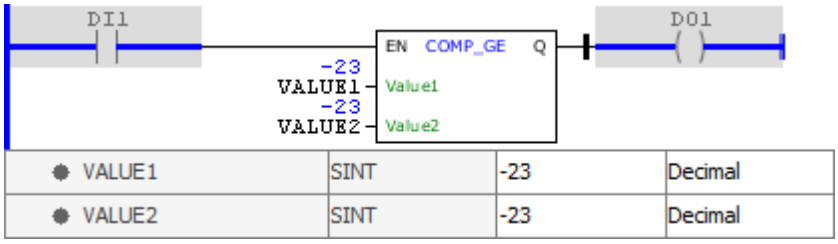


Example



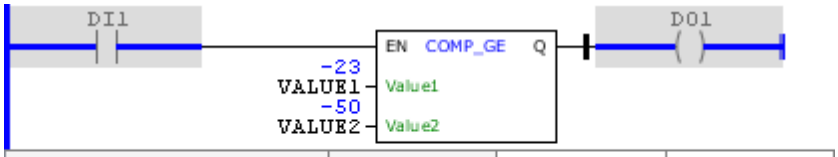
VALUE1	SINT	-23	Decimal
VALUE2	SINT	0	Decimal

The example above checks equality or majority of VALUE1 in relation to VALUE2. Since VALUE1 has lower value than VALUE2, the Q output is disabled.



VALUE1	SINT	-23	Decimal
VALUE2	SINT	-23	Decimal

The example above checks equality or majority of VALUE1 in relation to VALUE2. Since both variables have the same value, the Q output is activated.



VALUE1	SINT	-23	Decimal
VALUE2	SINT	-50	Decimal

The example above checks equality or majority of VALUE1 in relation to VALUE2. Since VALUE1 has higher value than VALUE2, the Q output is activated.

11.10.7.3.3 COMPGT

Block that compares the values of Value1 and Value2, enabling the output Q if Value1 is higher than Value2.

Ladder Representation

Block Structure

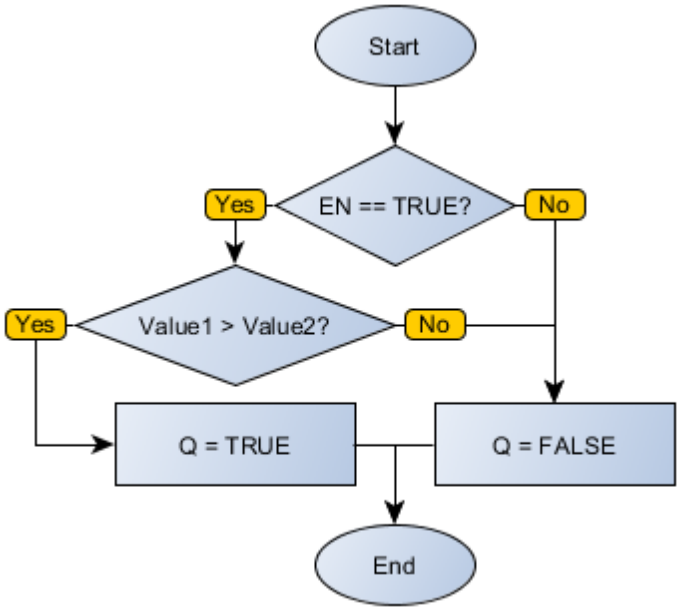
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	First value of comparison
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Second value of comparison
VAR_OUTPUT	Q	BOOL	Indicator of majority of Value1

Operation

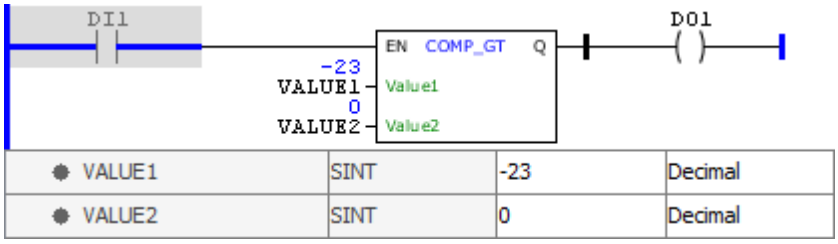
When this block has a TRUE value in EN, it sends to the Q output the TRUE value if Value1 is higher than Value2. Otherwise, Q receives FALSE.

When EN has FALSE value, Q remains in FALSE.

Block Flowchart

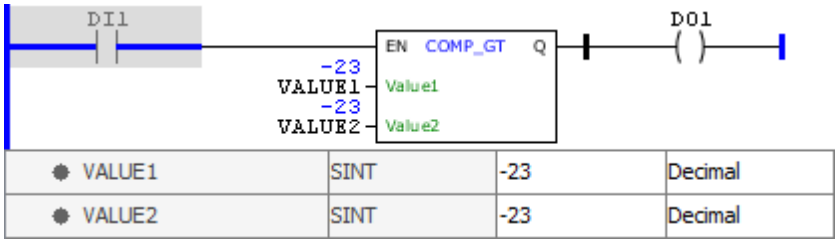


Example



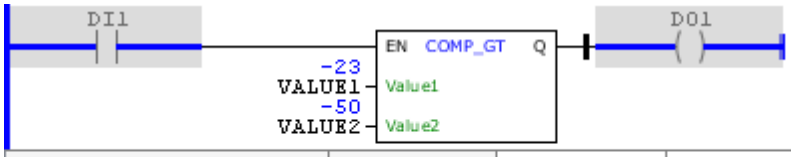
VALUE1	SINT	-23	Decimal
VALUE2	SINT	0	Decimal

The example above checks the majority of VALUE1 in relation to VALUE2. Since VALUE1 has lower value than VALUE2, the Q output is disabled.



VALUE1	SINT	-23	Decimal
VALUE2	SINT	-23	Decimal

The example above checks the majority of VALUE1 in relation to VALUE2. Since both variables have the same value, the Q output is disabled.



VALUE1	SINT	-23	Decimal
VALUE2	SINT	-50	Decimal

The example above checks the majority of VALUE1 in relation to VALUE2. Since VALUE1 has higher value than VALUE2, the Q output is activated.

11.10.7.3.4 COMPLE

Block that compares the values of Value1 and Value2, enabling the output Q if Value1 is lower than or equal to Value2.

Ladder Representation

Block Structure

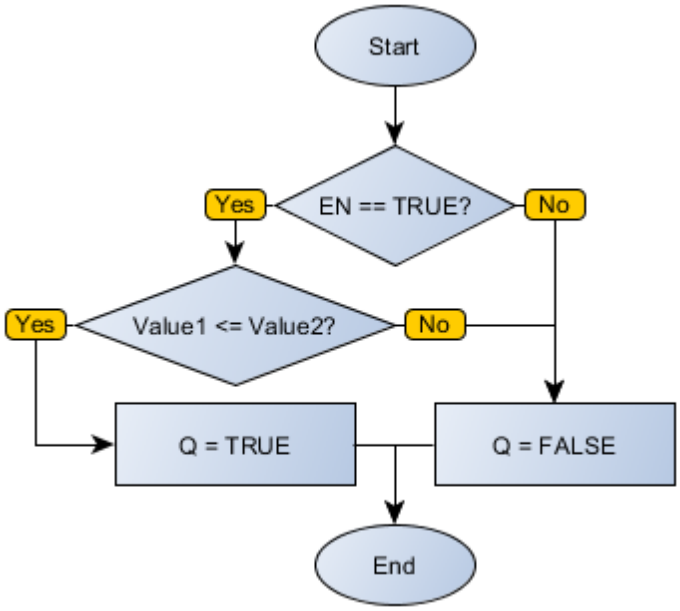
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	First value of comparison
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Second value of comparison
VAR_OUTPUT	Q	BOOL	Indicator of equality or minority of Value1

Operation

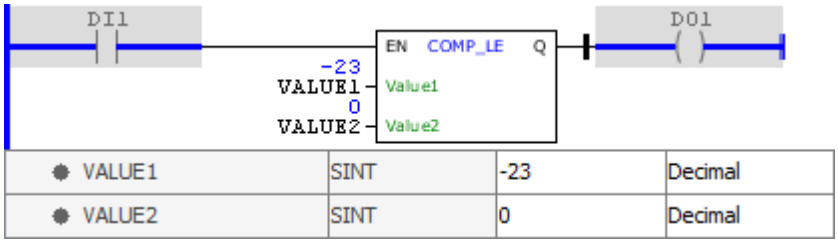
When this block has a TRUE value in EN, it sends to the Q output the TRUE value if Value1 is lower than or equal to Value2. Otherwise, Q receives FALSE.

When EN has FALSE value, Q remains in FALSE.

Block Flowchart

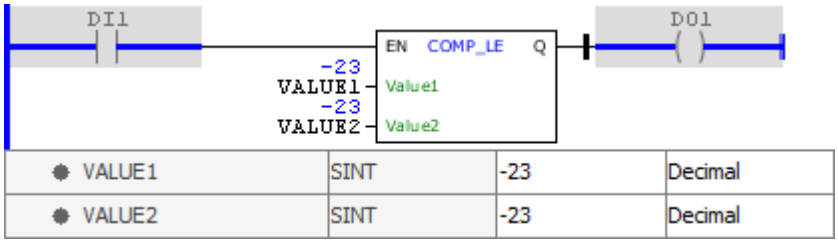


Example



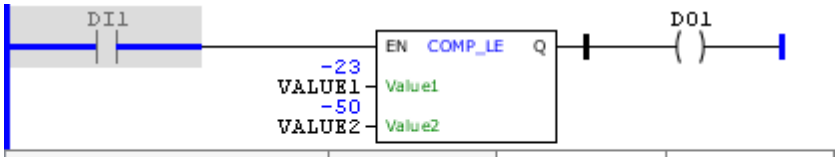
VALUE1	SINT	-23	Decimal
VALUE2	SINT	0	Decimal

The example above checks equality or minority of VALUE1 in relation to VALUE2. Since VALUE1 has lower value than VALUE2, the Q output is activated.



VALUE1	SINT	-23	Decimal
VALUE2	SINT	-23	Decimal

The example above checks equality or minority of VALUE1 in relation to VALUE2. Since both variables have the same value, the Q output is activated.



VALUE1	SINT	-23	Decimal
VALUE2	SINT	-50	Decimal

The example above checks equality or minority of VALUE1 in relation to VALUE2. Since VALUE1 has higher value than VALUE2, the Q output is disabled.

11.10.7.3.5 COMPLT

Block that compares the values of Value1 and Value2, enabling the output Q if Value1 is lower than Value2.

Ladder Representation

Block Structure

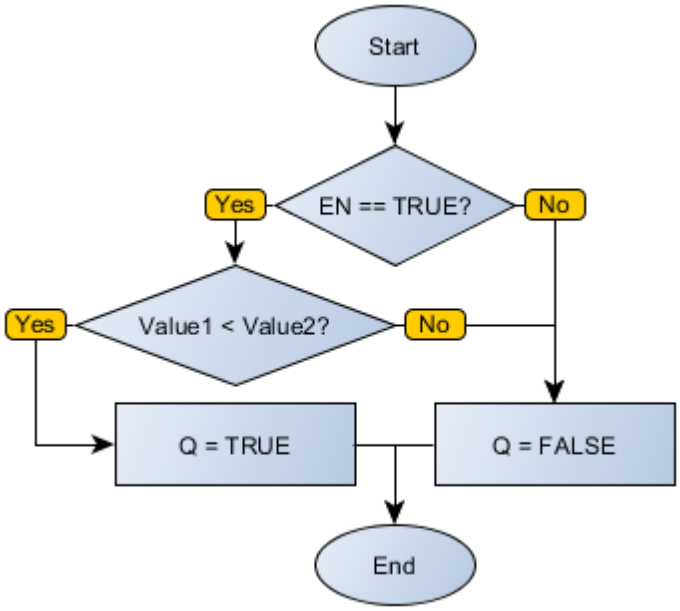
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	First value of comparison
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Second value of comparison
VAR_OUTPUT	Q	BOOL	Indicator of minority of Value1

Operation

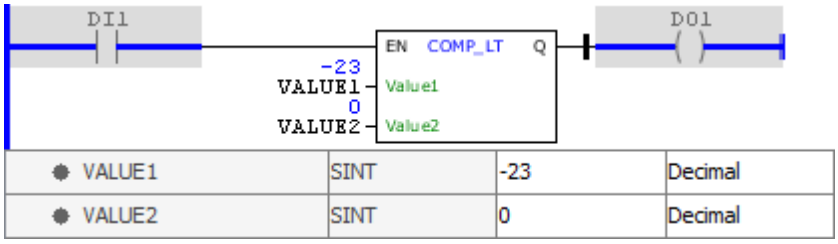
When this block has a TRUE value in EN, it sends to the Q output the TRUE value if Value1 is lower than or equal to Value2. Otherwise, Q receives FALSE.

When EN has FALSE value, Q remains in FALSE.

Block Flowchart

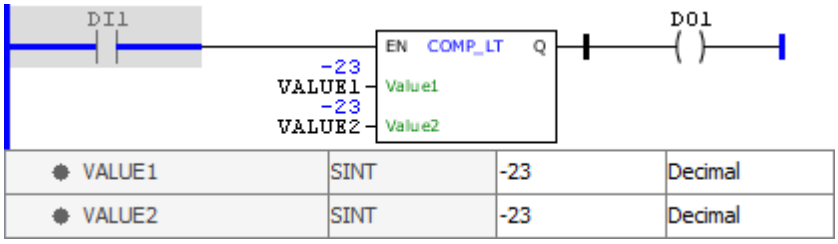


Example



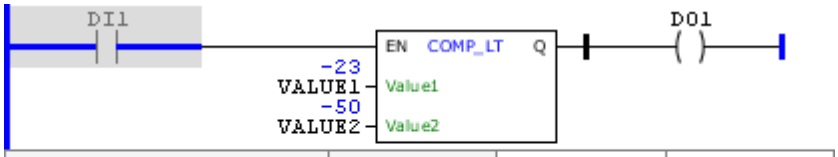
VALUE1	SINT	-23	Decimal
VALUE2	SINT	0	Decimal

The example above checks minority of VALUE1 in relation to VALUE2. Since VALUE1 has lower value than VALUE2, the Q output is activated.



VALUE1	SINT	-23	Decimal
VALUE2	SINT	-23	Decimal

The example above checks the minority of VALUE1 in relation to VALUE2. Since both variables have the same value, the Q output is disabled.



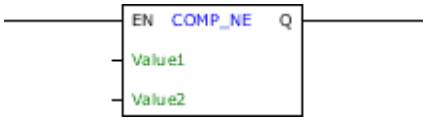
VALUE1	SINT	-23	Decimal
VALUE2	SINT	-50	Decimal

The example above checks the minority of VALUE1 in relation to VALUE2. Since VALUE1 has higher value than VALUE2, the Q output is disabled.

11.10.7.3.6 COMPNE

Block that compares the values of Value1 and Value2, enabling the Q output if Value1 is different from Value2.

Ladder Representation



Block Structure

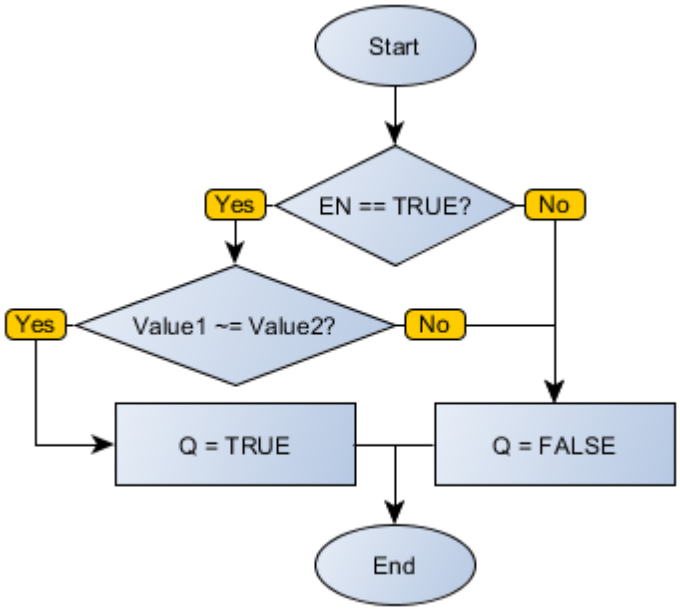
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	First value of comparison
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Second value of comparison
VAR_OUTPUT	Q	BOOL	Indicator of inequality

Operation

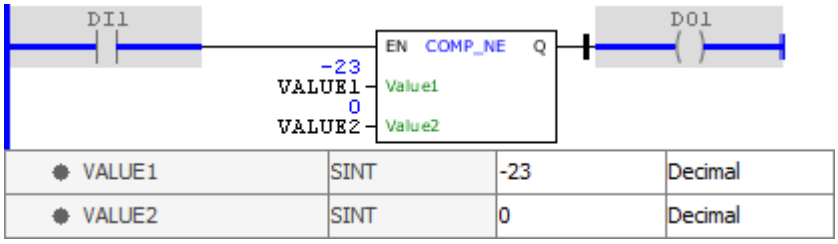
When this block has a TRUE value in EN, it sends to the Q output the TRUE value if Value1 is different from Value2. Otherwise, Q receives FALSE.

When EN has FALSE value, Q remains in FALSE.

Block Flowchart

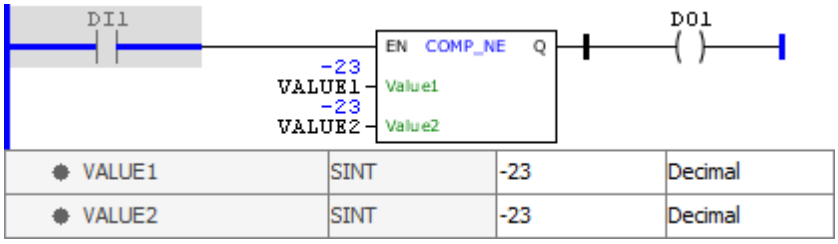


Example



VALUE1	SINT	-23	Decimal
VALUE2	SINT	0	Decimal

The example above checks inequality between VALUE1 and VALUE2. Since both variables have different values, the Q output is activated.



VALUE1	SINT	-23	Decimal
VALUE2	SINT	-23	Decimal

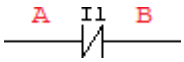
The example above checks equality between VALUE1 and VALUE2. Since both variables have the same value, the Q output is disabled.

11.10.7.4 Contact

11.10.7.4.1 NCCONTACT

Normally closed contact.

Ladder Representation



Block Structure

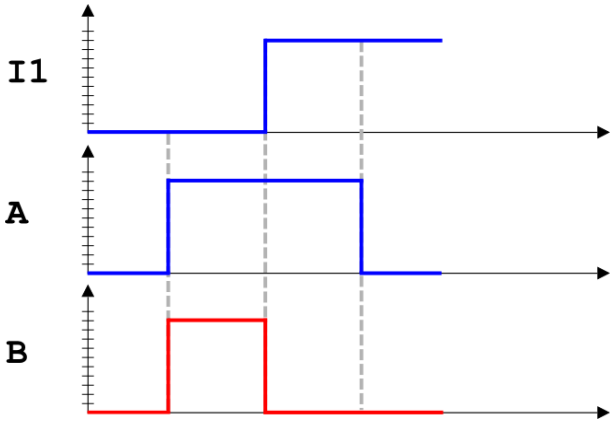
Variable Type	Name	Data Type	Description
VAR_OUTPUT	I1	BOOL	Block control input

Operation

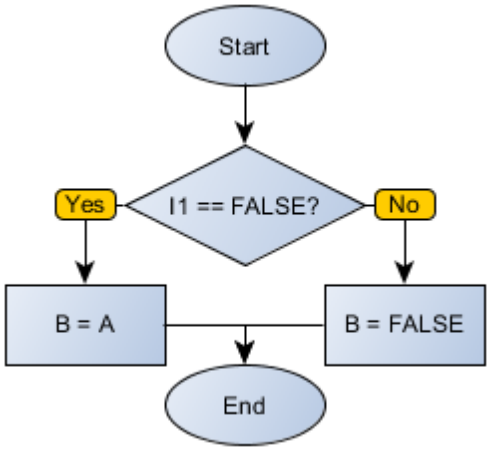
When variable I1 is with TRUE value, B receives FALSE.
When variable I1 is with FALSE value, B receives the value of A.

NOTE!
Watch out for series and parallel associations of contacts. Refer to section Contact Logic for further information.

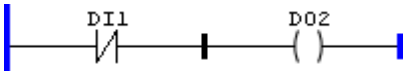
Diagram



Block Flowchart



Example

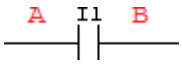


The above example performs the transfer of the opposite value of digital input DI1 to the digital output DO2.

11.10.7.4.2 NOCONTACT

Normally open contact.

Ladder Representation



Block Structure

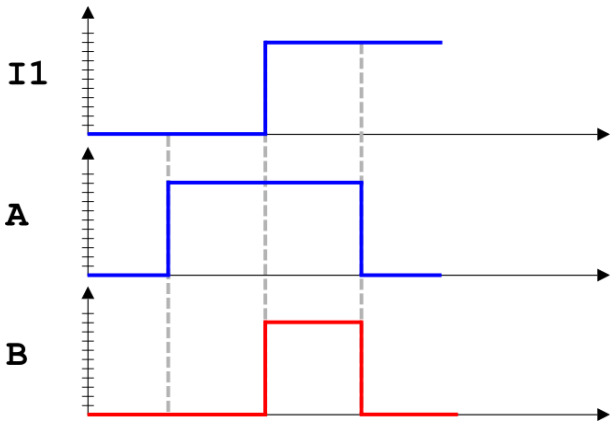
Variable Type	Name	Data Type	Description
VAR_OUTPUT	I1	BOOL	Block control input

Operation

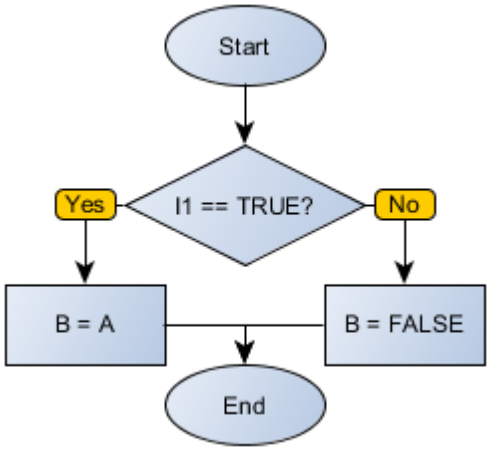
When variable I1 is with FALSE value, B receives FALSE.
When variable I1 is with TRUE value, B receives the value of A.

NOTE!
Watch out for series and parallel associations of contacts. Refer to section Contact Logic for further information.

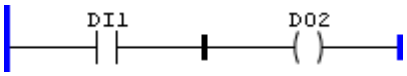
Diagram



Block Flowchart



Example



The above example performs the transfer of the value of digital input DI1 to the digital output DO2.

11.10.7.4.3 NTSCONTACT

Falling edge transition contact.

Ladder Representation

Block Structure

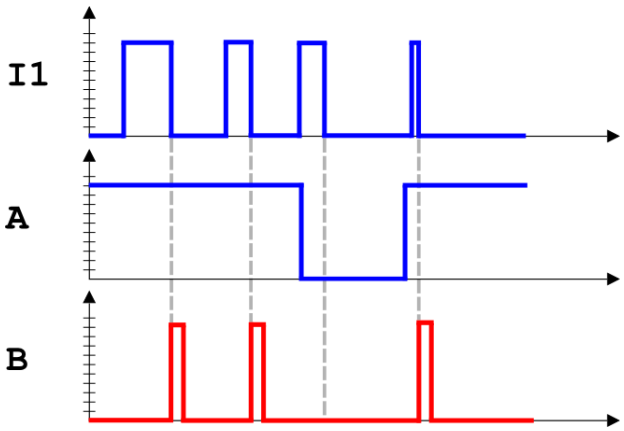
Variable Type	Name	Data Type	Description
VAR_INPUT	I1	BOOL	Block control input
VAR	NTSCONTACT_INST_0	NTSCONTACT	Instance of access to block structure

Operation

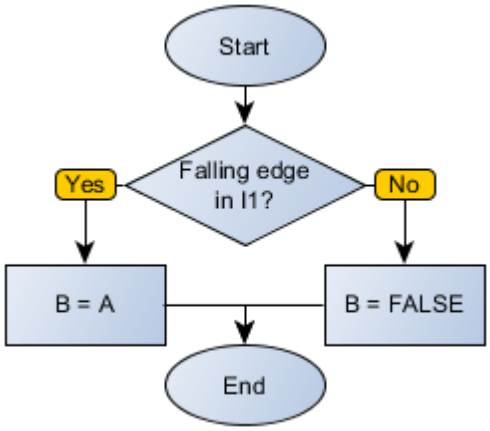
At the instant the variable I1 transitions from TRUE to FALSE (falling edge or negative edge transition), B receives the value of A for a scan cycle.
At all other times, B receives the FALSE value.

NOTE!
Watch out for series and parallel associations of contacts. Refer to section Contact Logic for further information.

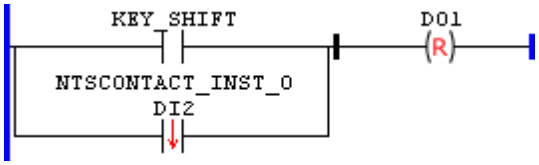
Diagram



Block Flowchart



Example

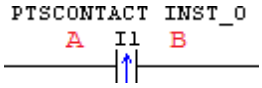


The above example resets the digital output DO1 if the SHIFT key is pressed or a positive pulse on the digital input DI2 is given.

11.10.7.4.4 PTSCONTACT

Leading edge transition contact.

Ladder Representation



Block Structure

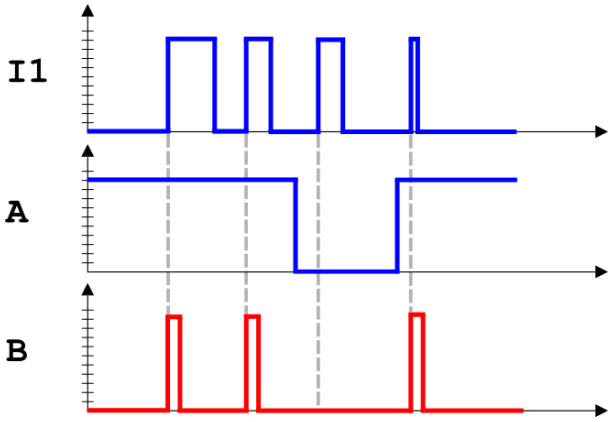
Variable Type	Name	Data Type	Description
VAR_INPUT	I1	BOOL	Block control input
VAR	PTSCONTACT_INST_0	PTSCONTACT	Instance of access to block structure

Operation

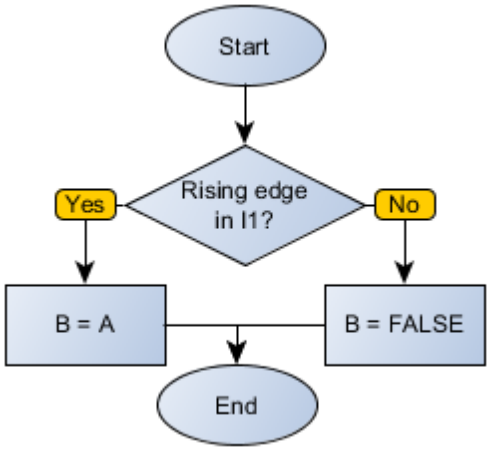
At the instant the variable I1 transitions from FALSE to TRUE (leading edge or positive edge transition), B receives the value of A for a scan cycle.
At all other times, B receives the FALSE value.

NOTE!
Watch out for series and parallel associations of contacts. Refer to section Contact Logic for further information.

Diagram



Block Flowchart



Example

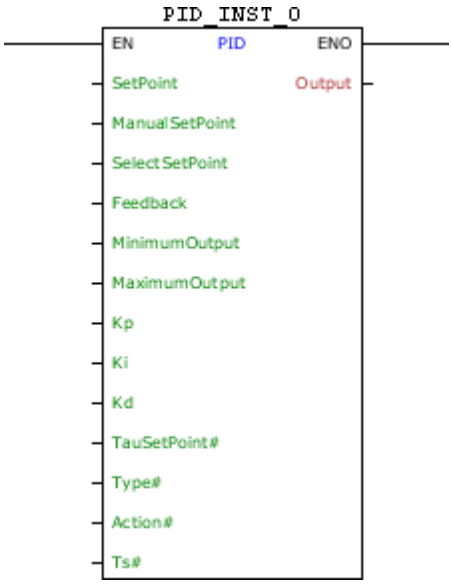
The above example resets the digital output DO1 if the SHIFT key is pressed and a positive pulse on the digital input DI2 is given.

11.10.7.5 Control

11.10.7.5.1 PID

Block that performs the function of a discrete PID controller. From the input variables, it calculates the corresponding controller output.

Ladder Representation



Block Structure

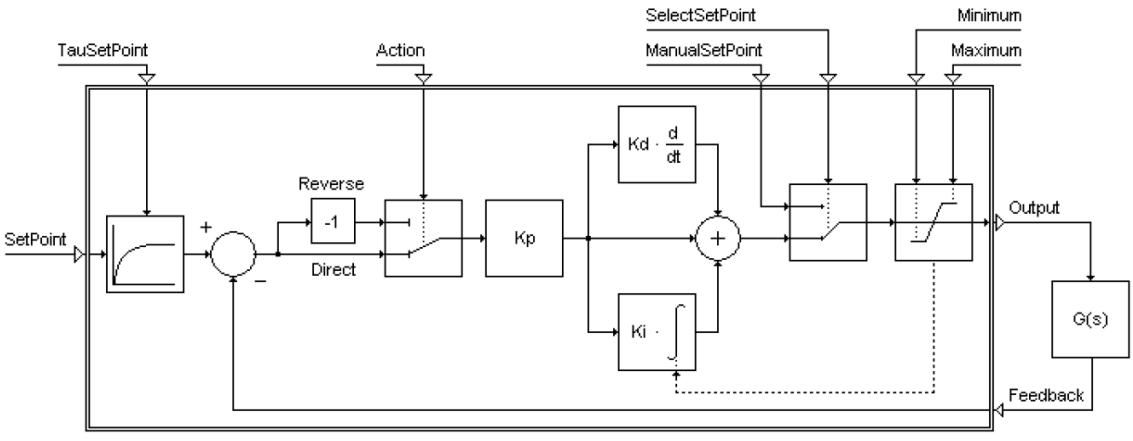
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	SetPoint	REAL	Automatic reference (pre-control)
	ManualSetPoint	REAL	Forced reference (post control)
	SelectSetPoint	BOOL	Selects w hich reference to use
	Feedback	REAL	Feedback loop variable
	MinimumOutput	REAL	Minimum value of the controller output
	MaximumOutput	REAL	Maximum value of the controller output
	Kp	REAL	Proportional gain
	Ki	REAL	Integral gain
	Kd	REAL	Derivative gain
	TauSetPoint#	REAL	Time constant of the automatic reference in put filter
	Type#	BYTE	Controller type
	Action#	BYTE	Control action
	Ts#	UINT	Sampling time [ms]
VAR_OUTPUT	ENO	BOOL	Output enabling
	Output	REAL	Controller output
VAR	PID_INST_0	PID	Instance of access to block structure

Operation

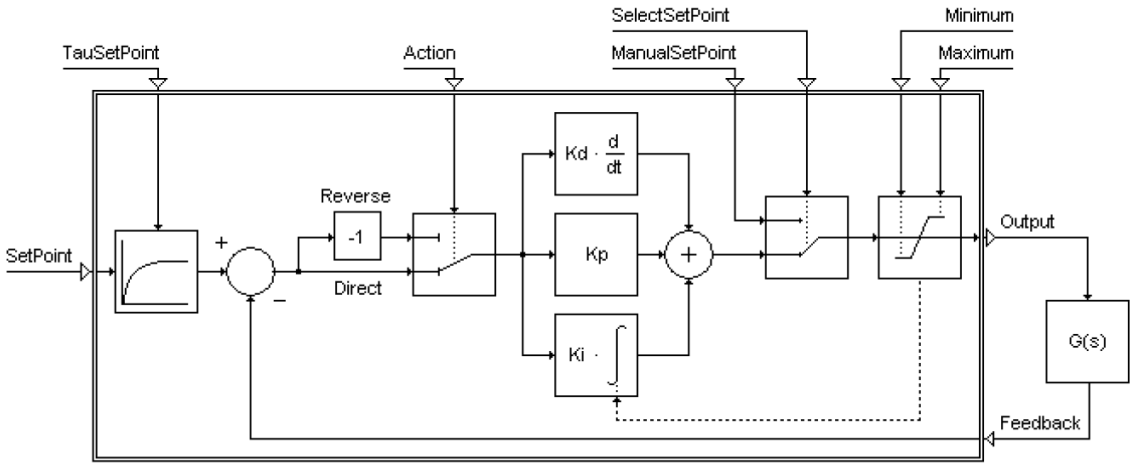
On the positive transition edge in EN, Output receives zero value, and the block executes its functionality as EN is at TRUE level.

When enabled, this block performs a routine PID control with the Kp, Ki and Kd parameters chosen. The PID topology used may be the Academic or Parallel, depending on what is chosen in Type#.

Academic Form:



Parallel Form:



The levels of the output signal of the controller are saturated at value MinimumOutput and MaximumOutput. The SelectSetPoint input level FALSE causes the SetPoint reference be adopted, allowing the controller maintains control over the process. When SelectSetPoint goes to TRUE level, the controller has no more domain, and ManualSetPoint becomes to be considered the output signal of the controller.

Action# will define the feedback operation. If Action# is DIRECT, the operation will be SetPoint – Feedback. If Action# is REVERSE, the operation will be Feedback – SetPoint.

Feedback receives the process variable considered as the plant output. Ts# receives the sampling time for the controller and # TauSetPoint receives the time constant for the input filter of the automatic reference.

When EN has FALSE value, Output remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

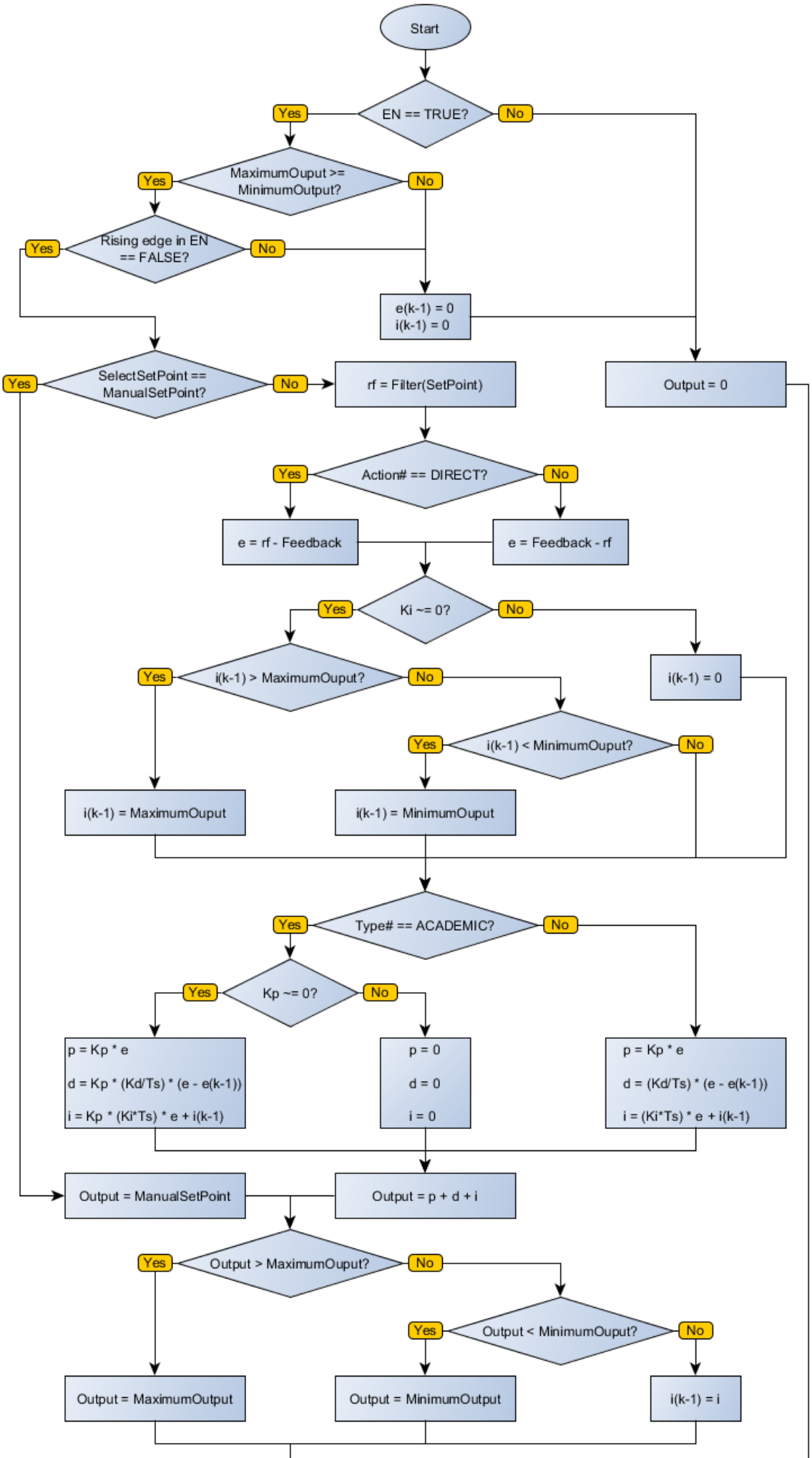
NOTE!
Effects of the alteration of gains on the process

- If Kp decreases, the process becomes slower; generally more stable or less oscillating; it has less overshoot.
- If Kp increases, the process responds faster; it may become more unstable or more oscillating; it has more overshoot.
- If Ki decreases, the process becomes slower, lagging to reach the "SetPoint"; it becomes more stable or less oscillating; it has less overshoot.
- If Ki increases, the process becomes faster, quickly reaching the "SetPoint"; it becomes more unstable or more oscillating; it has more overshoot.
- If Kd decreases, the process becomes slower; it has less overshoot.
- If Kd increases, it has more overshoot.

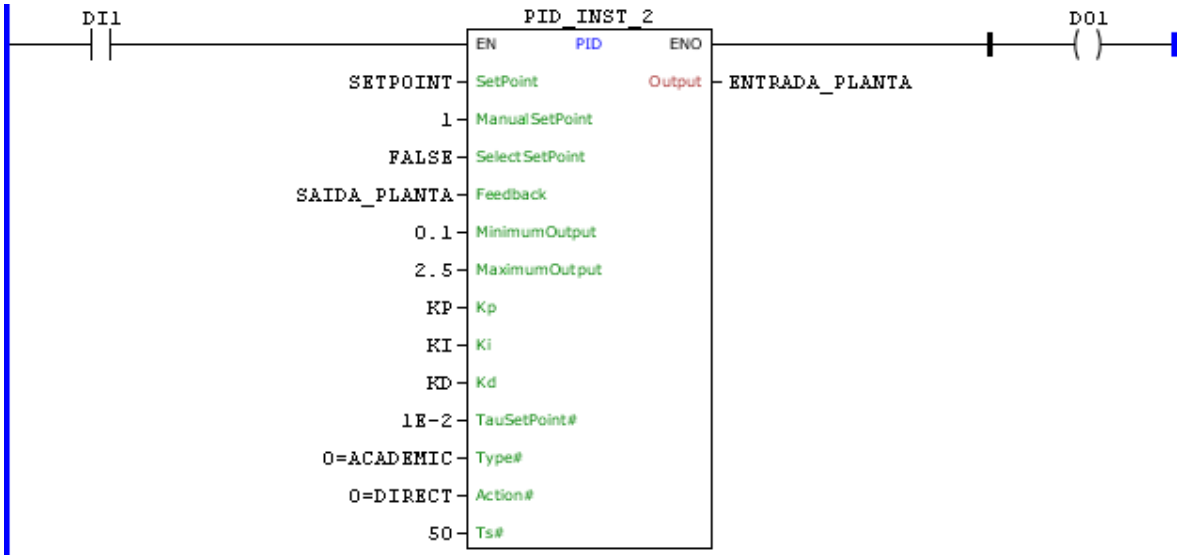
NOTE!
How to improve the performance of the process through the adjustment of gains (valid for the Academic PID)

- If the performance of the process is almost good, but the overshoot is a bit high, try to: (1) decrease Kp 20%, (2) decrease Ki 20% and/or (3) decrease Kd 50%.
- If the performance of the process is almost good, but it does not have overshoot and lags to reach the "SetPoint", try to: (1) increase Kp 20%, (2) increase Ki 20% and/or (3) increase Kd 50%.
- If the performance of the process is good, but the process output is varying too much, try to: (1) increase Kd 50%, (2) decrease Kp 20%.
- If the performance of the process is bad, i.e. after start up, the transitory lasts several periods of oscillation that reduce very slowly or never reduce at all, try to: (1) decrease Kp 50%.
- If the performance of the process is bad, i.e. after start up it slowly moves towards the "SetPoint" without overshoot, but is still very far and the process output is less than the rated value, try to: (1) increase Kp 50%, (2) increase Ki 50%, (3) increase Kd 70%.

Block Flowchart


Start
EN == TRUE?
Yes
No
MaximumOuput >= MinimumOutput?
Yes
No
Rising edge in EN == FALSE?
Yes
No
e(k-1) = 0
i(k-1) = 0
SelectSetPoint == ManualSetPoint?
Yes
No
rf = Filter(SetPoint)
Output = 0
Action# == DIRECT?
Yes
No
e = rf - Feedback
e = Feedback - rf
Ki ~= 0?
Yes
No
i(k-1) > MaximumOuput?
Yes
No
i(k-1) = 0
i(k-1) < MinimumOuput?
Yes
No
i(k-1) = MaximumOuput
i(k-1) = MinimumOuput
Type# == ACADEMIC?
Yes
No
Kp ~= 0?
Yes
No
p = Kp * e
d = Kp * (Kd/Ts) * (e - e(k-1))
i = Kp * (Ki*Ts) * e + i(k-1)
p = 0
d = 0
i = 0
p = Kp * e
d = (Kd/Ts) * (e - e(k-1))
i = (Ki*Ts) * e + i(k-1)
Output = ManualSetPoint
Output = p + d + i
Output > MaximumOuput?
Yes
No
Output < MinimumOuput?
Yes
No
Output = MaximumOutput
Output = MinimumOutput
i(k-1) = i

Example



The above example creates a loop of a digital PID form with sampling time 50 ms, using the constants KP, KI and KD for control. Automatic reference SETPOINT, filtered by a first order filter with time constant of 0:01 is used. The error signal is calculated as the difference between the filtered reference and variable SAIDA_PLANTA. The controller output is saturated between the values 0.1 and 2.5 and sent to the variable ENTRADA_PLANTA.

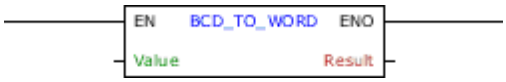
11.10.7.6 Conversion

11.10.7.6.1 BCD

11.10.7.6.1.1 BCD_TO_WORD

Block that performs the conversion of a BCD code into a WORD value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	WORD UINT INT	Value in BCD
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	WORD UINT INT	Value in WORD

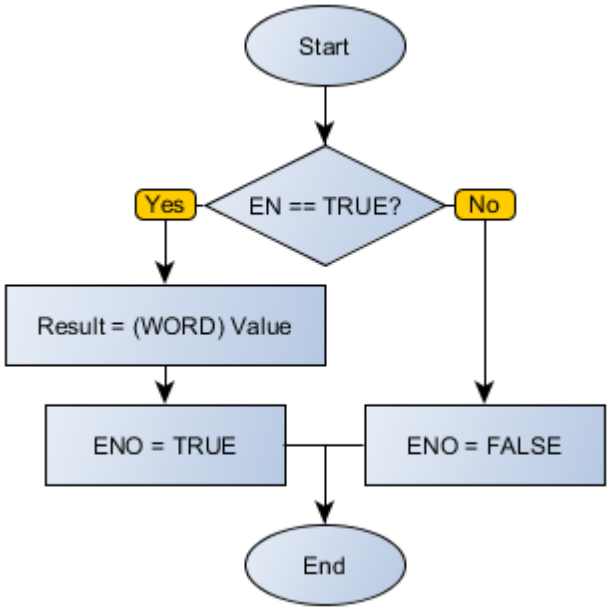
Operation

When this block has a TRUE value in EN, it interprets the Value value as BCD and converts it to WORD, storing in Result.

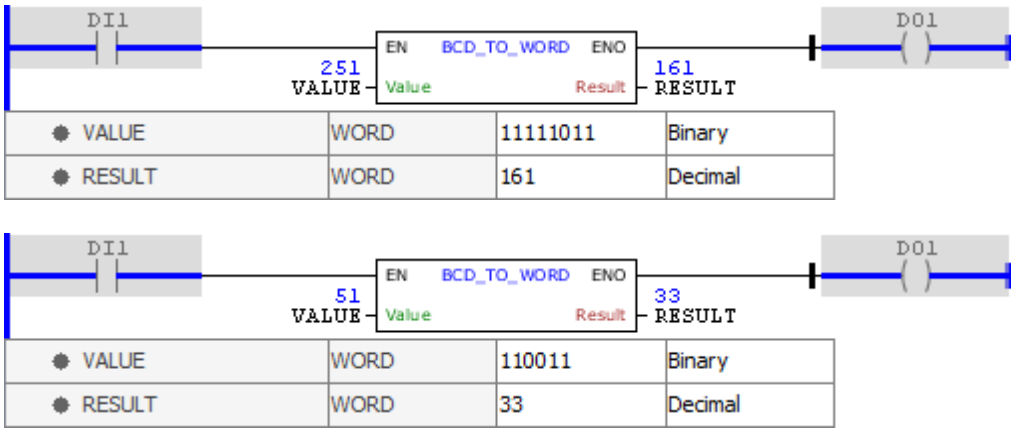
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example

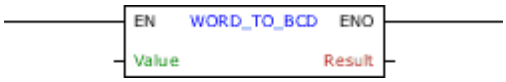


The above example converts the VALUE variable, in BCD, into a WORD value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.10.7.6.1.2 WORD_TO_BCD

Block that performs the conversion of a WORD value into a BCD code.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	WORD UINT INT	Value in WORD
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	WORD UINT INT	Value in BCD

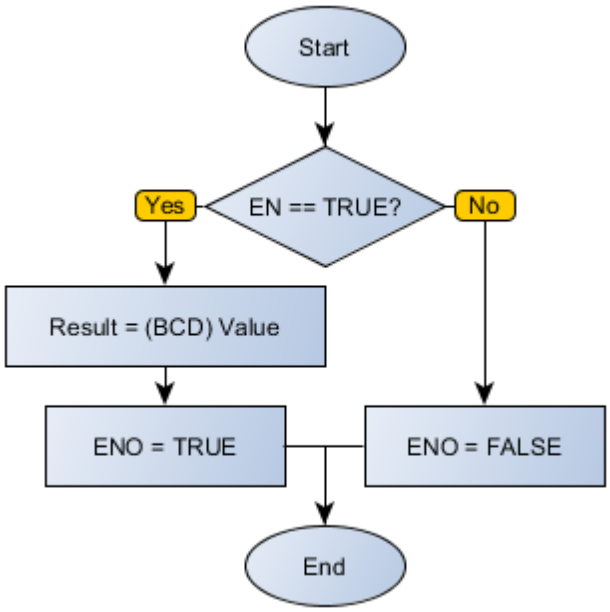
Operation

When this block has a TRUE value in EN, it interprets the Value value as WORD and converts it into BCD, storing in Result.

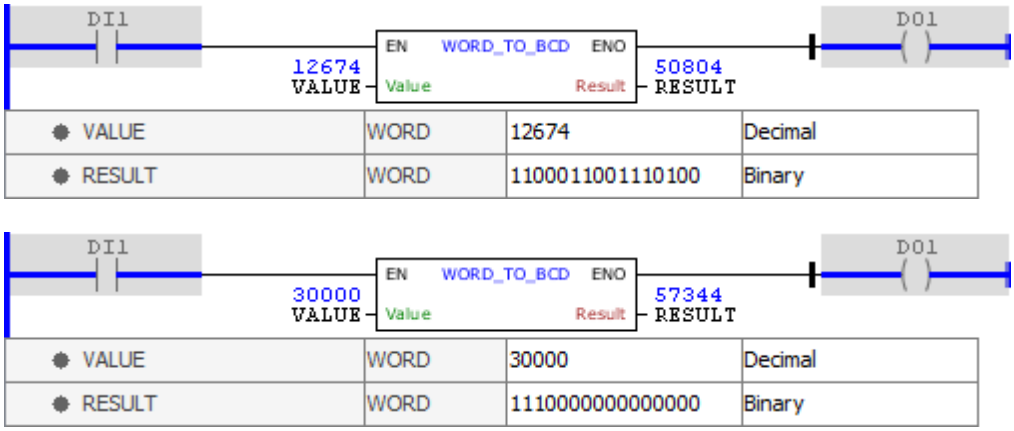
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



VALUE	WORD	12674	Decimal
RESULT	WORD	1100011001110100	Binary

VALUE	WORD	30000	Decimal
RESULT	WORD	1110000000000000	Binary

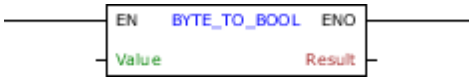
The examples above perform the conversion of VALUE variable, in WORD, into a BCD value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.10.7.6.2 BOOL

11.10.7.6.2.1 BYTE_TO_BOOL

Block that performs the conversion of a BYTE value into a BOOL value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT	Value in BYTE
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BOOL	Value in BOOL

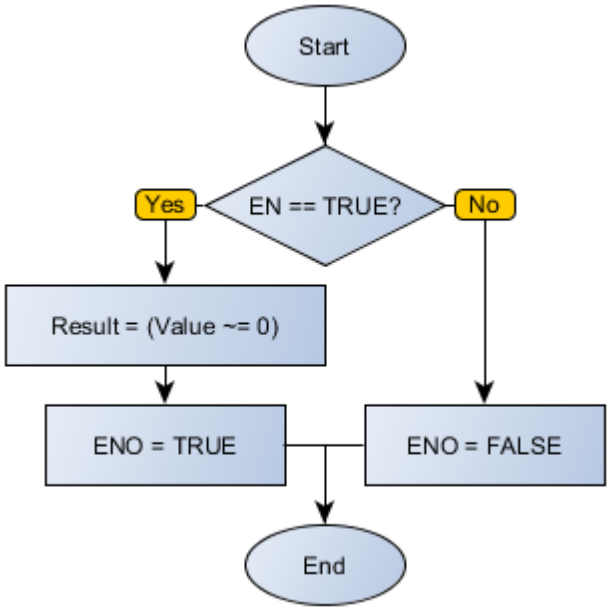
Operation

When this block has a TRUE value in EN, it interprets the Value value as BYTE and converts it into BOOL, storing in Result.

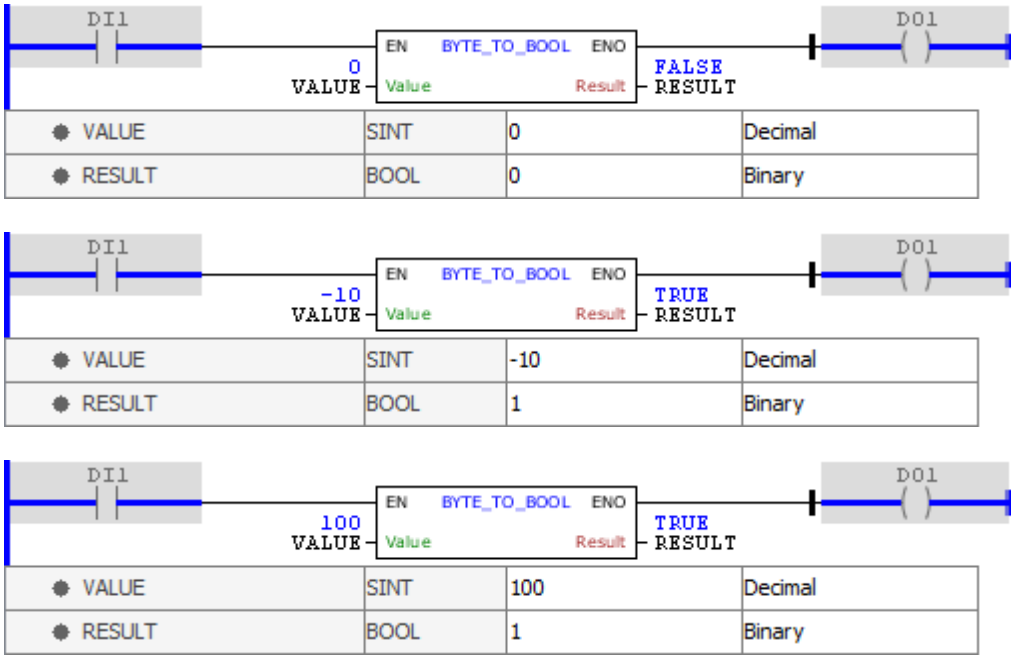
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example

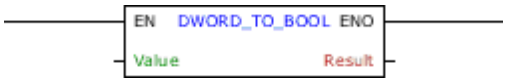


The examples above perform the conversion of VALUE variable, in BYTE, into a BOOL value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.10.7.6.2.2 DWORD_TO_BOOL

Block that performs the conversion of a DWORD value into a BOOL value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	DWORD UDINT DINT	Value in DWORD
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BOOL	Value in BOOL

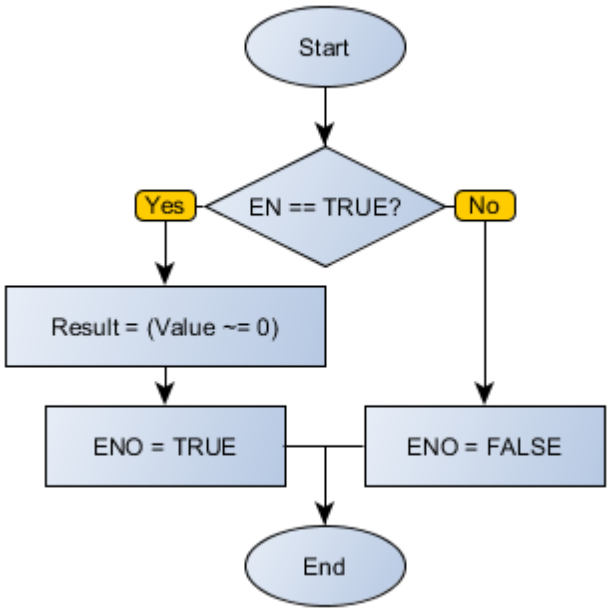
Operation

When this block has a TRUE value in EN, it interprets the Value value as DWORD and converts it into BOOL, storing in Result.

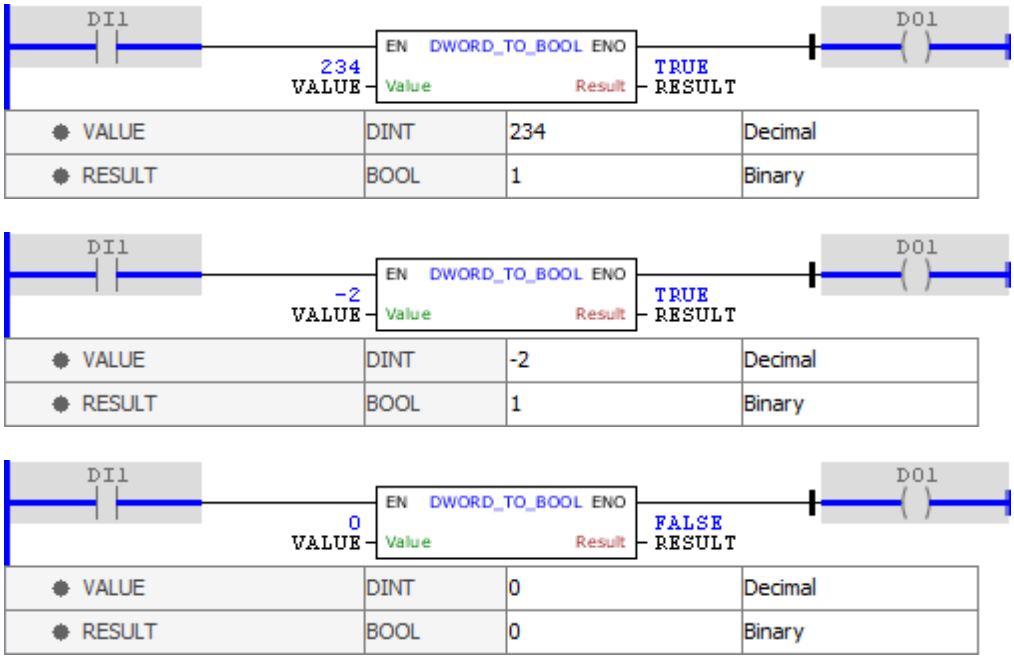
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example

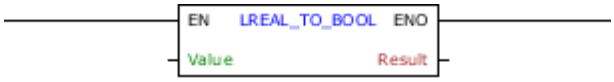


The examples above perform the conversion of VALUE variable, in DWORD, into a BOOL value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.10.7.6.2.3 LREAL_TO_BOOL

Block that performs the conversion of a LREAL value into a BOOL value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	LREAL	Value in LREAL
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BOOL	Value in BOOL

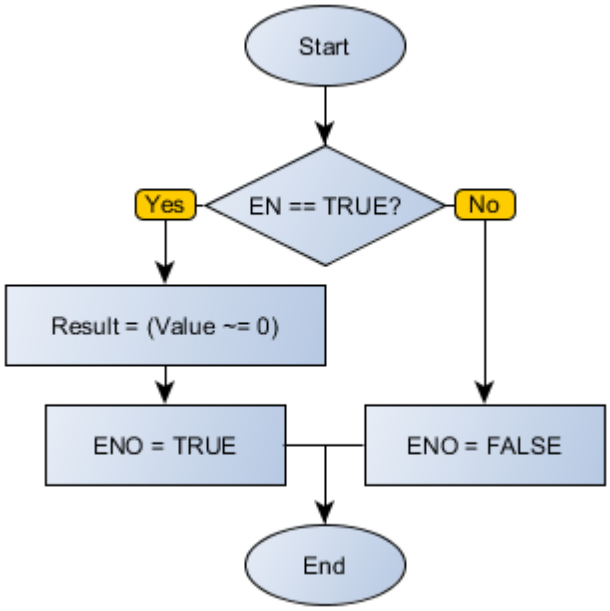
Operation

When this block has a TRUE value in EN, it interprets the Value value as LREAL and converts it into BOOL, storing in Result.

When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

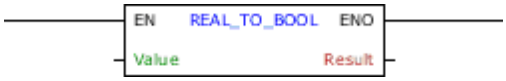
Block Flowchart



11.10.7.6.2.4 REAL_TO_BOOL

Block that performs the conversion of a REAL value into a BOOL value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Value in REAL
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BOOL	Value in BOOL

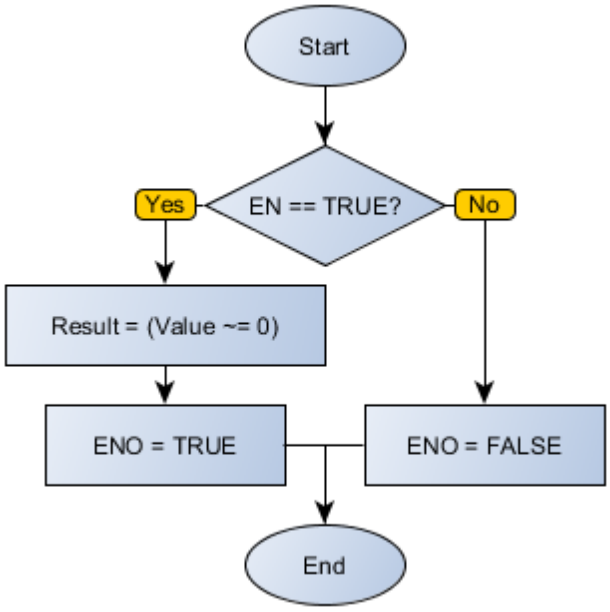
Operation

When this block has a TRUE value in EN, it interprets the Value value as REAL and converts it into BOOL, storing in Result.

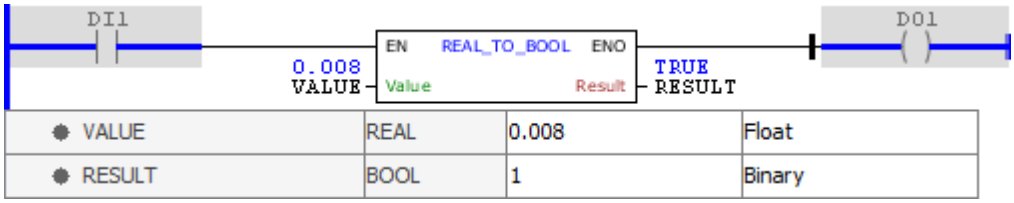
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

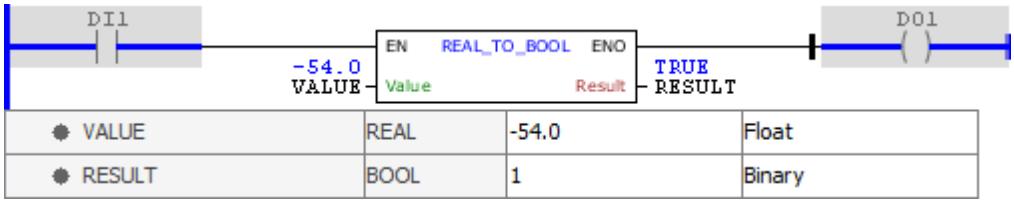
Block Flowchart



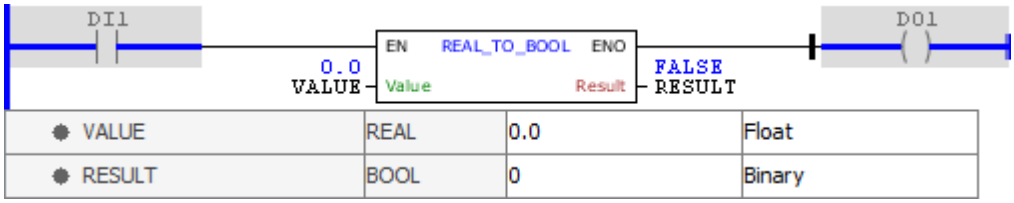
Example



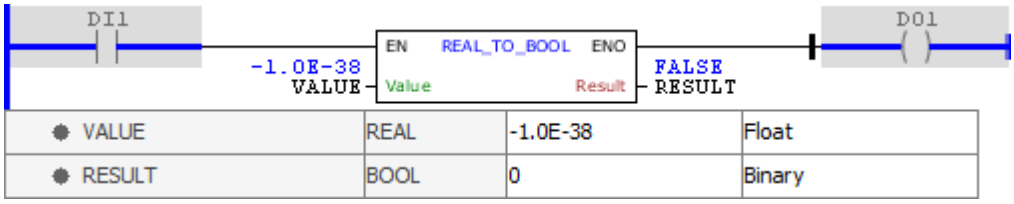
VALUE	REAL	0.008	Float
RESULT	BOOL	1	Binary



VALUE	REAL	-54.0	Float
RESULT	BOOL	1	Binary



VALUE	REAL	0.0	Float
RESULT	BOOL	0	Binary



VALUE	REAL	-1.0E-38	Float
RESULT	BOOL	0	Binary

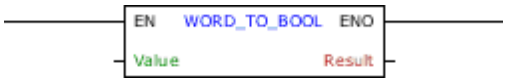
The examples above perform the conversion of VALUE variable, in REAL, into a BOOL value storing

the final result in RESULT. The block ends with success and ENO output is activated. Notice in the last example that the values very close to the machine epsilon may result in an interpretation of the FALSE value.

11.10.7.6.2.5 WORD_TO_BOOL

Block that performs the conversion of a WORD value into a BOOL value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	WORD UINT INT	Value in WORD
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BOOL	Value in BOOL

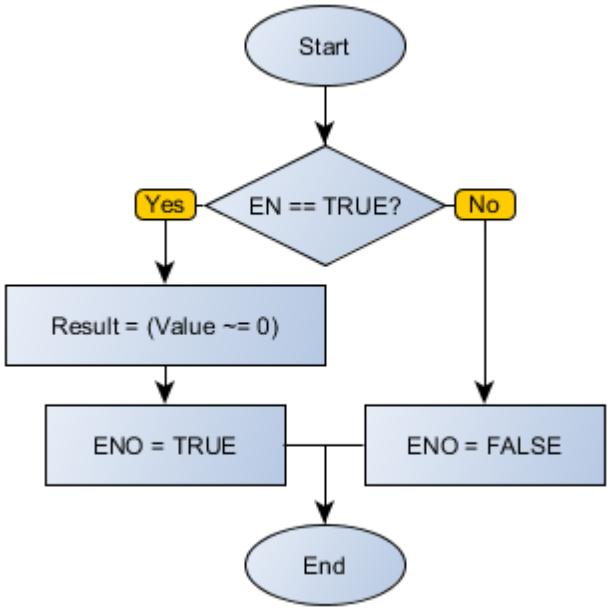
Operation

When this block has a TRUE value in EN, it interprets the Value value as WORD and converts it into BOOL, storing in Result.

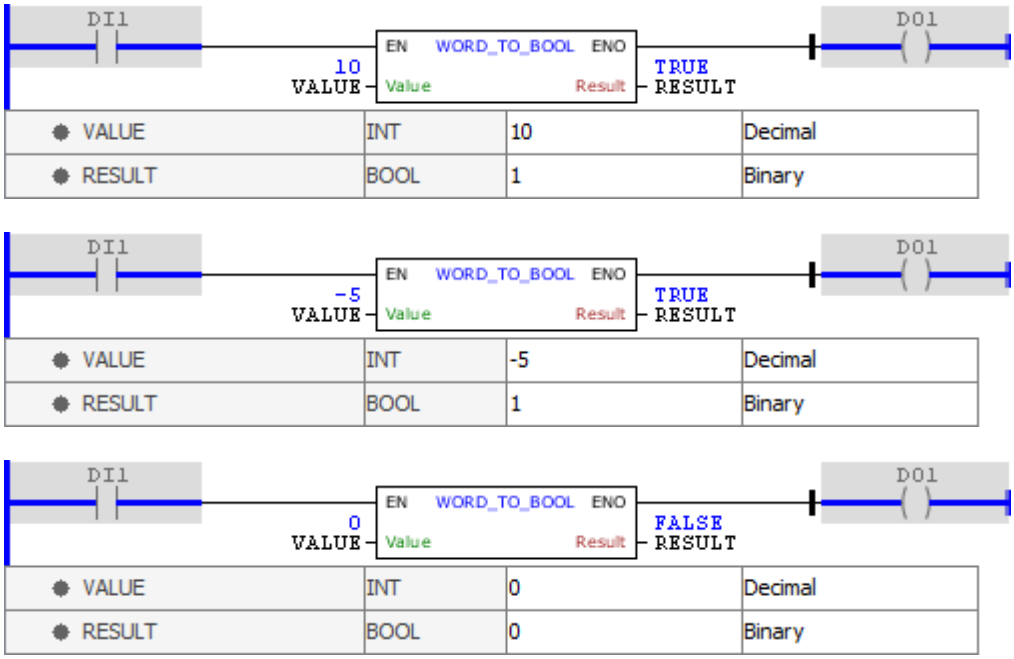
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



VALUE	INT	10	Decimal
RESULT	BOOL	1	Binary

VALUE	INT	-5	Decimal
RESULT	BOOL	1	Binary

VALUE	INT	0	Decimal
RESULT	BOOL	0	Binary

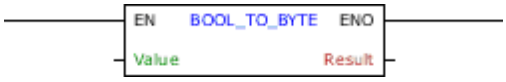
The examples above perform the conversion of VALUE variable, in WORD, into a BOOL value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.10.7.6.3 BYTE

11.10.7.6.3.1 BOOL_TO_BYTE

Block that performs the conversion of a BOOL value into a BYTE value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BOOL	Value in BOOL
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT	Value in BYTE

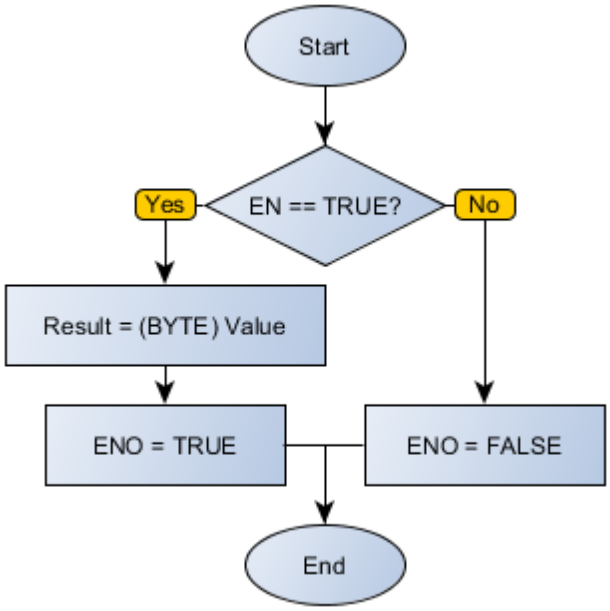
Operation

When this block has a TRUE value in EN, it interprets the Value value as BOOL and converts it into BYTE, storing in Result.

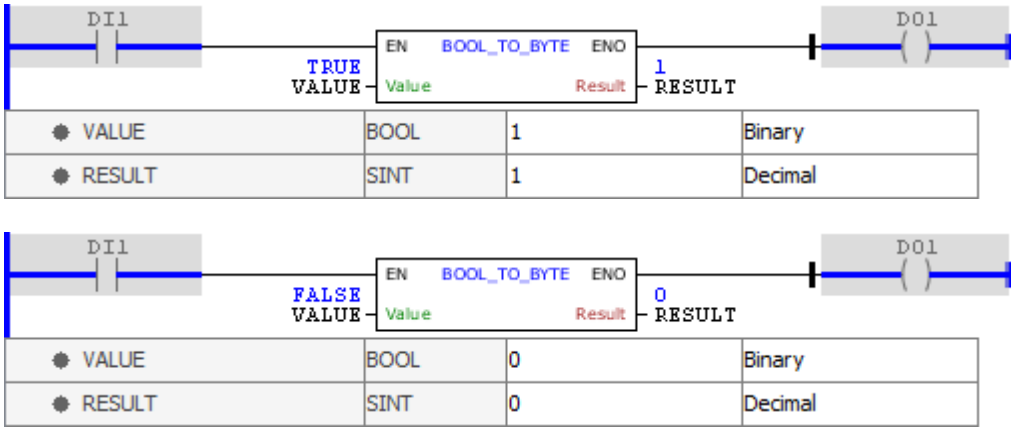
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



VALUE	BOOL	1	Binary
RESULT	SINT	1	Decimal

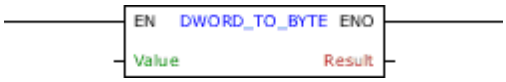
VALUE	BOOL	0	Binary
RESULT	SINT	0	Decimal

The examples above perform the conversion of variable VALUE, in BOOL, into a BYTE value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.10.7.6.3.2 DWORD_TO_BYTE

Block that performs the conversion of a DWORD value into a BYTE value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	DWORD UDINT DINT	Value in DWORD
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT	Value in BYTE

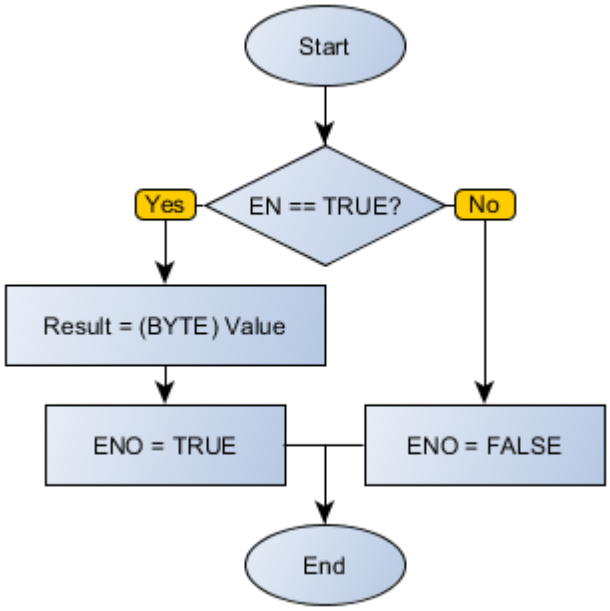
Operation

When this block has a TRUE value in EN, it interprets the Value value as DWORD and converts it into BYTE, storing in Result.

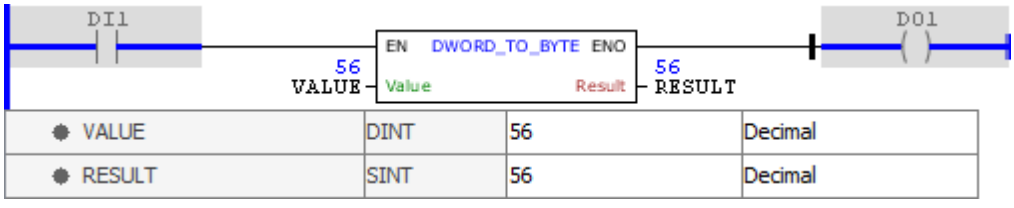
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

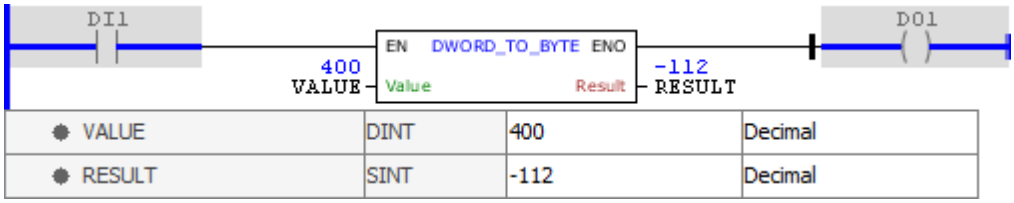
Block Flowchart



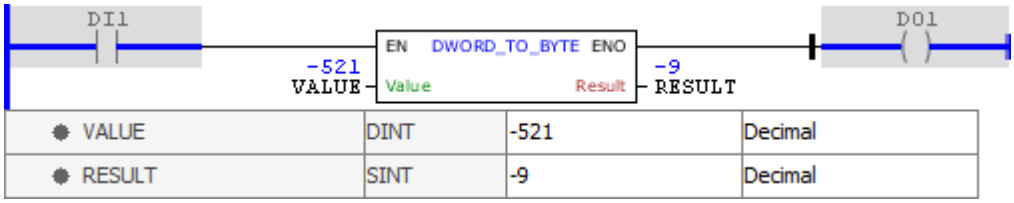
Example



VALUE	DINT	56	Decimal
RESULT	SINT	56	Decimal



VALUE	DINT	400	Decimal
RESULT	SINT	-112	Decimal



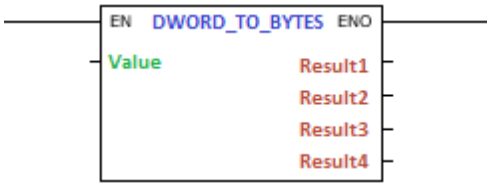
VALUE	DINT	-521	Decimal
RESULT	SINT	-9	Decimal

The examples above perform the conversion of variable VALUE, in DWORD, into a BYTE value storing the final result in RESULT. The block ends with success and ENO output is activated. Notice that only the eight least significant bits are taken into account.

11.10.7.6.3.3 DWORD_TO_BYTES

Block that performs the conversion of a 32 bits (DWORD) value into four 8 bits (4 BYTES) value.

Ladder Representation



Block Structure

Variable Type	Name	Data type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	DWORD UDINT DINT	Value in DWORD
VAR_OUTPUT	ENO	BOOL	End of operation
	Result1	BYTE USINT SINT	Value in BYTE (LSB)
	Result2	BYTE USINT SINT	Value in BYTE
	Result3	BYTE USINT SINT	Value in BYTE
	Result4	BYTE USINT SINT	Value in BYTE (MSB)

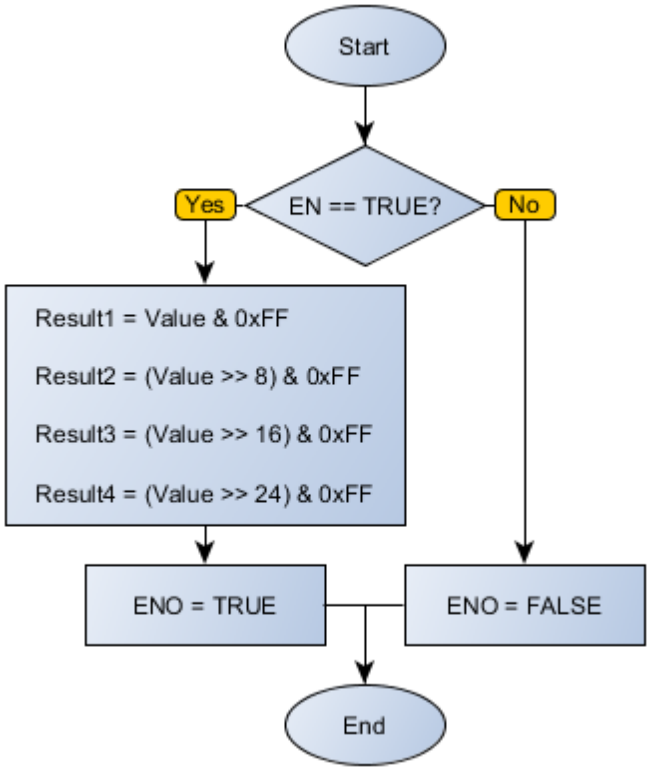
Operation

When this block has a TRUE value in EN, it interprets the Value value as DWORD and converts it into four BYTE values (from Result1 to Result4, where Result 1 is LSB), storing in Result.

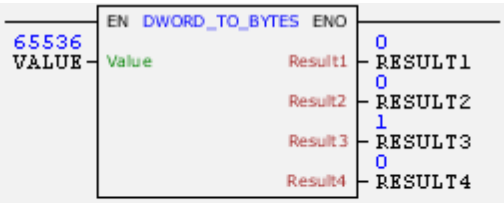
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

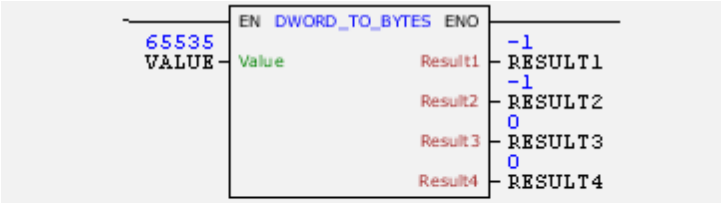
Block Flowchart



Example



RESULT1	SINT	0	0	Decimal
RESULT2	SINT	0	0	Decimal
RESULT3	SINT	0	1	Decimal
RESULT4	SINT	0	0	Decimal
VALUE	DINT	0	65536	Decimal



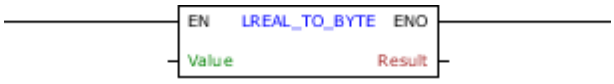
RESULT1	SINT	0	-1	Decimal
RESULT2	SINT	0	-1	Decimal
RESULT3	SINT	0	0	Decimal
RESULT4	SINT	0	0	Decimal
VALUE	DINT	0	65535	Decimal

The examples above perform the conversion of variable VALUE, in DWORD, into four BYTE value storing the final result in RESULT1, RESULT2, RESULT3 and RESULT4. The block ends with success and ENO output is activated.

11.10.7.6.3.4 LREAL_TO_BYTE

Block that performs the conversion of a LREAL value into a BYTE value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	LREAL	Value in LREAL
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT	Value in BYTE

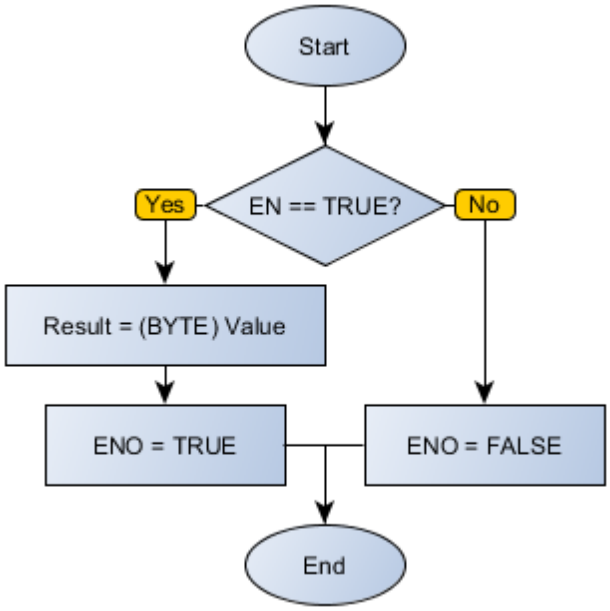
Operation

When this block has a TRUE value in EN, it interprets the Value value as LREAL and converts it into BYTE, storing in Result.

When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

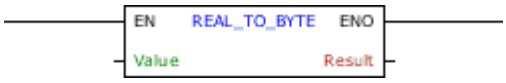
Block Flowchart



11.10.7.6.3.5 REAL_TO_BYTE

Block that performs the conversion of a REAL value into a BYTE value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Value in REAL
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT	Value in BYTE

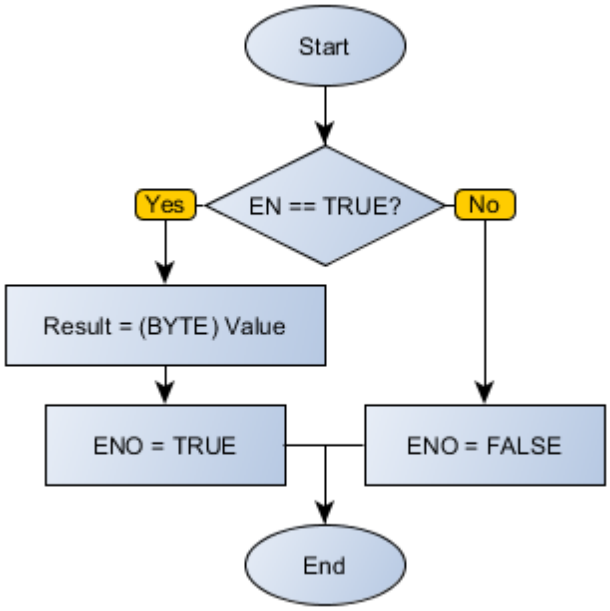
Operation

When this block has a TRUE value in EN, it interprets the Value value as REAL and converts it into BYTE, storing in Result.

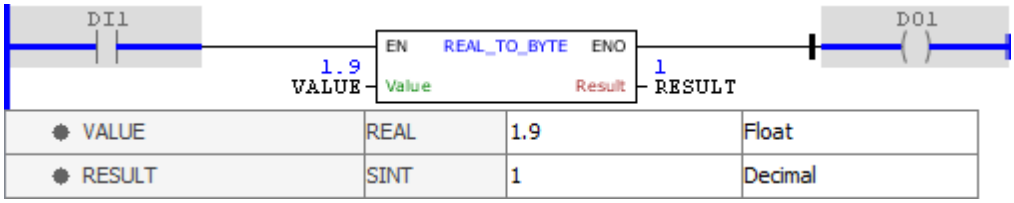
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

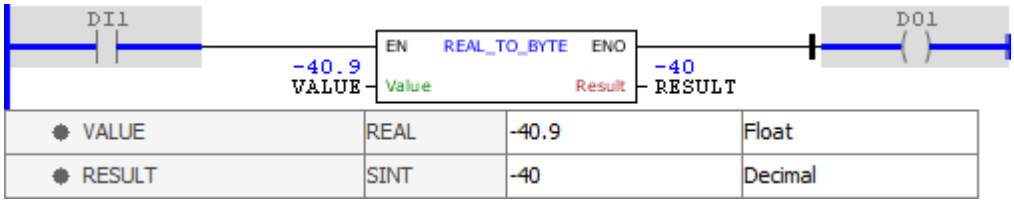
Block Flowchart



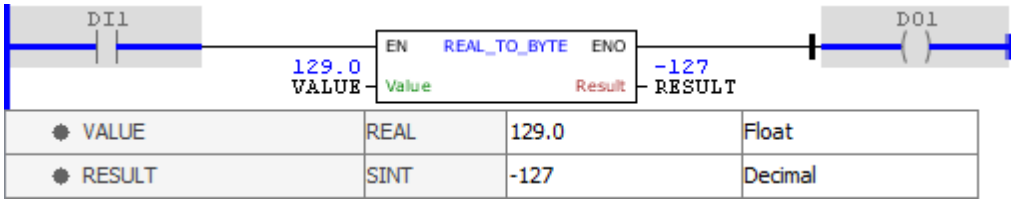
Example



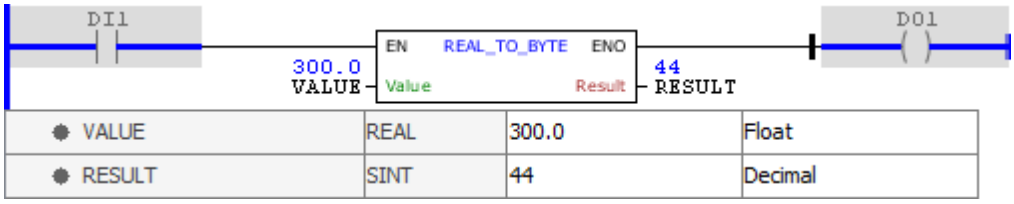
VALUE	REAL	1.9	Float
RESULT	SINT	1	Decimal



VALUE	REAL	-40.9	Float
RESULT	SINT	-40	Decimal



VALUE	REAL	129.0	Float
RESULT	SINT	-127	Decimal



VALUE	REAL	300.0	Float
RESULT	SINT	44	Decimal

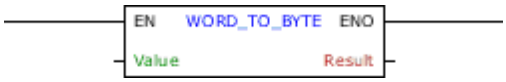
The examples above perform the conversion of variable VALUE, in REAL, into a BYTE value storing

the final result in RESULT. The block ends with success and ENO output is activated. Notice that the results are truncated in decimal and only the eight least significant bits are taken into account.

11.10.7.6.3.6 WORD_TO_BYTE

Block that performs the conversion of a WORD value into a BYTE value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	WORD UINT INT	Value in WORD
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT	Value in BYTE

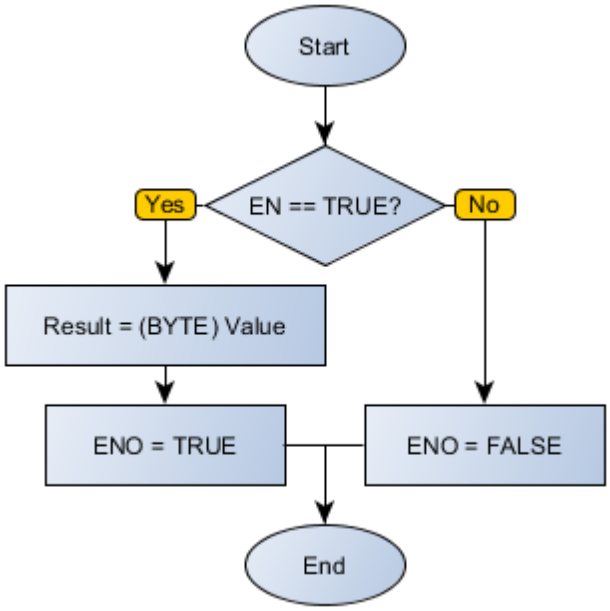
Operation

When this block has a TRUE value in EN, it interprets the Value value as WORD and converts it into BYTE, storing in Result.

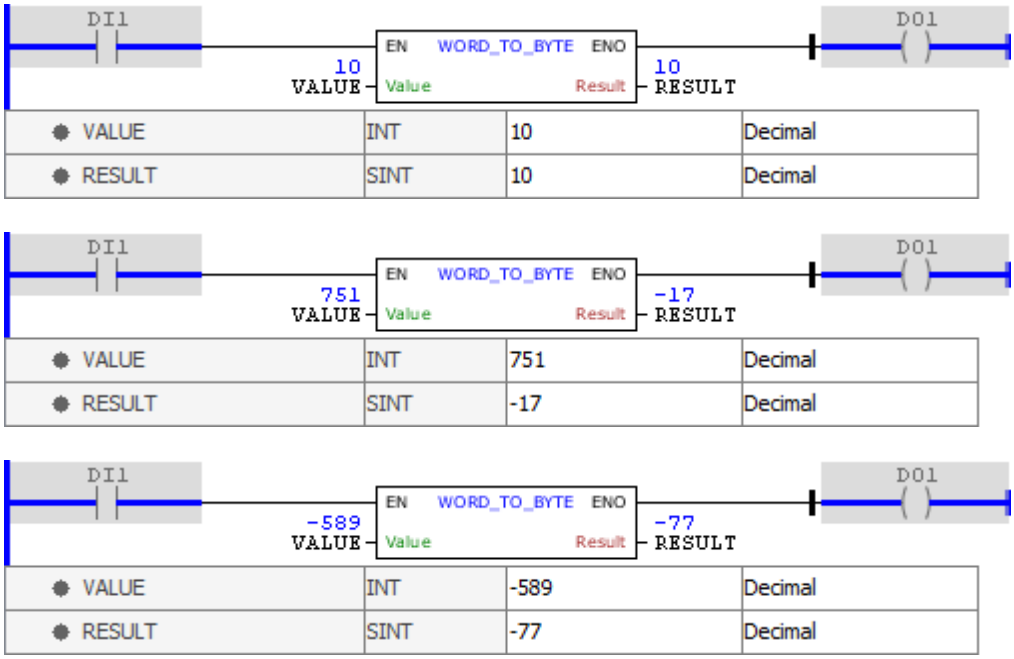
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



The examples above perform the conversion of variable VALUE, in WORD, into a BYTE value storing the final result in RESULT. The block ends with success and ENO output is activated. Notice that only the eight least significant bits are taken into account.

11.10.7.6.3.7 WORD_TO_BYTES

Block that performs the conversion of a 16 bits (WORD) value in two 8 bits (2 BYTES) value.

Ladder Representation

Block Structure

Variable Type	Name	Data type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	WORD UINT INT	Value in WORD
VAR_OUTPUT	ENO	BOOL	End of operation
	Result1	BYTE USINT SINT	Value in BYTE (LSB)
	Result2	BYTE USINT SINT	Value in BYTE (MSB)

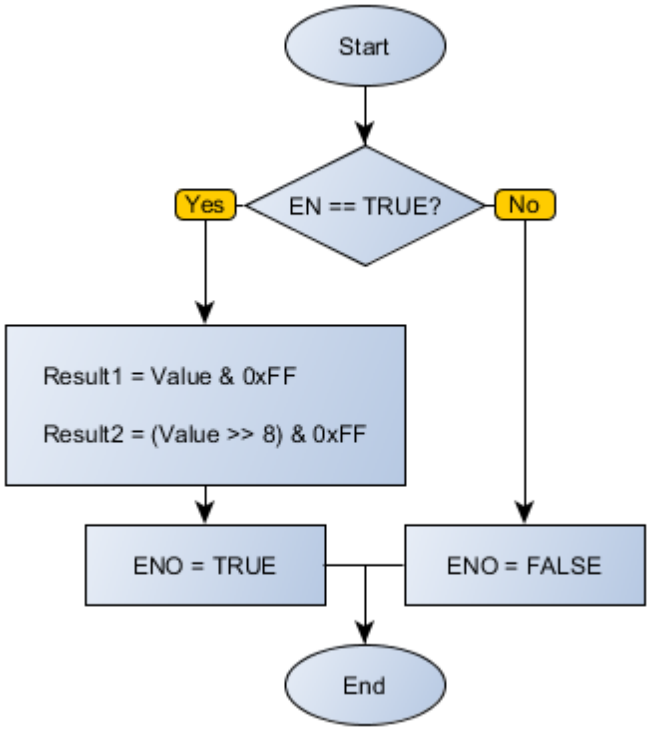
Operation

When this block has a TRUE value in EN, it interprets the Value value as WORD and converts it in two BYTE variables, storing in Result.

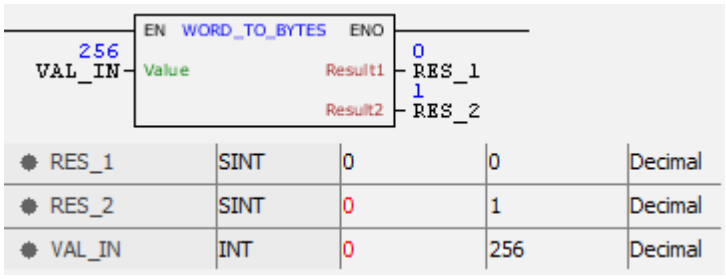
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

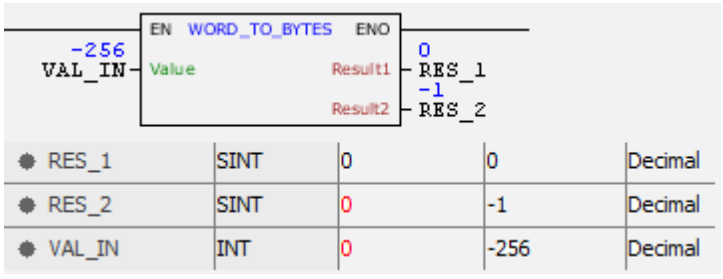
Block Flowchart



Example



RES_1	SINT	0	0	Decimal
RES_2	SINT	0	1	Decimal
VAL_IN	INT	0	256	Decimal



RES_1	SINT	0	0	Decimal
RES_2	SINT	0	-1	Decimal
VAL_IN	INT	0	-256	Decimal

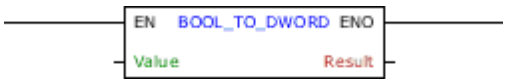
The examples above perform the conversion of variable VALUE "VAL_IN", in WORD, in two BYTE values storing the final result in RESULT1 and RESULT2. The block ends with success and ENO output is activated.

11.10.7.6.4 DWORD

11.10.7.6.4.1 BOOL_TO_DWORD

Block that performs the conversion of a BOOL value into a DWORD value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BOOL	Value in BOOL
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	DWORD UDINT DINT	Value in DWORD

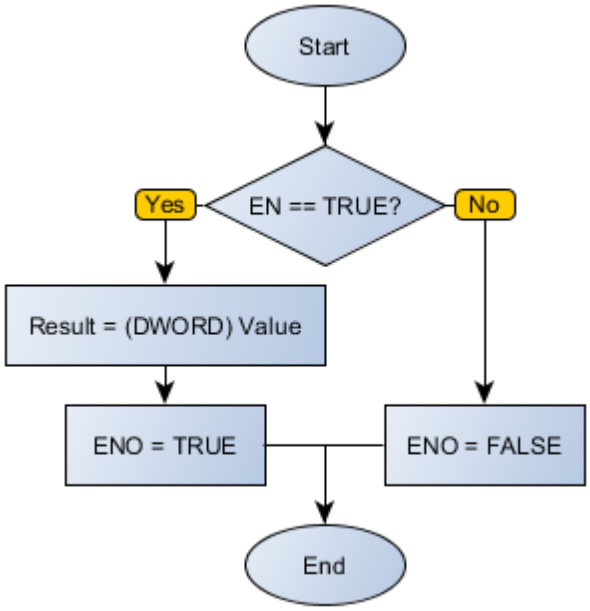
Operation

When this block has a TRUE value in EN, it interprets the Value value as BOOL and converts it into DWORD, storing in Result.

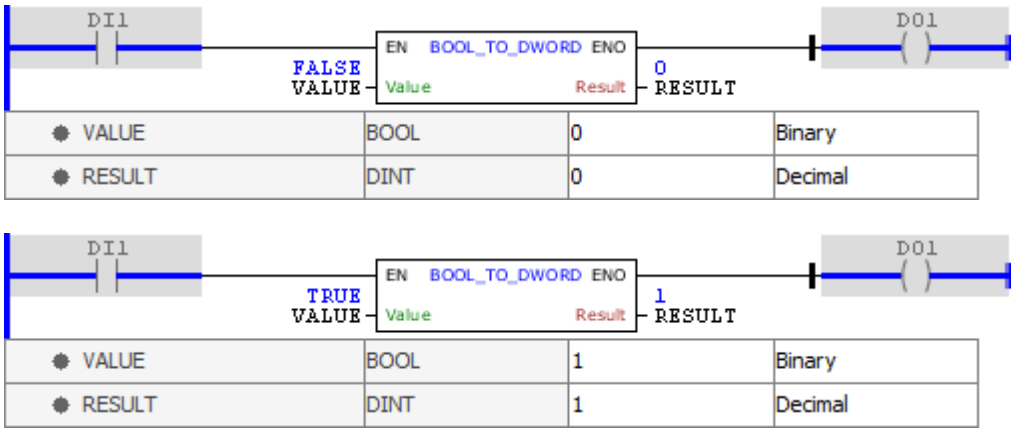
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example

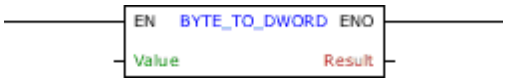


The examples above perform the conversion of VALUE variable, in BOOL, into a DWORD value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.10.7.6.4.2 BYTE_TO_DWORD

Block that performs the conversion of a BYTE value into a DWORD value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT	Value in BYTE
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	DWORD UDINT DINT	Value in DWORD

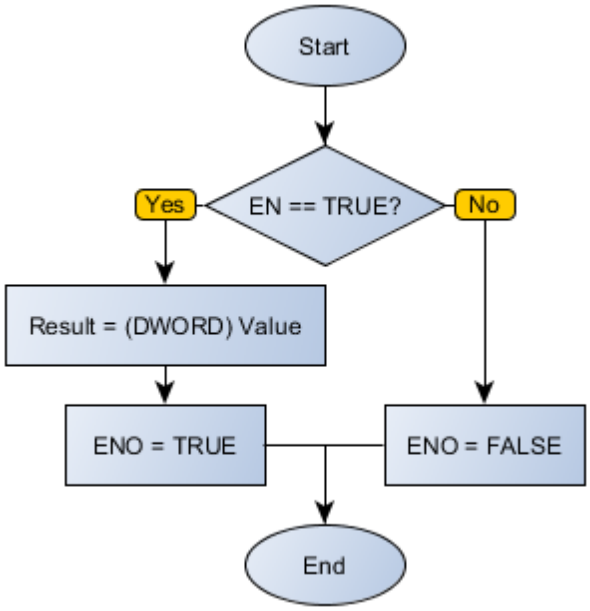
Operation

When this block has a TRUE value in EN, it interprets the Value value as BYTE and converts it into DWORD, storing in Result.

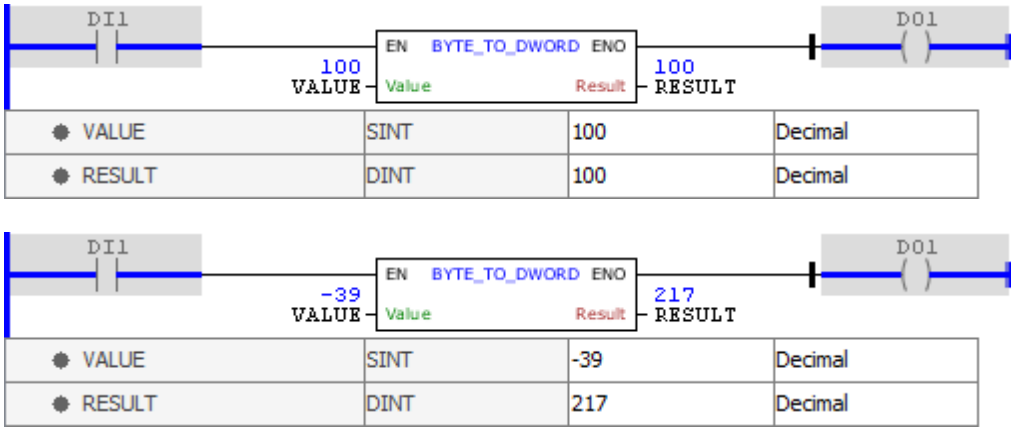
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example

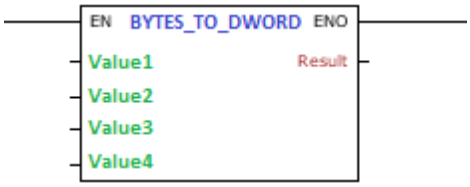


The examples above perform the conversion of variable VALUE, in BYTE, into a DWORD value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.10.7.6.4.3 BYTES_TO_DWORD

Block that performs the conversion of four 8 bits (BYTE) values into a 32 bits (DWORD) value.

Ladder Representation



Block Structure

Variable Type	Name	Data type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT	Value in BYTE (1st byte - LSB)
	Value2	BYTE USINT SINT	Value in BYTE
	Value3	BYTE USINT SINT	Value in BYTE
	Value4	BYTE USINT SINT	Value in BYTE (4th byte - MSB)
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	DWORD UDINT DINT	Value in DWORD

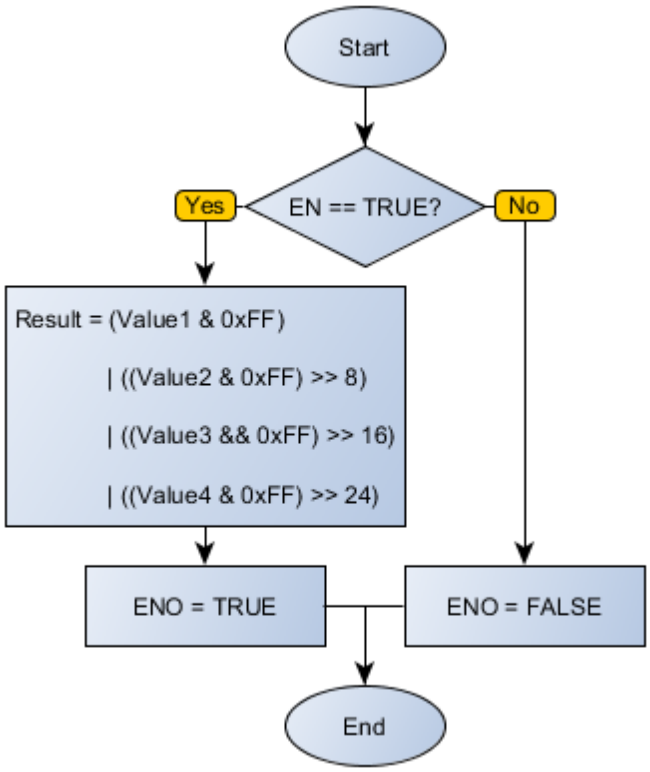
Operation

When this block has a TRUE value in EN, it interprets the Value1, Value2, Value3 and Value4 values as BYTE and converts it into a DWORD variable, storing in Result.

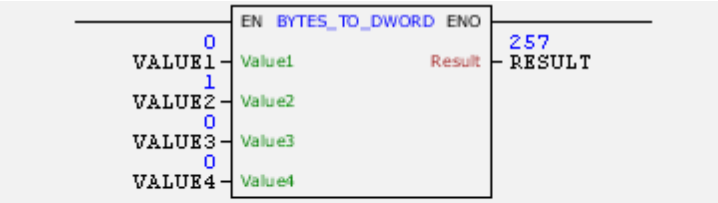
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



VALUE1	SINT	0	0	Decimal
VALUE2	SINT	0	1	Decimal
VALUE3	SINT	0	0	Decimal
VALUE4	SINT	0	0	Decimal
RESULT	DINT	0	257	Decimal

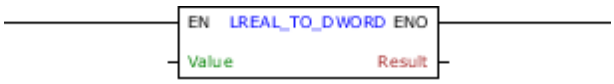
VALUE1	SINT	0	-1	Decimal
VALUE2	SINT	0	0	Decimal
VALUE3	SINT	0	0	Decimal
VALUE4	SINT	0	0	Decimal
RESULT	DINT	0	255	Decimal

The examples above perform the conversion of four variables VALUE1..4, in BYTE, into a DWORD value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.10.7.6.4.4 LREAL_TO_DWORD

Block that performs the conversion of a LREAL value into a DWORD value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	LREAL	Value in LREAL
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	DWORD UDINT DINT	Value in DWORD

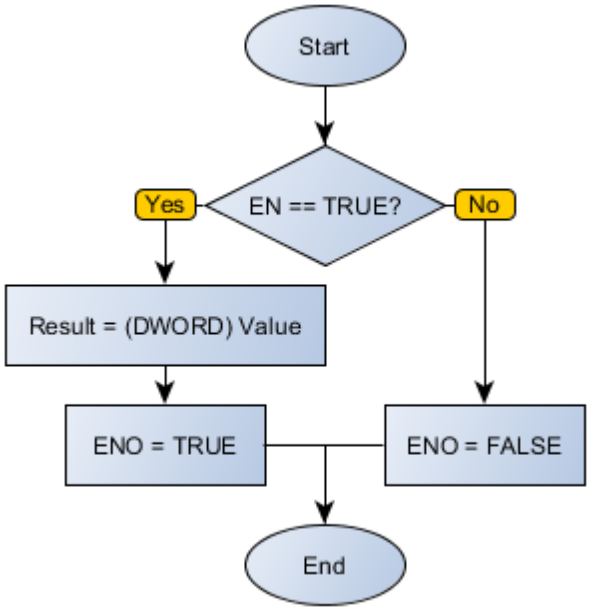
Operation

When this block has a TRUE value in EN, it interprets the Value value as LREAL and converts it into DWORD, storing in Result.

When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

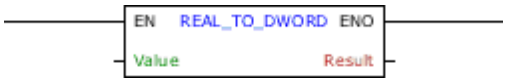
Block Flowchart



11.10.7.6.4.5 REAL_TO_DWORD

Block that performs the conversion of a REAL value into a DWORD value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Value in REAL
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	DWORD UDINT DINT	Value in DWORD

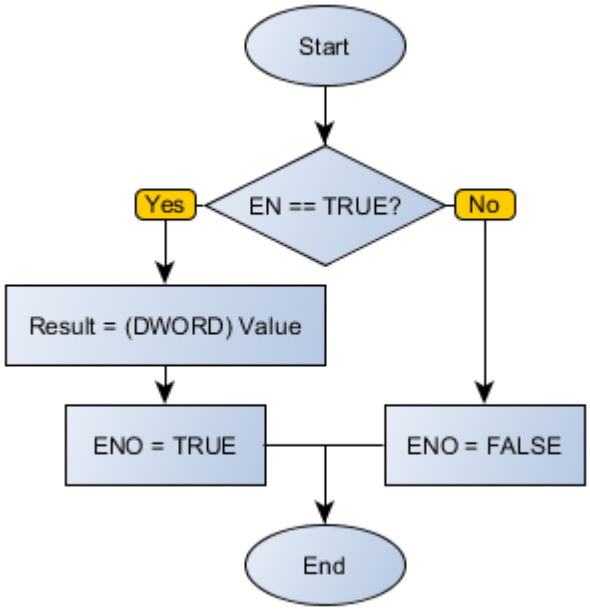
Operation

When this block has a TRUE value in EN, it interprets the Value value as REAL and converts it into DWORD, storing in Result.

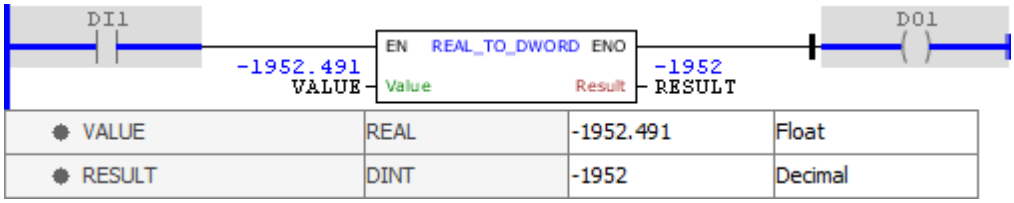
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

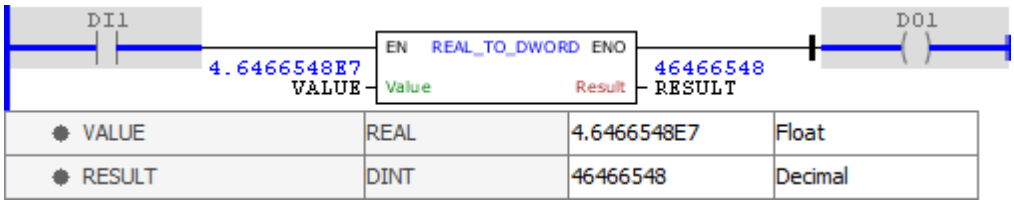
Block Flowchart



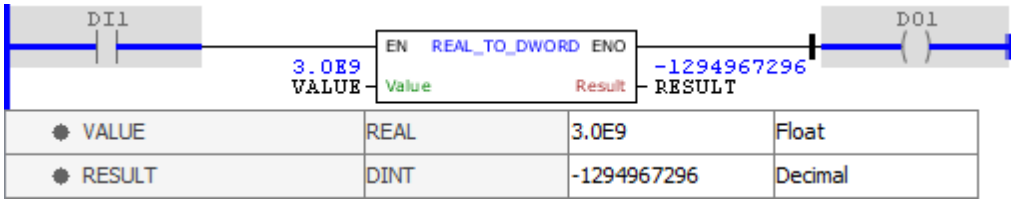
Example



VALUE	REAL	-1952.491	Float
RESULT	DINT	-1952	Decimal



VALUE	REAL	4.6466548E7	Float
RESULT	DINT	46466548	Decimal



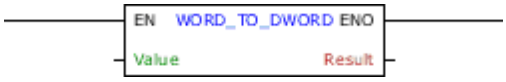
VALUE	REAL	3.0E9	Float
RESULT	DINT	-1294967296	Decimal

The examples above perform the conversion of variable VALUE, in REAL, into a DWORD value storing the final result in RESULT. The block ends with success and ENO output is activated. Note that the results are truncated in decimal and only the thirty-two least significant bits are taken into account.

11.10.7.6.4.6 WORD_TO_DWORD

Block that performs the conversion of a WORD value into a DWORD value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	WORD UINT INT	Value in WORD
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	DWORD UDINT DINT	Value in DWORD

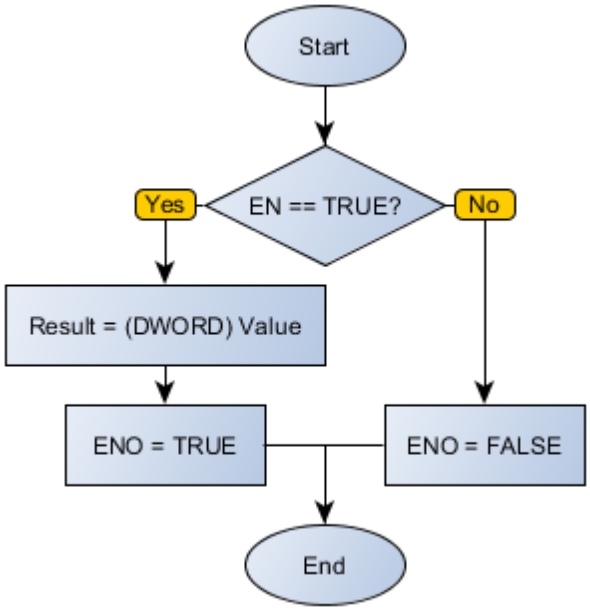
Operation

When this block has a TRUE value in EN, it interprets the Value value as WORD and converts it into DWORD, storing in Result.

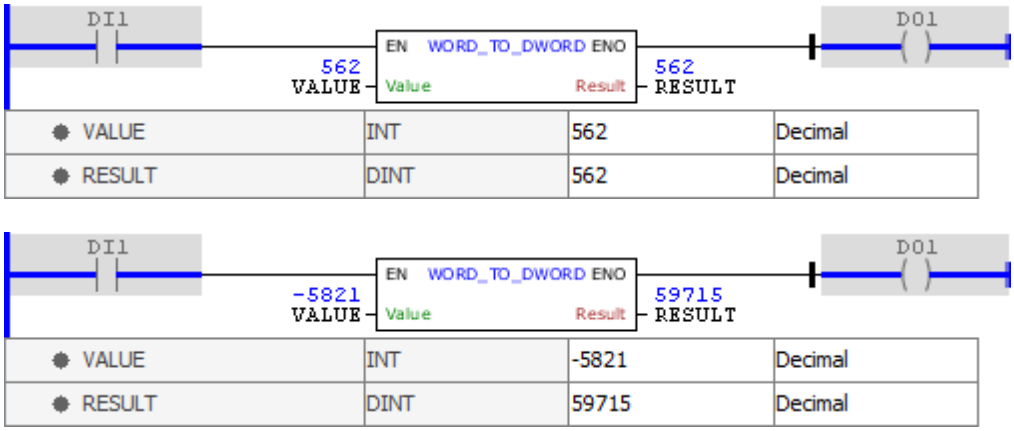
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example

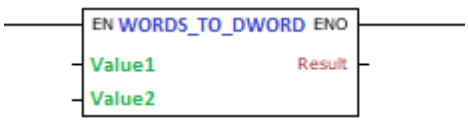


The examples above convert the VALUE variable, in WORD, into a DWORD value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.10.7.6.4.7 WORDS_TO_DWORD

Block that performs the conversion of two 16 bits (WORD) values into a 32 bits (DWORD) value.

Ladder Representation



Block Structure

Variable Type	Name	Data type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	WORD UINT INT	1st WORD (Less Significant Word)
	Value2	WORD UINT INT	2nd WORD (More Significant Word)
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	DWORD UDINT DINT	Value in DWORD

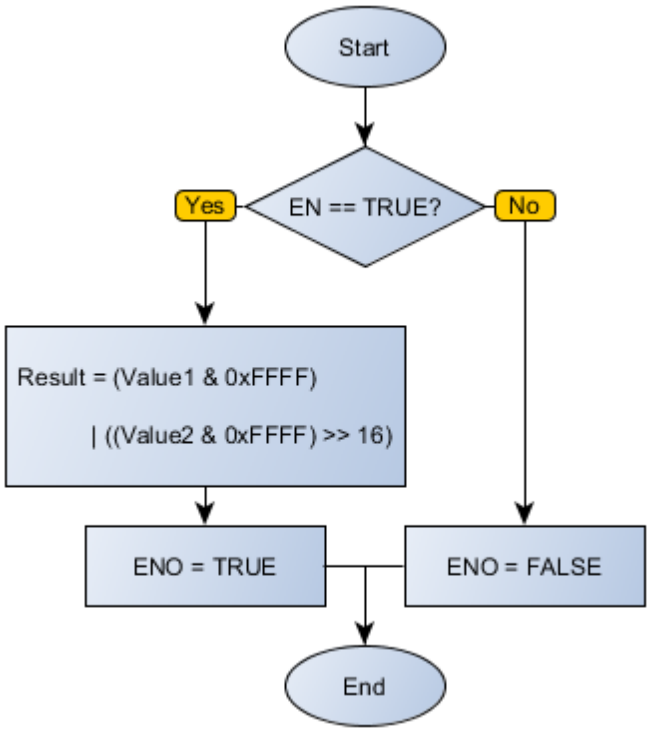
Operation

When this block has a TRUE value in EN, it interprets the Value1 and Value2 values as WORD and converts it into a DWORD variable, storing in Result.

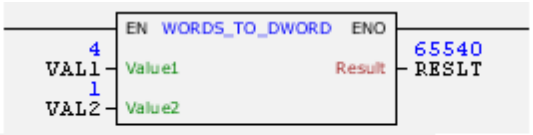
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

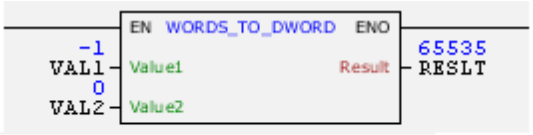
Block Flowchart



Example



RESLT	DINT	0	65540	Decimal
VAL1	INT	0	4	Decimal
VAL2	INT	0	1	Decimal



RESLT	DINT	0	65535	Decimal
VAL1	INT	0	-1	Decimal
VAL2	INT	0	0	Decimal

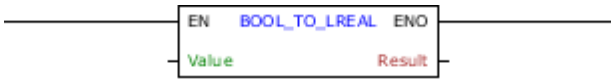
The examples above perform the conversion of two variable VALUE1 and VALUE2, in WORD, into a DWORD value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.10.7.6.5 LREAL

11.10.7.6.5.1 BOOL_TO_LREAL

Block that performs the conversion of a BOOL value into a LREAL value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BOOL	Value in BOOL
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	LREAL	Value in LREAL

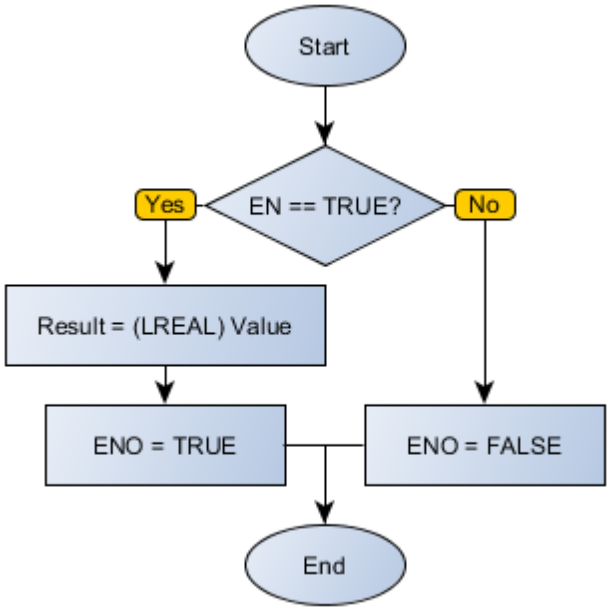
Operation

When this block has a TRUE value in EN, it interprets the Value value as BOOL and converts it into LREAL, storing in Result.

When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

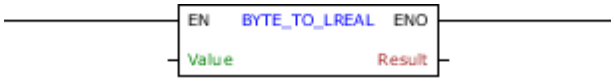
Block Flowchart



11.10.7.6.5.2 BYTE_TO_LREAL

Block that performs the conversion of a BYTE value into a LREAL value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT	Value in BYTE
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	LREAL	Value in LREAL

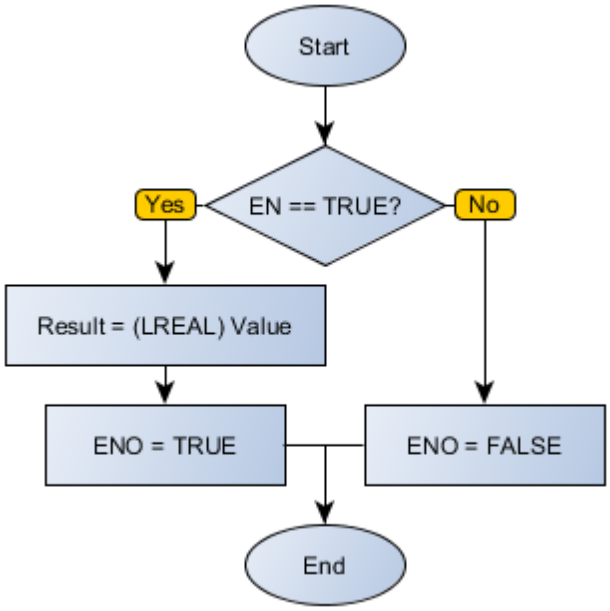
Operation

When this block has a TRUE value in EN, it interprets the Value value as BYTE and converts it into LREAL, storing in Result.

When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

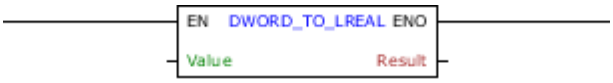
Block Flowchart



11.10.7.6.5.3 DWORD_TO_LREAL

Block that performs the conversion of a DWORD value into a LREAL value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	DWORD UDINT DINT	Value in DWORD
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	LREAL	Value in LREAL

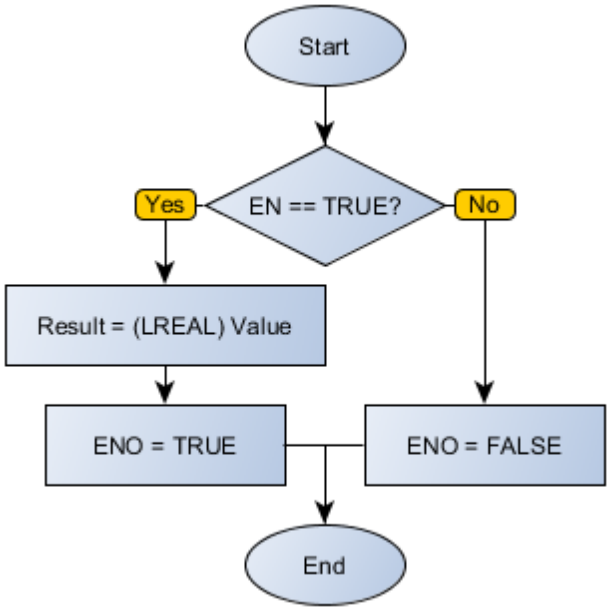
Operation

When this block has a TRUE value in EN, it interprets the Value value as DWORD and converts it into LREAL, storing in Result.

When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

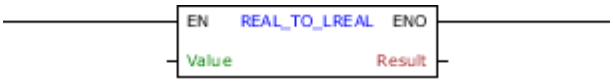
Block Flowchart



11.10.7.6.5.4 REAL_TO_LREAL

Block that performs the conversion of a REAL value into a LREAL value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL USINT SINT	Value in REAL
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	LREAL	Value in LREAL

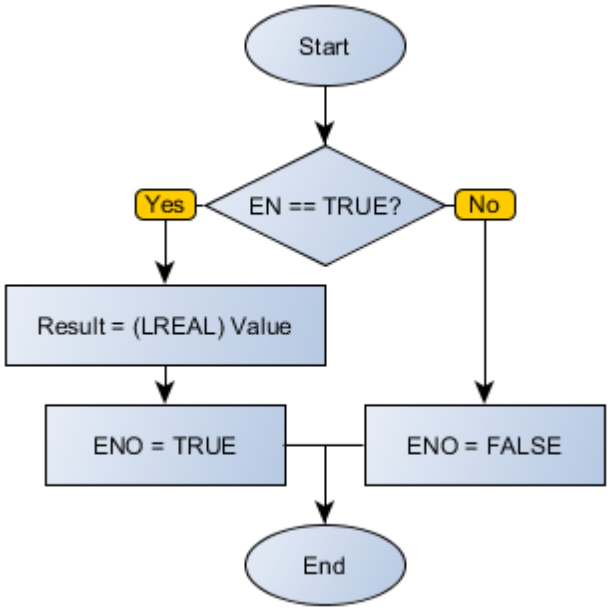
Operation

When this block has a TRUE value in EN, it interprets the Value value as REAL and converts it into LREAL, storing in Result.

When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

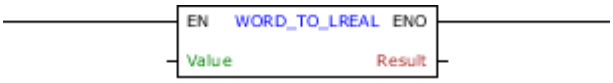
Block Flowchart



11.10.7.6.5.5 WORD_TO_LREAL

Block that performs the conversion of a WORD value into a LREAL value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	WORD UINT INT	Value in WORD
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	LREAL	Value in LREAL

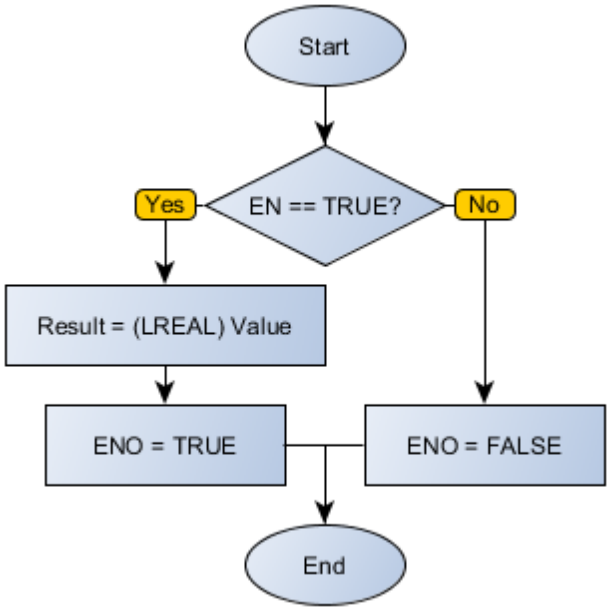
Operation

When this block has a TRUE value in EN, it interprets the Value value as WORD and converts it into LREAL, storing in Result.

When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart

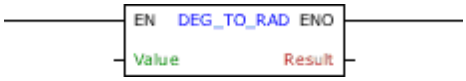


11.10.7.6.6 Rad-Deg

11.10.7.6.6.1 DEG_TO_RAD

Block that performs the conversion of a value in degrees into a value in radians.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Value in degrees
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	REAL	Value in radians

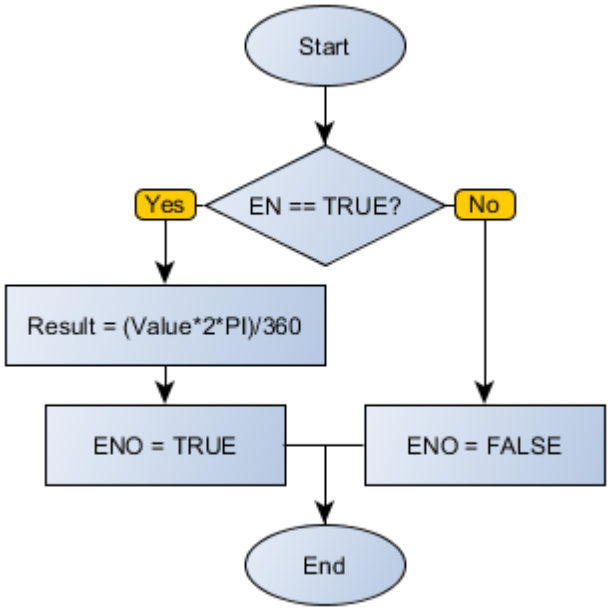
Operation

When this block has a TRUE value in EN, it interprets the Value value as in degrees and converts it into radians, storing in Result.

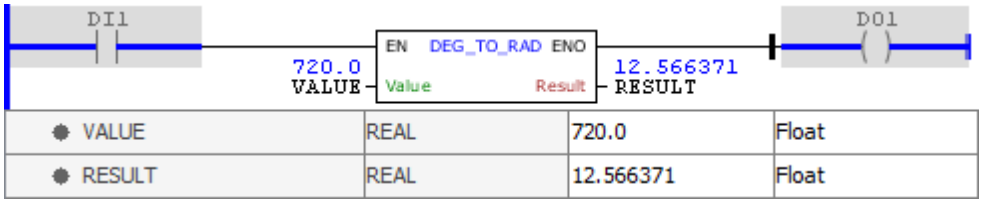
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

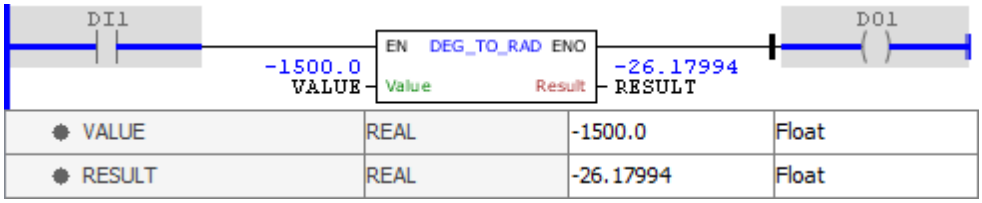
Block Flowchart



Example



VALUE	REAL	720.0	Float
RESULT	REAL	12.566371	Float



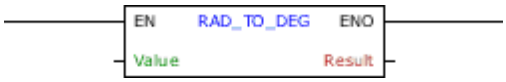
VALUE	REAL	-1500.0	Float
RESULT	REAL	-26.17994	Float

The examples above perform the conversion of variable VALUE, in degrees, into a corresponding value in radians storing the final result in RESULT. The block ends with success and ENO output is activated.

11.10.7.6.6.2 RAD_TO_DEG

Block that performs the conversion of a value in radians into a value in degrees.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Value in radianos
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	REAL	Value in graus

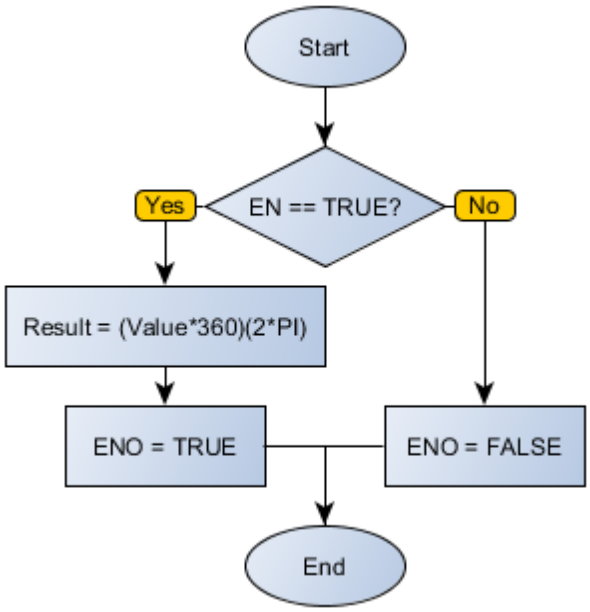
Operation

When this block has a TRUE value in EN, it interprets the Value value as in radians and converts it into degrees, storing in Result.

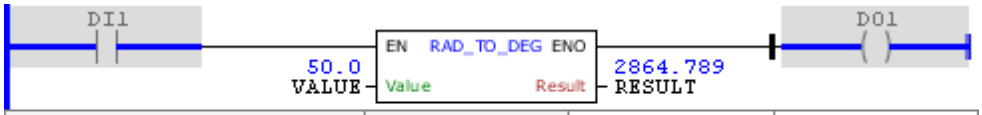
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

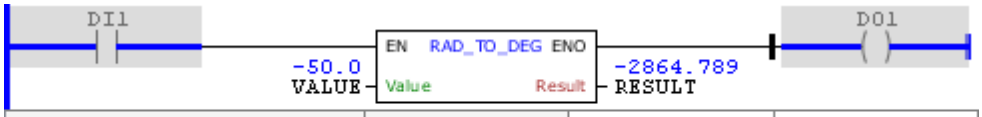
Block Flowchart



Example



VALUE	REAL	50.0	Float
RESULT	REAL	2864.789	Float



VALUE	REAL	-50.0	Float
RESULT	REAL	-2864.789	Float

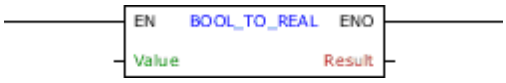
The examples above perform the conversion of variable VALUE, in radians, into a corresponding value in degrees storing the final result in RESULT. The block ends with success and ENO output is activated.

11.10.7.6.7 REAL

11.10.7.6.7.1 BOOL_TO_REAL

Block that performs the conversion of a BOOL value into a REAL value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BOOL	Value in BOOL
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	REAL	Value in REAL

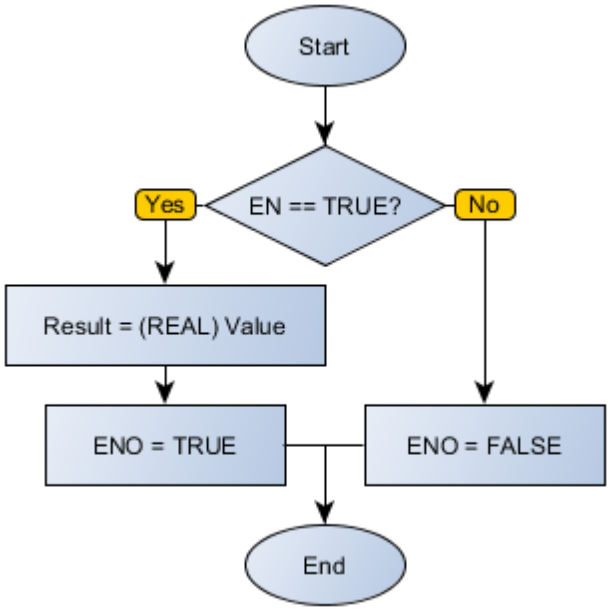
Operation

When this block has a TRUE value in EN, it interprets the Value value as BOOL and converts it into REAL, storing in Result.

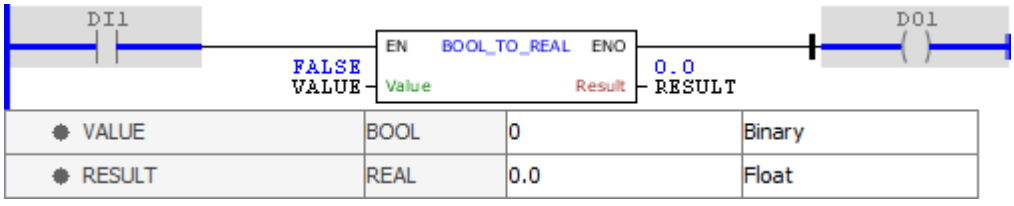
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

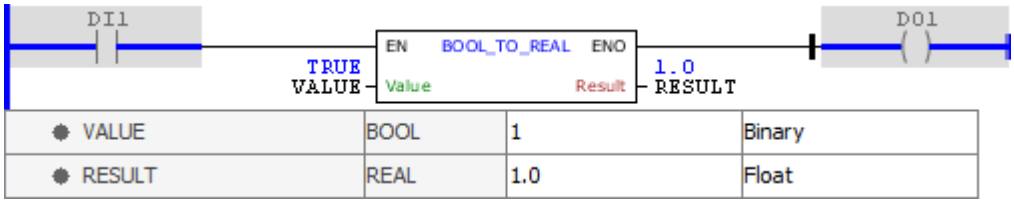
Block Flowchart



Example



VALUE	BOOL	0	Binary
RESULT	REAL	0.0	Float



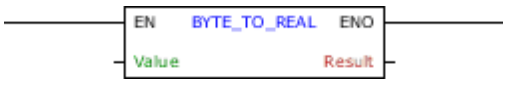
VALUE	BOOL	1	Binary
RESULT	REAL	1.0	Float

The examples above perform the conversion of variable VALUE, in BOOL, into a REAL value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.10.7.6.7.2 BYTE_TO_REAL

Block that performs the conversion of a BYTE value into a REAL value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT	Value in BYTE
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	REAL	Value in REAL

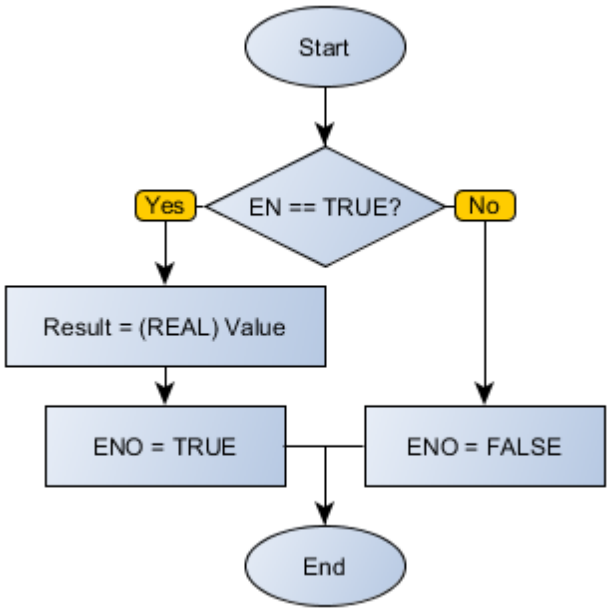
Operation

When this block has a TRUE value in EN, it interprets the Value value as BYTE and converts it into REAL, storing in Result.

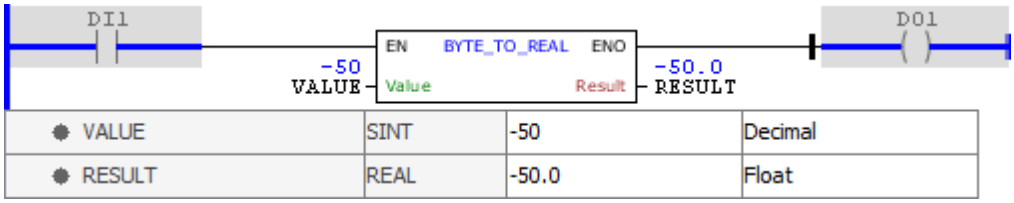
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

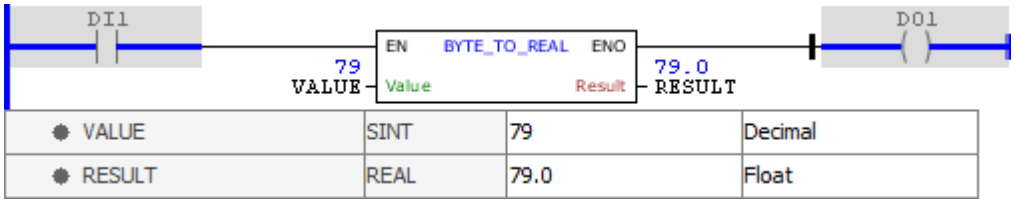
Block Flowchart



Example



VALUE	SINT	-50	Decimal
RESULT	REAL	-50.0	Float



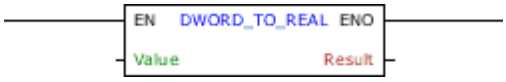
VALUE	SINT	79	Decimal
RESULT	REAL	79.0	Float

The examples above perform the conversion of variable VALUE, in BYTE, into a REAL value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.10.7.6.7.3 DWORD_TO_REAL

Block that performs the conversion of a DWORD value into a REAL value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	DWORD UDINT DINT	Value in DWORD
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	REAL	Value in REAL

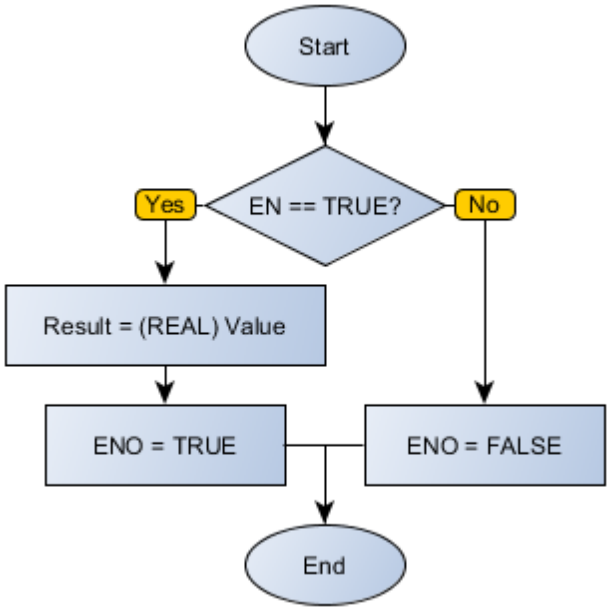
Operation

When this block has a TRUE value in EN, it interprets the Value value as DWORD and converts it into REAL, storing in Result.

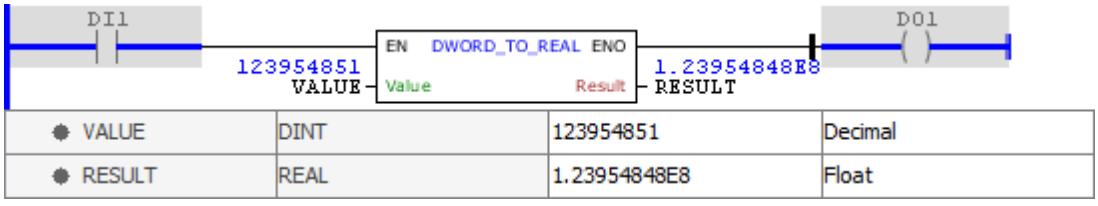
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

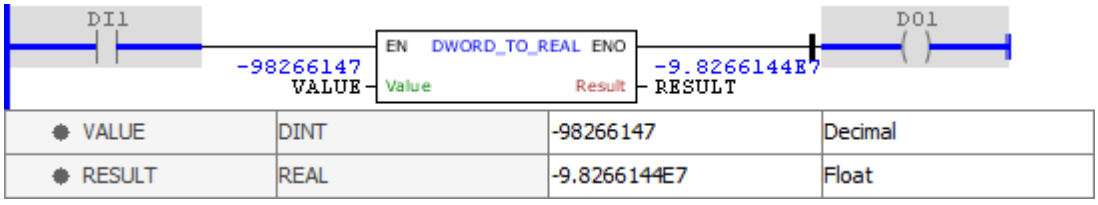
Block Flowchart



Example



VALUE	DINT	123954851	Decimal
RESULT	REAL	1.23954848E8	Float



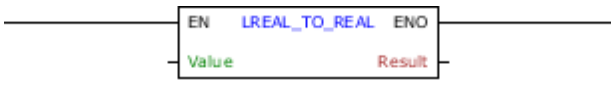
VALUE	DINT	-98266147	Decimal
RESULT	REAL	-9.8266144E7	Float

The examples above perform the conversion of variable VALUE, in DWORD, into a REAL value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.10.7.6.7.4 LREAL_TO_REAL

Block that performs the conversion of a LREAL value into a REAL value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	LREAL USINT SINT	Value in LREAL
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	REAL	Value in REAL

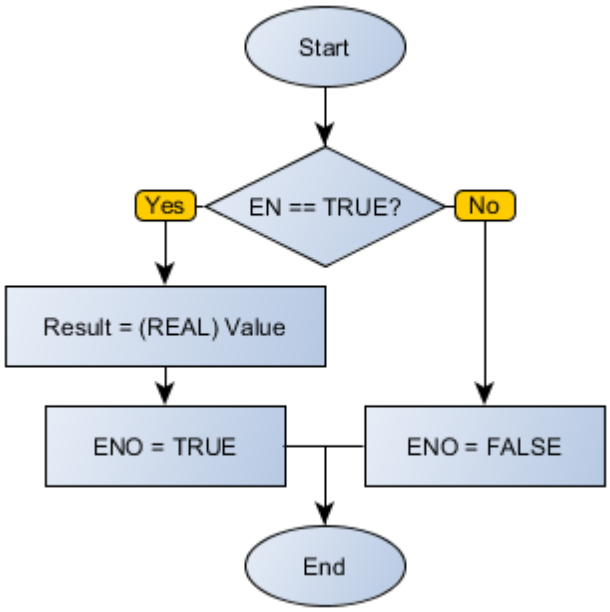
Operation

When this block has a TRUE value in EN, it interprets the Value value as LREAL and converts it into REAL, storing in Result.

When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

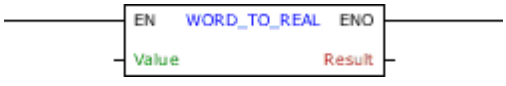
Block Flowchart



11.10.7.6.7.5 WORD_TO_REAL

Block that performs the conversion of a WORD value into a REAL value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	WORD UINT INT	Value in WORD
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	REAL	Value in REAL

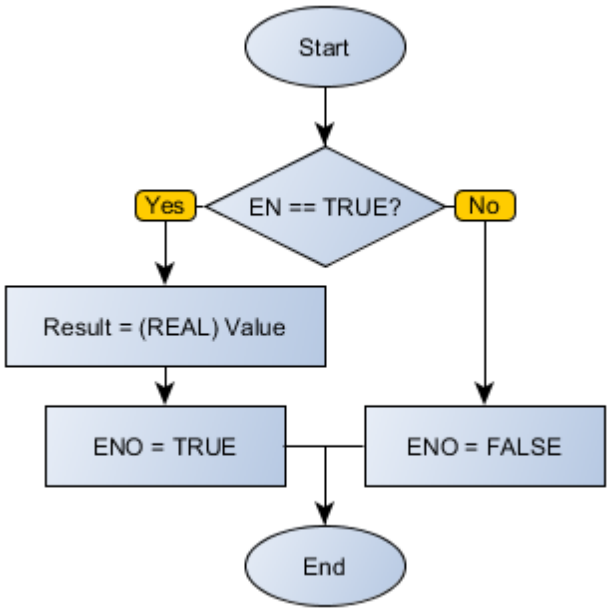
Operation

When this block has a TRUE value in EN, it interprets the Value value as WORD and converts it into REAL, storing in Result.

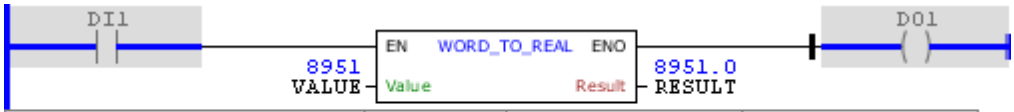
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

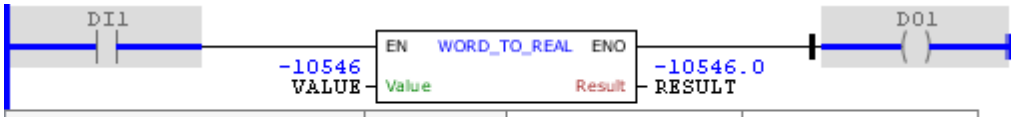
Block Flowchart



Example



VALUE	INT	8951	Decimal
RESULT	REAL	8951.0	Float



VALUE	INT	-10546	Decimal
RESULT	REAL	-10546.0	Float

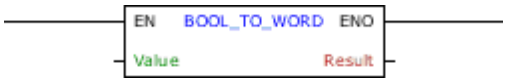
The examples above perform the conversion of variable VALUE, in WORD, into a REAL value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.10.7.6.8 WORD

11.10.7.6.8.1 BOOL_TO_WORD

Block that performs the conversion of a BOOL value into a WORD value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BOOL	Value in BOOL
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	WORD UINT INT	Value in WORD

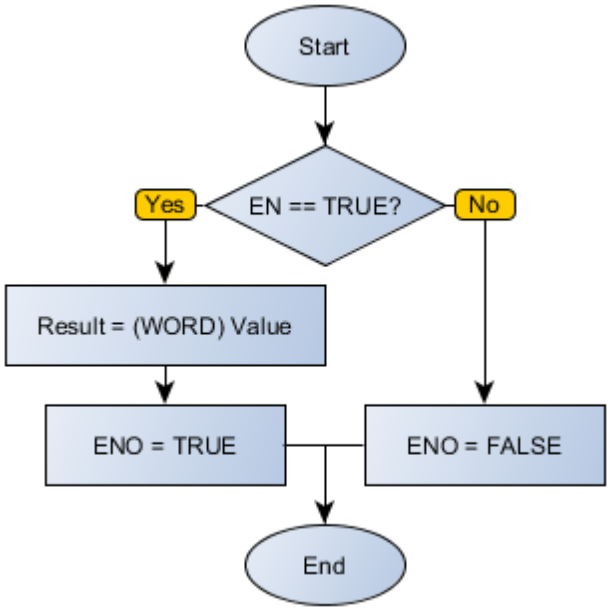
Operation

When this block has a TRUE value in EN, it interprets the Value value as BOOL and converts it into WORD, storing in Result.

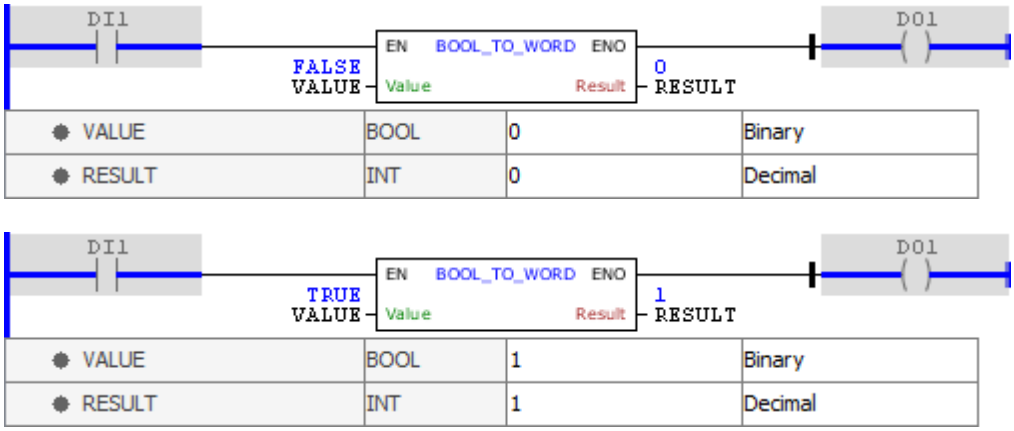
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example

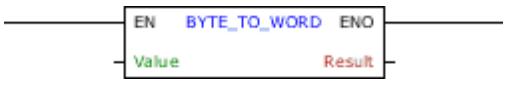


The examples above perform the conversion of VALUE variable, in BOOL, into a WORD value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.10.7.6.8.2 BYTE_TO_WORD

Block that performs the conversion of a BYTE value into a WORD value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT	Value in BYTE
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	WORD UINT INT	Value in WORD

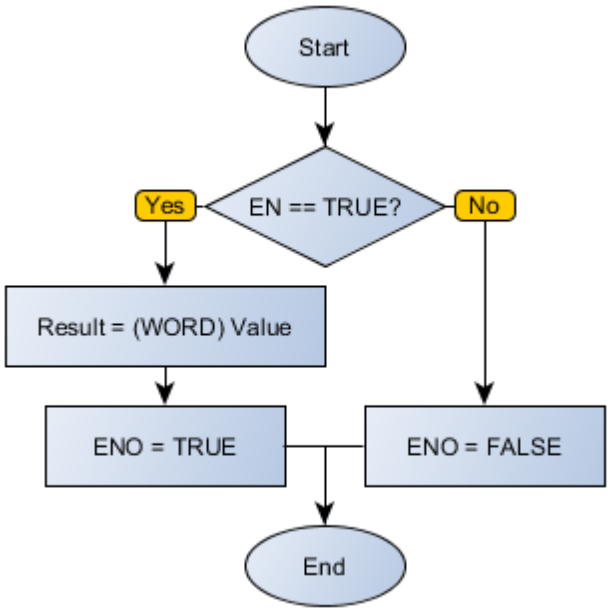
Operation

When this block has a TRUE value in EN, it interprets the Value value as BYTE and converts it into WORD, storing in Result.

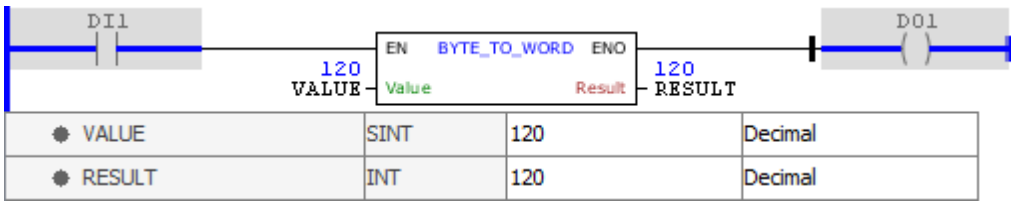
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

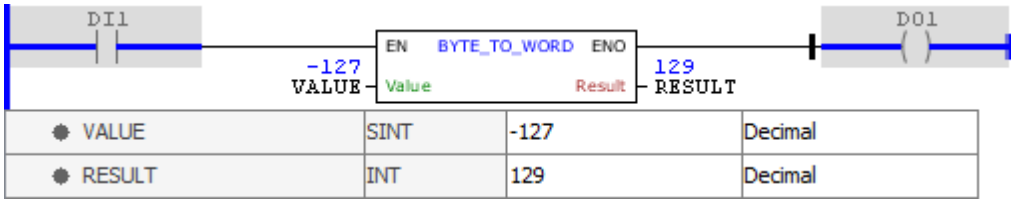
Block Flowchart



Example



VALUE	SINT	120	Decimal
RESULT	INT	120	Decimal



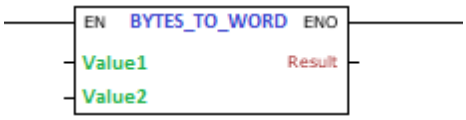
VALUE	SINT	-127	Decimal
RESULT	INT	129	Decimal

The examples above perform the conversion of variable VALUE, in BYTE, into a WORD value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.10.7.6.8.3 BYTES_TO_WORD

Block that performs the conversion of two 8 bits (BYTE) values into a 16 bits (WORD) value.

Ladder Representation



Block Structure

Variable Type	Name	Data type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT	1st BYTE (LSB)
	Value2	BYTE USINT SINT	2nd BYTE (MSB)
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	WORD UINT INT	Value in WORD

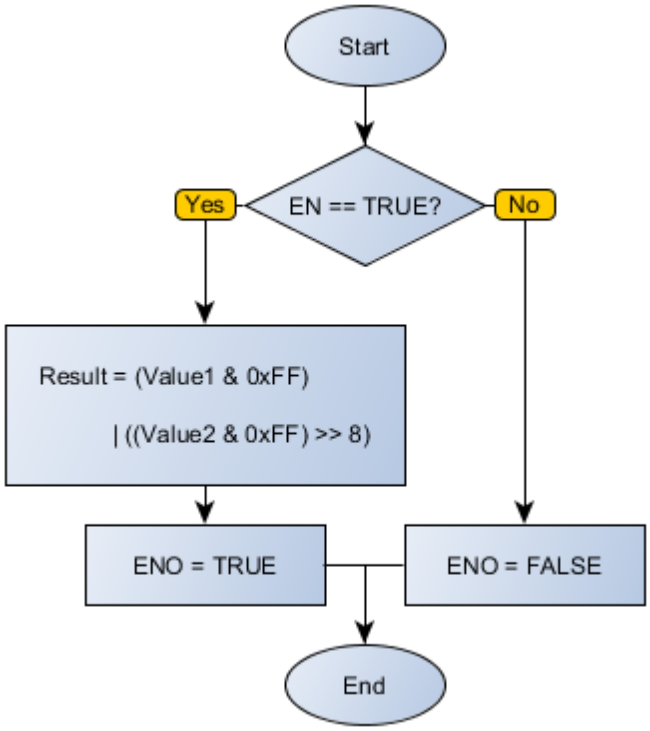
Operation

When this block has a TRUE value in EN, it interprets the Value1 and Value2 values as BYTE and converts it into a WORD variable, storing in Result.

When EN has FALSE value, Result remains unchanged.

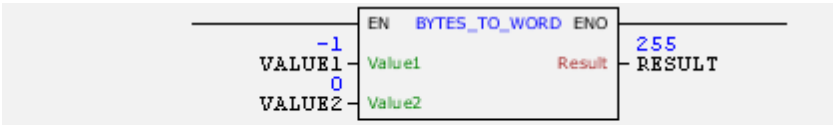
The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example

VALUE1	SINT	0	5	Decimal
VALUE2	SINT	0	10	Decimal



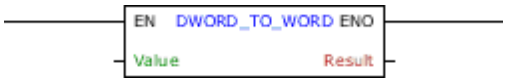
VALUE1	SINT	0	-1	Decimal
VALUE2	SINT	0	0	Decimal

The examples above perform the conversion of two variable VALUE1 and VALUE2, in BYTE, into a WORD value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.10.7.6.8.4 DWORD_TO_WORD

Block that performs the conversion of a DWORD value into a WORD value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	DWORD UDINT DINT	Value in DWORD
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	WORD UINT INT	Value in WORD

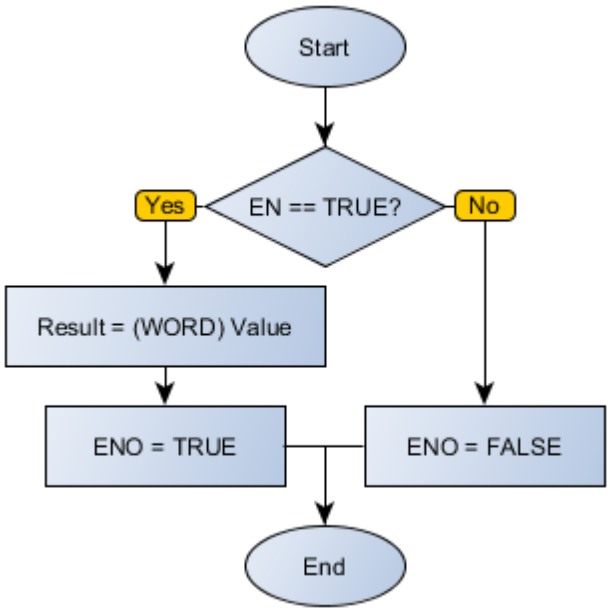
Operation

When this block has a TRUE value in EN, it interprets the Value value as DWORD and converts it into WORD, storing in Result.

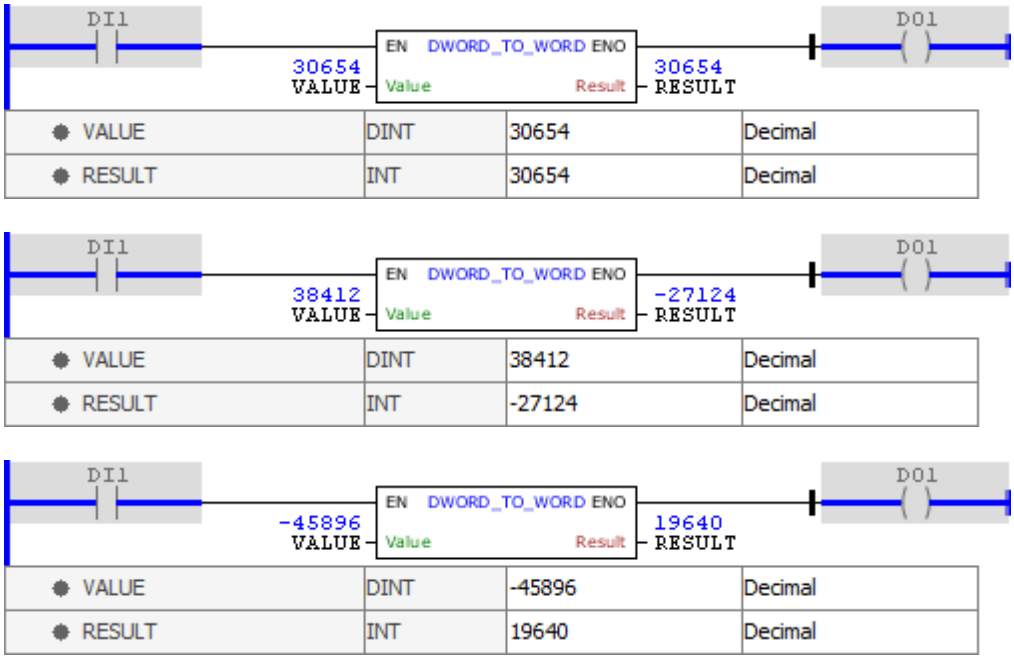
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example

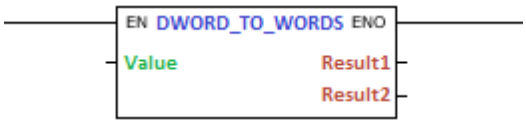


The examples above convert the VALUE variable, in DWORD, into a WORD value storing the final result in RESULT. The block ends with success and ENO output is activated. Notice that only the sixteen least significant bits are taken into account.

11.10.7.6.8.5 DWORD_TO_WORDS

Block that performs the conversion of a 32 bits (DWORD) value in two 16 bits (2 WORD) value.

Ladder Representation



Block Structure

Variable Type	Name	Data type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	DWORD UDINT DINT	Value in DWORD
VAR_OUTPUT	ENO	BOOL	End of operation
	Result1	WORD UINT INT	Value in WORD (Less Significant Word)
	Result2	WORD UINT INT	Value in WORD (More Significant Word)

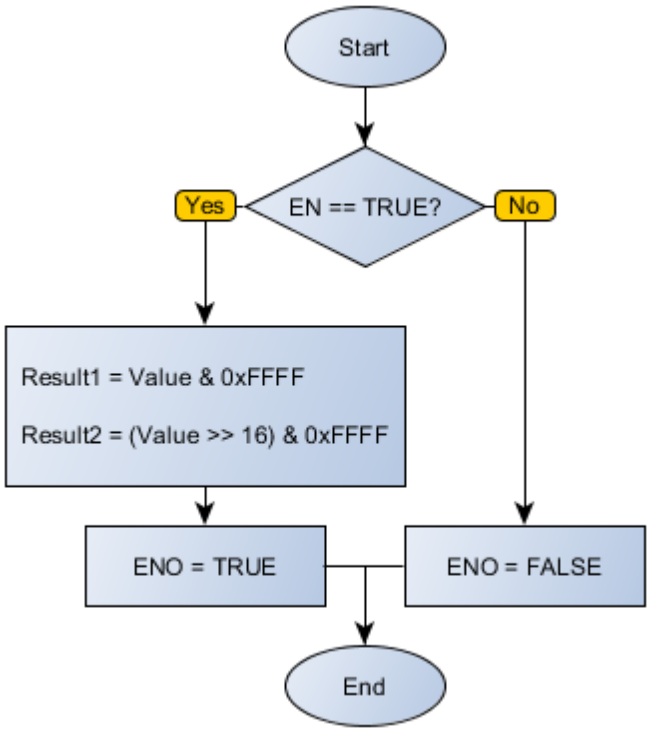
Operation

When this block has a TRUE value in EN, it interprets the value as DWORD and converts it in two WORD variables (Result1 and Result2), storing in Result.

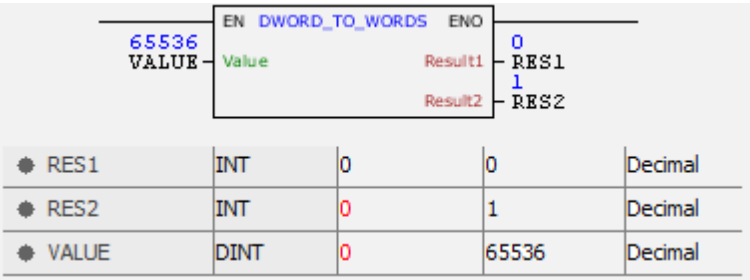
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

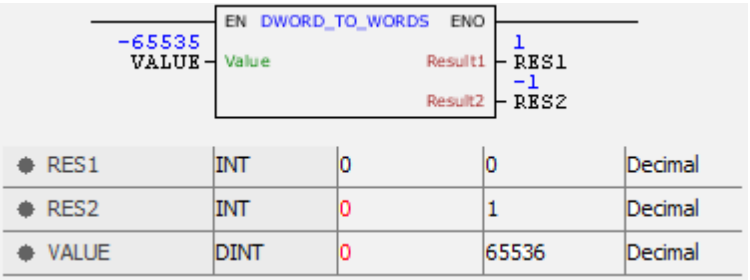
Block Flowchart



Example



RES1	INT	0	0	Decimal
RES2	INT	0	1	Decimal
VALUE	DINT	0	65536	Decimal



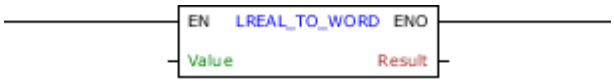
RES1	INT	0	0	Decimal
RES2	INT	0	1	Decimal
VALUE	DINT	0	65536	Decimal

The examples above perform the conversion of a variable VALUE, in DWORD, in two WORD values storing the final result in RESULT1 and RESULT2. The block ends with success and ENO output is activated.

11.10.7.6.8.6 LREAL_TO_WORD

Block that performs the conversion of a LREAL value into a WORD value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	LREAL	Value in LREAL
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	WORD UINT INT	Value in WORD

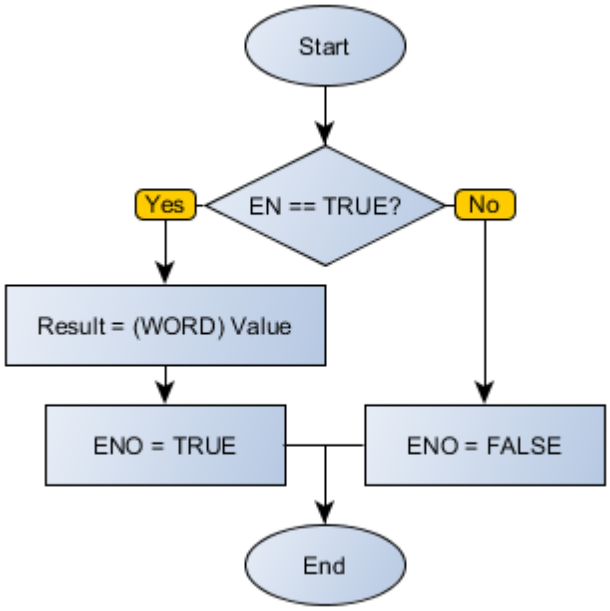
Operation

When this block has a TRUE value in EN, it interprets the Value value as LREAL and converts it into WORD, storing in Result.

When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

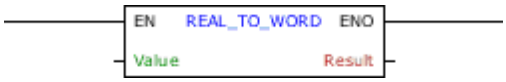
Block Flowchart



11.10.7.6.8.7 REAL_TO_WORD

Block that performs the conversion of a REAL value into a WORD value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Value in REAL
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	WORD UINT INT	Value in WORD

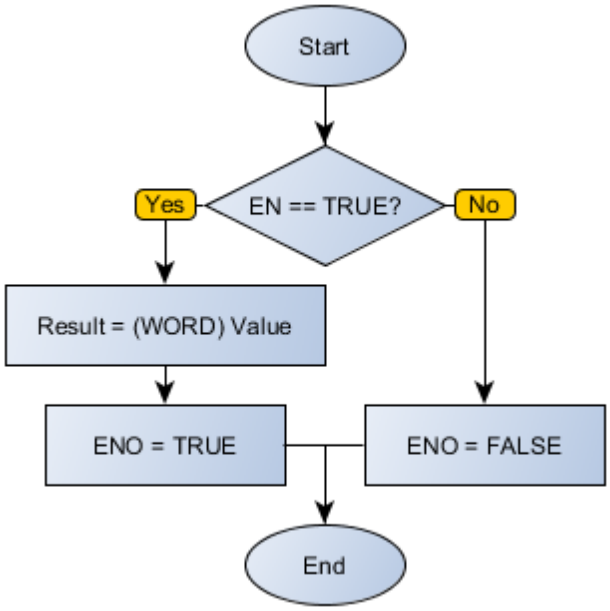
Operation

When this block has a TRUE value in EN, it interprets the Value value as REAL and converts it into WORD, storing in Result.

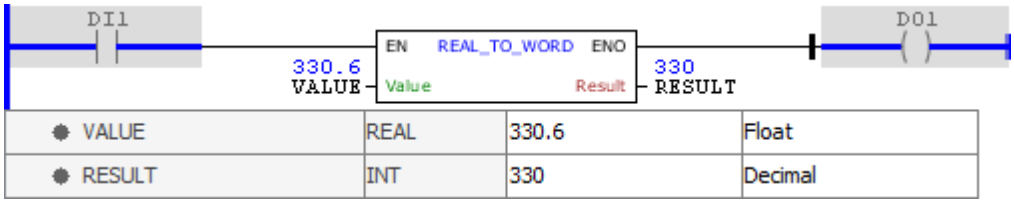
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

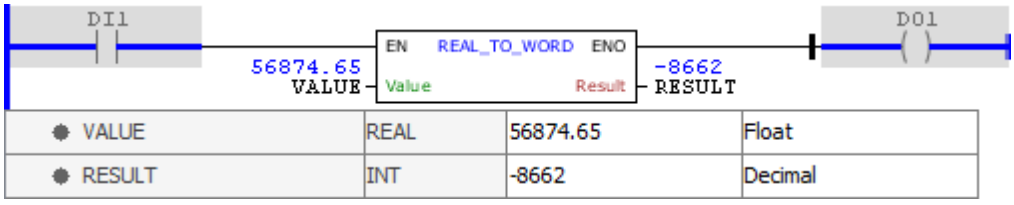
Block Flowchart



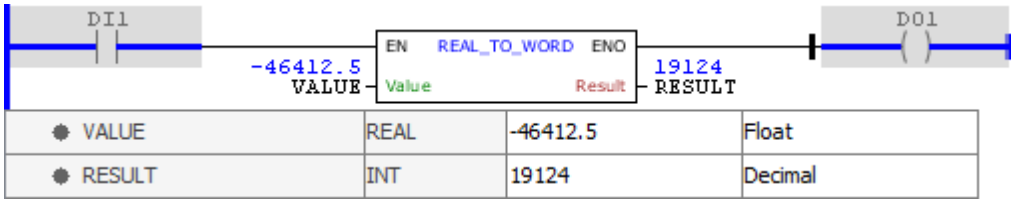
Example



VALUE	REAL	330.6	Float
RESULT	INT	330	Decimal



VALUE	REAL	56874.65	Float
RESULT	INT	-8662	Decimal



VALUE	REAL	-46412.5	Float
RESULT	INT	19124	Decimal

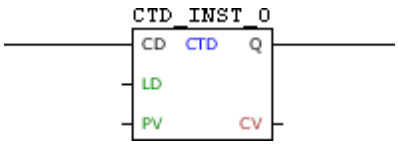
The examples above convert the VALUE variable, in DWORD, into a WORD value storing the final result in RESULT. The block ends with success and ENO output is activated. Note that the results are truncated in decimal and only the sixteen least significant bits are taken into account.

11.10.7.7 Counter

11.10.7.7.1 CTD

Countdown block of input pulses.

Ladder Representation



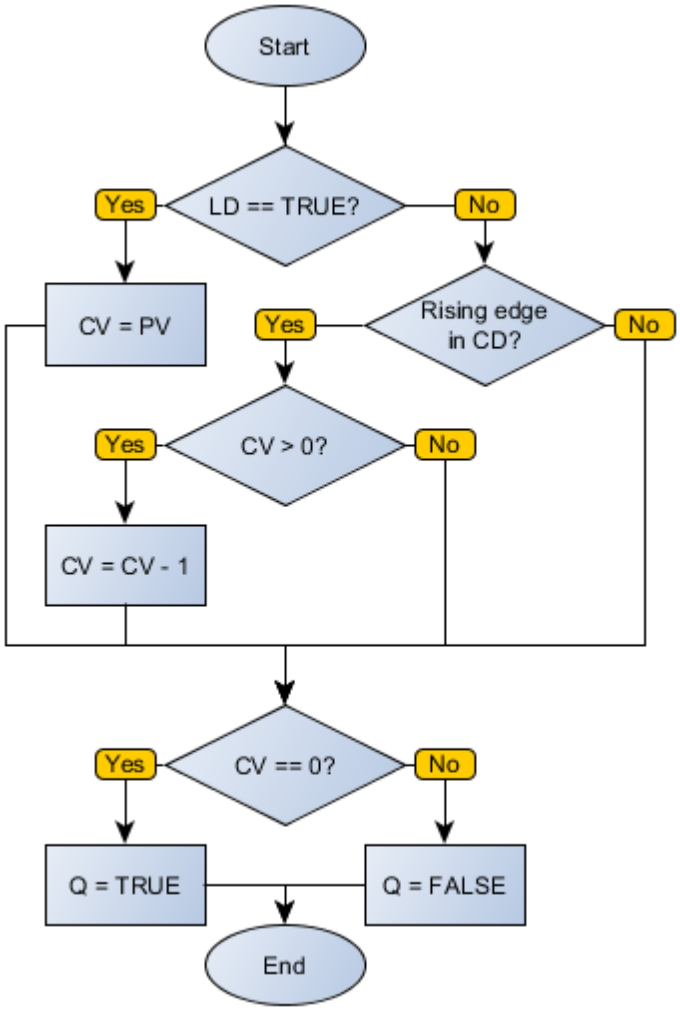
Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	CD	BOOL	Pulse identifier
	LD	BOOL	Loads the value of PV in CV
	PV	WORD UINT	Value of initial configuration
VAR_OUTPUT	Q	BOOL	Counter zeroed flag
	CV	WORD UINT	Current count value
VAR	CTD_INST_0	CTD	Instance of access to block structure

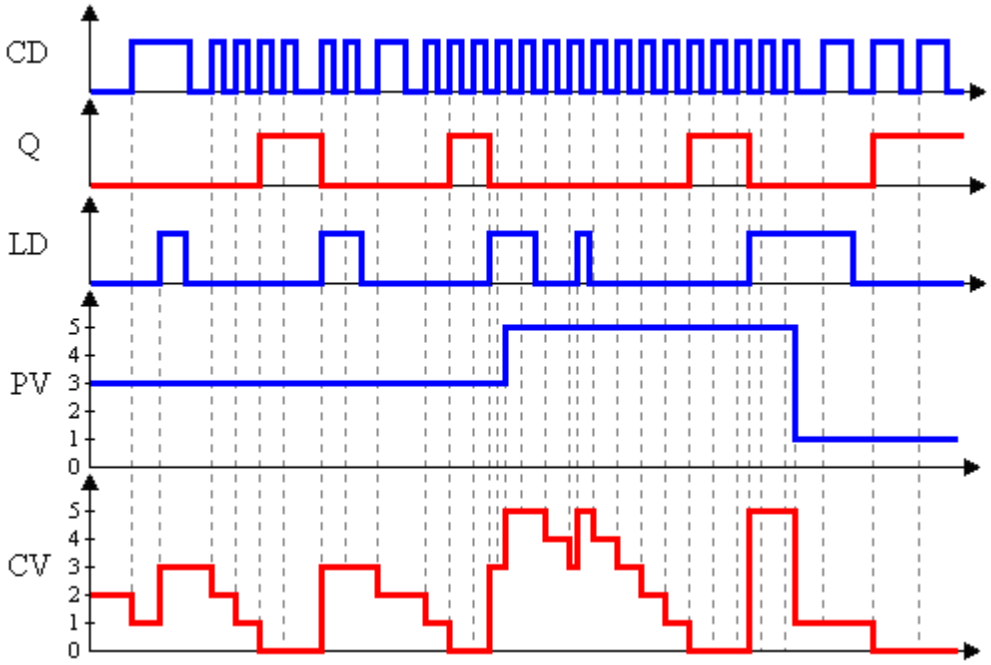
Operation

When this block identifies a leading edge in CD, it decrements the CV variable until it is zero. While CV equals zero, the output Q remains at TRUE level. By detecting high-level LD, the block loads the PV value in CV.

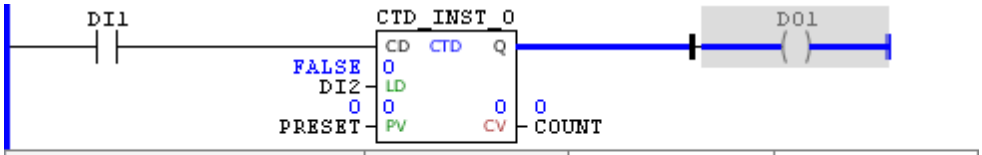
Block Flowchart



Operation Diagram

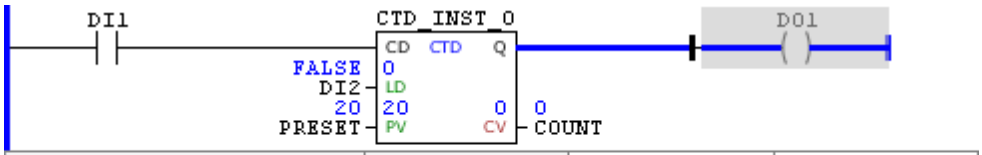


Example



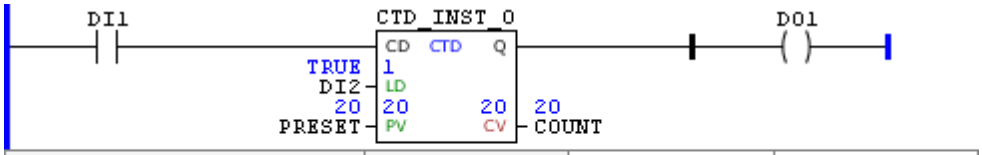
PRESET	UINT	0	Decimal
COUNT	UINT	0	Decimal

The above example shows the initial conditions of routine. As CV has a value of zero, the Q output is enabled.



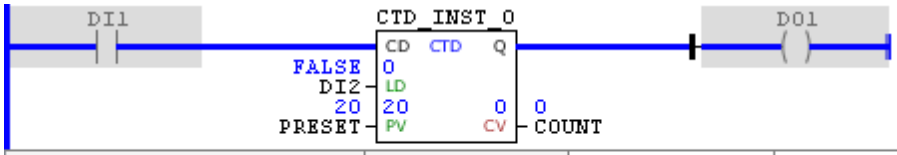
PRESET	UINT	20	Decimal
COUNT	UINT	0	Decimal

The value of the PV variable was changed to 20, but not yet loaded.



PRESET	UINT	20	Decimal
COUNT	UINT	20	Decimal

By identifying TRUE level in LD, the block loads the PV value to CV. Since this value is greater than zero, the Q output is disabled.



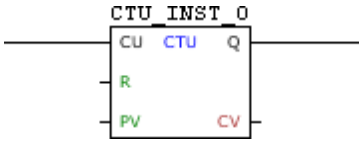
PRESET	UINT	20	Decimal
COUNT	UINT	0	Decimal

At each leading edge identified in CD, the value of COUNT is decremented until it reaches zero, when the Q output is enabled.

11.10.7.7.2 CTU

Block for gradual count of input pulses.

Ladder Representation



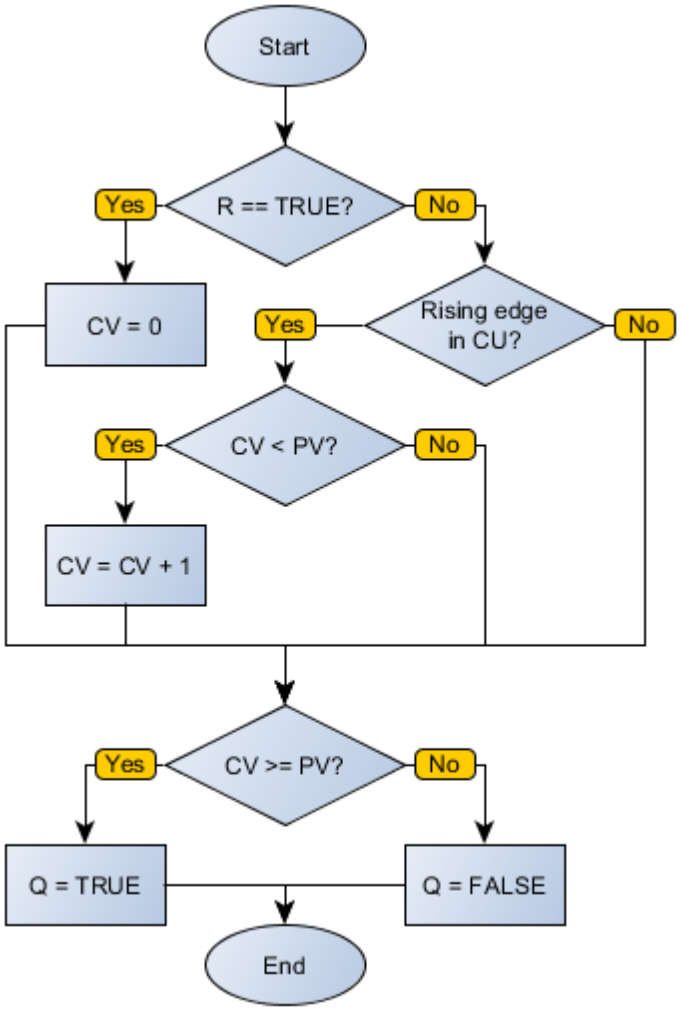
Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	CU	BOOL	Pulse identifier
	R	BOOL	Loads the zero value in CV
	PV	WORD UINT	Maximum count value
VAR_OUTPUT	Q	BOOL	Counter overrun flag
	CV	WORD UINT	Current count value
VAR	CTU_INST_0	CTU	Instance of access to block structure

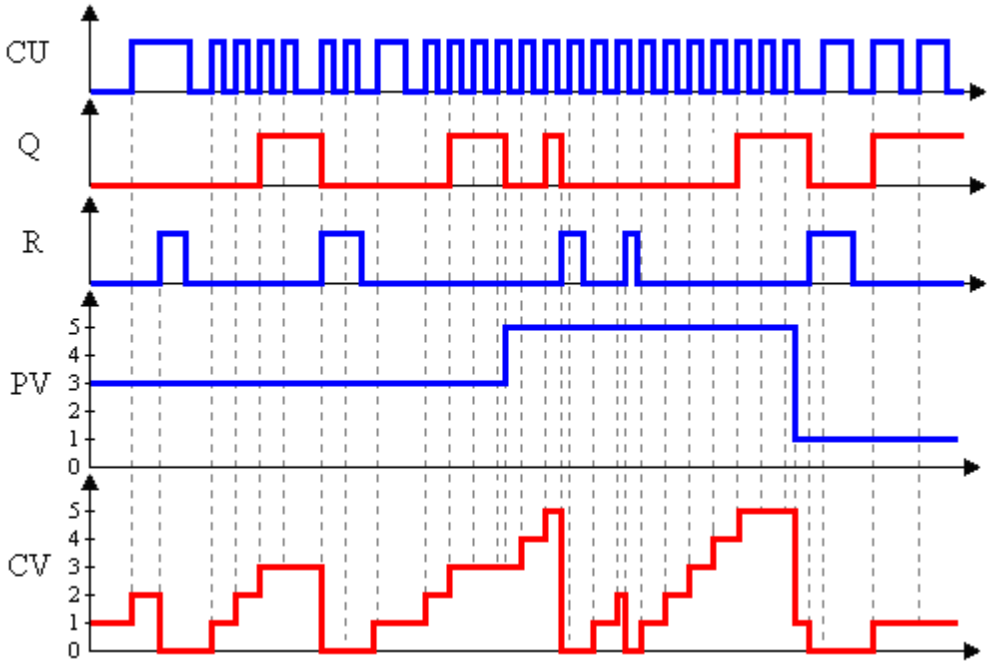
Operation

When this block identifies a leading edge in CD, it increments the CV variable until it is equal to PV. While CV equals PV, the output Q remains at TRUE level. By detecting high-level R, the block loads the zero value in CV.

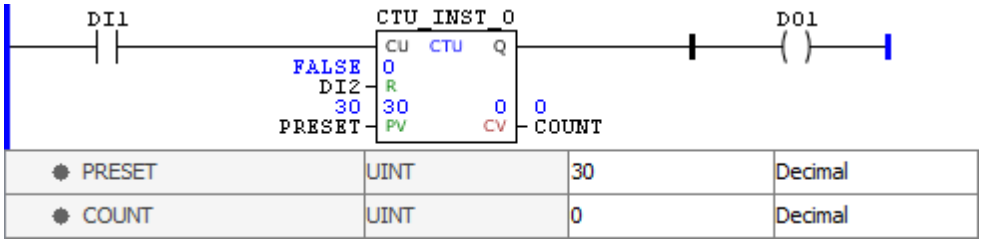
Block Flowchart



Operation Diagram

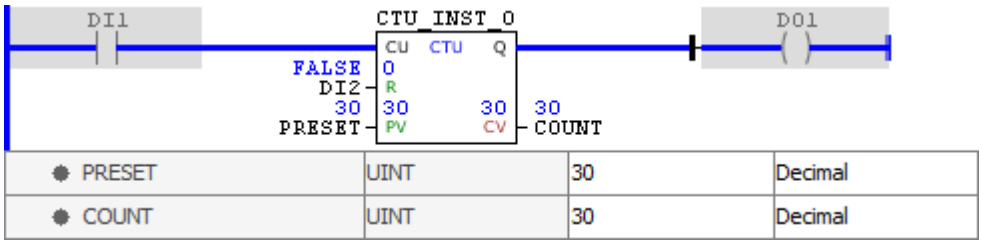


Example



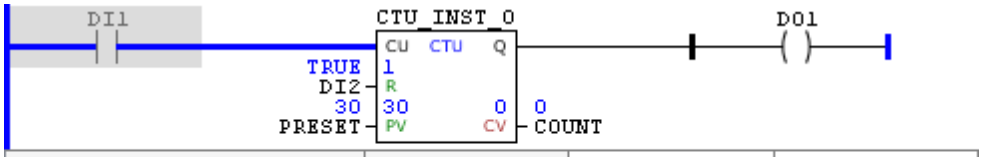
PRESET	UINT	30	Decimal
COUNT	UINT	0	Decimal

The above example shows the initial conditions of routine. Since CV has a lower value than of PV, the Q output is disabled.



PRESET	UINT	30	Decimal
COUNT	UINT	30	Decimal

At each leading edge identified in CU, the value of CV is incremented until it reaches the PV value, when the Q output is enabled.



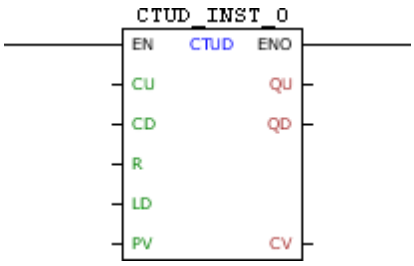
PRESET	UINT	30	Decimal
COUNT	UINT	0	Decimal

By identifying TRUE level in R, the block loads the zero value to CV. Since this value is lower than of PV, the Q output is disabled.

11.10.7.7.3 CTUD

Block for gradual count and countdown of input pulses.

Ladder Representation



Block Structure

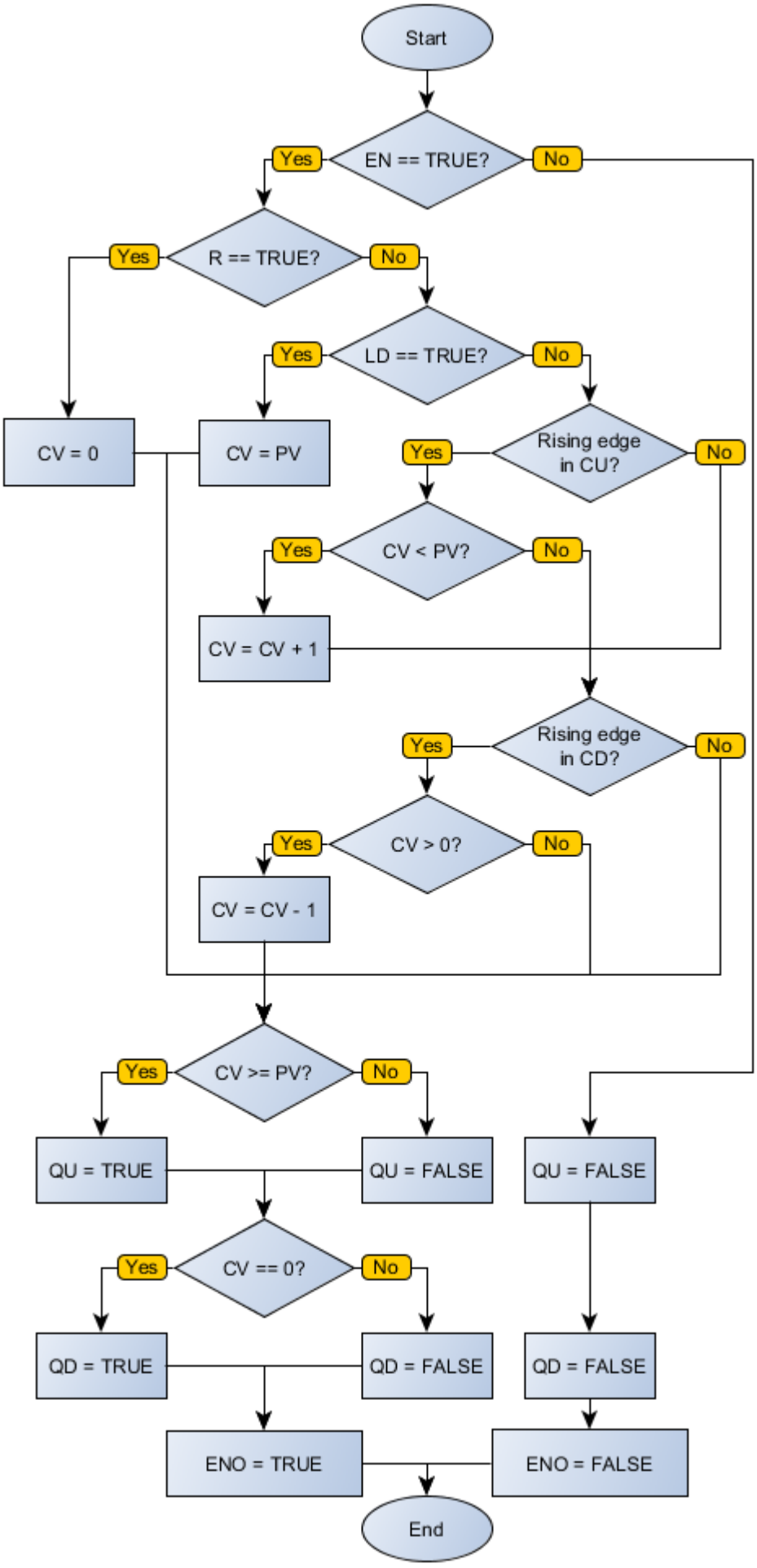
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	CU	BOOL	Pulse identifier for incremental
	CD	BOOL	Pulse identifier for decremental
	R	BOOL	Loads the zero value in CV
	LD	BOOL	Loads the value of PV in CV
	PV	WORD UINT	Reference value
VAR_OUTPUT	ENO	BOOL	Output enabling
	QU	BOOL	Counter overrun flag
	QD	BOOL	Counter zeroed flag
	CV	WORD UINT	Current count value
VAR	CTUD_INST_0	CTUD	Instance of access to block structure

Operation

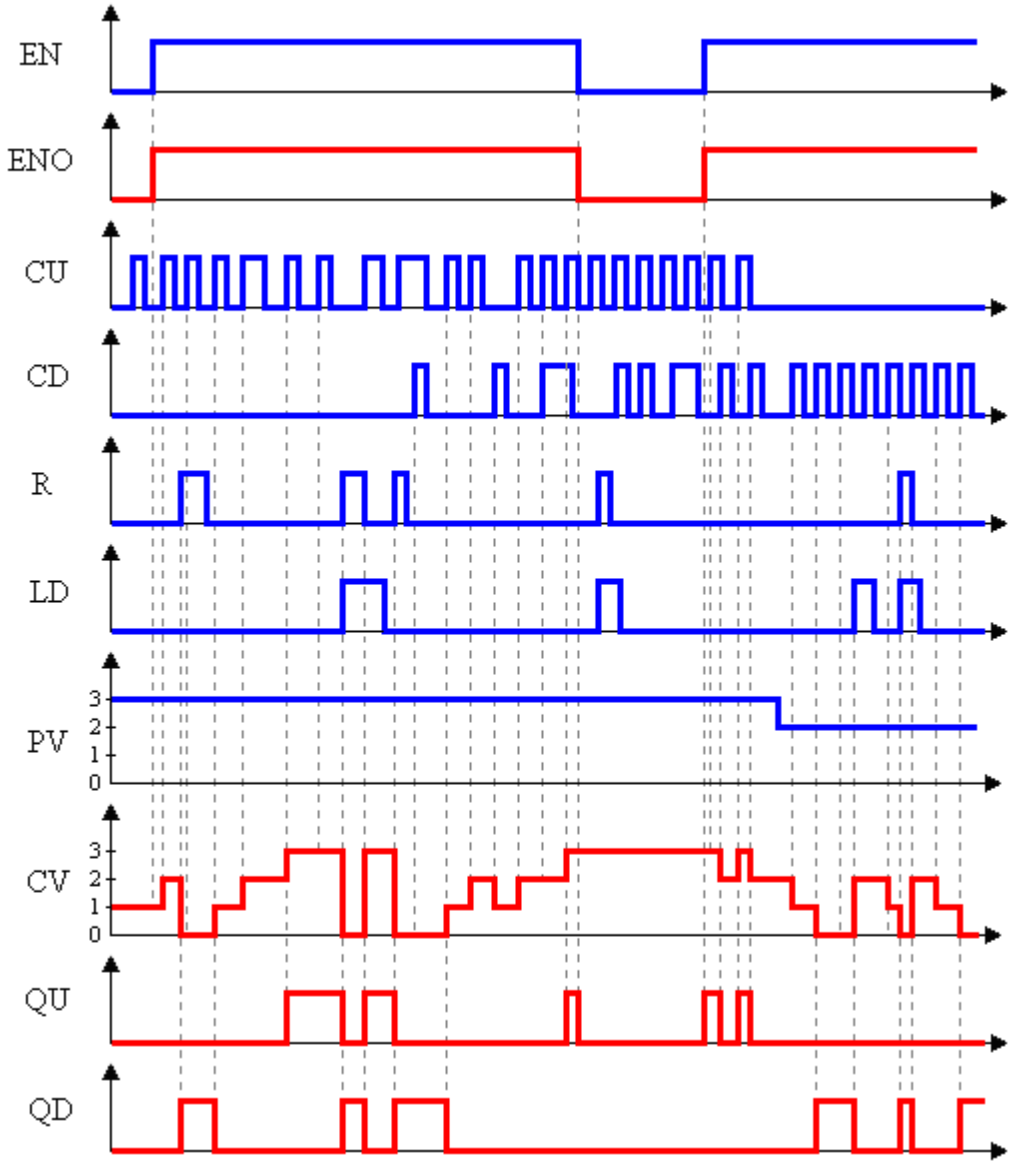
When this block has a TRUE value in EN, it acts as a CTD block and block CTU at the same time acting on the same CV counter. That is: increments CV until it reaches PV to the leading edges in CU and decrements CV until it reaches zero to the leading edges in CD. A positive transition in R carries zero in CV, while a leading edge in LD loads the PV value in CV. If CV has zero value, QD receives TRUE, and if CV has value equal to PV, QU receives TRUE.

The ENO value forwards to the next Ladder block the EN value.

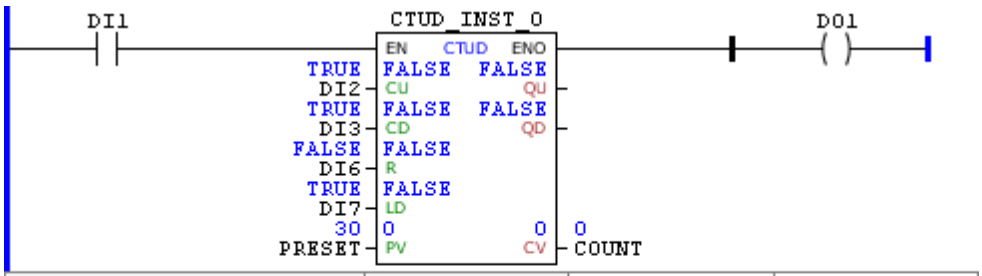
Block Flowchart


Start
EN == TRUE?
Yes
No
R == TRUE?
Yes
No
LD == TRUE?
Yes
No
CV = 0
CV = PV
Rising edge in CU?
Yes
No
CV < PV?
Yes
No
CV = CV + 1
Rising edge in CD?
Yes
No
CV > 0?
Yes
No
CV = CV - 1
CV >= PV?
Yes
No
QU = TRUE
QU = FALSE
QU = FALSE
CV == 0?
Yes
No
QD = TRUE
QD = FALSE
QD = FALSE
ENO = TRUE
ENO = FALSE
End

Operation Diagram

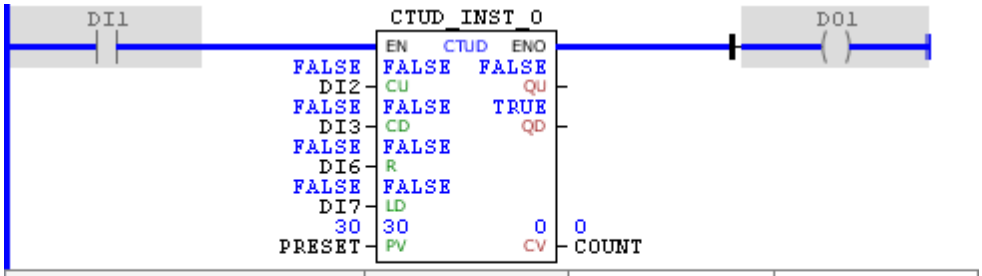


Example



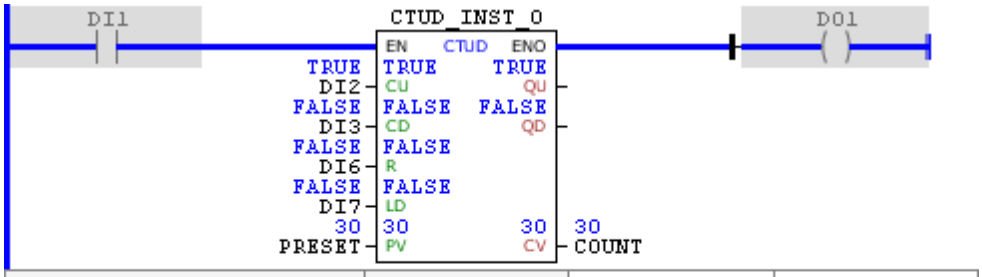
PRESET	UINT	30	Decimal
COUNT	UINT	0	Decimal

The example above shows the disabled block, with all its internal variables zeroed. Although the external controls are activated, these values are not forwarded to the instance of the block.



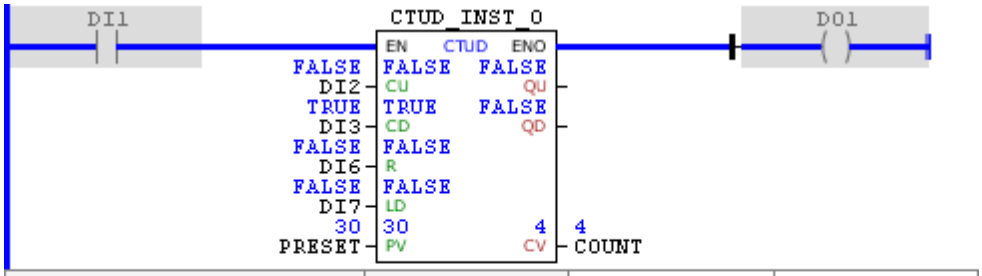
PRESET	UINT	30	Decimal
COUNT	UINT	0	Decimal

When activated, the block identifies the value of PRESET, loading it in PV, and identifies that the output is at zero, enabling the QD output. When execution is completed, the ENO output is activated.



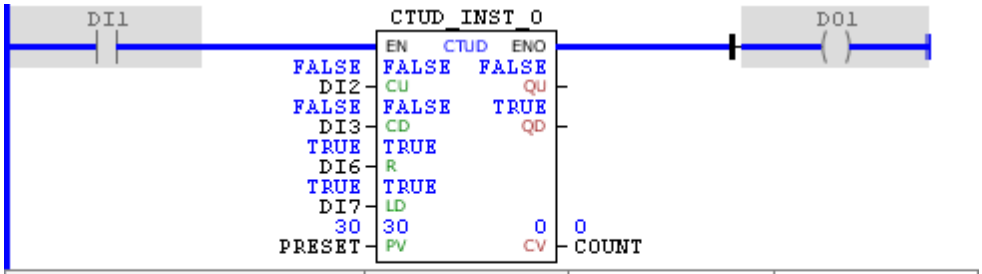
PRESET	UINT	30	Decimal
COUNT	UINT	30	Decimal

At each leading edge identified in CU, the value of CV is incremented until it reaches the PV value, when the QU output is enabled. When execution is completed, the ENO output is activated.



PRESET	UINT	30	Decimal
COUNT	UINT	4	Decimal

At each leading edge detected in CD, the CV value is decremented. When CV is a value between zero and PV, both QD and QU outputs are deactivated. When execution is completed, the ENO output is activated.



PRESET	UINT	30	Decimal
COUNT	UINT	0	Decimal

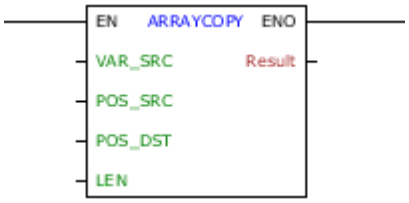
A TRUE value in R resets CV, while a TRUE value in LD loads the value of PV to CV. As we can see, R prevails over LD, leaving CV and enabling the QD output. When execution is completed, the ENO output is activated.

11.10.7.8 Data Transfer

11.10.7.8.1 ARRAYCOPY

Block that copies an array from a certain position to another array or to itself.

Ladder Representation



Block Structure

Variable Type	Name	Data type	Description
VAR_INPUT	EN	BOOL	Block enabling
	VAR_SRC	Array: BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Input Array
	POS_SRC	BYTE USINT WORD UINT	Position of the input array from which the copy will be made
	POS_DST	BYTE USINT WORD UINT	Position of the output array from which it will be replaced
	LEN	BYTE USINT WORD UINT	Number of array positions to be copied
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	Array: BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Output Array

Operation

This block, when it has a value of TRUE in EN, copies LEN values from the POS_SRC position from the input array (VAR_SRC) to the position POS_DST into the destination array (Result).

Comments:
- POS_SRC, POS_DST and LEN input variables only accept positive integers. If a negative value is assigned to any of them, the value zero will be considered.
- The Input Array can be repeated on the output without worrying about data being overwritten.
- If the amount of data to be copied defined by LEN exceeds the last position of the input array, only valid data will be copied to the last position of the input array, thus avoiding any garbage being assigned to the output array.
- If the amount of data to be copied defined by LEN exceeds the last position of the output array, only the data required to complete it will be copied, preventing subsequent memory from receiving unwanted values.
- The block will not execute if LEN has a value greater than the size of the input array.

When EN has FALSE value, Result remains unchanged.

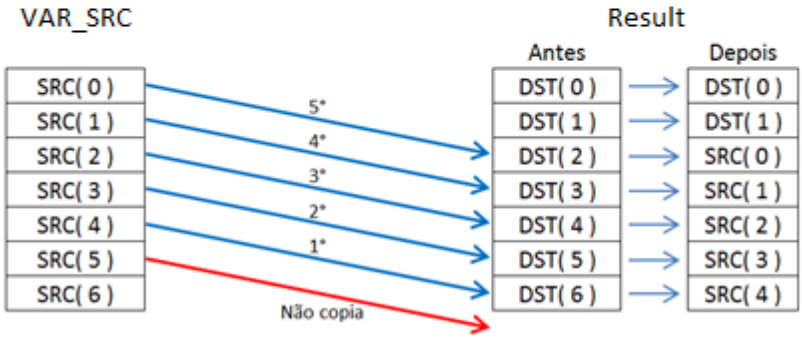
The ENO value forwards to the next Ladder block the EN value after the operation is completed.

NOTE!

- It is important to notice that not only LEN but also POS_SRC will not exceed the VAR_SRC array's size. The same must be noticed when setting values to POS_DST, related to the output array **Result**.
- To learn how to create arrays please go to: Ladder > Editor > Variables > Editing in the Rung

Block Flowchart

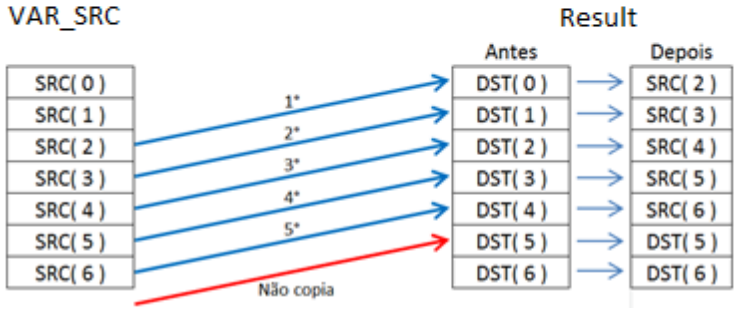
Example



1

Main Ladder				
SRC				
SRC[0]	INT	1	1	Decimal
SRC[1]	INT	2	2	Decimal
SRC[2]	INT	3	3	Decimal
SRC[3]	INT	4	4	Decimal
SRC[4]	INT	5	5	Decimal
SRC[5]	INT	6	6	Decimal
SRC[6]	INT	7	7	Decimal
DST				
DST[0]	INT	0	0	Decimal
DST[1]	INT	0	0	Decimal
DST[2]	INT	0	3	Decimal
DST[3]	INT	0	4	Decimal
DST[4]	INT	0	5	Decimal
DST[5]	INT	0	6	Decimal
DST[6]	INT	0	7	Decimal

2



2

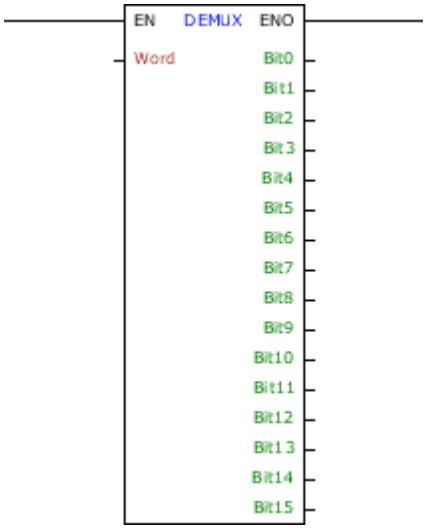
Main Ladder				
SRC				
SRC[0]	INT	1	1	Decimal
SRC[1]	INT	2	2	Decimal
SRC[2]	INT	3	3	Decimal
SRC[3]	INT	4	4	Decimal
SRC[4]	INT	5	5	Decimal
SRC[5]	INT	6	6	Decimal
SRC[6]	INT	7	7	Decimal
DST				
DST[0]	INT	0	3	Decimal
DST[1]	INT	0	4	Decimal
DST[2]	INT	0	5	Decimal
DST[3]	INT	0	6	Decimal
DST[4]	INT	0	7	Decimal
DST[5]	INT	0	0	Decimal
DST[6]	INT	0	0	Decimal

In the examples above the value of the variable SRC is copied to DST array, according to source position (POS_SRC), destination (POS_DST) and the lenght to be copied (LEN). The block ends with success and ENO output is activated.

11.10.7.8.2 DEMUX

Block that creates 16 new BOOL variables from the decomposition of a WORD variable.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Word	WORD UINT INT	Input variable of 15 bits
VAR_OUTPUT	ENO	BOOL	End of operation
	Bit0 – Bit15	BOOL	Bit of the corresponding position of Word

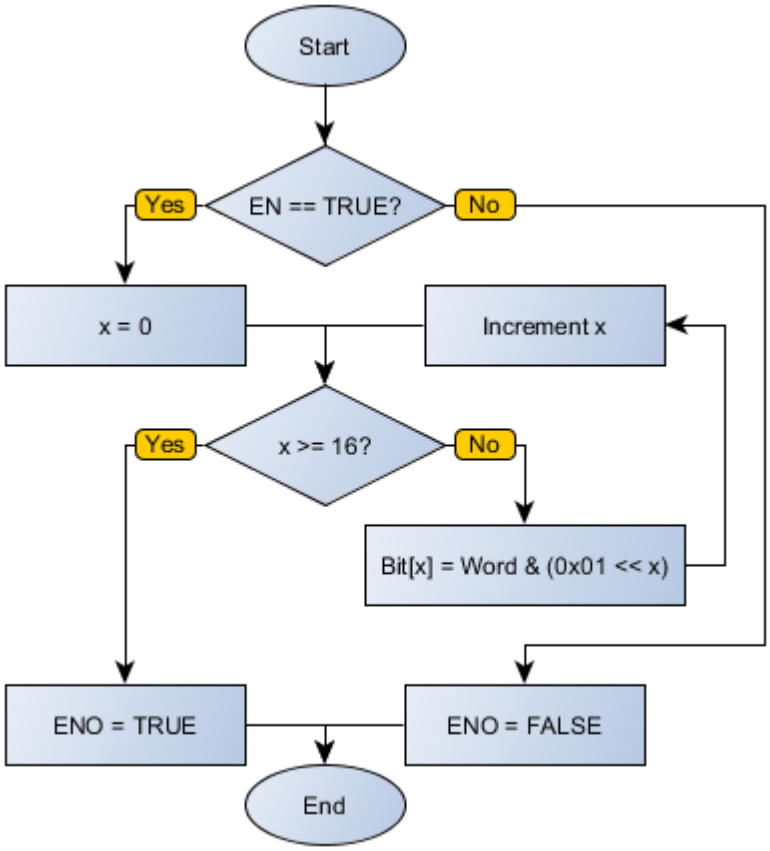
Operation

When this block has a TRUE value in EN, it decomposes the input variable in Word 15 Boolean values stored in Bit0 to Bit15 variables. Bit0 corresponds to the LSB (least significant bit) and Bit15 corresponds to the MSB (most significant bit).

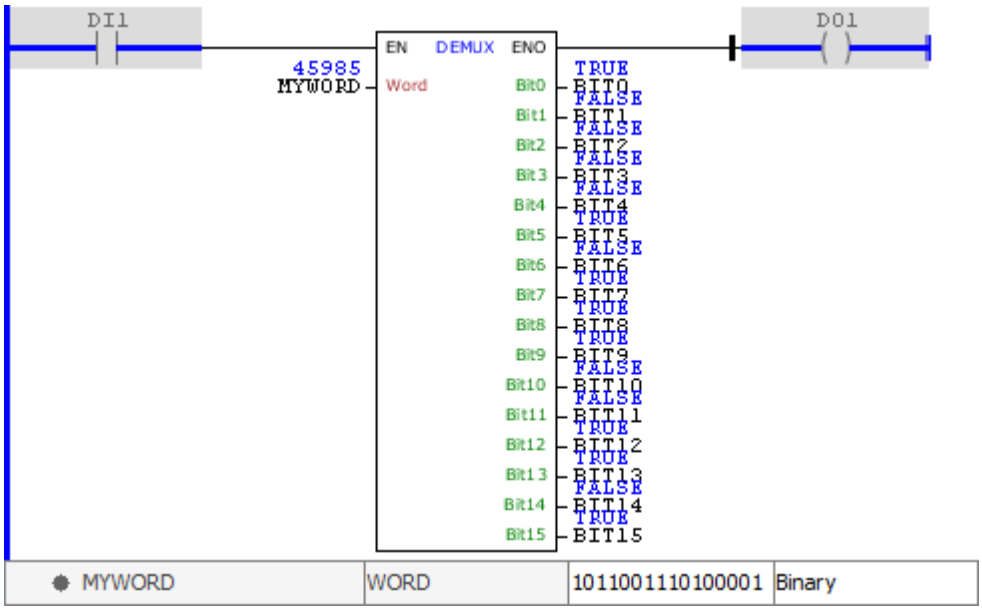
When EN has FALSE value, output variables remain unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example

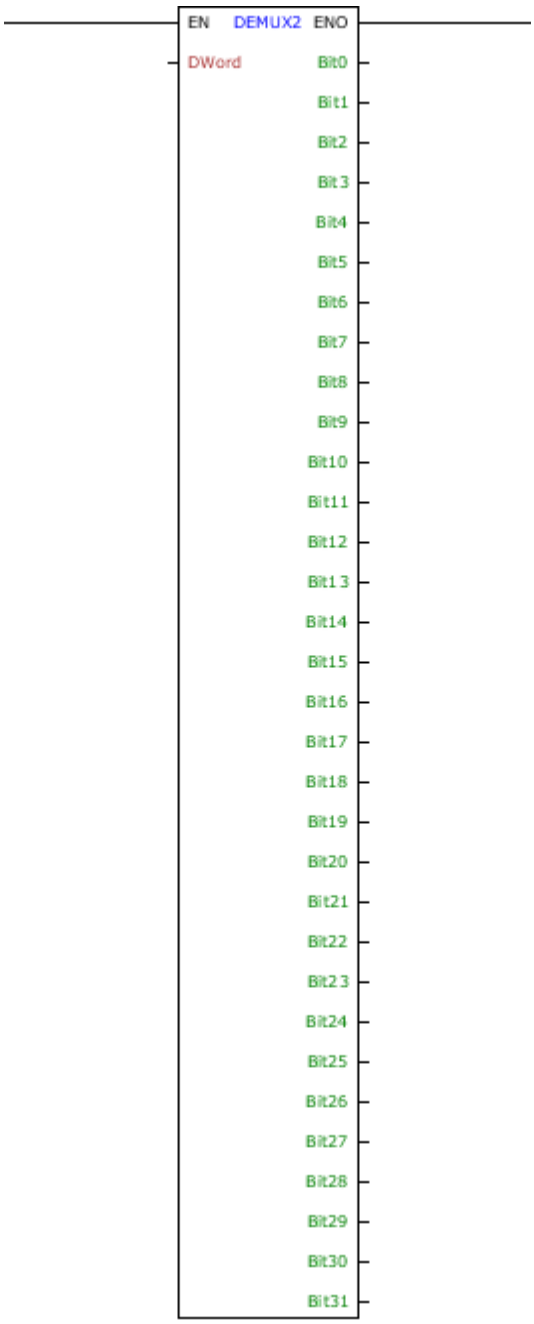


The example above decomposes the value of MYWORD in Boolean values, which are stored in the output variables BIT0 to Bit15. The block ends successfully and the ENO output is activated.

11.10.7.8.3 DEMUX2

Block that creates 32 new BOOL variables from the decomposition of a DWORD variable.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	DWord	DWORD UDINT DINT	Input variable of 15 bits
VAR_OUTPUT	ENO	BOOL	End of operation
	Bit0 – Bit31	BOOL	Bit of the corresponding position of Word

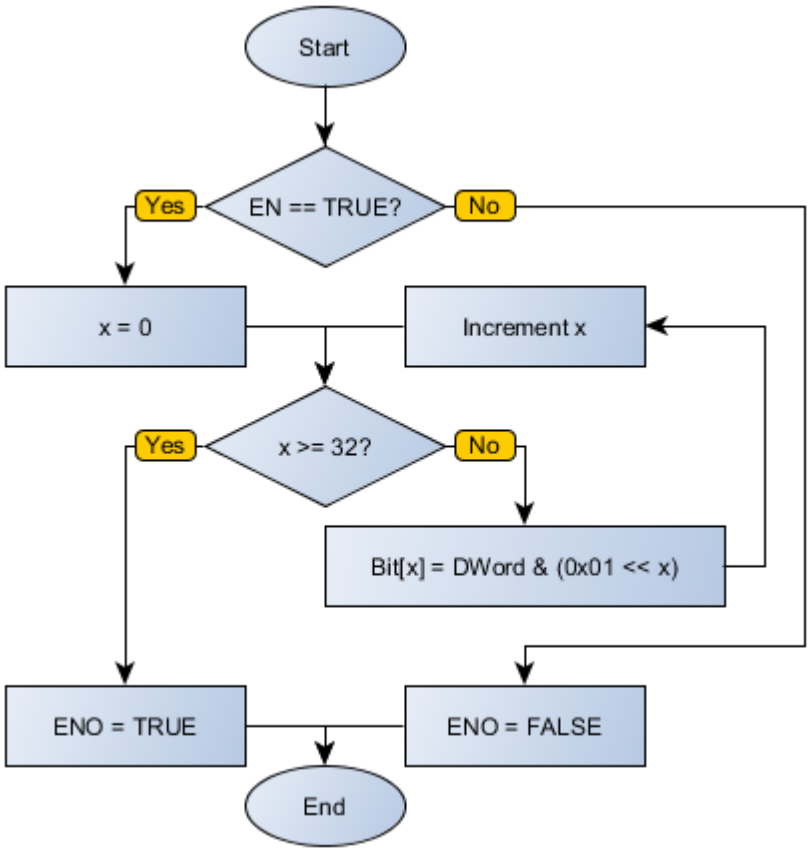
Operation

When this block has a TRUE value in EN, it decomposes the input variable in DWord 32 Boolean values stored in Bit0 to Bit31 variables. Bit0 corresponds to the LSB (least significant bit) and Bit15 corresponds to the MSB (most significant bit).

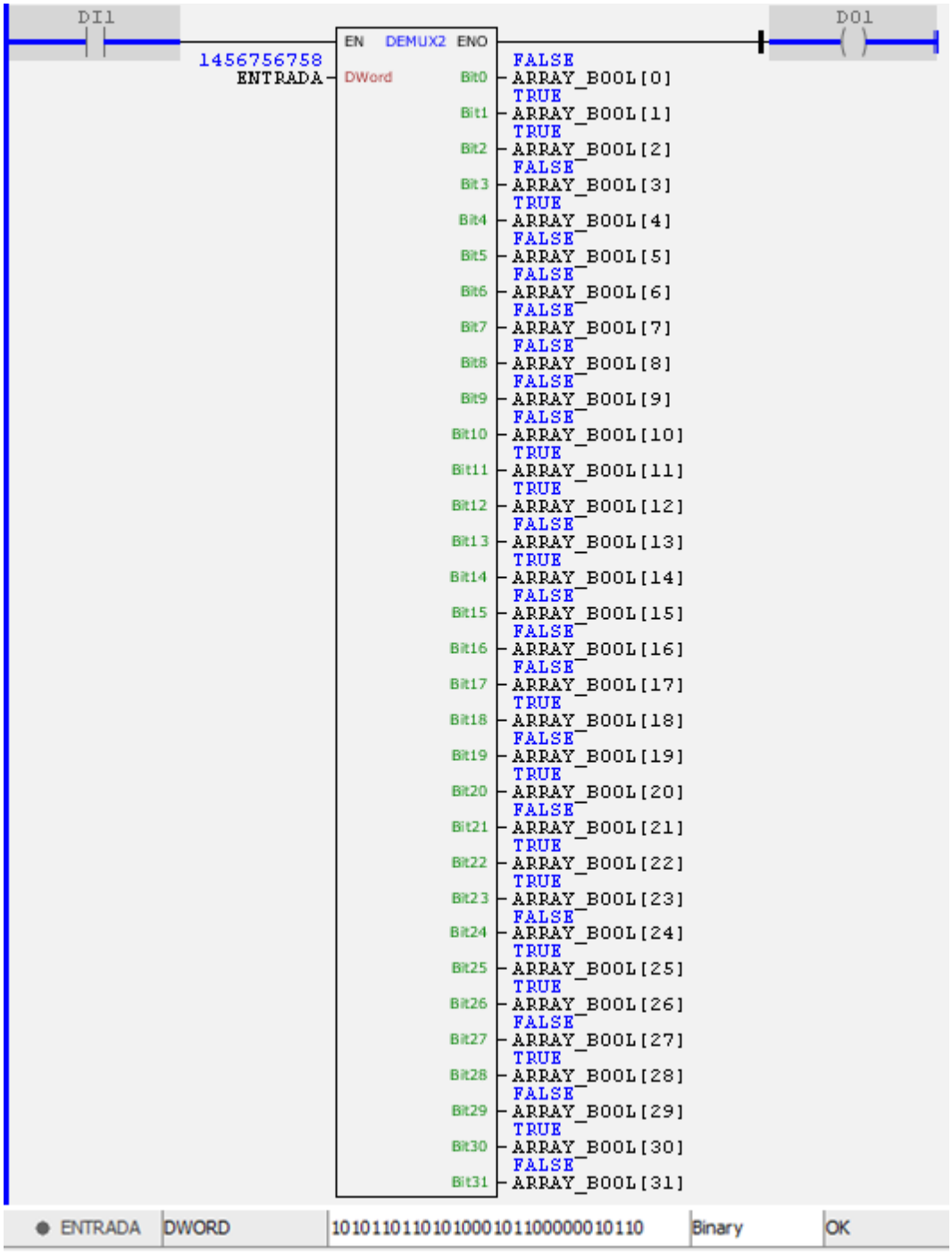
When EN has FALSE value, output variables remain unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example

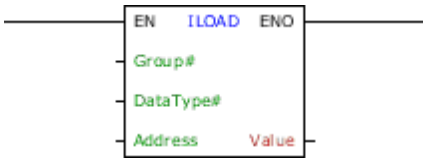


The example above decomposes the value of MYDWORD in Boolean values, which are stored in the output variables BIT0 to Bit31. The block ends successfully and the ENO output is activated

11.10.7.8.4 ILOAD

Block which indirectly loads the value of a variable and transfers it to Value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Group#	BYTE	Group w here the variable is stored
	DataType#	BYTE	Data type of the selected variable
	Address	DWORD UDINT DINT	Address of the global variable, as its group
VAR_OUTPUT	ENO	BOOL	End of operation
	Value	As selected in DataType#	Value of the selected variable

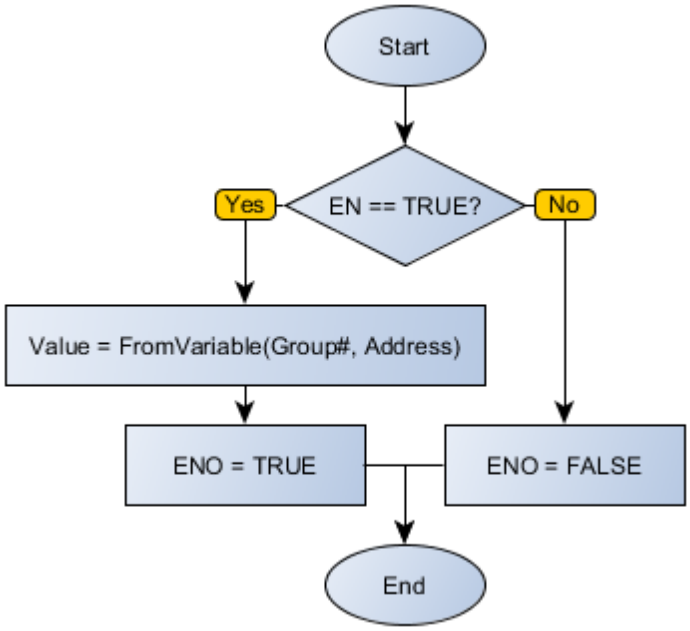
Operation

When this block has a TRUE value in EN, it loads, in Value, the of the Address variable belonging to the Group# group, as the selected DataType#.

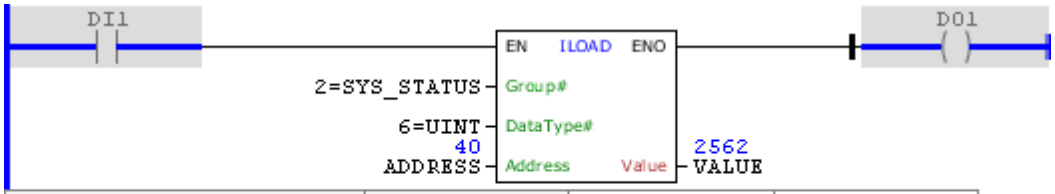
When EN has FALSE value, Value remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



ADDRESS	DINT	40	Decimal
VALUE	UINT	2562	Decimal

The above example loads the value of the address 40 of group 2 (GLOBAL_SYSTEM%S), which represents the status of ESC key in UINT format for the VALUE variable. The block ends with success and ENO output is activated.

11.10.7.8.5 ILOADBOOL

Block that indirectly loads the value of a bit in a global variable address.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Group#	BYTE	Group where the variable is stored
	Address	DWORD UDINT DINT	Address of the global variable, as its group
	Bit	BYTE USINT SINT	Position of the bit to be checked
VAR_OUTPUT	ENO	BOOL	End of operation
	Value	BOOL	Value of the bit selected by the input arguments

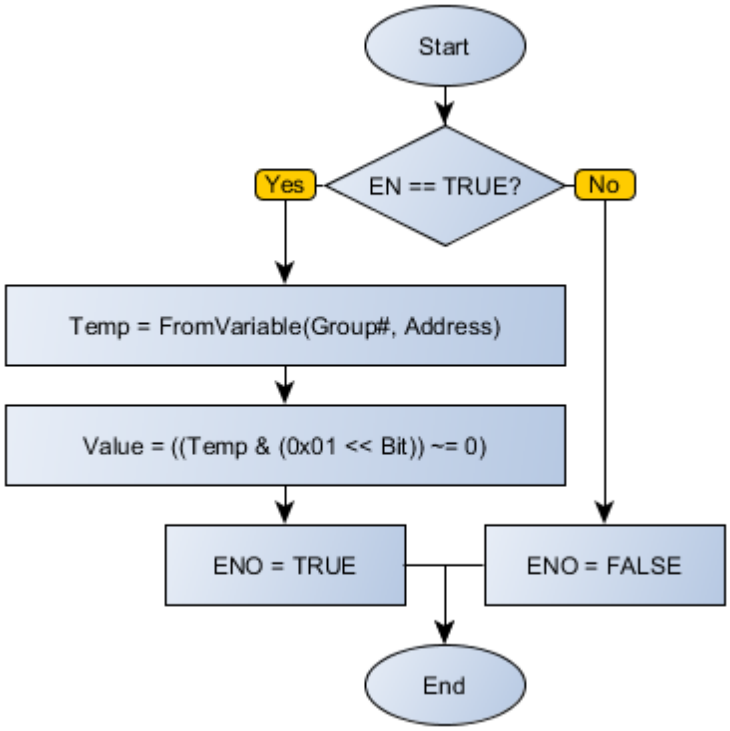
Operation

When this block has a TRUE value in EN, it loads, in Value, the Bit contents of the Address variable belonging to the Group# group.

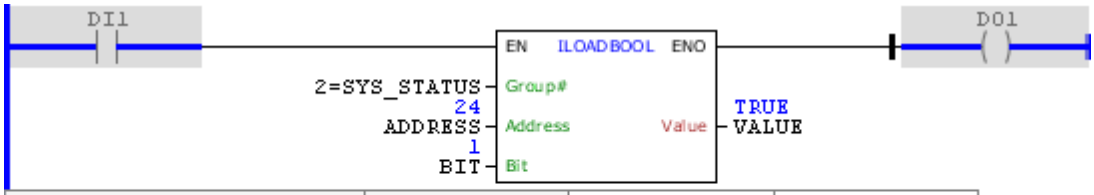
When EN has FALSE value, Value remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



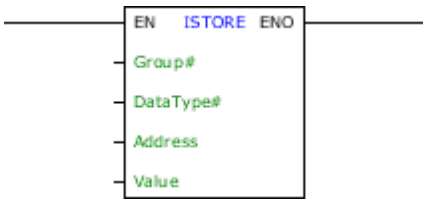
ADDRESS	DINT	24	Decimal
BIT	BYTE	1	Decimal
VALUE	BOOL	1	Binary

The above example loads the value of bit 1 of the address 24 of group 2 (S GLOBAL_SYSTEM%), which represents the status of ESC key for the VALUE variable. The block ends with success and ENO output is activated.

11.10.7.8.6 ISTORE

Block that indirectly loads the Value value in a variable.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Group#	BYTE	Group where the variable is stored
	DataType#	BYTE	Data type of the selected variable
	Address	DWORD UDINT DINT	Address of the global variable, as its group
	Value	Depending on the selection of the DataType#	Value to be written in the selected variable
VAR_OUTPUT	ENO	BOOL	End of operation

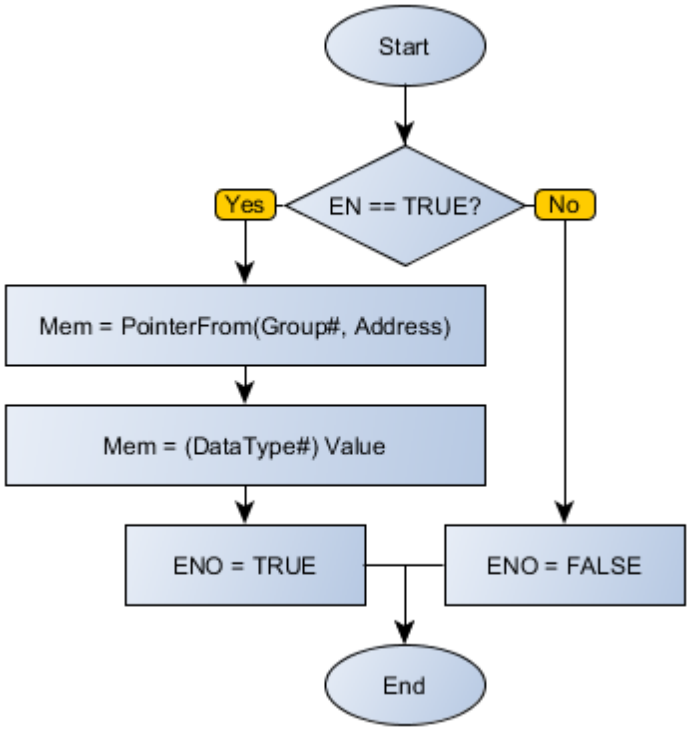
Operation

When this block has a TRUE value in EN, it loads the Value value in the contents of the Address variable belonging to the Group# group, as the selected DataType#.

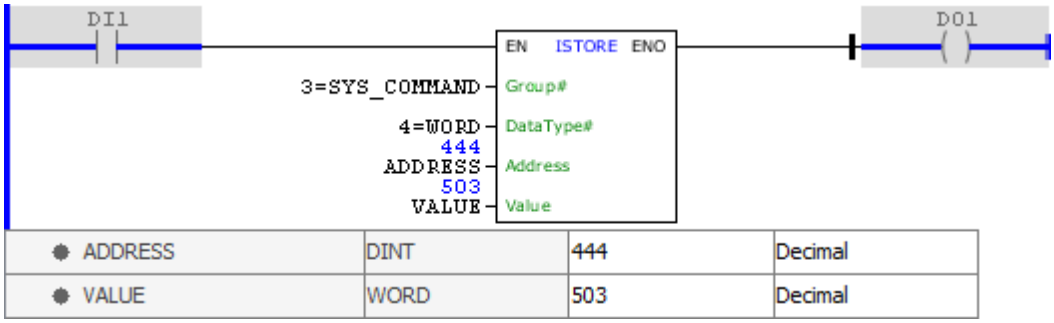
When EN has FALSE value, Value remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example

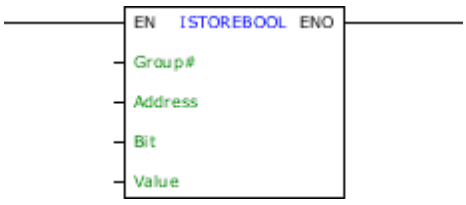


The example above stores the VALUE value in WORD format in address 444 of group 3 (GLOBAL_SYSTEM% C), which represents the index of the communication port Modbus TCP. The block ends with success and ENO output is activated.

11.10.7.8.7 ISTOREBOOL

Block that indirectly loads the Value value in a bit in a global variable address.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Group#	BYTE	Group w here the variable is stored
	Address	DWORD UDINT DINT	Address of the global variable, as its group
	Bit	BYTE USINT SINT	Position of the bit to be modified
	Value	BOOL	New value of the selected bit
VAR_OUTPUT	ENO	BOOL	End of operation

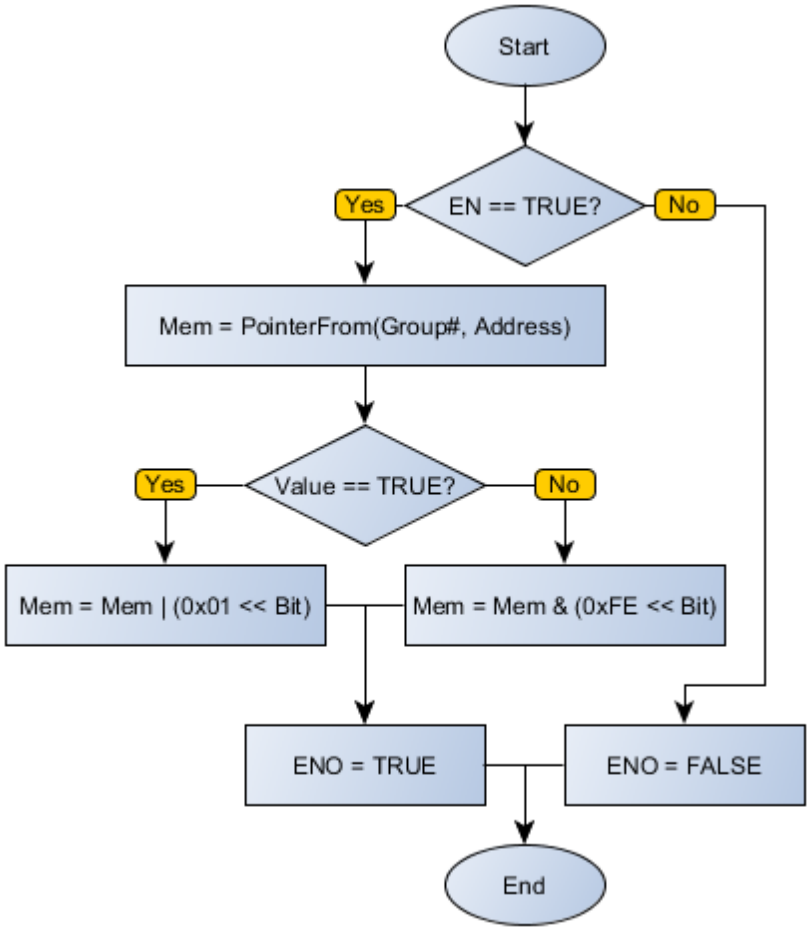
Operation

When this block has a TRUE value in EN, it loads the Value value in the Bit contents of the Address variable belonging to the Group# group.

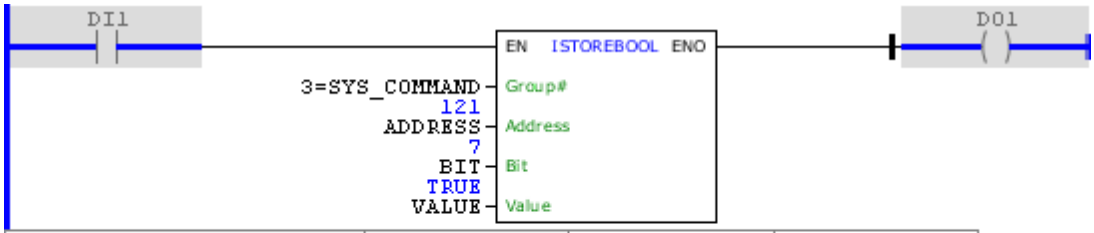
When EN has FALSE value, Value remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



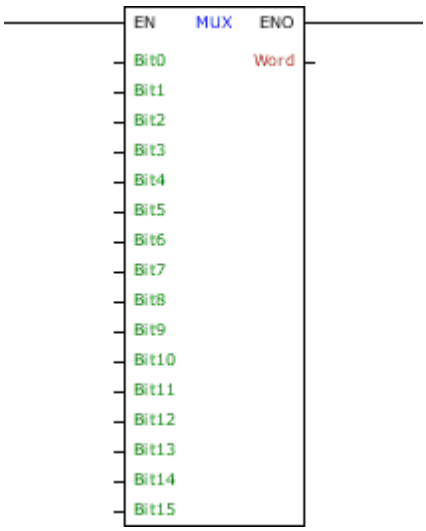
ADDRESS	DINT	121	Decimal
BIT	BYTE	7	Decimal
VALUE	BOOL	1	Binary

The example above stores the value of VALUE in bit 7 of the address 121 in group 3 (GLOBAL_SYSTEM% C), which represents the disable command of CANopen communication. The block ends with success and ENO output is activated.

11.10.7.8.8 MUX

Block that creates a new WORD variable from the concatenation of 16 BOOL variables.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Bit0 – Bit15	BOOL	Bit of the corresponding position in the new word
VAR_OUTPUT	ENO	BOOL	End of operation
	Word	WORD UINT INT	New word formed from the input bits

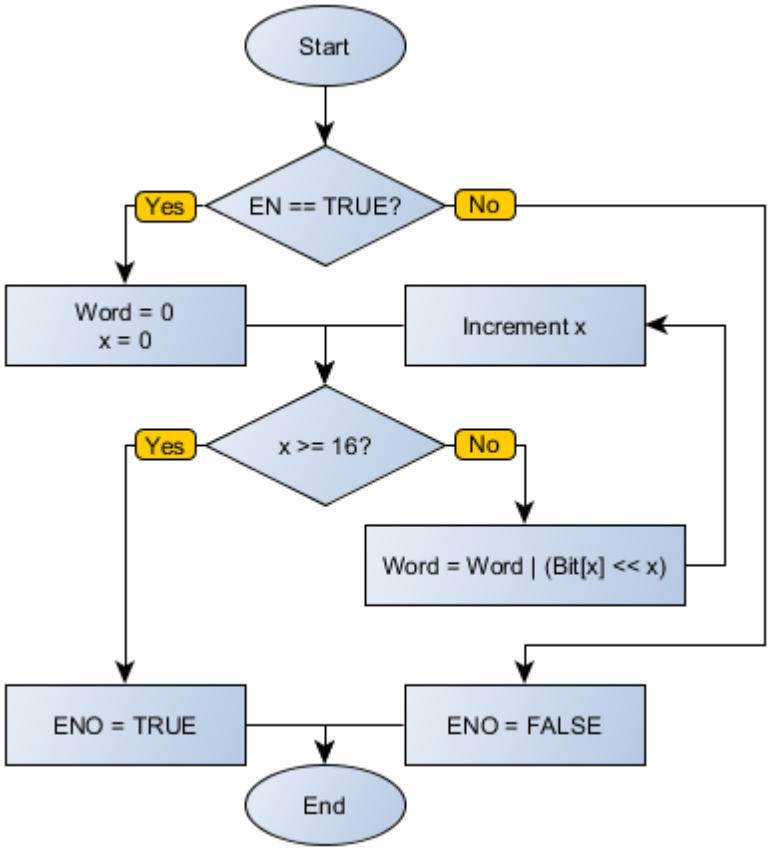
Operation

When this block has a TRUE value in EN, it concatenates Boolean values of the input variables and stores this value in the variable Word. Bit0 corresponds to LSB (least significant bit) and Bit15 corresponds to the MSB (most significant bit).

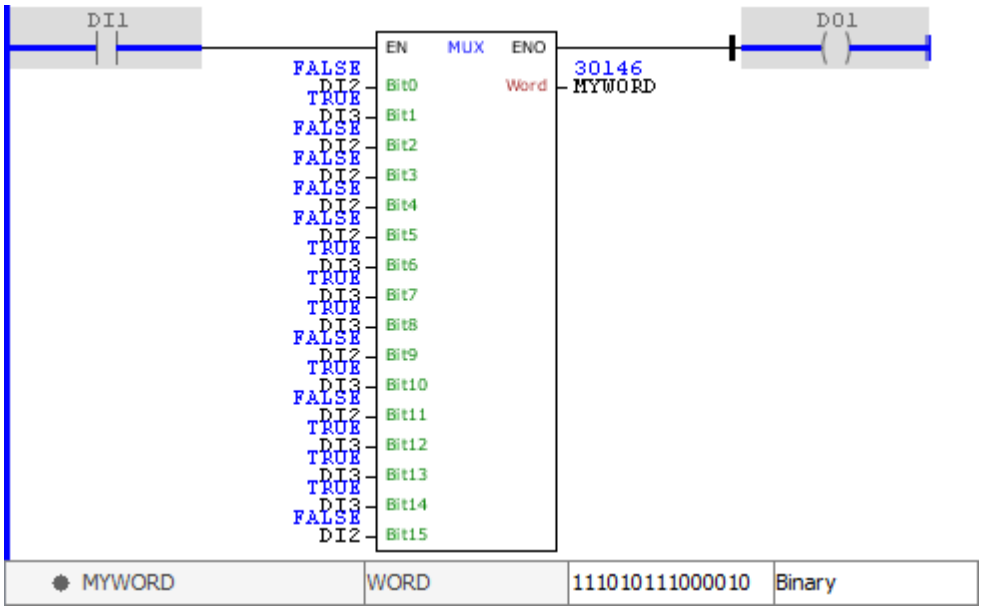
When EN has FALSE value, Word remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example

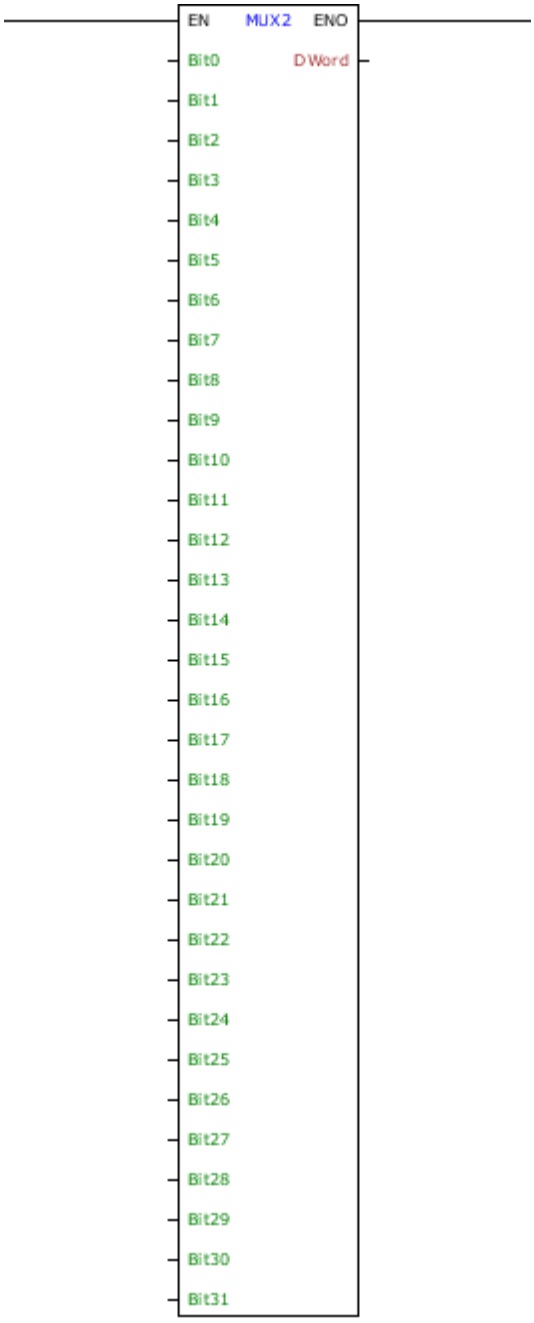


The above example concatenates the Boolean values of the input bits of the block to form the output word stored in MYWORD. The block ends with success and ENO output is activated.

11.10.7.8.9 MUX2

Block that creates a new DWORD variable from the concatenation of 32 BOOL variables.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Bit0 – Bit31	BOOL	Bit of the corresponding position in the new word
VAR_OUTPUT	ENO	BOOL	End of operation
	DWord	DWORD UDINT DINT	New word formed from the input bits

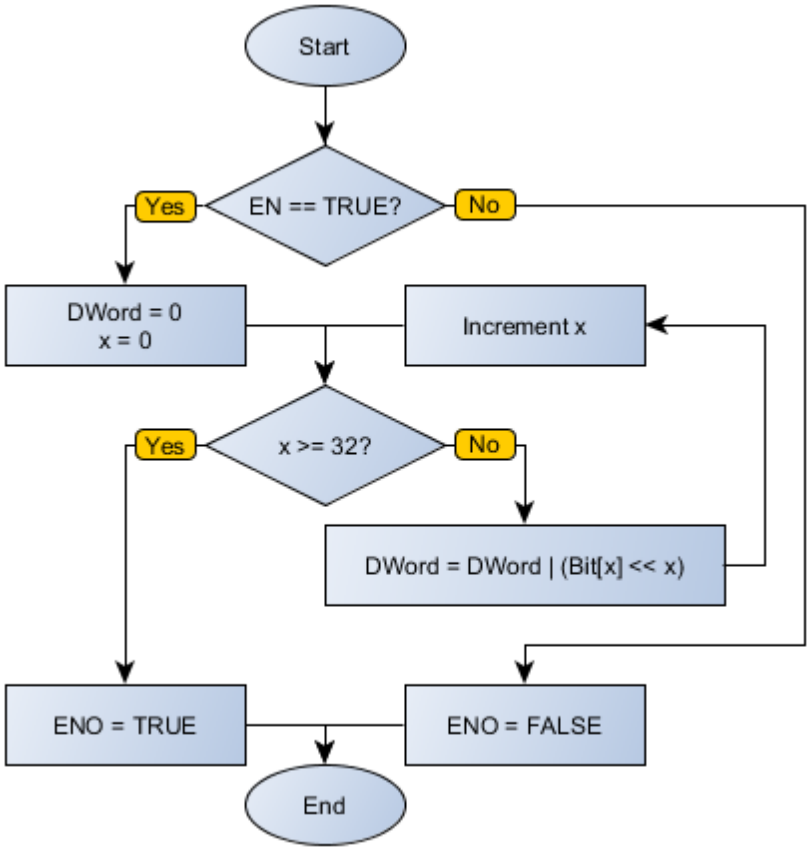
Operation

When this block has a TRUE value in EN, it concatenates Boolean values of the input variables and stores this value in the variable DWord. Bit0 corresponds to LSB (least significant bit) and Bit31 corresponds to the MSB (most significant bit).

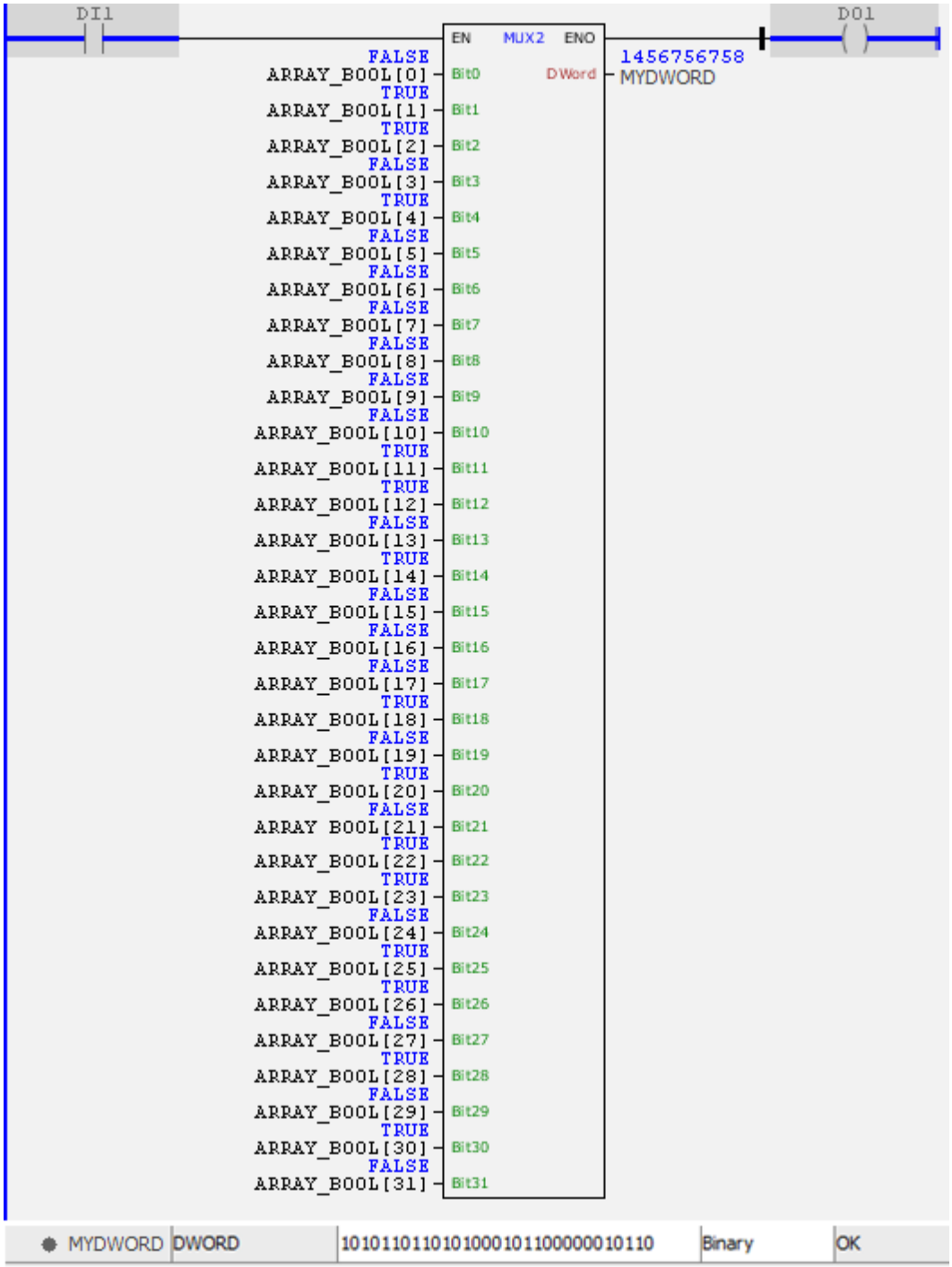
When EN has FALSE value, Word remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example

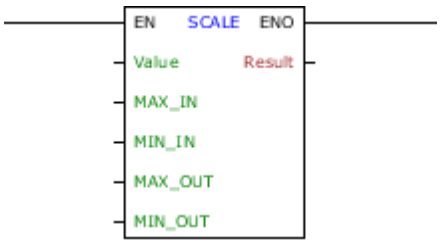


The above example concatenates the Boolean values of the input bits of the block to form the output word stored in MYDWORD. The block ends with success and ENO output is activated.

11.10.7.8.10 SCALE

Block that converts a value from a scale to another one.

Ladder Representation



Block Structure

Variable Type	Name	Data type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Input value to be converted
	MAX_IN	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Maximum value of input scale
	MIN_IN	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Minimum value of input scale
	MAX_OUT	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Maximum value of output scale
	MIN_OUT	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Minimum value of output scale
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Output value on new scale

Operation

This block, when it has a TRUE value in EN, by setting the minimum and maximum values of the variable to be converted and the minimum and maximum values of the new scale variable, defined by the user, performs the Scale function for the conversion of the variable according to equation:

$$VAR_{OUT} = a \times (Value - MIN_{in}) + b$$

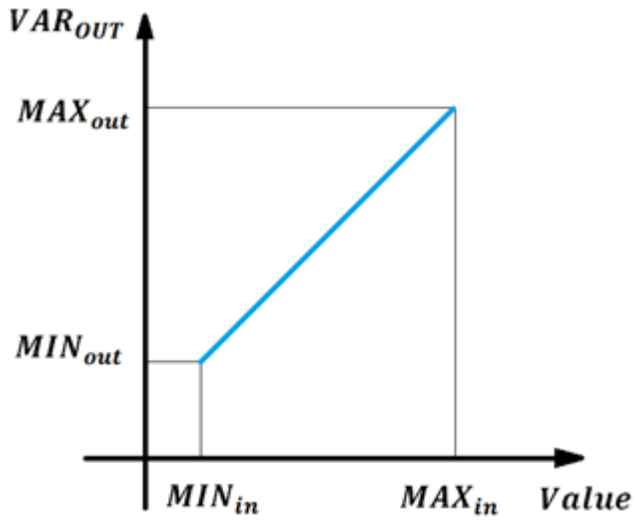
Where:

$$a = \left(\frac{MAX_{out} - MIN_{out}}{MAX_{in} - MIN_{in}}\right)$$

and

$$b = MIN_{out}$$

The graph below represents the straight linearized:



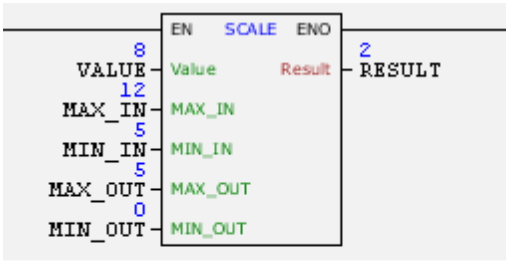
When EN has FALSE value, DST remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

NOTE!
- The value in MAX_{in} must be greater than value in MIN_{in};
- The value in MAX_{out} must be greater than value in MIN_{out};
- Value in **Value** according to: **MIN_{in} = Value = MAX_{in}.**

Block Flowchart

Example



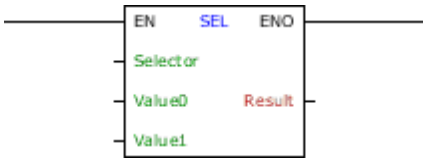
VALUE	INT	0	8
RESULT	INT	0	2

The example above stores the value of the variable VALUE in Result. The block considers the equation described above and ends with success and ENO output is activated.

11.10.7.8.11 SEL

Block that replicates to the output the value of an input variable (Value0 or Value1) according to the Selector selection.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Selector	BOOL	Variable that selects the input
	Value0	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Multiplexed input number 1
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Multiplexed input number 2
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Output value selected

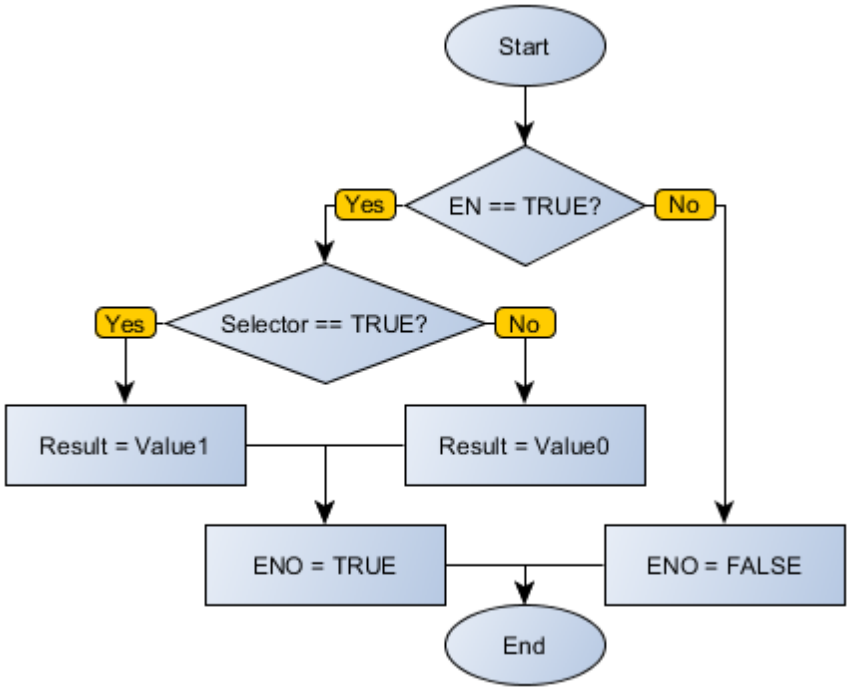
Operation

When this block has a TRUE value in EN, it replicates to the Result variable the Value0 value if selector is FALSE or the Value1 value if Selector is TRUE.

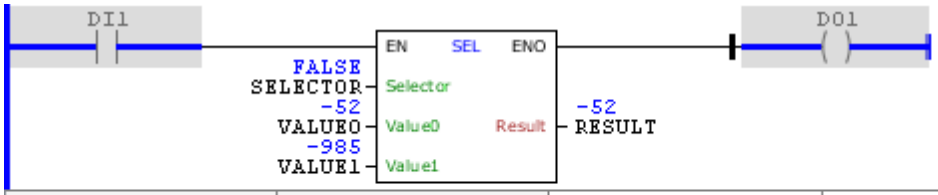
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart

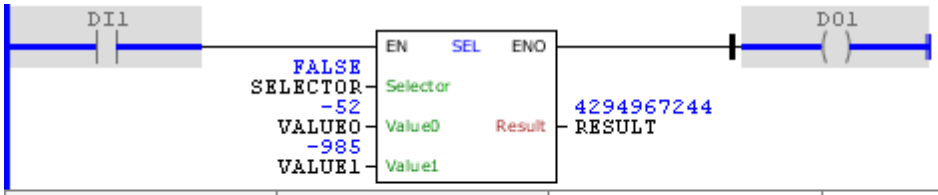


Example



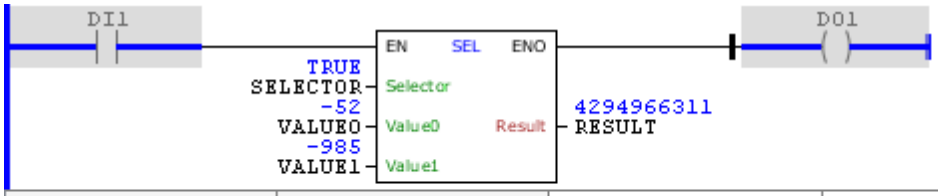
SELECTOR	BOOL	0	Binary
VALUE0	SINT	-52	Decimal
VALUE1	INT	-985	Decimal
RESULT	INT	-52	Decimal

The above example uses the SELECTOR variable as multiplexing channel selector. When it is at FALSE level, the RESULT output gets the value of VALUE0. The block ends successfully and the ENO output is activated.



SELECTOR	BOOL	0	Binary
VALUE0	SINT	-52	Decimal
VALUE1	INT	-985	Decimal
RESULT	DWORD	4294967244	Decimal

The above example uses the SELECTOR variable as multiplexing channel selector. When it is at FALSE level, the RESULT output gets the value of VALUE0. The block ends successfully and the ENO output is activated. Note that the binary pattern has been maintained even though the decimal representation is corrupted, since DWORD does not accept negative values.



SELECTOR	BOOL	1	Binary
VALUE0	SINT	-52	Decimal
VALUE1	INT	-985	Decimal
RESULT	DWORD	4294966311	Decimal

The above example uses the SELECTOR variable as multiplexing channel selector. When it is at TRUE level, the RESULT output gets the value of VALUE1. The block ends successfully and the ENO output is activated. Note that the binary pattern has been maintained even though the decimal representation is corrupted, since DWORD does not accept negative values.

11.10.7.8.12 STORE

Block that performs direct storage of data from a source to a destination.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	SRC	BYTE USINT SINT WORD UINT INT DWORD DINT DINT REAL	Data source
VAR_OUTPUT	ENO	BOOL	End of operation
	DST	BYTE USINT SINT WORD UINT INT DWORD DINT DINT REAL	Data destination

Operation

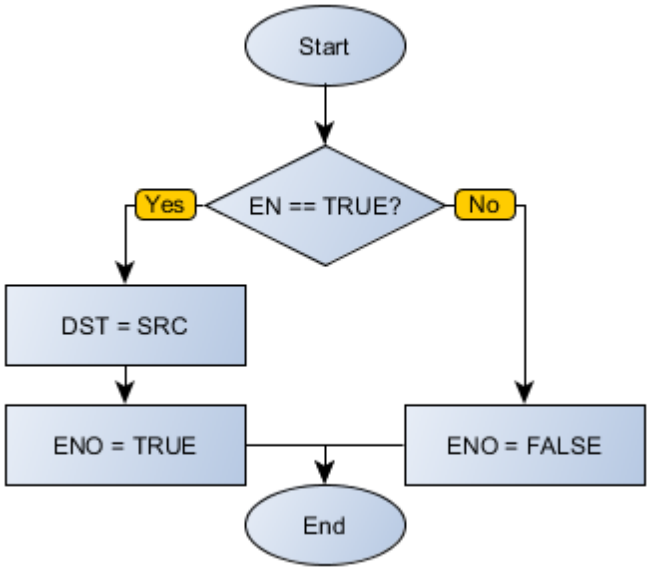
When this block has a TRUE value in EN, it stores the contents from SRC into DST.

NOTE!
SRC and DST must have data types of the same size.

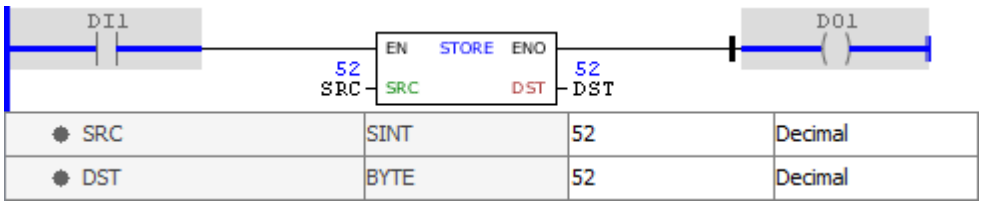
When EN has FALSE value, DST remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



The example above stores the value of the variable SRC in DST. The block ends with success and ENO output is activated.

SRC	SINT	-15	Decimal
DST	BYTE	241	Decimal

The example above stores the value of the variable SRC in DST. The block ends with success and ENO output is activated. Note that the binary pattern is maintained between variables of different types.

11.10.7.8.13 SWAP

Block that performs a swap between the odd bytes and consecutive even bytes into Value and sends the value to Result.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	WORD UINT INT DWORD UDINT DINT	Input variable to be swapped
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	WORD UINT INT DWORD UDINT DINT REAL(*)	Output value

Operation

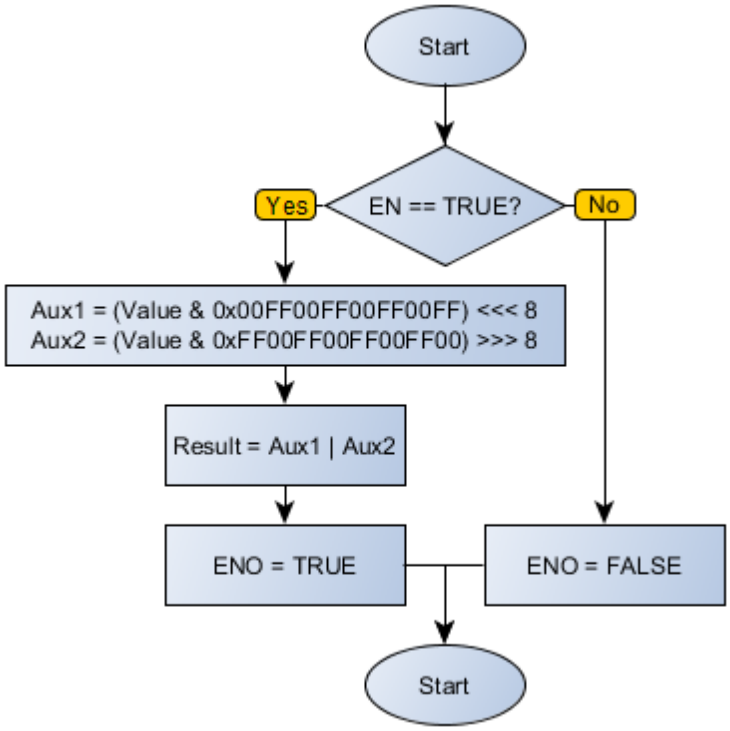
When this block has a TRUE value in EN, it changes the values of the odd bytes (1, 3, 5 and 7) and the consecutive even bytes (2, 4, 6 and 8, respectively) of the Value variable, storing the result in Result.

When EN has FALSE value, Result remains unchanged.

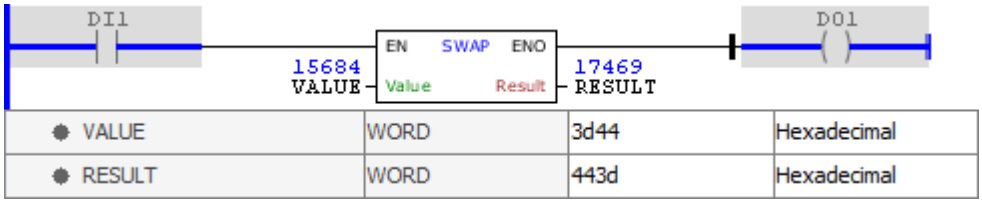
The ENO value forwards to the next Ladder block the EN value after the operation is completed.

NOTA!
Caution when using in Result a variable of REAL type, because the block does not perform type conversion, that is, it only reverses the bytes in memory.

Block Flowchart

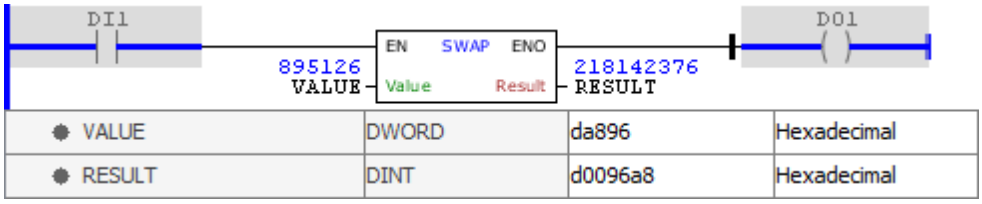


Example



VALUE	WORD	3d44	Hexadecimal
RESULT	WORD	443d	Hexadecimal

The example changes the position of byte 1 value of VALUE (0x44) with byte 2 of VALUE (0x3D), storing the final result (0x44_3D) in RESULT. The block ends with success and ENO output is activated.



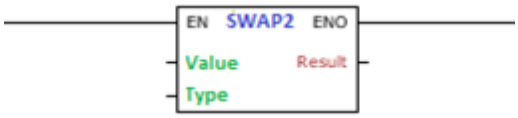
VALUE	DWORD	da896	Hexadecimal
RESULT	DINT	d0096a8	Hexadecimal

The example changes the position of byte 1 value of VALUE (0x96) with byte 2 of VALUE (0xA8) and byte 3 of VALUE (0x0D) with byte 4 of VALUE (0x00), storing the final result (0x0D_00_96_A8) in RESULT. The block ends with success and ENO output is activated.

11.10.7.8.14 SWAP2

Block that rearranges the bytes of a variable.

Ladder Representation



Block Structure

Variable type	Name	Data type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	WORD UINT INT DWORD UDINT DINT	Input variable to be rearranged
	Type	BYTE	Variable that defines the conversion type according to Table 2
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL(*)	Output value

Table 1. Block variables.

Type	WORD UINT INT	ENO	DWORD UDINT DINT	ENO
0	AB->AB*	TRUE	ABCD->ABCD	TRUE
1	AB->BA	TRUE	ABCD->DCBA	TRUE
2	-	FALSE	ABCD->CDAB	TRUE
3	-	FALSE	ABCD->BADC	TRUE
4 ...	-	FALSE	-	FALSE

Table 2. Conversion type (*characters A, B, C and D represents BYTES).

Operation

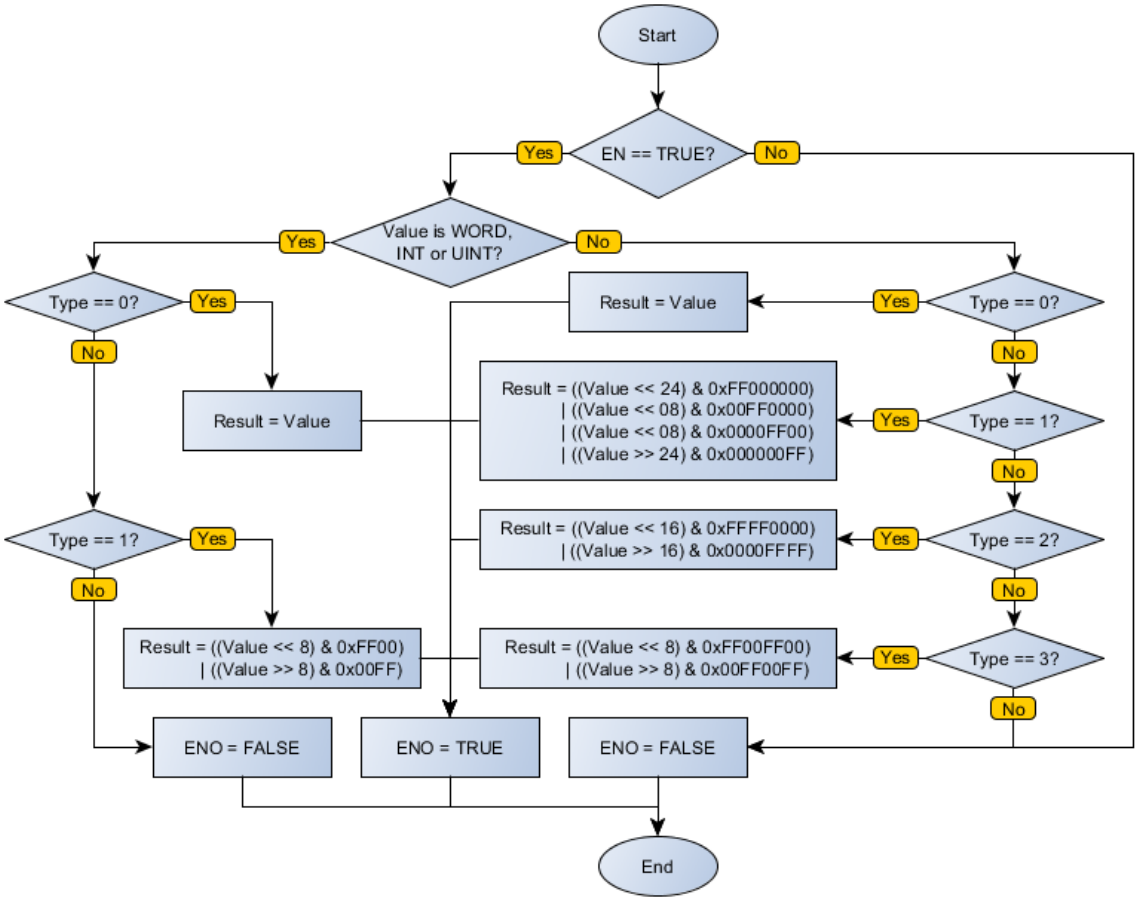
When this block has a TRUE value in EN, it rearranges the bytes from Value variable, storing the result in Result.
Type defines how the bytes will be rearranged, as shown in Table 2.
Note that for 16-bit variables, only options 0 and 1 are valid.
For 32-bit variables the options 0, 1, 2, and 3 are valid.

When EN has FALSE value, Result remains unchanged.

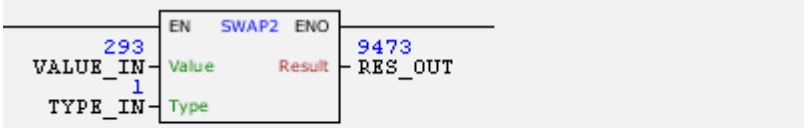
The ENO value forwards to the next Ladder block the EN value after the operation is completed.
Invalid TYPE options assigns FALSE to ENO, and Result value is not changed.

NOTA!
Caution when using in Result a variable of REAL type, because the block does not perform type conversion, that is, it only reverses the bytes in memory.

Block Flowchart



Example



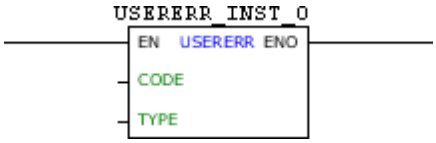
RES_OUT	INT	2501	Hexadecimal	OK
TYPE_IN	BYTE	1	Decimal	OK
VALUE_IN	INT	125	Hexadecimal	OK

The example rearranges the position of value VALUE_IN according to the type set in TYPE_IN = 1 (AB->BA), storing the final result in RESULT. The block ends with success and ENO output is activated.

11.10.7.8.15 USERERR

Block that generates an alarm or fault with the number programmed by the user.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	CODE	WORD UINT	Error code generated (950 - 999)
	TYPE	BYTE	Error type generated (0 - Alarm) (1 - Fault)
VAR_OUTPUT	ENO	BOOL	Success in the generation of error
VAR	USERERR_INST_0	USERERR	(*) Instance of access to block structure

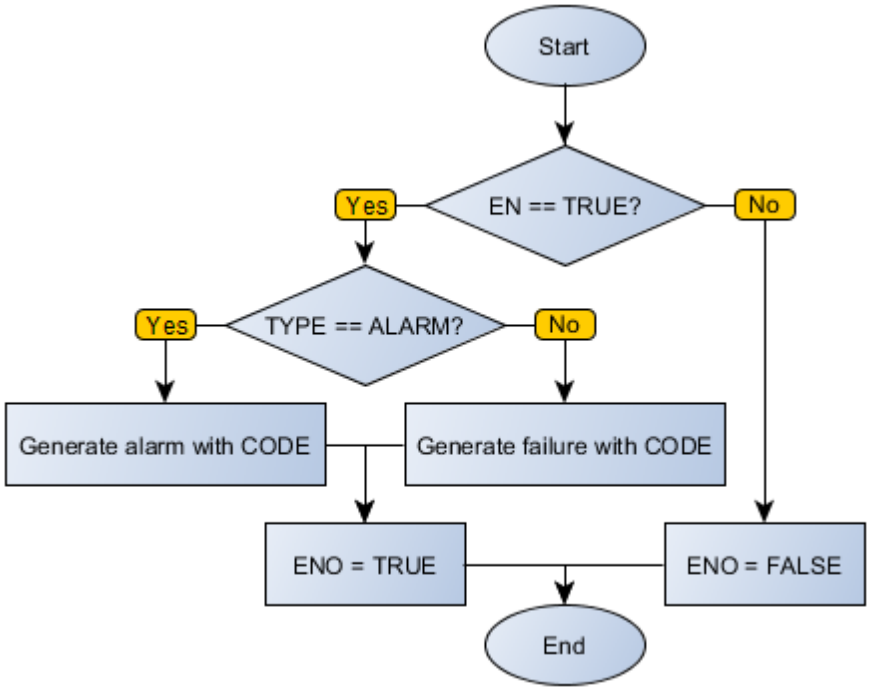
NOTE!
(*) USERERR_INST_0 instance must be configurated to SCA06 and LDW900.

Operation

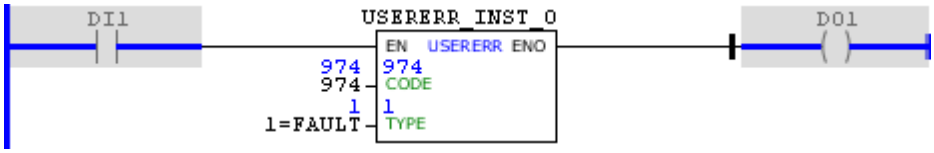
When this block has a TRUE value in EN, it generates an alarm or equipment failure, depending on the type defined in TYPE with CODE code.

The value of ENO informs if the generation of alarm or fault has been executed successfully.

Block Flowchart



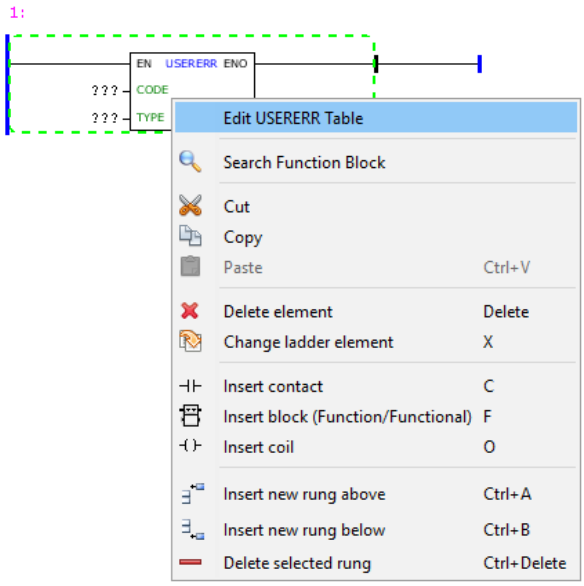
Example



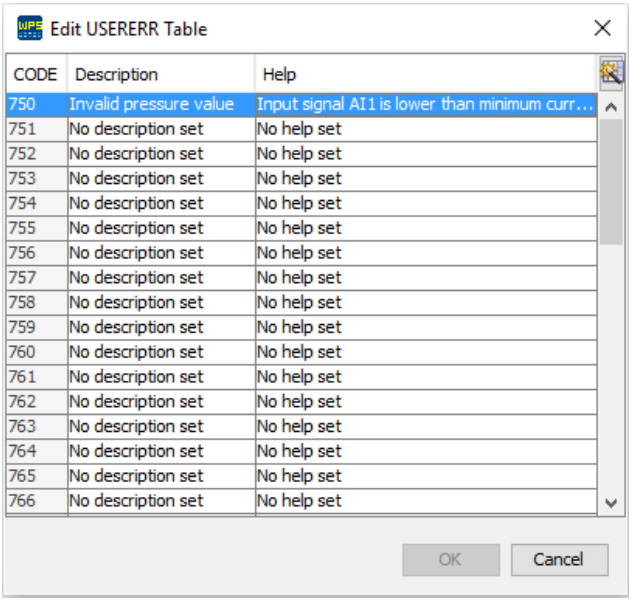
The above example, when identifying TRUE level in DI1, generates a fault with the code 974 and sets the DO1 output.

USERERR table configuration

On devices that have text-based HMI, messages can be configured through an editor. To access the editor, right click on the USERERR block and select the "Edit USERERR Table" option.



The texts configured in the table will be displayed on the HMI when the block USERERR is enabled.



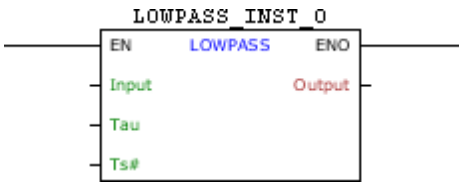
After editing the table, select the argument CODE of the block equal to the CODE column of the table.

11.10.7.9 Filter

11.10.7.9.1 LOWPASS

Block that filters the input using a low pass filter of first order and inserts the result in the output.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Input	REAL	Input signal
	Tau	REAL	Filter time constant
	Ts#	UINT	Sampling time [ms]
VAR_OUTPUT	ENO	BOOL	Output enabling
	Output	REAL	Filter output
VAR	LOWPASS_INST_0	LOWPASS	Instance of access to block structure

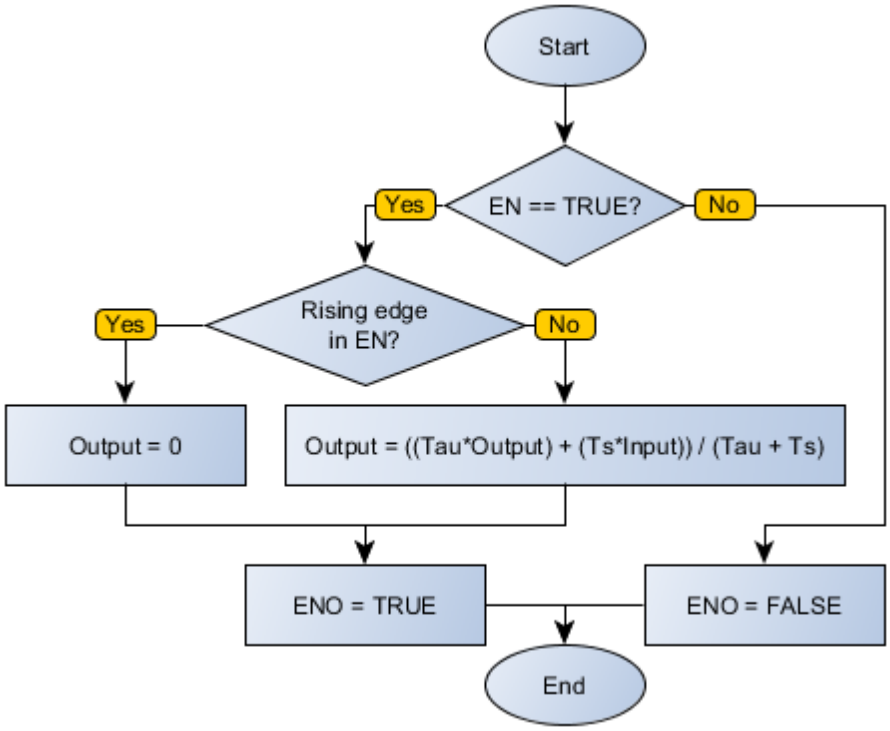
Operation

When this block has a TRUE value in EN, filters the input value of Input using a low pass first order filter described by Tau and Ts#, inserting the result in Output. On the leading edge of EN, Output receives zero.

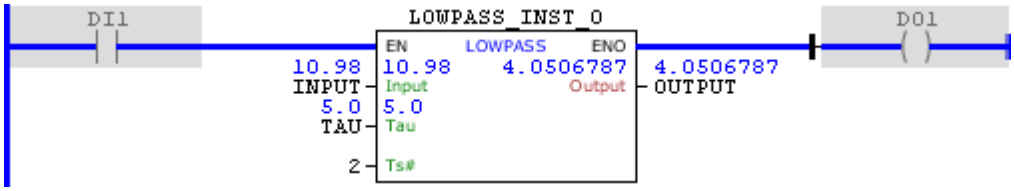
When EN has FALSE value, Output remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

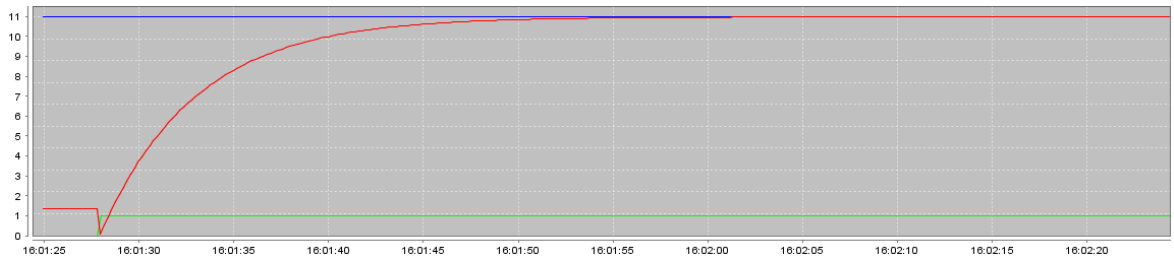
Block Flowchart



Example



The above example causes OUTPUT, by identifying a leading edge in EN, to display a behavior of first order with time constant equal to Tau and the sampling time of 2 ms, in order to achieve the reference set to INPUT. At each calculation completed successfully, the ENO output is activated.

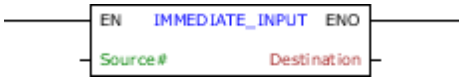


11.10.7.10 Hardware

11.10.7.10.1 IMMEDIATE_INPUT

Block that performs an instantaneous reading of the selected input value, without changing the value of images (GLOBAL_IO variables).

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Source#	BYTE	Inputs to be read (digital or analog)
VAR_OUTPUT	ENO	BOOL	Output enabling
	Destination	WORD INT UINT	Variable mapped w ith the values of the inputs selected

Operation

When this block has a TRUE value in EN, it gets the immediate value of the selected input in Source#. If selected the analog input AI1, its value is passed on to Destination. If the digital input is selected, its bits are concatenated so that DI1 be the least significant bit and DI10 be the most significant bit and the result is sent to Destination.

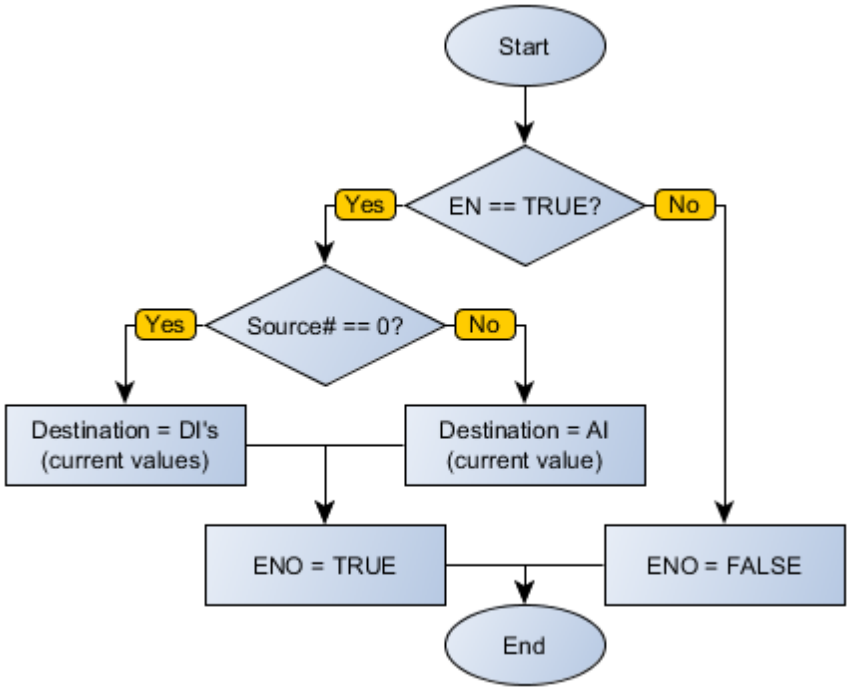
When EN has FALSE value, Destination remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

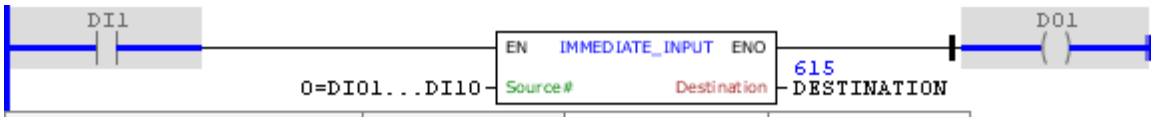
Compatibility

Device	Version
PLC300	1.20 or higher
SCA06	2.00 or higher

Block Flowchart



Example



DESTINATION	WORD	1001100111	Binary
DI1	BOOL	1	Binary
DI2	BOOL	1	Binary
DI3	BOOL	1	Binary
DI4	BOOL	0	Binary
DI5	BOOL	0	Binary
DI6	BOOL	1	Binary
DI7	BOOL	1	Binary
DI8	BOOL	0	Binary
DI9	BOOL	0	Binary
DI10	BOOL	1	Binary

The example above is an immediate reading of the signs of the digital inputs DI1 to DI10 of the PLC300. This reading is then interpreted as a binary sequence with DI1 being the least significant bit and the result is sent to the DESTINATION variable. The block ends with success, ENO output is activated.

11.10.7.10.2 IMMEDIATE_OUTPUT

Block that performs an instantaneous reading of the selected output port, without changing the value of images (GLOBAL_IO variables).

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Source	WORD INT UINT	Variable mapped with the values to be sent to the selected outputs
VAR_OUTPUT	ENO	BOOL	Output enabling
	Destination#	BYTE	Outputs to be written (digital or analog)

Operation

When this block has a TRUE value in EN, it writes immediately in the selected output the value of Source. If selected analog output AO1, the Source value is passed on to it. If the digital outputs are selected, DO1 will receive the zero bit of Source, DO2 bit one, DO3 bit two, and so on.

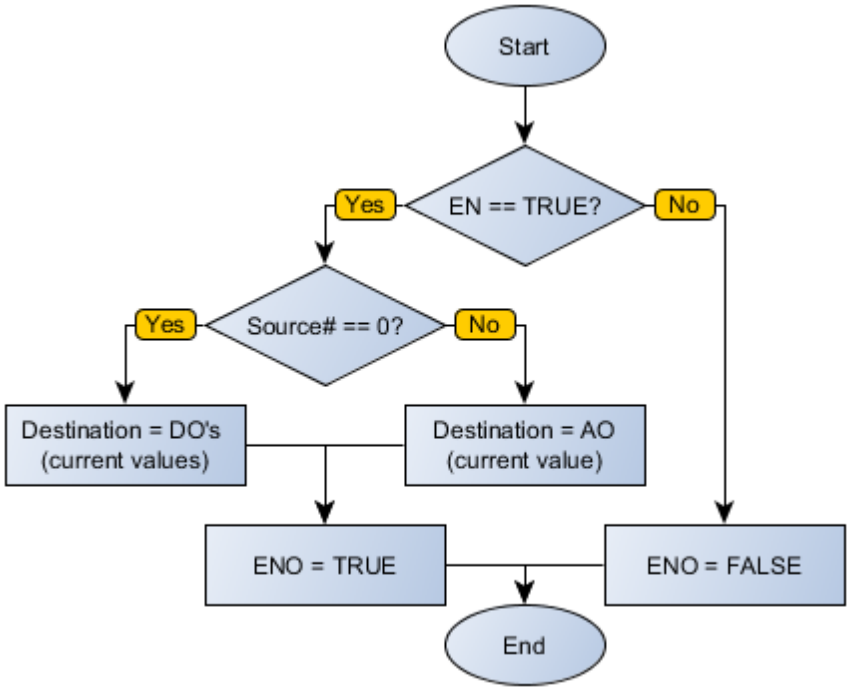
When EN has FALSE value, Destination# remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

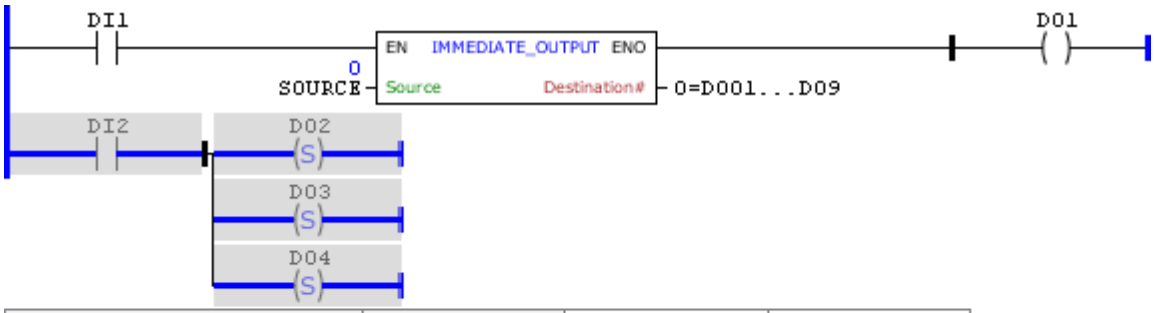
Compatibility

Device	Version
PLC300	1.20 or higher
SCA06	2.00 or higher

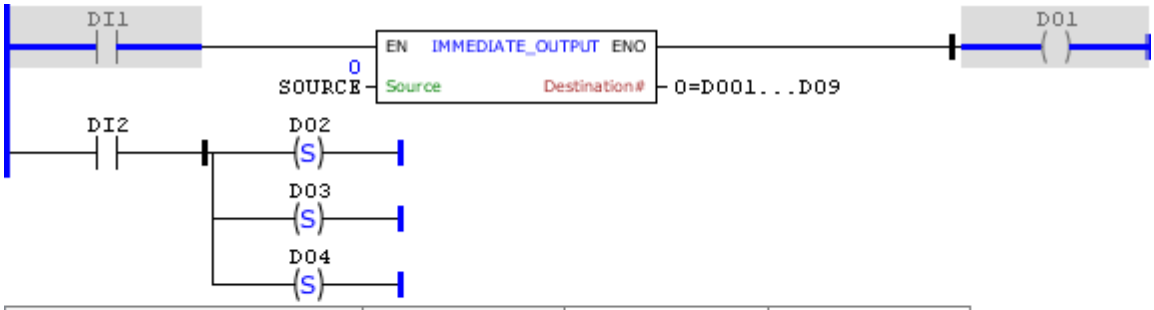
Block Flowchart



Example



SOURCE	WORD	0	Binary
DO1	BOOL	0	Binary
DO2	BOOL	1	Binary
DO3	BOOL	1	Binary
DO4	BOOL	1	Binary
DO5	BOOL	0	Binary
DO6	BOOL	0	Binary
DO7	BOOL	0	Binary
DO8	BOOL	0	Binary
DO9	BOOL	0	Binary



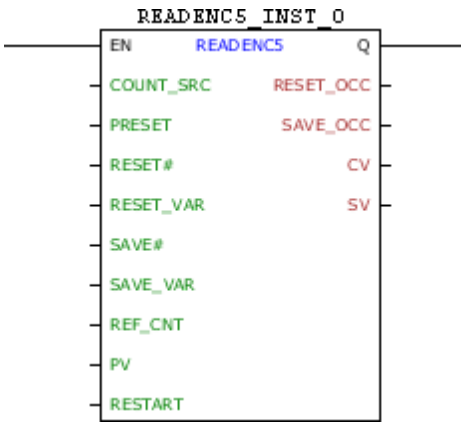
SOURCE	WORD	0	Binary
DO1	BOOL	1	Binary
DO2	BOOL	0	Binary
DO3	BOOL	0	Binary
DO4	BOOL	0	Binary
DO5	BOOL	0	Binary
DO6	BOOL	0	Binary
DO7	BOOL	0	Binary
DO8	BOOL	0	Binary
DO9	BOOL	0	Binary

The above example is for immediate SOURCE written value, interpreted as a binary sequence, the digital outputs DO1 to DO9 of the PLC300 and DO1 receives the value of the least significant bit. The block ends with success, ENO output is activated. Note that the immediate writing does not prevail over direct coil DO1 or over enabling coils in DO2, DO3 and DO4.

11.10.7.10.3 READENC5

Block that performs counting of encoder pulses.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	COUNT_SRC	BYTE	It determines which encoder will be used in counting of pulses
	PRESET	BOOL	Attributes value from PV to CV
	RESET#	BYTE	Chooses the CV reset control
	RESET_VAR	BOOL	If the choice of RESET # is by variable, it performs the reset of CV when in TRUE value
	SAVE#	BYTE	Chooses the saving control of the counter
	SAVE_VAR	BOOL	If the choice of SAVE# is by variable, it performs the saving of CV in SV when in TRUE value
	REF_CNT	LREAL	Value of the reference pulse for output enabling
	PV	LREAL	Value of initial configuration
	RESTART	LREAL	Reference value for automatic reset of CV
VAR_OUTPUT	Q	BOOL	Output enabling
	RESET_OCC	BOOL	Reset flag
	SAVE_OCC	BOOL	Saving flag
	CV	LREAL	Value of pulse counter
	SV	LREAL	Last saved counter value
VAR	READENC5_INST_0	READENC5	Instance of access to block structure

Operation

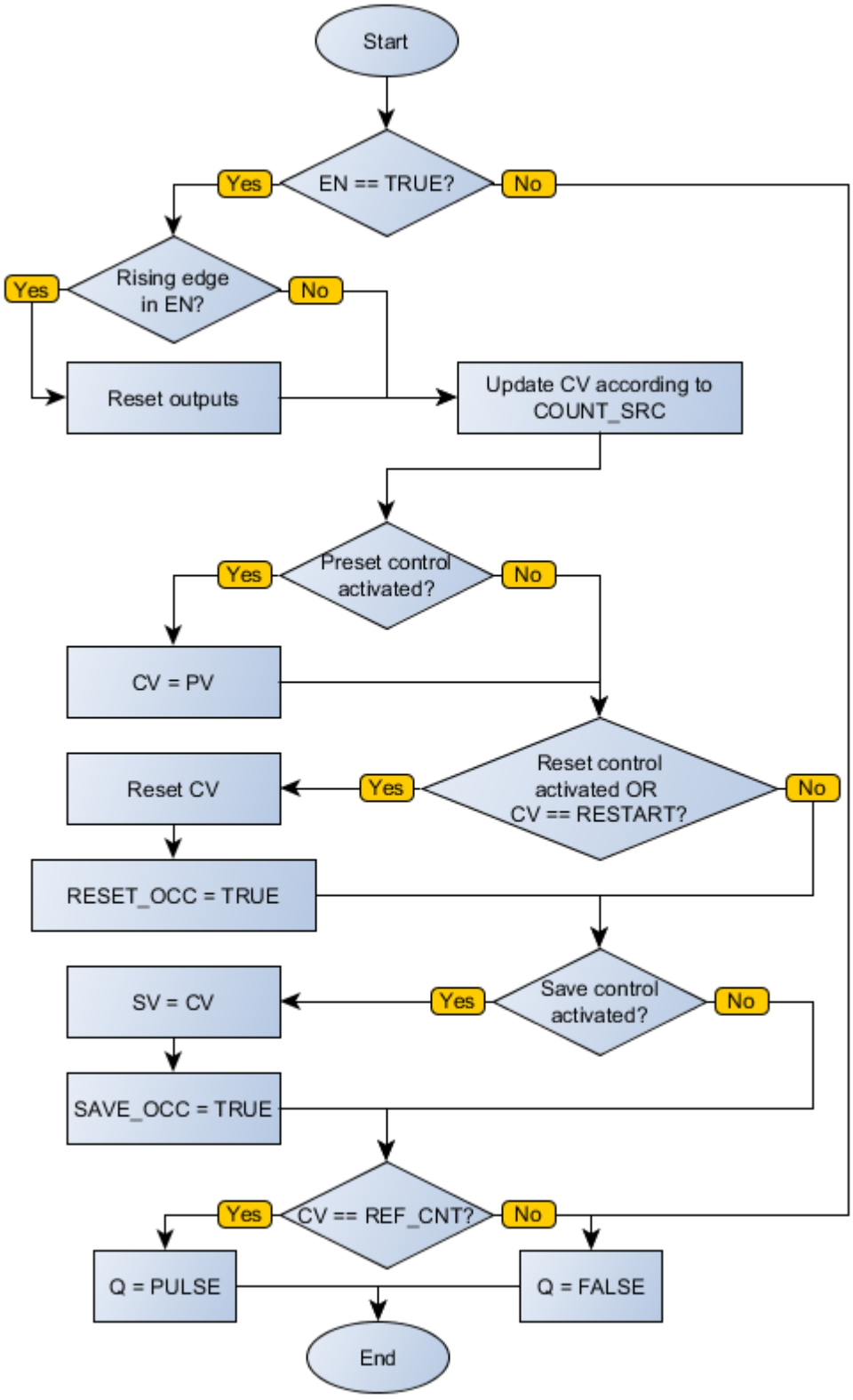
When this block identifies a leading edge in EN, it resets its outputs and counts the pulses from the encoder specified in COUNT_SRC while enabled. This count value is stored in CV.

The specified control in RESET# resets the counter when enabled, while the specified control in SAVE#, when activated, stores the value of CV in SV. Each of these controls sets its respective flag (RESET_OCC or SAVE_OCC) for one scan cycle.

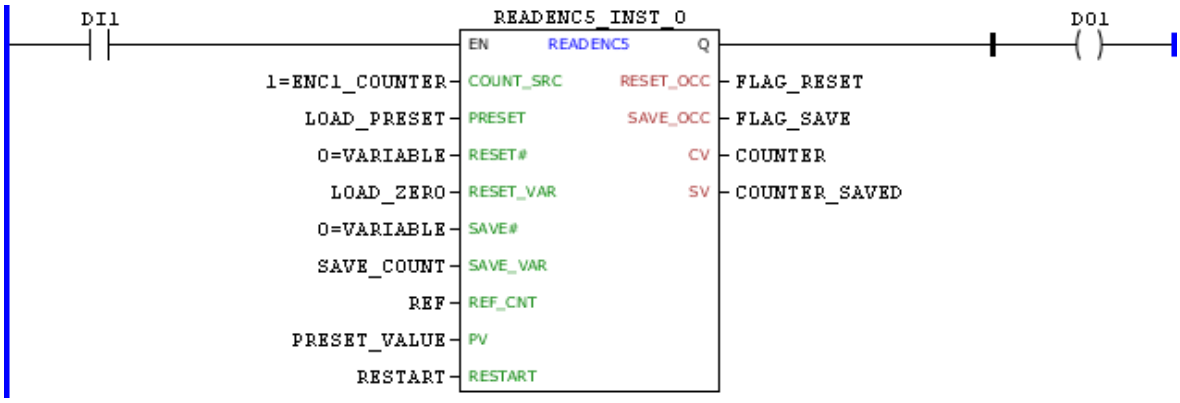
The block also allows you to configure an automatic reset when CV reaches the RESTART value.

The value of Q is activated by one scan cycle when CV reaches the value of REF_CNT.

Block Flowchart


Start
EN == TRUE?
Yes
No
Rising edge in EN?
Yes
No
Reset outputs
Update CV according to COUNT_SRC
Preset control activated?
Yes
No
CV = PV
Reset CV
Yes
Reset control activated OR CV == RESTART?
No
RESET_OCC = TRUE
SV = CV
Yes
Save control activated?
No
SAVE_OCC = TRUE
Yes
CV == REF_CNT?
No
Q = PULSE
Q = FALSE
End

Example



The above example reads the encoder 1 and stores its value in COUNTER, with count reference set in REF and RESET and SAVE control done by the variables LOAD_ZERO and SAVE_COUNT, respectively. When COUNTER is equals to REF, the output Q is set for one scan cycle.

11.10.7.11 Logic

11.10.7.11.1 Logic Bit

11.10.7.11.1.1 RESETBIT

Logical block used to perform reset of a specific bit in a field.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_IN_OUT	Data	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable w hose bit w ill be changed
VAR_INPUT	EN	BOOL	Block enabling
	Position	BYTE USINT	Position of the bit that w ill be changed
VAR_OUTPUT	DONE	BOOL	Operation successful

Operation

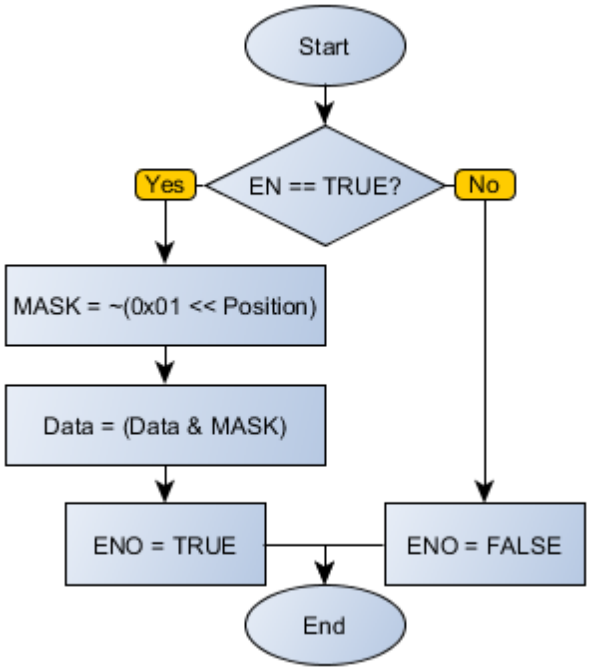
This block when it has a TRUE value in EN, resets the bit indicated in Position in the Data variable that is forwarded to the output already with its updated value.

When EN has FALSE value, Data remains unchanged.

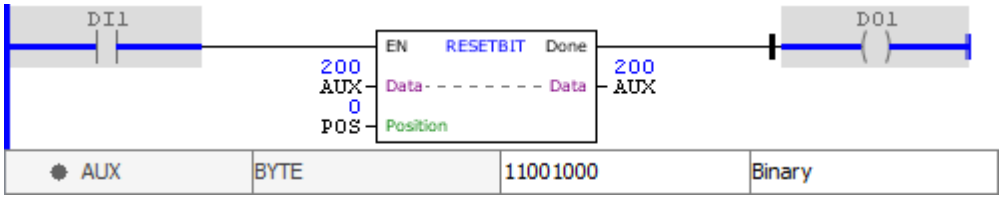
The DONE variable receives the same EN value, except when there is an error in the reset of the bit, then getting a FALSE value.

NOTE!
It is important to notice that Position is within the range of values of bits corresponding to variable type in Data. For example: if Data is a BYTE, it has 8 bits, and Position must contain a value between 0 and 7.

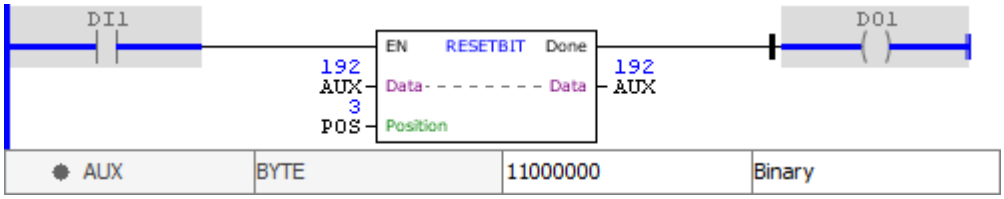
Block Flowchart



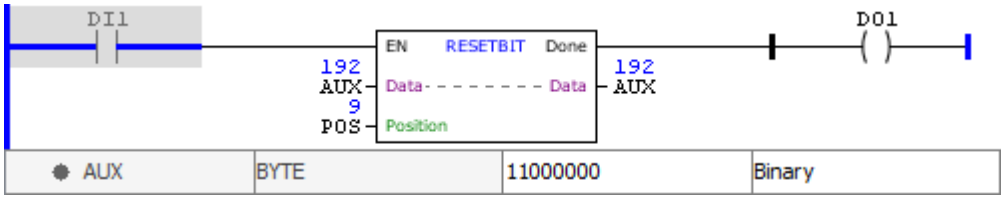
Example



The example above resets the bit of AUX zero position, whose initial value is 200 (1100 1000, in binary). Since this bit already had FALSE value, nothing has changed.



The example above resets the bit in position three of AUX by changing its binary value and, therefore, its decimal representation.



The example above resets the bit in position nine of AUX. Since AUX is a variable BYTE type, it has only eight bits. Thus, the example above creates a runtime error in the block and therefore the output is not enabled.

11.10.7.11.1.2 SETBIT

Logical block used to perform the set of a specific bit in a field.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_IN_OUT	Data	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable w hose bit w ill be changed
VAR_INPUT	EN	BOOL	Block enabling
	Position	BYTE USINT	Position of the bit that w ill be changed
VAR_OUTPUT	DONE	BOOL	Operation successful

Operation

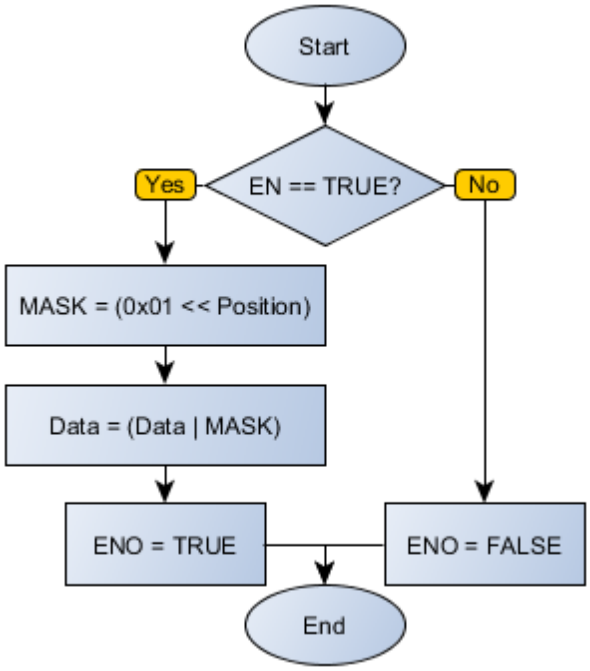
This block when it has a TRUE value in EN, sets the bit indicated in Position in the Data variable that is forwarded to the output already with its updated value.

When EN has FALSE value, Data remains unchanged.

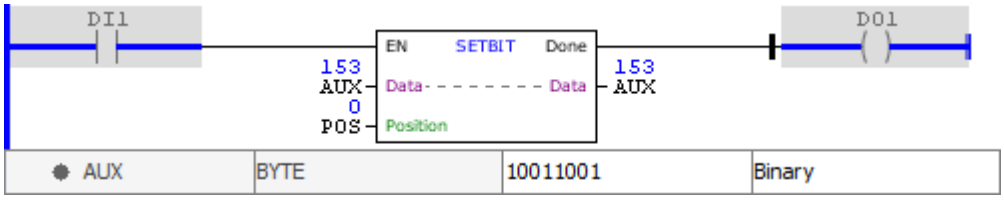
The DONE variable receives the same EN value, except when there is an error in the set of the bit, then getting a FALSE value.

NOTE!
It is important to notice that Position is within the range of values of bits corresponding to variable type in Data. For example: if Data is a BYTE, it has 8 bits, and Position must contain a value between 0 and 7.

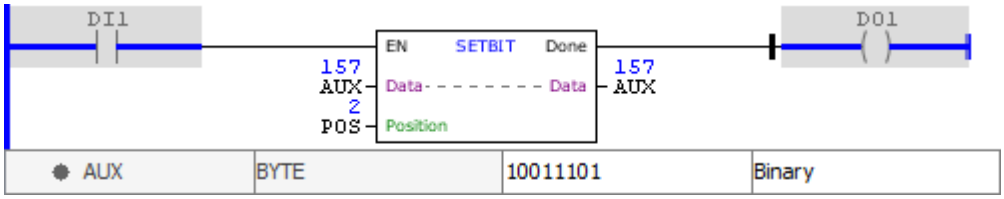
Block Flowchart



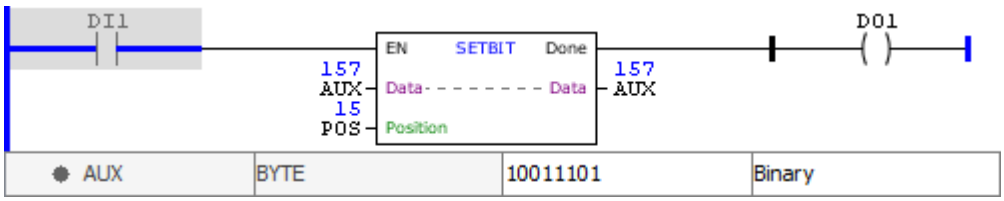
Example



The example above sets the bit of AUX zero position, whose initial value is 153 (1001 1001, in binary). Since this bit already had TRUE value, nothing has changed.



The example above sets the bit in position three of AUX by changing its binary value and, therefore, its decimal representation.



The example above sets the bit in position fifteen of AUX. Since AUX is a variable BYTE type, it has

only eight bits. Thus, the example above creates a runtime error in the block and therefore the output is not enabled.

11.10.7.11.1.3 TESTBIT

Logical block that revolutions the value of a specific bit in a field.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	Data	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable w hose bit w ill be tested
	EN	BOOL	Block enabling
	Position	BYTE USINT	Position of the bit that w ill be changed
VAR_OUTPUT	Q	BOOL	Value of the tested bit

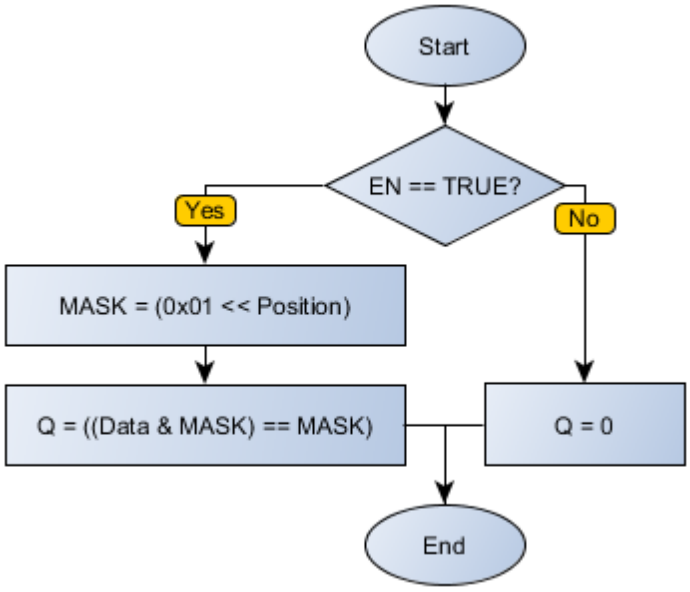
Operation

This block when it has a TRUE value in EN, sends to the output Q the bit value indicated in Position in the Data variable.

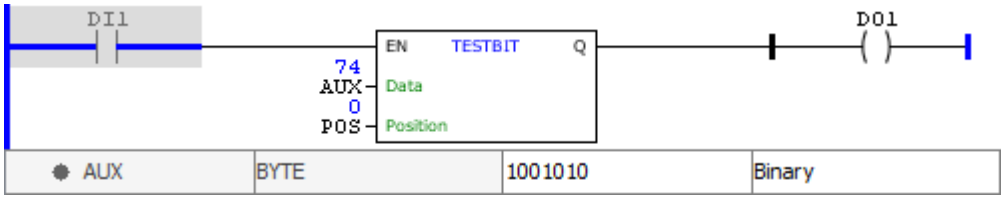
When EN has FALSE value, Q also receives FALSE.

NOTE!
It is important to notice that Position is within the range of values of bits corresponding to variable type in Data. For example: if Data is a BYTE, it has 8 bits, and Position must contain a value between 0 and 7.

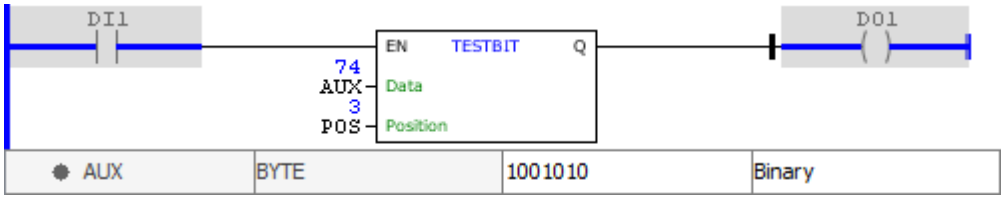
Block Flowchart



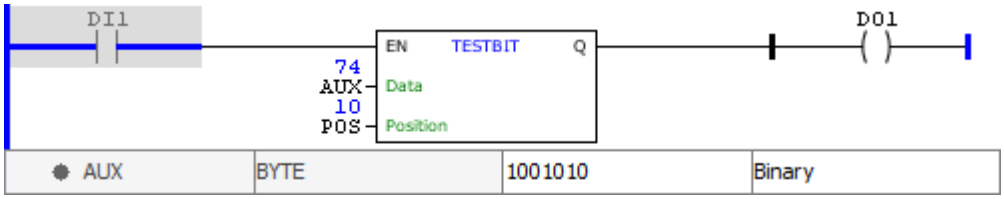
Example



The example above sets the bit value of zero position of AUX, whose initial value is 74 (0100 1010 in binary) to the output Q. Since this bit has value 0, the output is disabled.



The example above sets the value of the bit of position three of AUX to the output Q. Since this bit has value 1, the output is enabled.



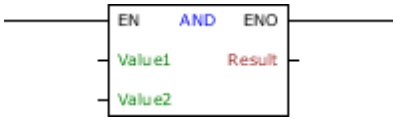
The example above sets the bit value of position ten of AUX to output Q. Since AUX is a variable of BYTE type, it has only eight bits. Thus, the example above creates a runtime error in the block and therefore the output is disabled.

11.10.7.11.2 Logic Boolean

11.10.7.11.2.1 AND

Logical block that performs an boolean "and" operation between two variables, storing the result in a third one.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable for the operation
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable for the operation
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable that stores the result of the operation

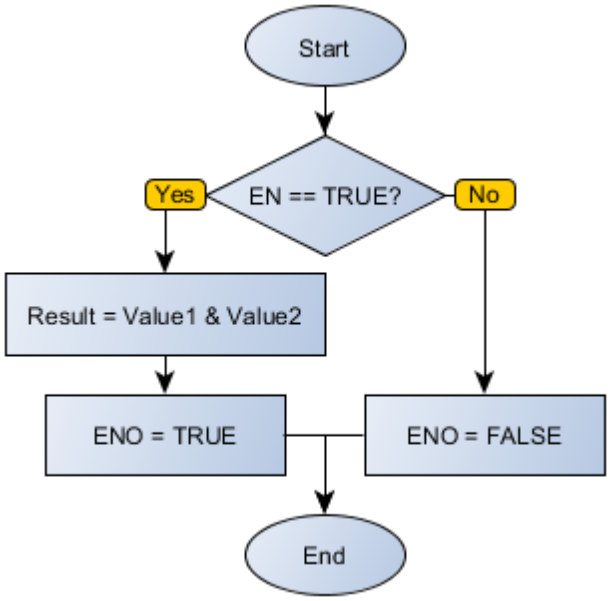
Operation

When this block has a TRUE value in EN, it sends to the Result output the “and” Boolean operation of input variables Value1 and Value2.

When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example

AUX	BYTE	11001000	Binary
AUX2	BYTE	10111001	Binary
AUX3	BYTE	10001000	Binary

The example above performs an "and" Boolean operation between AUX and AUX2, storing the result in AUX3.

11.10.7.11.2.2 NOT

Block that performs a logical operation of boolean "not" in a variable, storing the result in another.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Reference variable for the operation
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable that stores the result of the operation

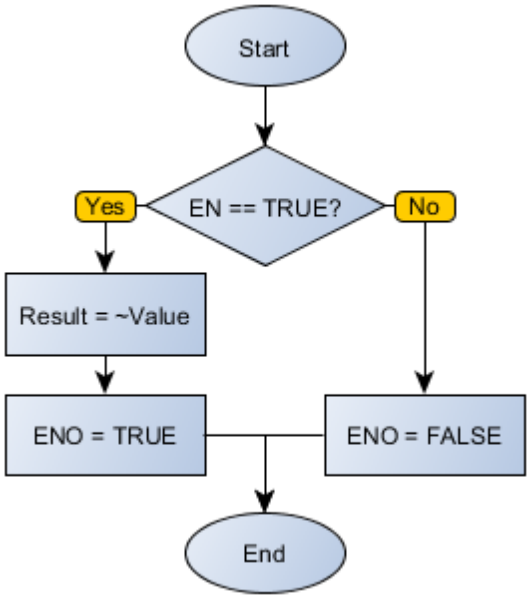
Operation

When this block has a TRUE value in EN, it sends to the Result output the denied Boolean value of the Value input variable.

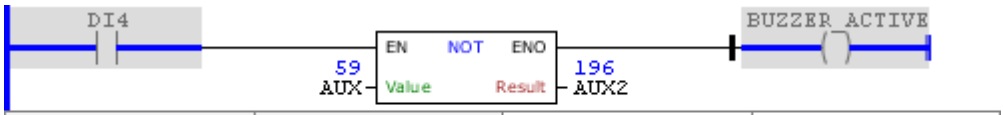
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



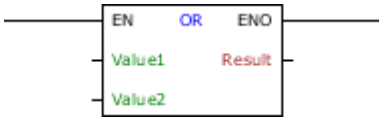
AUX	BYTE	111011	Binary
AUX2	BYTE	11000100	Binary

The example above performs a boolean "not" operation in AUX, storing the result in AUX2.

11.10.7.11.2.3 OR

Logical block that performs an Boolean "or" operation between two variables, storing the result in a third one.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable for the operation
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable for the operation
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable that stores the result of the operation

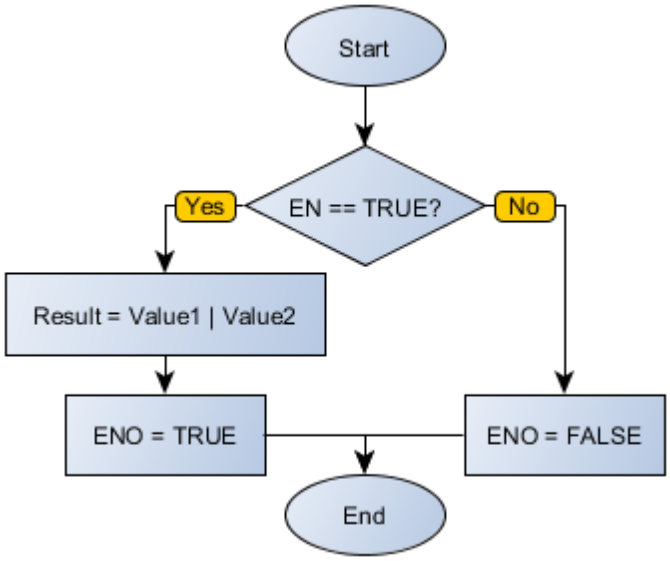
Operation

When this block has a TRUE value in EN, it sends to the Result output the “or” Boolean operation of input variables Value1 and Value2.

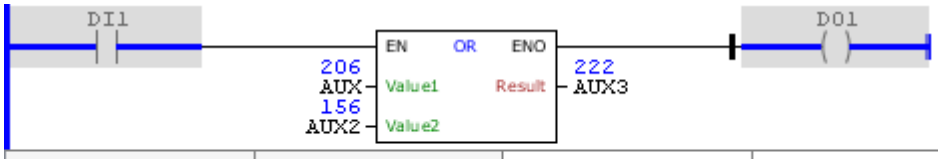
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



AUX	BYTE	11001110	Binary
AUX2	BYTE	10011100	Binary
AUX3	BYTE	11011110	Binary

The example above performs an "or" Boolean operation between AUX and AUX2, storing the result in AUX3.

11.10.7.11.2.4 XNOR

Logical block that performs an Boolean "not exclusive or" operation between two variables, storing the result in a third one.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable for the operation
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable for the operation
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable that stores the result of the operation

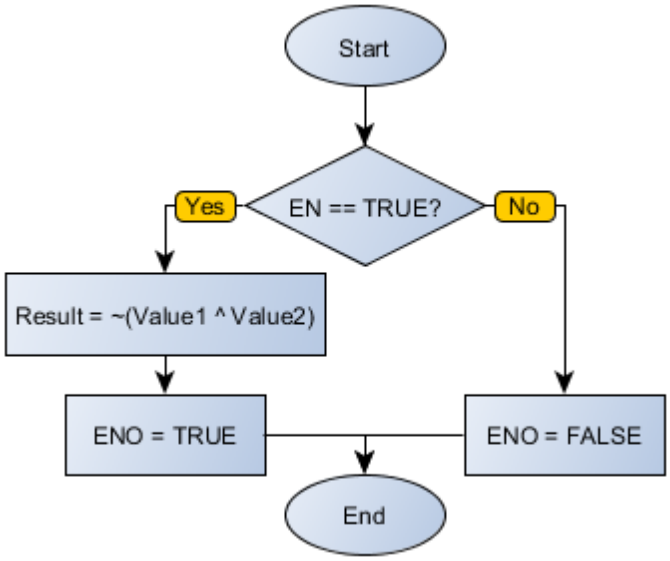
Operation

When this block has a TRUE value in EN, it sends to the Result output the “denied exclusive or” Boolean operation of input variables Value1 and Value2.

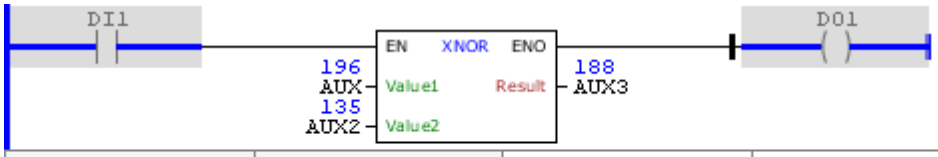
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



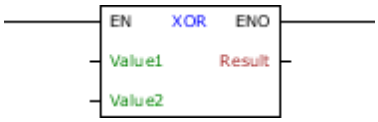
AUX	BYTE	11000100	Binary
AUX2	BYTE	10000111	Binary
AUX3	BYTE	10111100	Binary

The example above performs a "denied exclusive or" Boolean operation between AUX and AUX2, storing the result in AUX3.

11.10.7.11.2.5 XOR

Logical block that performs an Boolean "exclusive or" operation between two variables, storing the result in a third one.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable for the operation
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable for the operation
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable that stores the result of the operation

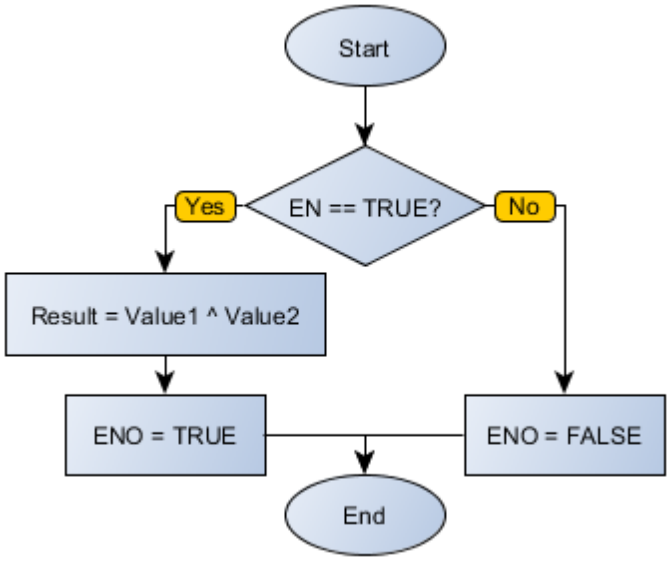
Operation

When this block has a TRUE value in EN, it sends to the Result output the “xor” Boolean operation of input variables Value1 and Value2.

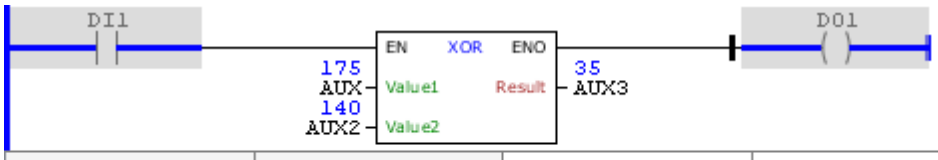
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



AUX	BYTE	10101111	Binary
AUX2	BYTE	10001100	Binary
AUX3	BYTE	100011	Binary

The example above performs a "xor" Boolean operation between AUX and AUX2, storing the result in AUX3.

11.10.7.11.3 Logic Rotate

11.10.7.11.3.1 ROL

Block that performs a logical left rotation operation in a value passed by Value, storing the result in Result.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable to undergo rotation
	Shift	BYTE USINT	Shift index
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable that stores the result of the operation

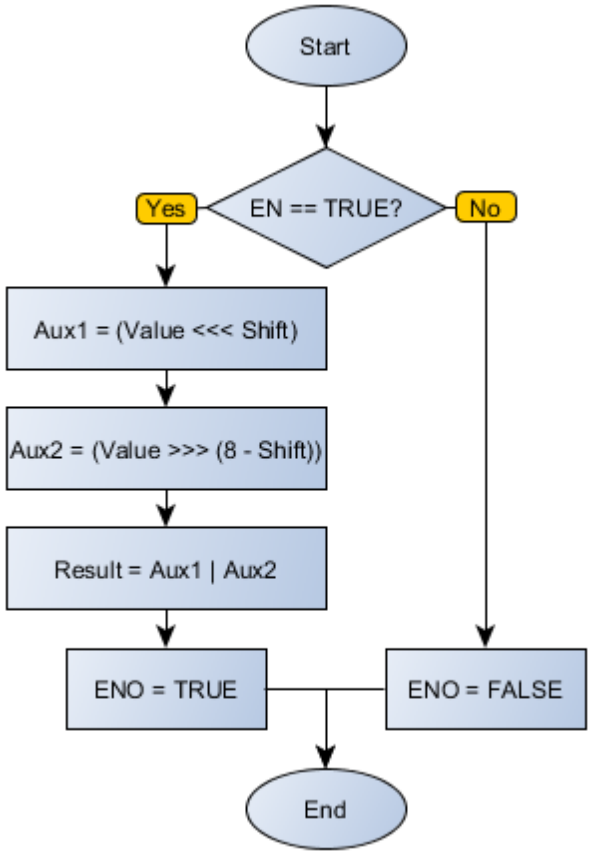
Operation

When this block has a TRUE value in EN, it sends to the Result output the value of the Value variable after performing a number of logical left shifts, according to the Shift value. The most significant bits that are being discarded are returned to the least significant bits, characterizing the rotation.

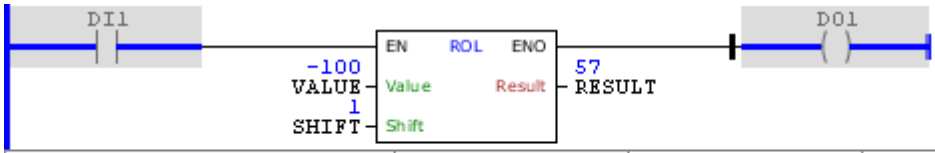
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart

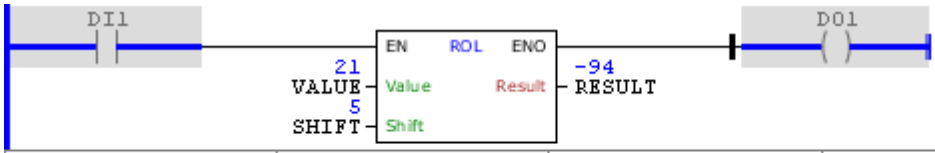


Example



VALUE	SINT	10011100	Binary
SHIFT	BYTE	1	Decimal
RESULT	SINT	111001	Binary

The above example performs a logical left shift by one position in the VALUE variable whose initial value is -100 (1001 1100 in binary). The discarded bits on the left are reinserted on the right. The final result (0011 1001 in binary) is stored in RESULT.



VALUE	SINT	10101	Binary
SHIFT	BYTE	5	Decimal
RESULT	SINT	10100010	Binary

The above example performs a logical left rotation by five positions in the VALUE variable whose initial value is 21 (0001 0101 in binary). The discarded bits on the left are reinserted on the right. The final result (1010 0010 in binary) is stored in RESULT.

11.10.7.11.3.2 ROR

Block that performs a logical right rotation operation in a value passed by Value, storing the result in Result.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable to undergo rotation
	Shift	BYTE USINT	Shift index
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable that stores the result of the operation

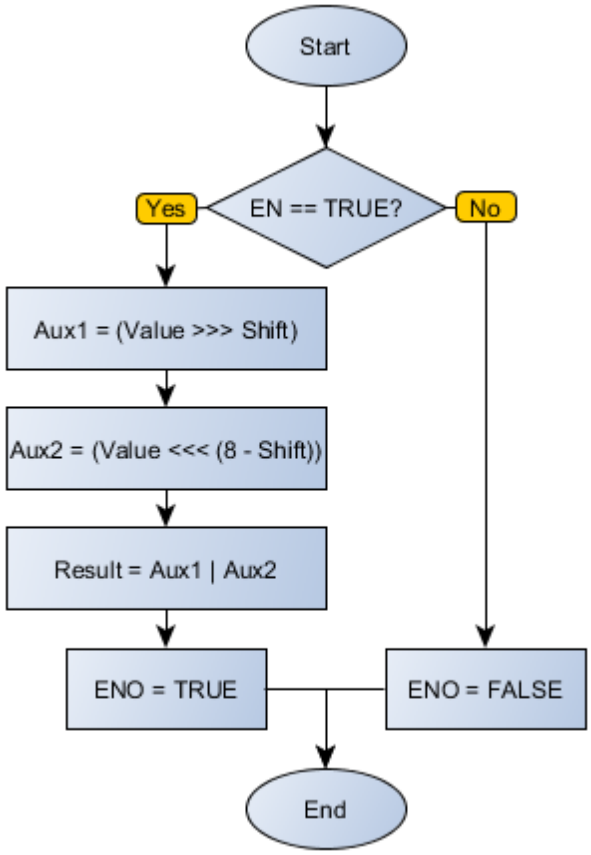
Operation

When this block has a TRUE value in EN, it sends to the Result output the value of the Value variable after performing a number of logical right shifts, according to the Shift value. The least significant bits that are being discarded are returned to the most significant bits, characterizing the rotation.

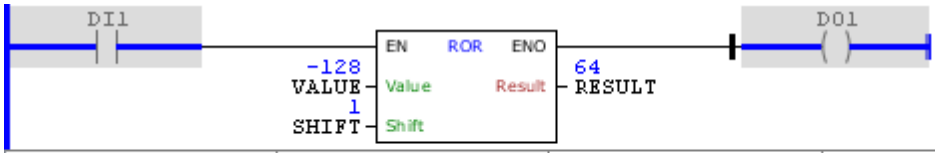
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart

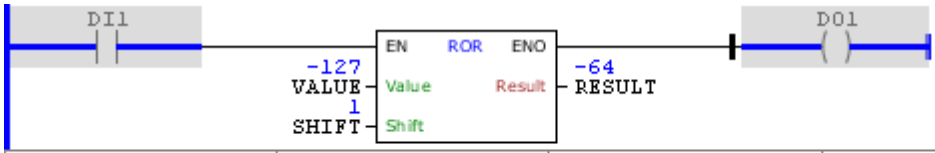


Example



VALUE	SINT	10000000	Binary
SHIFT	BYTE	1	Decimal
RESULT	SINT	1000000	Binary

The above example performs a logic right shift by one position in the VALUE variable whose initial value is -128 (1000 0000 in binary). The discarded bits on the right are reinserted on the left. The final result (0100 0000 in binary) is stored in RESULT. Notice that the sign is not preserved in this operation.



VALUE	SINT	10000001	Binary
SHIFT	BYTE	1	Decimal
RESULT	SINT	11000000	Binary

The above example performs a logical right rotation by one position in the VALUE variable whose initial value is -127 (1000 0001 in binary). The discarded bits on the right are reinserted on the left. The final result (1100 0000 in binary) is stored in RESULT.

11.10.7.11.4 Logic Shift

11.10.7.11.4.1 ASHL

Block that performs a binary left shift operation in a value passed by Value, storing the result in Result.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	SINT INT DINT	Variable to undergo shift
	Shift	BYTE USINT	Shift index
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	SINT INT DINT	Variable that stores the result of the operation

Operation

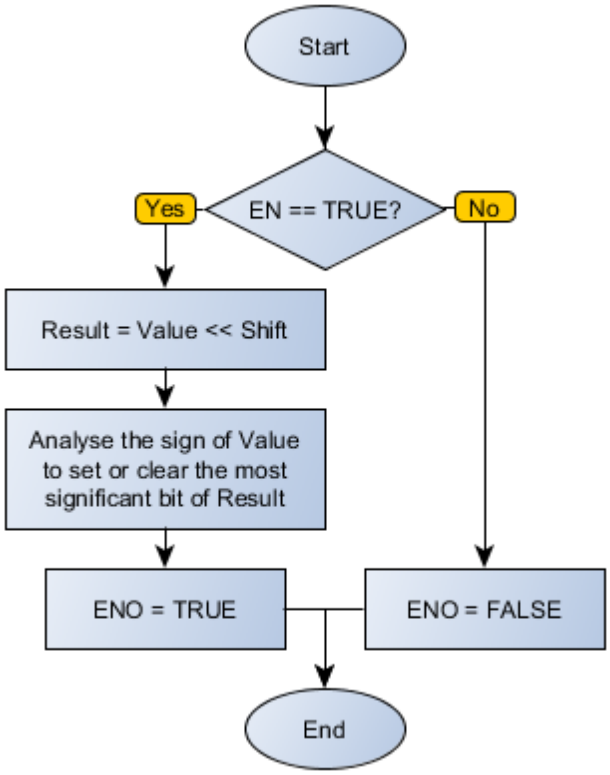
When this block has a TRUE value in EN, it sends to the Result output the value of the Value variable after performing a number of arithmetic left shifts, according to the Shift value.

NOTE!
All arithmetic shifts implemented maintain the sign of the variable.

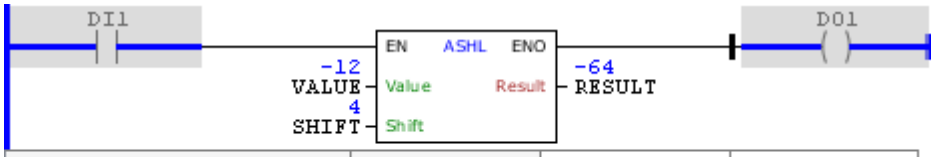
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart

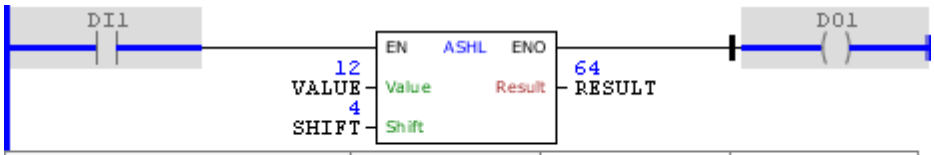


Example



VALUE	SINT	11110100	Binary
SHIFT	USINT	100	Binary
RESULT	SINT	11000000	Binary

Description of exemple.



VALUE	SINT	1100	Binary
SHIFT	USINT	100	Binary
RESULT	SINT	1000000	Binary

Description of exemple.

11.10.7.11.4.2 ASHR

Block that performs arithmetic left shift operation in a value passed by Value, storing the result in Result.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	SINT INT DINT	Variable to undergo shift
	Shift	BYTE USINT	Shift index
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	SINT INT DINT	Variable that stores the result of the operation

Operation

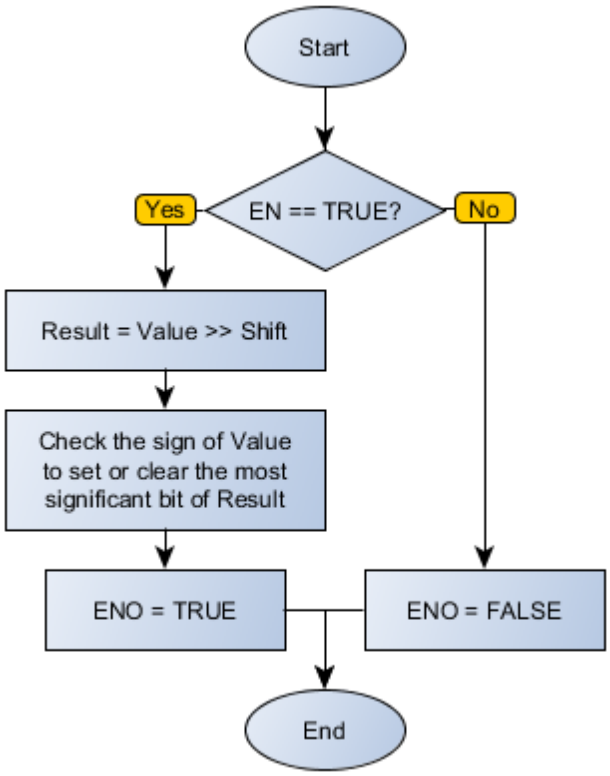
When this block has a TRUE value in EN, it sends to the Result output the value of the Value variable after performing a number of arithmetic right shifts, according to the Shift value.

NOTE!
All arithmetic shifts implemented maintain the sign of the variable.

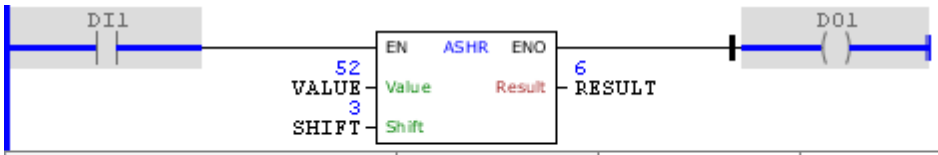
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart

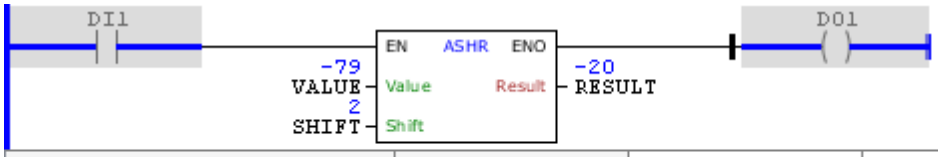


Example



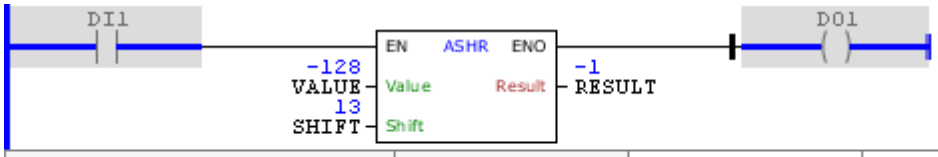
VALUE	SINT	110100	Binary
SHIFT	BYTE	3	Decimal
RESULT	INT	110	Binary

The above example performs an arithmetic right shift by three positions in the VALUE variable whose initial value is 52 (0011 0100 in binary). The bits on the right are being discarded, and on the left new zeros are inserted. The final result (0000 0110 in binary) is stored in RESULT.



VALUE	SINT	10110001	Binary
SHIFT	BYTE	2	Decimal
RESULT	INT	1111111111101100	Binary

The above example performs an arithmetic right shift by two positions in the VALUE variable whose initial value is -79 (1011 0001 in binary). The bits on the right will be discarded and new ones on the left are inserted, since the arithmetic right shifts preserve the sign of the variable. The final result (1111 0110 in binary) is stored in RESULT.



VALUE	SINT	10000000	Binary
SHIFT	BYTE	13	Decimal
RESULT	INT	1111111111111111	Binary

The above example performs an arithmetic right shift by thirteen positions in the VALUE variable whose initial value is -128 (1000 0000 in binary). The bits on the right are being discarded, and on the left new ones are inserted. The final result (1111 1111 in binary) is stored in RESULT.

11.10.7.11.4.3 SHL

Block that performs a binary logical left shift operation in a value passed by Value, storing the result in Result.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable to undergo shift
	Shift	BYTE USINT	Shift index
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable that stores the result of the operation

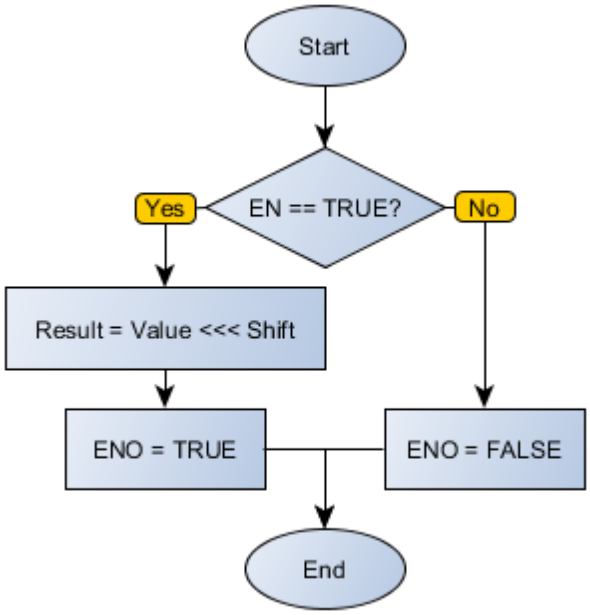
Operation

When this block has a TRUE value in EN, it sends to the Result output the value of the Value variable after performing a number of logical shifts left, according to the Shift value.

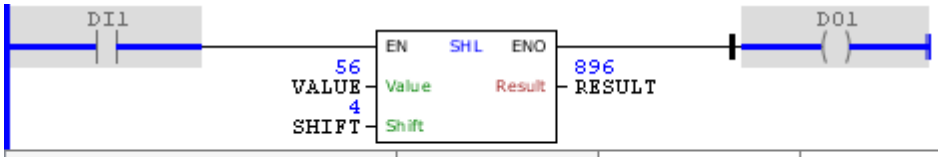
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart

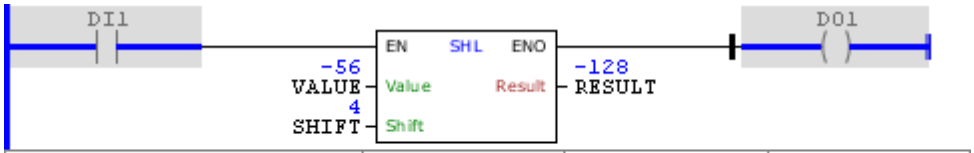


Example



VALUE	INT	111000	Binary
SHIFT	BYTE	4	Decimal
RESULT	INT	1110000000	Binary

The above example performs a logical right shift by four positions in the VALUE variable whose initial value is 56 (0011 1000 in binary). The bits on the left are being discarded, and on the left new zeros are inserted. The final result (0011 1000 0000 in binary) is stored in RESULT.



VALUE	SINT	11001000	Binary
SHIFT	USINT	4	Decimal
RESULT	SINT	10000000	Binary

The above example performs a logical right shift by four positions in the VALUE variable whose initial value is -56 (1100 1000 in binary). The bits on the left are being discarded, and on the left new zeros are inserted. The final result (1100 1000 0000 in binary) is stored in RESULT. Since RESULT is SINT type, it only accepts the first eight bits (1000 0000).

11.10.7.11.4.4 SHR

Block that performs a binary logical right shift operation in a value passed by Value, storing the result in Result.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable to undergo shift
	Shift	BYTE USINT	Shift index
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable that stores the result of the operation

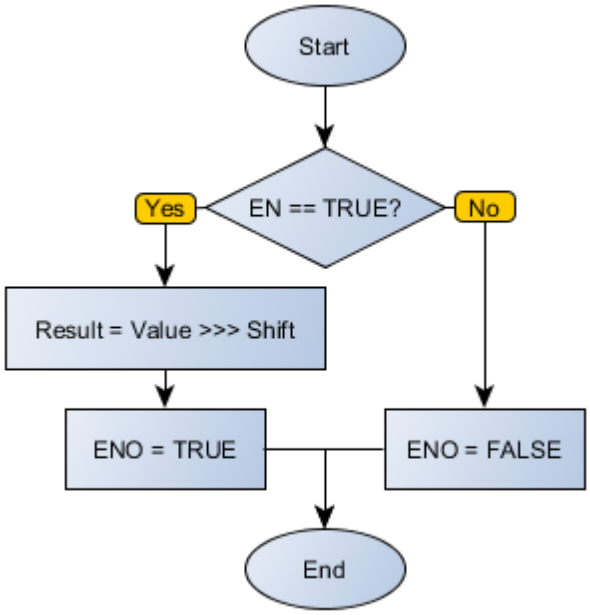
Operation

When this block has a TRUE value in EN, it sends to the Result output the value of the Value variable after performing a number of logical shifts right, according to the Shift value.

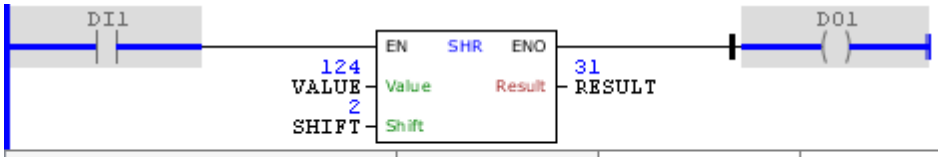
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart

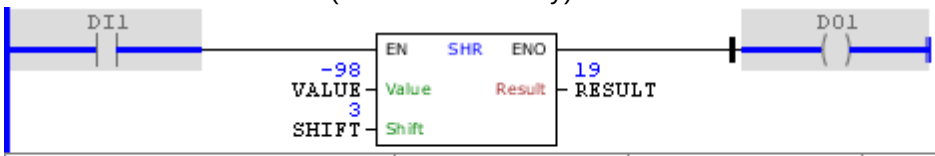


Example



VALUE	SINT	1111100	Binary
SHIFT	BYTE	2	Decimal
RESULT	SINT	11111	Binary

The above example performs a logical right shift by two positions in the VALUE variable whose initial value is 124 (0111 1100 in binary). The bits on the right are being discarded, and on the left new zeros are inserted. The final result (0001 1111 in binary) is stored in RESULT.



VALUE	SINT	10011110	Binary
SHIFT	BYTE	3	Decimal
RESULT	INT	10011	Binary

The above example performs a logical right shift by three positions in the VALUE variable whose initial value is -98 (1001 1110 in binary). The bits on the right are being discarded, and on the left new zeros are inserted. The final result (0001 0011 in binary) is stored in RESULT.

11.10.7.12 Math

11.10.7.12.1 Math Basic

11.10.7.12.1.1 ABS

Block that calculates the Value module, storing the result in Result.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Reference variable for the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Variable that stores the result of the operation

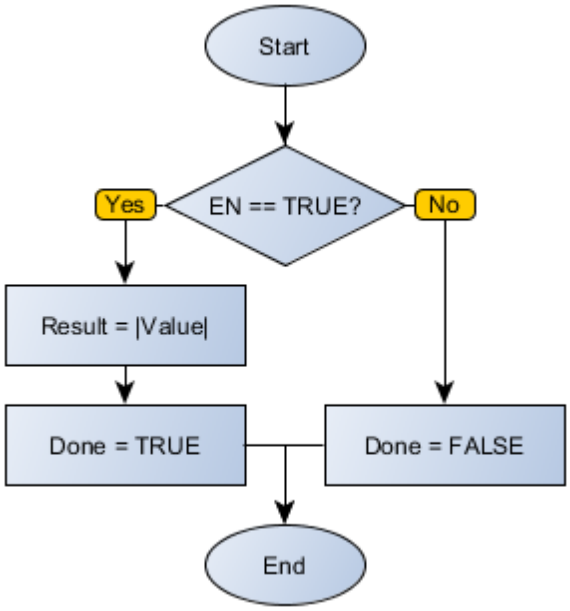
Operation

When this block has a TRUE value in EN, it sends to the Result output the absolute value of the

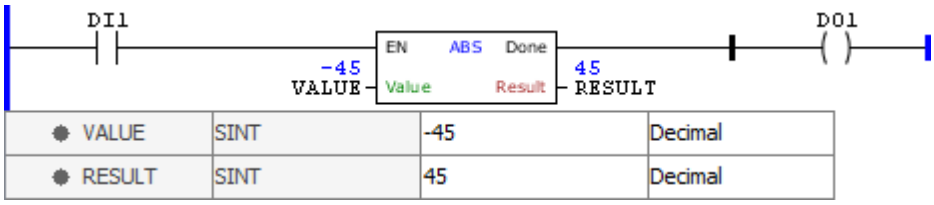
Value variable. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

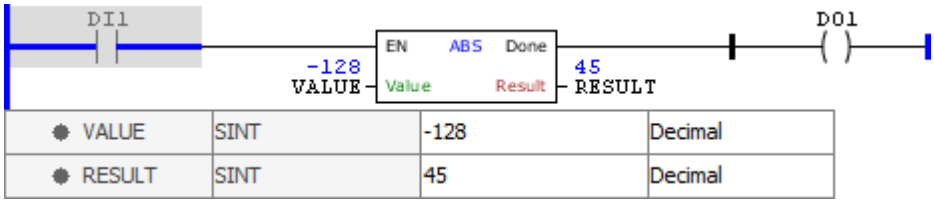


Example



VALUE	SINT	-45	Decimal
RESULT	SINT	45	Decimal

The above example calculates the absolute value of the VALUE variable whose initial value is -45, storing the final result, 45, in RESULT.



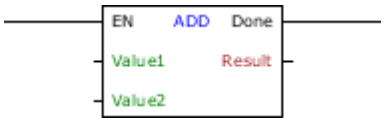
VALUE	SINT	-128	Decimal
RESULT	SINT	45	Decimal

The above example calculates the absolute value of the VALUE variable whose initial value is -45. The final result, 128, cannot be stored in RESULT, because it is outside the limits of accepted values by SINT type. Therefore, RESULT remains unchanged and the output is disabled.

11.10.7.12.1.2 ADD

Block that calculates the sum of the values of Value1 and Value2, storing the result in Result.

Ladder Representation



Block Structure

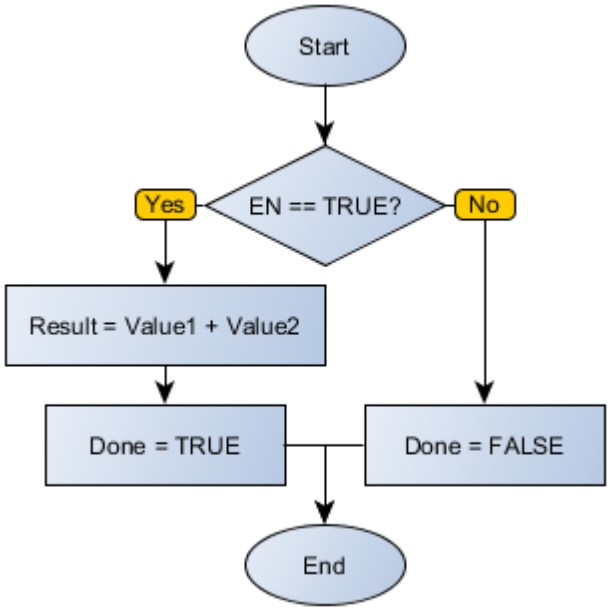
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	First addend of the operation
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Second addend of the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Variable that stores the result of the operation

Operation

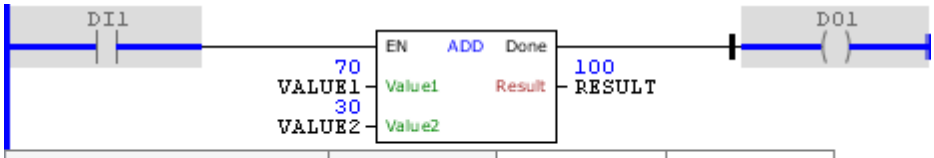
When this block has a TRUE value in EN, it sends to the Result output the sum of Value1 and Value2 variables. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

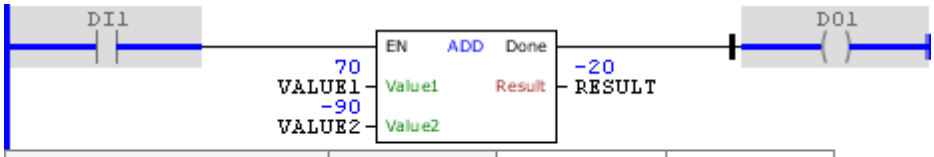


Example



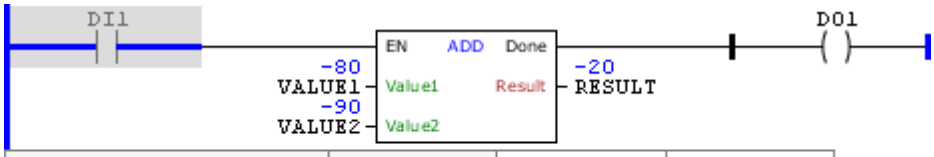
VALUE1	SINT	70	Decimal
VALUE2	SINT	30	Decimal
RESULT	SINT	100	Decimal

The above example calculates the sum of VALUE 1 and VALUE2 variables, storing the final result in RESULT.



VALUE1	SINT	70	Decimal
VALUE2	SINT	-90	Decimal
RESULT	SINT	-20	Decimal

The above example calculates the sum of VALUE 1 and VALUE2 variables, storing the final result in RESULT. Notice that the block accepts arguments of both signs.



VALUE1	SINT	-80	Decimal
VALUE2	SINT	-90	Decimal
RESULT	SINT	-20	Decimal

The above example calculates the sum of VALUE1 and VALUE2 variables. The final result -170 cannot be stored in RESULT, because it is outside the limits of accepted values by SINT type. Therefore, RESULT remains unchanged and the output is disabled.

11.10.7.12.1.3 DIV

Block that calculates the division of the values of Value1 and Value2, storing the result in Result.

Ladder Representation

Block Structure

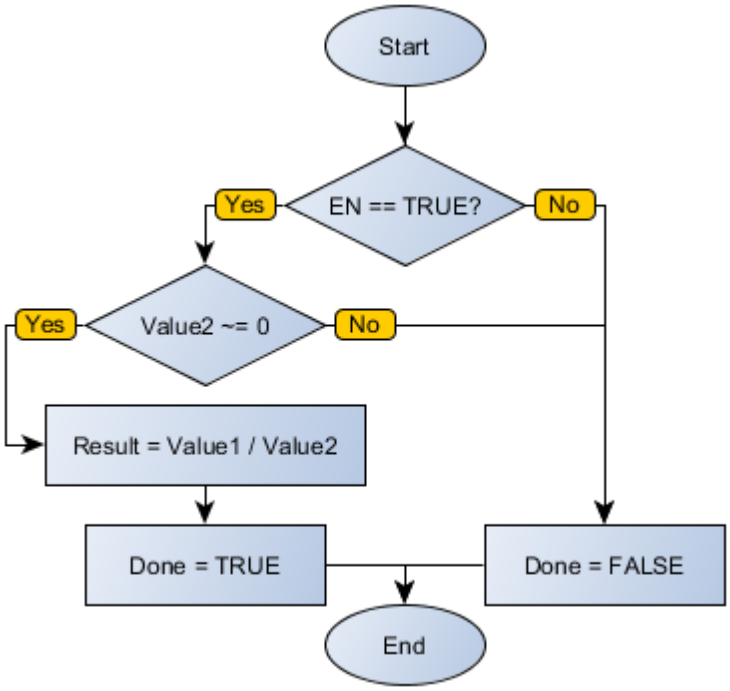
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Dividend of the operation
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Divisor of the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Variable that stores the result of the operation

Operation

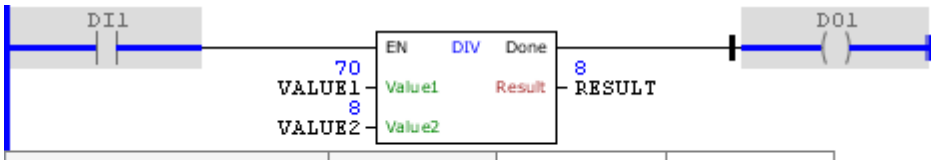
When this block has a TRUE value in EN, it sends to the Result output the division of Value1 and Value2 variables. The value stored will be the exact division if Result is REAL, or, in other cases, only the quotient. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

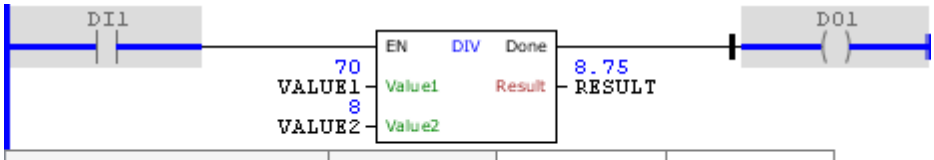


Example



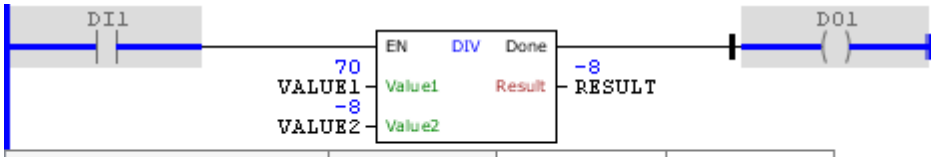
VALUE1	SINT	70	Decimal
VALUE2	SINT	8	Decimal
RESULT	SINT	8	Decimal

The above example calculates the division of VALUE 1 and VALUE2 variables, storing the final result in RESULT. Since RESULT is SINT type, only the quotient is stored in it.



VALUE1	SINT	70	Decimal
VALUE2	SINT	8	Decimal
RESULT	REAL	8.75	Float

The above example calculates the division of VALUE 1 and VALUE2 variables, storing the final result in RESULT. Since RESULT is of REAL type, the exact value of the division is stored in it.



VALUE1	SINT	70	Decimal
VALUE2	SINT	-8	Decimal
RESULT	SINT	-8	Decimal

The above example calculates the division of VALUE 1 and VALUE2 variables, storing the final result in RESULT. Since RESULT is SINT type, only the quotient is stored in it. Notice that the block accepts arguments of both signs.

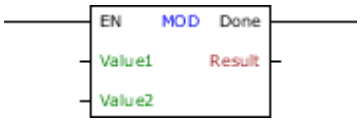
VALUE1	SINT	70	Decimal
VALUE2	SINT	0	Decimal
RESULT	SINT	-8	Decimal

The above example calculates the division of VALUE 1 and VALUE2 variables, storing the final result in RESULT. Since VALUE2 is zero, the block generates a runtime error, RESULT remains unchanged and the output is disabled.

11.10.7.12.1.4 MOD

Block that calculates the remainder of the values of Value1 and Value2, storing the result in Result.

Ladder Representation



Block Structure

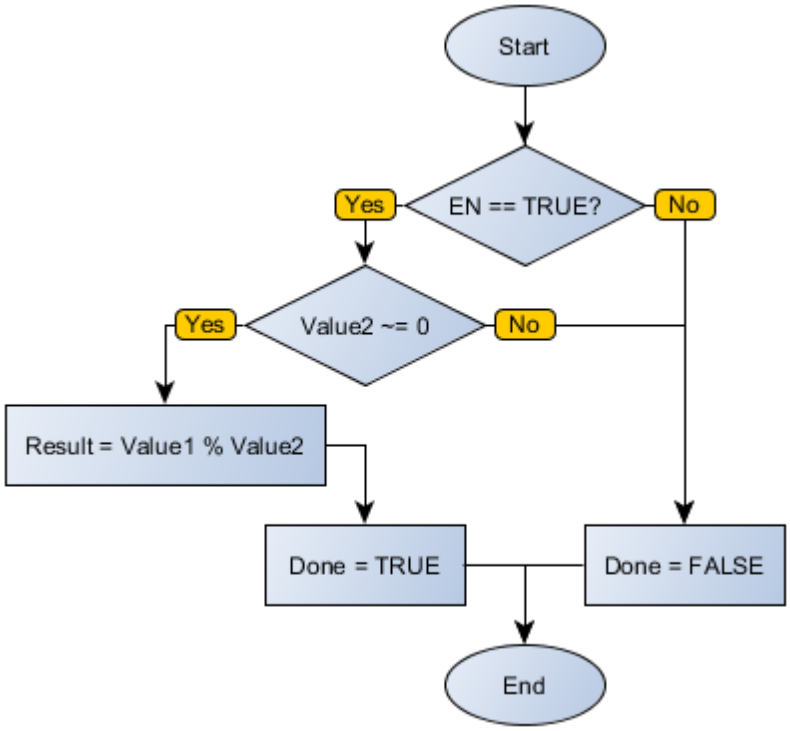
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Dividend of the operation
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Divisor of the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable that stores the result of the operation

Operation

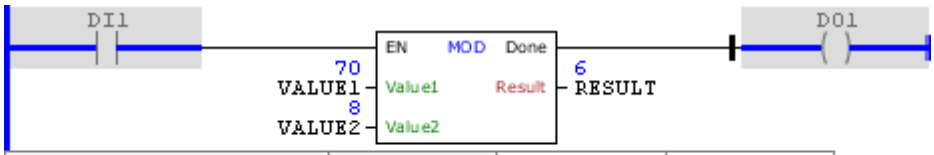
When this block has a TRUE value in EN, it sends to the Result output the remainder of Value1 and Value2 variables. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

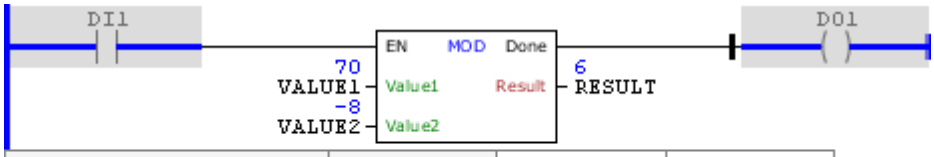


Example



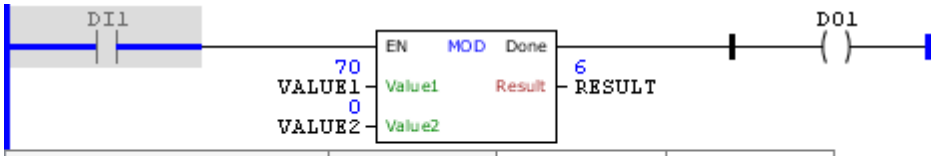
VALUE1	SINT	70	Decimal
VALUE2	SINT	8	Decimal
RESULT	SINT	6	Decimal

The above example calculates the remainder of VALUE 1 and VALUE2 variables, storing the final result in RESULT.



VALUE1	SINT	70	Decimal
VALUE2	SINT	-8	Decimal
RESULT	SINT	6	Decimal

The above example calculates the remainder of VALUE 1 and VALUE2 variables, storing the final result in RESULT. Notice that the block accepts arguments of both signs.



VALUE1	SINT	70	Decimal
VALUE2	SINT	0	Decimal
RESULT	SINT	6	Decimal

The above example calculates the remainder of VALUE 1 and VALUE2 variables, storing the final result in RESULT. Since VALUE2 is zero, the block generates a runtime error, RESULT remains unchanged and the output is disabled.

11.10.7.12.1.5 MUL

Block that calculates the multiplication of the values of Value1 and Value2, storing the result in Result.

Ladder Representation

Block Structure

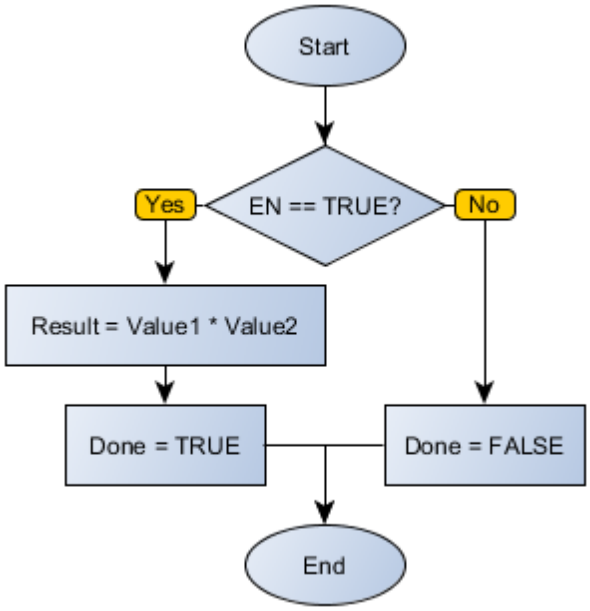
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	First factor of the operation
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Second factor of the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Variable that stores the result of the operation

Operation

When this block has a TRUE value in EN, it sends to the Result output the multiplication of Value1 and Value2 variables. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

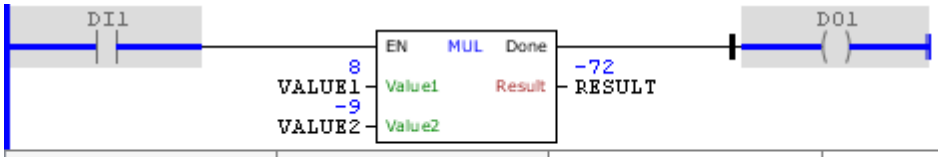
Block Flowchart



Example

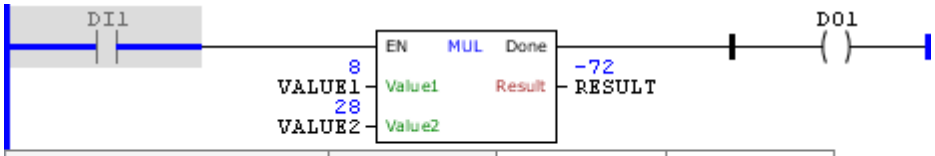
VALUE1	SINT	8	Decimal
VALUE2	SINT	12	Decimal
RESULT	SINT	96	Decimal

The above example calculates the product of VALUE 1 and VALUE2 variables, storing the final result in RESULT.



VALUE1	SINT	8	Decimal
VALUE2	SINT	-9	Decimal
RESULT	SINT	-72	Decimal

The above example calculates the product of VALUE 1 and VALUE2 variables, storing the final result in RESULT. Notice that the block accepts arguments of both signs.



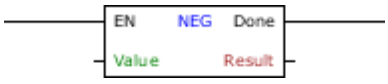
VALUE1	SINT	8	Decimal
VALUE2	SINT	28	Decimal
RESULT	SINT	-72	Decimal

The above example calculates the product of VALUE1 and VALUE2 variables. The final result 224 cannot be stored in RESULT, because it is outside the limits of accepted values by SINT type. Therefore, RESULT remains unchanged and the output is disabled.

11.10.7.12.1.6 NEG

Block that calculates the opposite (i.e., the product with -1) of a value passed by Value, storing the result in Result.

Ladder Representation



Block Structure

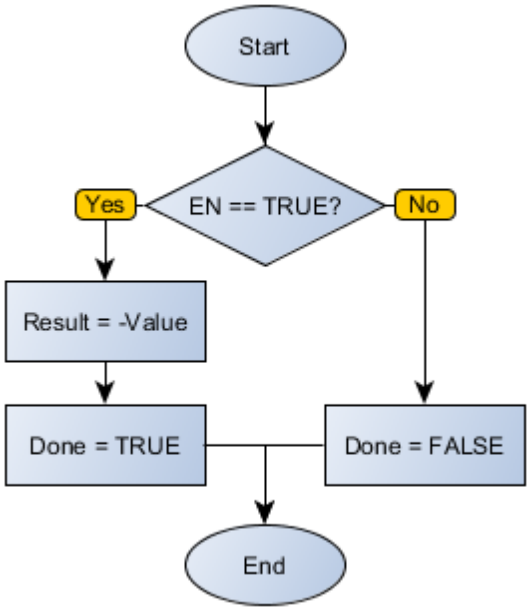
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Reference variable for the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Variable that stores the result of the operation

Operation

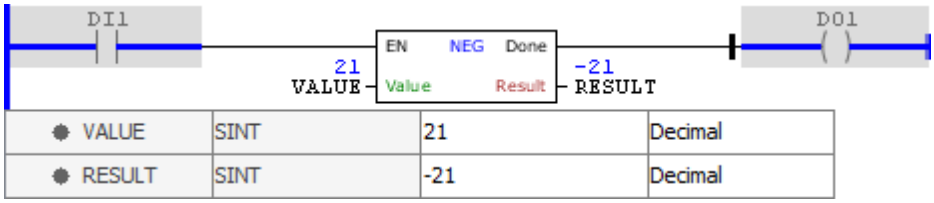
When this block has a TRUE value in EN, it sends to the Result output the opposite of the Value variable. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

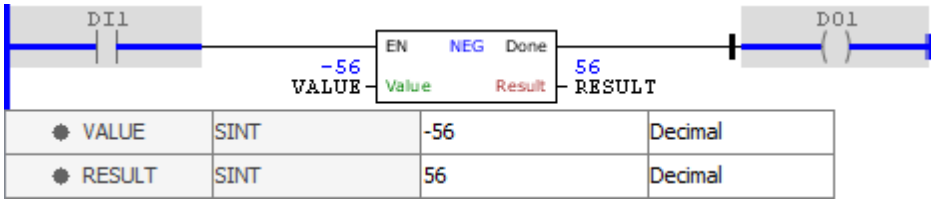


Example



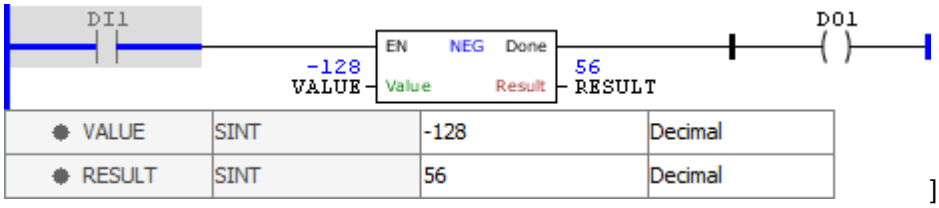
VALUE	SINT	21	Decimal
RESULT	SINT	-21	Decimal

The above example calculates the opposite of the VALUE variable whose initial value is 21, storing the final result, -21, in RESULT.



VALUE	SINT	-56	Decimal
RESULT	SINT	56	Decimal

The above example calculates the opposite of the VALUE variable whose initial value is -56, storing the final result, 56, in RESULT.



VALUE	SINT	-128	Decimal
RESULT	SINT	56	Decimal

]

The above example calculates the opposite of the VALUE variable whose initial value is -128. The final result, 128, cannot be stored in RESULT, because it is outside the limits of accepted values by SINT type. Therefore, RESULT remains unchanged and the output is disabled.

11.10.7.12.1.7 SUB

Block that calculates the subtraction between the Value1 and Value2 values, storing the result in Result.

Ladder Representation

Block Structure

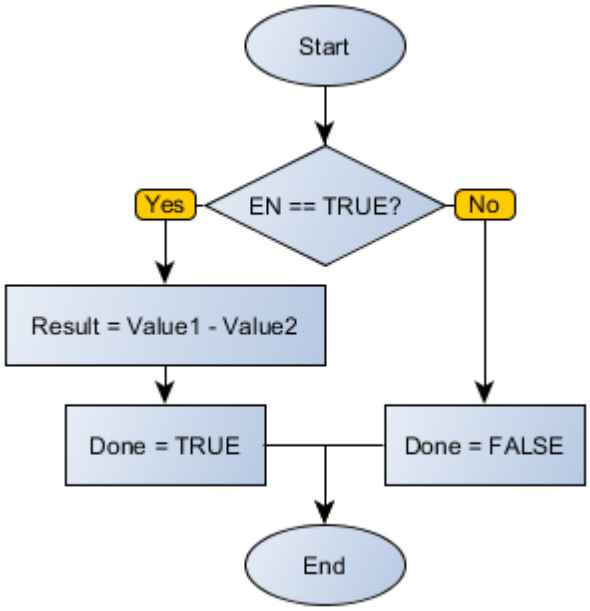
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Minuend of operation
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Subtrahend of operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Variable that stores the result of the operation

Operation

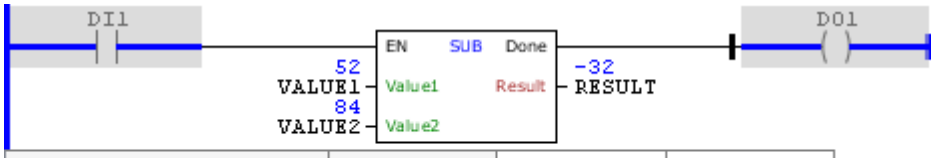
When this block has a TRUE value in EN, it sends to the Result output the subtraction of Value1 and Value2 variables. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

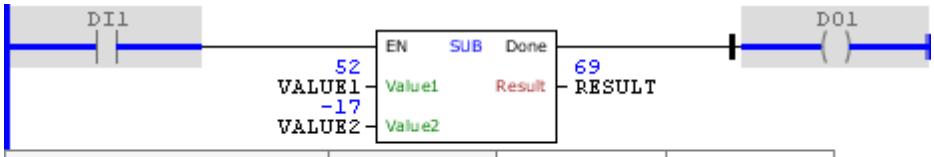


Example



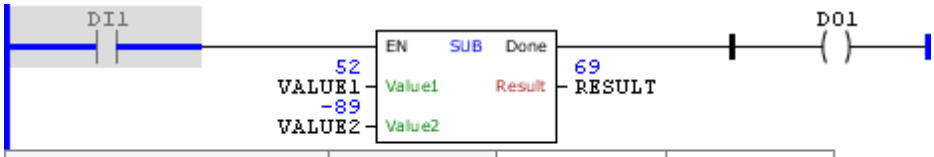
VALUE1	SINT	52	Decimal
VALUE2	SINT	84	Decimal
RESULT	SINT	-32	Decimal

The above example calculates the subtraction of VALUE 1 and VALUE2 variables, storing the final result in RESULT.



VALUE1	SINT	52	Decimal
VALUE2	SINT	-17	Decimal
RESULT	SINT	69	Decimal

The above example calculates the subtraction of VALUE 1 and VALUE2 variables, storing the final result in RESULT. Notice that the block accepts arguments of both signs.



VALUE1	SINT	52	Decimal
VALUE2	SINT	-89	Decimal
RESULT	SINT	69	Decimal

The above example calculates the subtraction of VALUE1 and VALUE2 variables. The final result 141 cannot be stored in RESULT, because it is outside the limits of accepted values by SINT type. Therefore, RESULT remains unchanged and the output is disabled.

11.10.7.12.2 Math Extended

11.10.7.12.2.1 ALOG10

Block that calculates the antilogarithm (exponent with base 10) of the Value value, storing the result in Result.

Ladder Representation

Block Structure

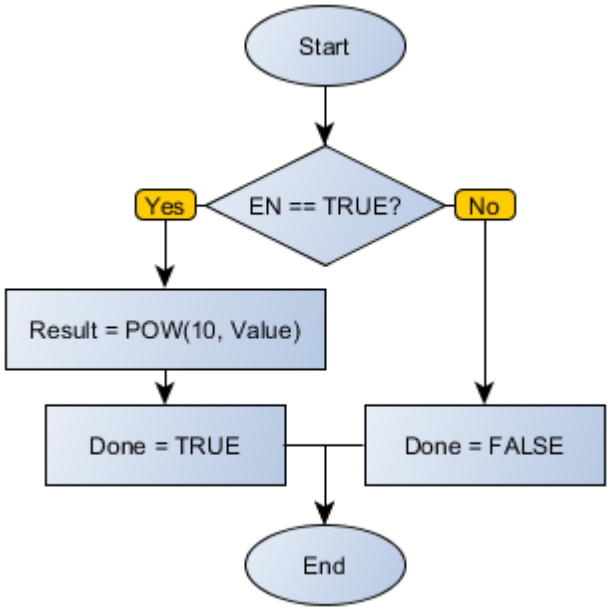
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Reference variable for the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	REAL	Variable that stores the result of the operation

Operation

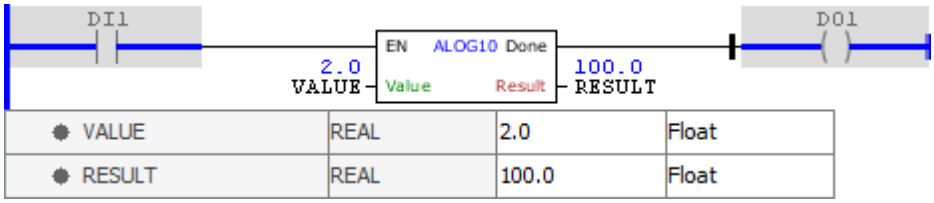
When this block has a TRUE value in EN, it sends to the Result output the antilogarithm of the Value variable. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

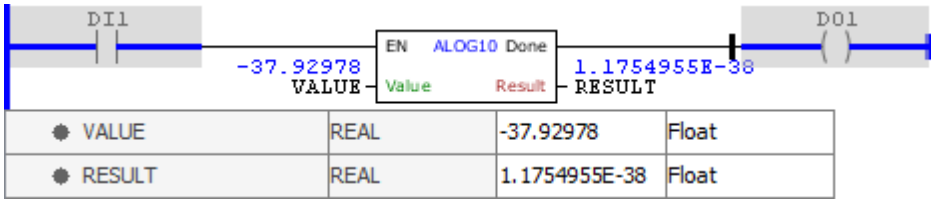


Example



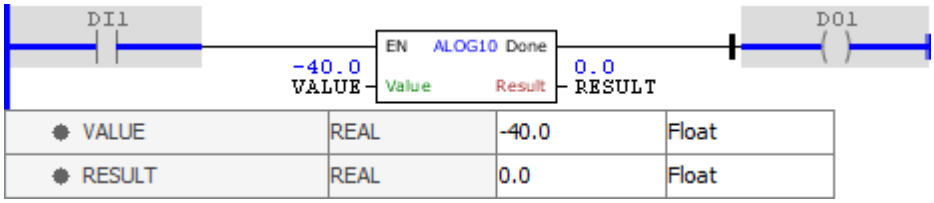
VALUE	REAL	2.0	Float
RESULT	REAL	100.0	Float

The above example calculates the antilogarithm of the VALUE variable, storing the final result in RESULT. The block ends with success and Done output is activated.



VALUE	REAL	-37.92978	Float
RESULT	REAL	1.1754955E-38	Float

The above example calculates the antilogarithm of the VALUE variable, storing the final result in RESULT. The indicated value is the minimum input value for which the block revolutions a nonzero result. The block ends with success and Done output is activated.



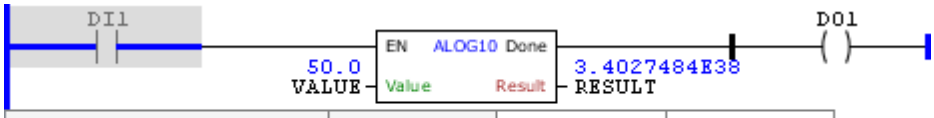
VALUE	REAL	-40.0	Float
RESULT	REAL	0.0	Float

The above example calculates the antilogarithm of the VALUE variable, storing the final result in

RESULT. Below the minimum values cause the block to return a null value. The block ends with success and Done output is activated.

VALUE	REAL	38.53183	Float
RESULT	REAL	3.4027484E38	Float

The above example calculates the antilogarithm of the VALUE variable, storing the final result in RESULT. The indicated value is the maximum input value for which the block revolutions a valid result. The block ends with success and Done output is activated.



VALUE	REAL	50.0	Float
RESULT	REAL	3.4027484E38	Float

The above example calculates the antilogarithm of the VALUE variable, storing the final result in RESULT. Values higher than the maximum cause the block to generate an error, the RESULT output remains unchanged and Done output is disabled.

11.10.7.12.2.2 EXP

Block that calculates the exponential of the Euler number "and" raised to the value of Value, storing the result in Result.

Ladder Representation

Block Structure

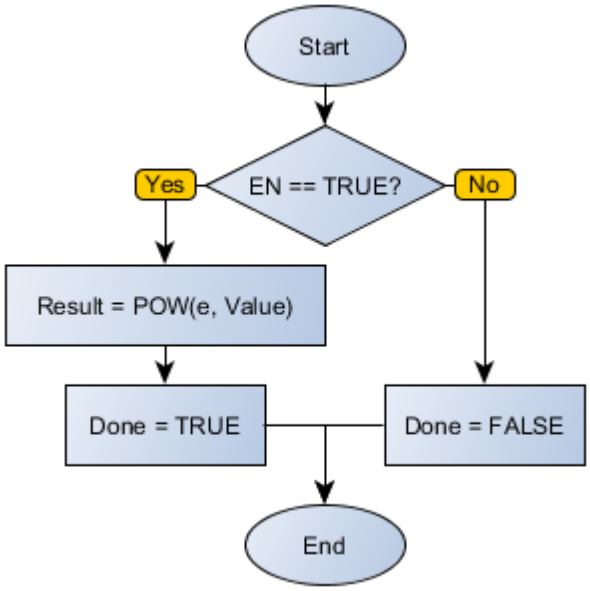
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Reference variable for the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	REAL	Variable that stores the result of the operation

Operation

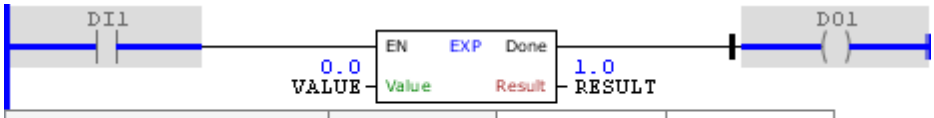
When this block has a TRUE value in EN, it sends to the Result output the exponent of the Euler number "and" raised to the Value variable. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart



Example



VALUE	REAL	0.0	Float
RESULT	REAL	1.0	Float

The above example calculates the exponent of the VALUE variable, storing the final result in RESULT. The block ends with success and Done output is activated.

VALUE	REAL	-87.33654	Float
RESULT	REAL	1.1754997E-38	Float

The above example calculates the exponent of the VALUE variable, storing the final result in RESULT. The indicated value is the minimum input value for which the block revolutions a nonzero result. The block ends with success and Done output is activated.

VALUE	REAL	-50000.0	Float
RESULT	REAL	0.0	Float

The above example calculates the exponent of the VALUE variable, storing the final result in RESULT. Values below the minimum cause the block to return to a null value. The block ends with success and Done output is activated.

VALUE	REAL	88.72283	Float
RESULT	REAL	3.4027985E38	Float

The above example calculates the exponent of the VALUE variable, storing the final result in RESULT. The indicated value is the maximum input value for which the block revolutions a valid result. The block ends with success and Done output is activated.

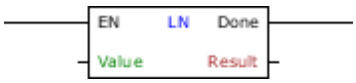
VALUE	REAL	90.0	Float
RESULT	REAL	3.4027985E38	Float

The above example calculates the exponent of the VALUE variable, storing the final result in RESULT. Values higher than the maximum cause the block to generate an error, the RESULT output remains unchanged and Done output is disabled.

11.10.7.12.2.3 LN

Block that calculates the natural logarithm of the Value value, storing the result in Result.

Ladder Representation



Block Structure

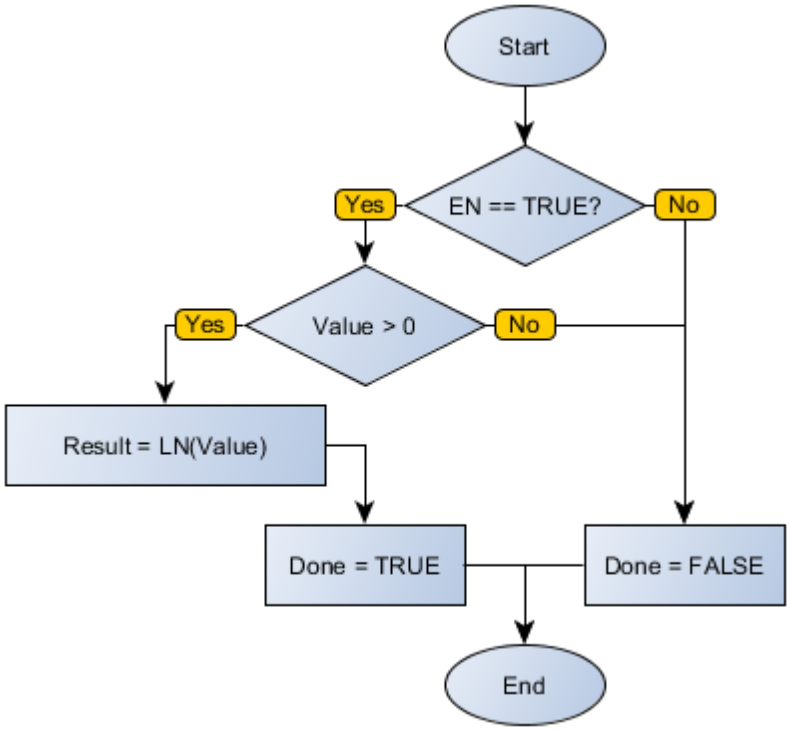
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Reference variable for the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	REAL	Variable that stores the result of the operation

Operation

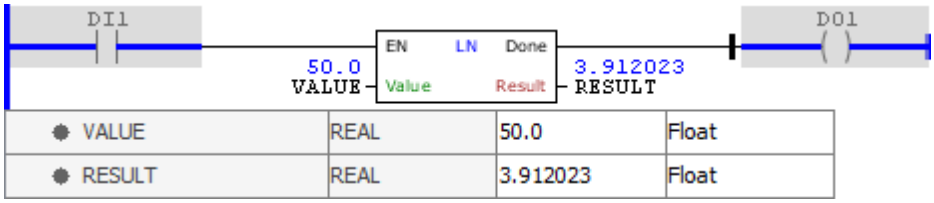
When this block has a TRUE value in EN, it sends to the Result output the natural logarithm of the Value variable. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart



Example



VALUE	REAL	50.0	Float
RESULT	REAL	3.912023	Float

The above example calculates the natural logarithm of the VALUE variable, storing the final result in RESULT. The block ends with success and Done output is activated.

VALUE	REAL	0.0	Float
RESULT	REAL	3.912023	Float

The above example calculates the natural logarithm of the VALUE variable, storing the final result in RESULT. The block generates a runtime error, since VALUE has value zero, and Done output is disabled.

11.10.7.12.2.4 LOG10

Block that calculates the common logarithm (base 10) of the Value value, storing the result in Result.

Ladder Representation

Block Structure

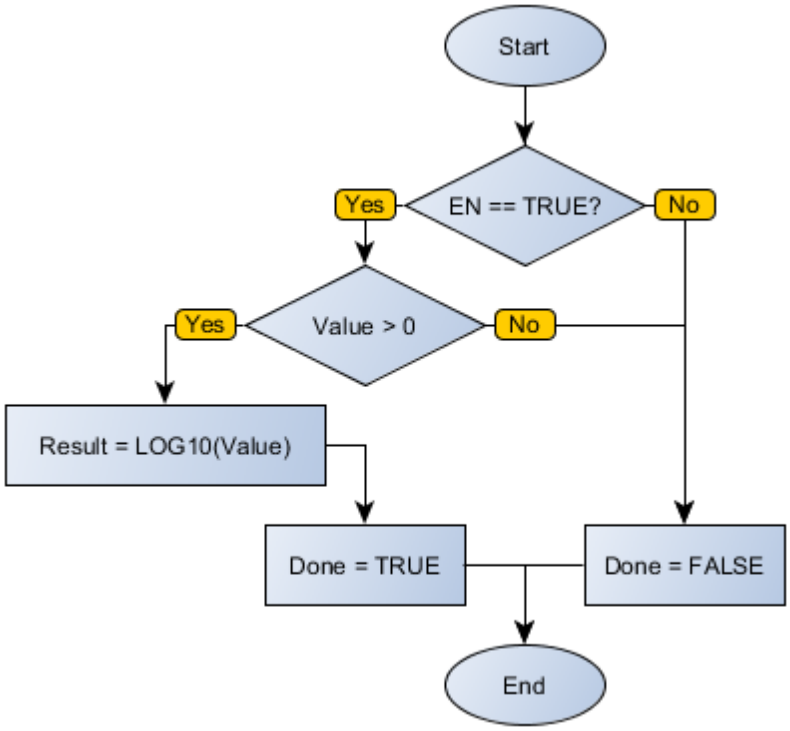
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Reference variable for the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	REAL	Variable that stores the result of the operation

Operation

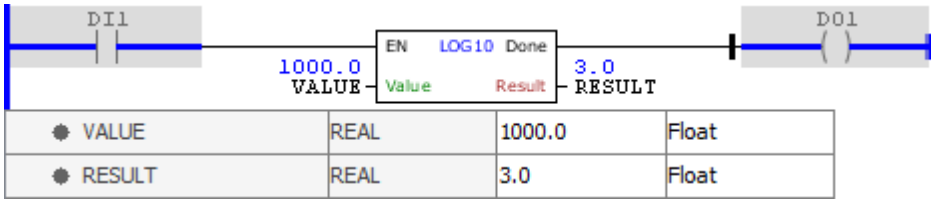
When this block has a TRUE value in EN, it sends to the Result output the common logarithm of the Value variable. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

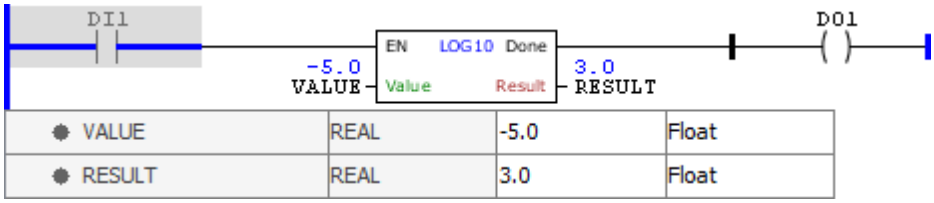


Example



VALUE	REAL	1000.0	Float
RESULT	REAL	3.0	Float

The above example calculates the common logarithm of the VALUE variable, storing the final result in RESULT. The block ends with success and Done output is activated.



VALUE	REAL	-5.0	Float
RESULT	REAL	3.0	Float

The above example calculates the common logarithm of the VALUE variable, storing the final result in RESULT. The block generates a runtime error, since VALUE has negative value, and Done output is disabled.

11.10.7.12.2.5 POW

Block that calculates the value of Value raised to the exponent Power, storing the result in Result.

Ladder Representation

Block Structure

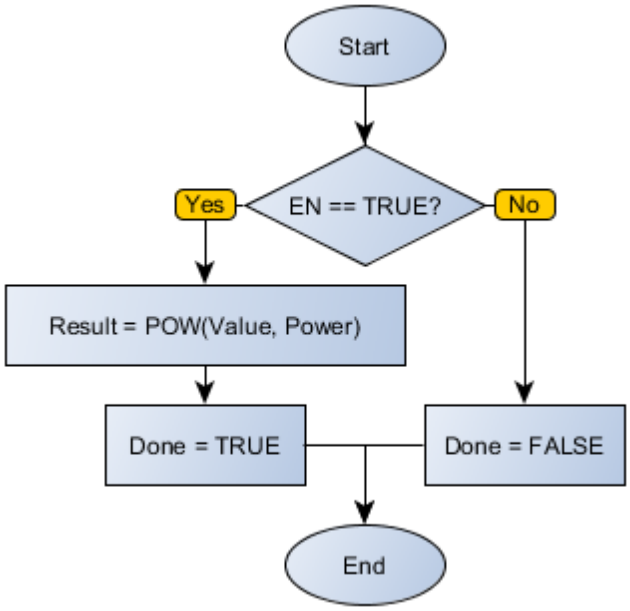
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Base of the operation
	Power	REAL	Exponent of the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	REAL	Variable that stores the result of the operation

Operation

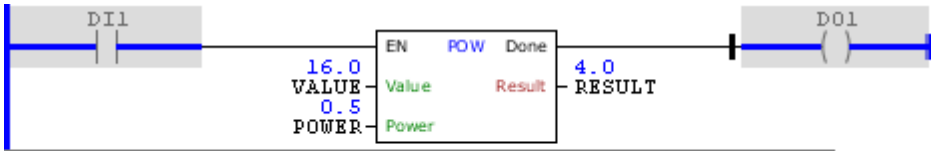
When this block has a TRUE value in EN, it sends to the Result output the value of Value raised to the exponent Power. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

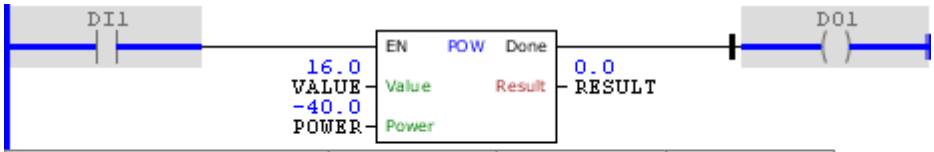


Example



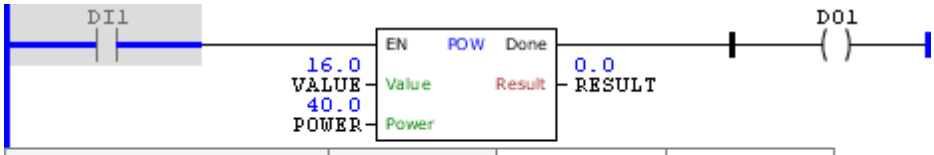
VALUE	REAL	16.0	Float
POWER	REAL	0.5	Float
RESULT	REAL	4.0	Float

The above example calculates the value of VALUE raised to the POWER variable, storing the final result in RESULT. The block ends with success and Done output is activated.



VALUE	REAL	16.0	Float
POWER	REAL	-40.0	Float
RESULT	REAL	0.0	Float

The above example calculates the value of VALUE raised to the POWER variable, storing the final result in RESULT. The block ends with success and Done output is activated.



VALUE	REAL	16.0	Float
POWER	REAL	40.0	Float
RESULT	REAL	0.0	Float

The above example calculates the value of VALUE raised to the POWER variable, storing the final result in RESULT. Since the result is higher than the maximum supported by REAL type, the block generates an error and Done output is disabled.

11.10.7.12.2.6 ROUND

Block that rounds the value of Value, storing the result in Result.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Reference variable for the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	REAL	Variable that stores the result of the operation

Operation

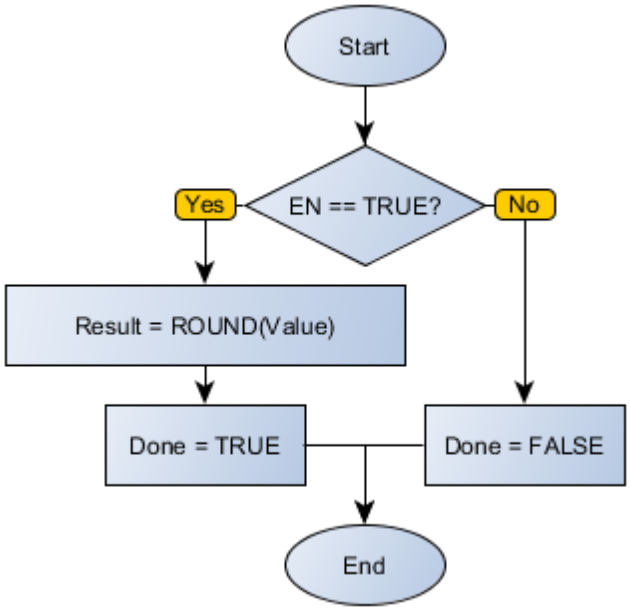
When this block has a TRUE value in EN, it sends to the Result output the rounded value of Value. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

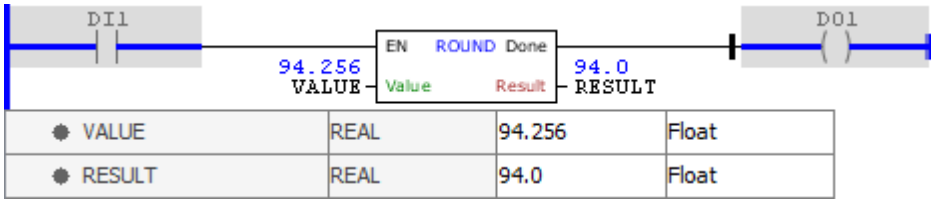
Compatibility

Device	Version
PLC300	2.10 or higher
SCA06	2.00 or higher

Block Flowchart

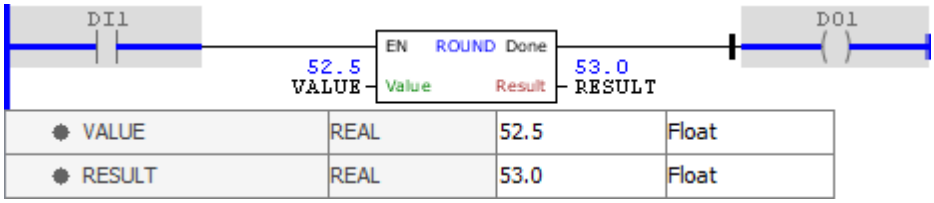


Example



VALUE	REAL	94.256	Float
RESULT	REAL	94.0	Float

The above example rounds the value of the VALUE variable, storing the final result in RESULT. Decimals less than 0.5 are discarded. The block ends with success and Done output is activated.



VALUE	REAL	52.5	Float
RESULT	REAL	53.0	Float

The above example rounds the value of the VALUE variable, storing the final result in RESULT. Decimals greater than or equal to 0.5 promote unity value immediately above. The block ends with success and Done output is activated.

11.10.7.12.2.7 SQRT

Block that calculates the square root value of Value, storing the result in Result.

Ladder Representation

Block Structure

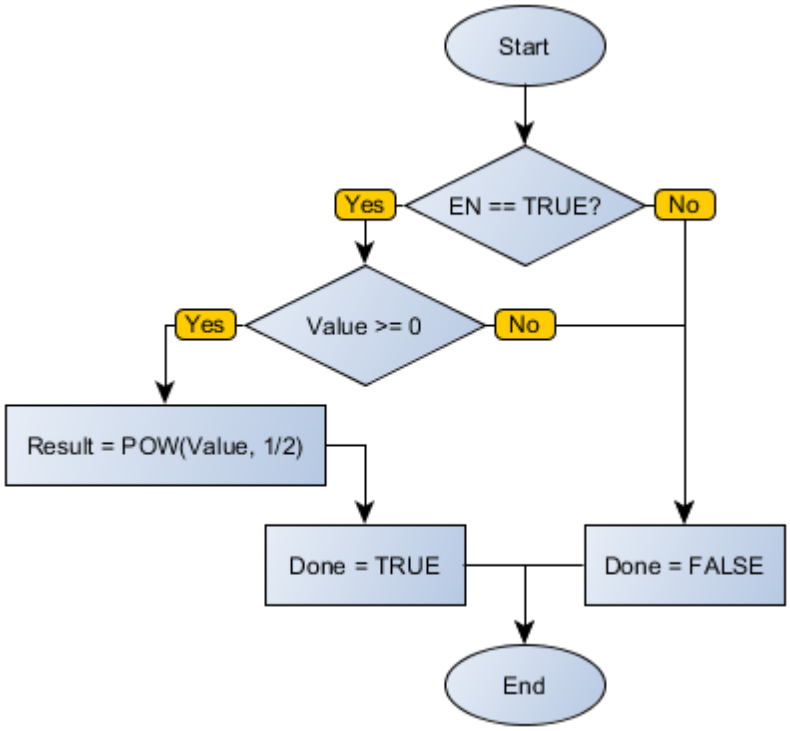
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Reference variable for the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	REAL	Variable that stores the result of the operation

Operation

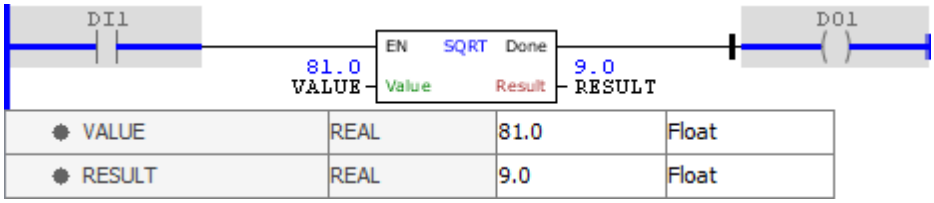
When this block has a TRUE value in EN, it sends to the Result output the square root value of Value. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

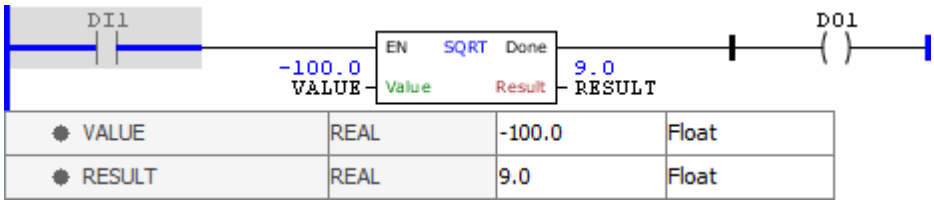


Example



VALUE	REAL	81.0	Float
RESULT	REAL	9.0	Float

The above example calculates the square root value of the VALUE variable, storing the final result in RESULT. The block ends with success and Done output is activated.



VALUE	REAL	-100.0	Float
RESULT	REAL	9.0	Float

The above example calculates the square root value of the VALUE variable, storing the final result in RESULT. The block generates a runtime error, since VALUE has negative value, and Done output is disabled.

11.10.7.12.2.8 TRUNC

Block that truncates the value of Value, storing the result in Result.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Reference variable for the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	REAL	Variable that stores the result of the operation

Operation

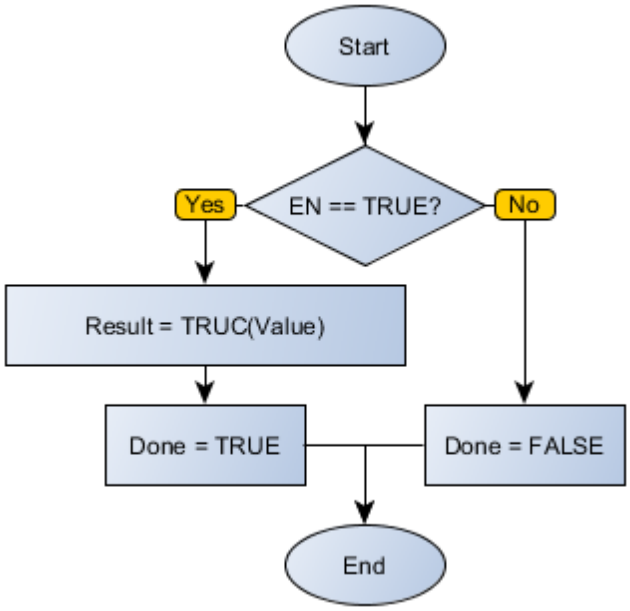
When this block has a TRUE value in EN, it sends to the Result output the truncated value of Value. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

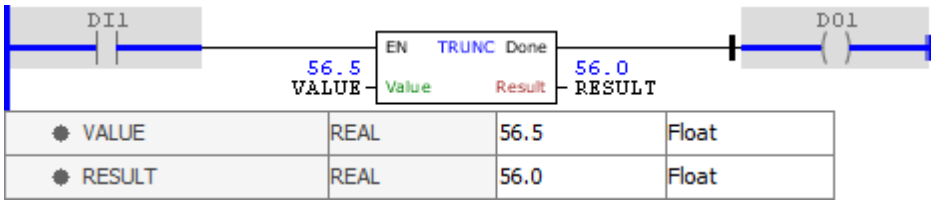
Compatibility

Device	Version
PLC300	2.10 or higher
SCA06	2.00 or higher

Block Flowchart



Example



VALUE	REAL	56.5	Float
RESULT	REAL	56.0	Float

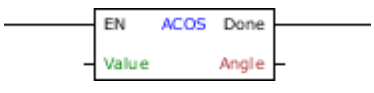
The above example truncates the value of the VALUE variable, storing the final result in RESULT. Decimals are discarded. The block ends with success and Done output is activated.

11.10.7.12.3 Math Trigonometry

11.10.7.12.3.1 ACOS

Block that calculates the arccosine of Value, storing the result in Angle.

Ladder Representation



Block Structure

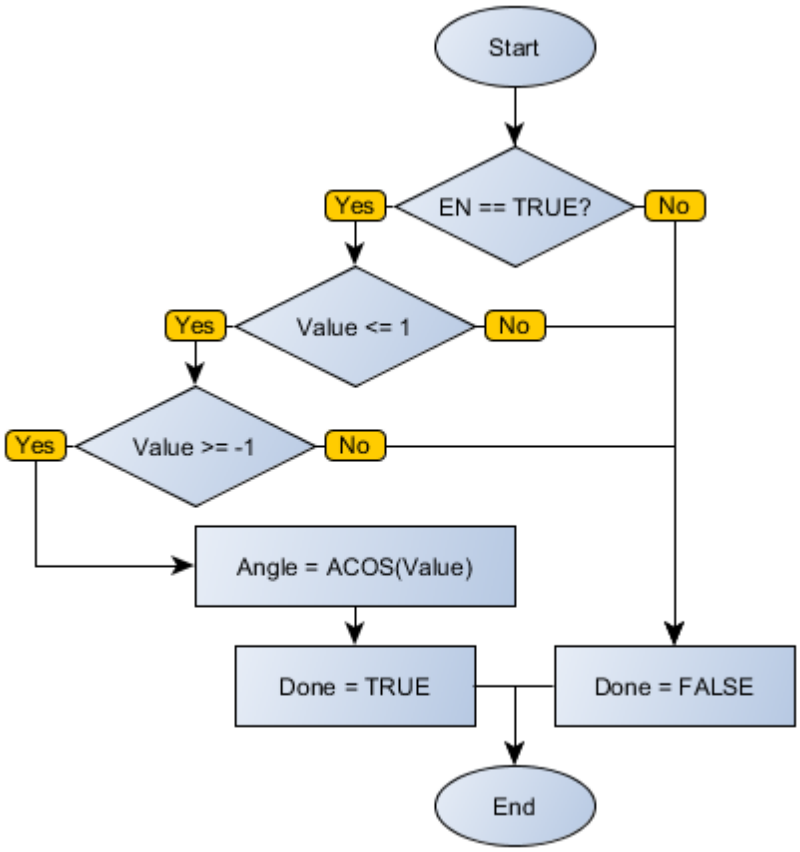
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Value of cosine
VAR_OUTPUT	Done	BOOL	End of operation
	Angle	REAL	Value of the angle w hose cosine is equal to Value (in radians)

Operation

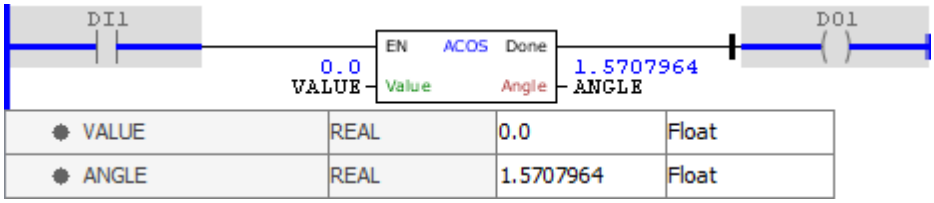
When this block has a TRUE value in EN, it sends to the Angle output the arccosine of Value. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Angle remains with its value unchanged.

When EN has FALSE value, Angle remains unchanged and Done remains in FALSE.

Block Flowchart

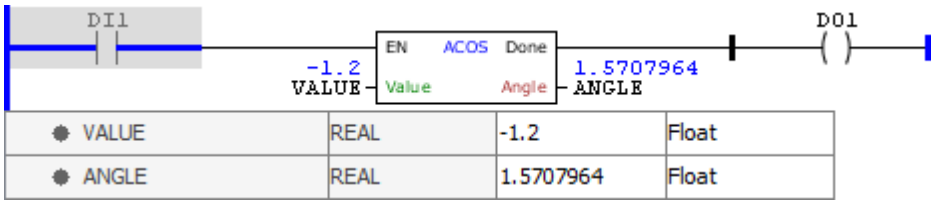


Example



VALUE	REAL	0.0	Float
ANGLE	REAL	1.5707964	Float

The above example calculates the arc, in radians, whose cosine is the VALUE variable, storing the final result in RESULT. The block ends with success and Done output is activated.



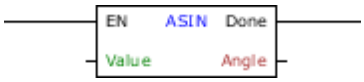
VALUE	REAL	-1.2	Float
ANGLE	REAL	1.5707964	Float

The above example calculates the arc, in radians, whose cosine is the VALUE variable, storing the final result in RESULT. The block generates a runtime error, since VALUE has value inferior to 1, and Done output is disabled.

11.10.7.12.3.2 ASIN

Block that calculates the arcsine of Value, storing the result in Angle.

Ladder Representation



Block Structure

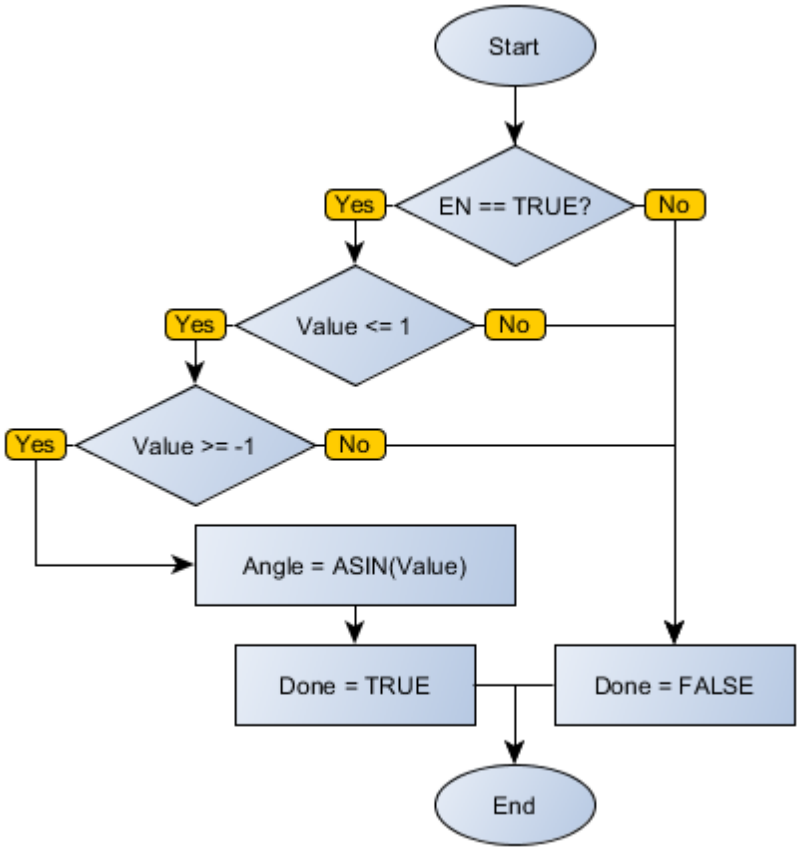
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Value of sine
VAR_OUTPUT	Done	BOOL	End of operation
	Angle	REAL	Value of the angle whose sine is equal to Value (in radians)

Operation

When this block has a TRUE value in EN, it sends to the Angle output the arcsine of Value. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Angle remains with its value unchanged.

When EN has FALSE value, Angle remains unchanged and Done remains in FALSE.

Block Flowchart



Example

VALUE	REAL	-1.0	Float
ANGLE	REAL	-1.5707964	Float

The above example calculates the arc, in radians, whose sine is the VALUE variable, storing the final result in RESULT. The block ends with success and Done output is activated.

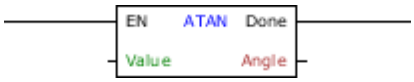
VALUE	REAL	2.0	Float
ANGLE	REAL	-1.5707964	Float

The above example calculates the arc, in radians, whose sine is the VALUE variable, storing the final result in RESULT. The block generates a runtime error, since VALUE has value superior to 1, and Done output is disabled.

11.10.7.12.3.3 ATAN

Block that calculates the arctangent of Value, storing the result in Angle.

Ladder Representation



Block Structure

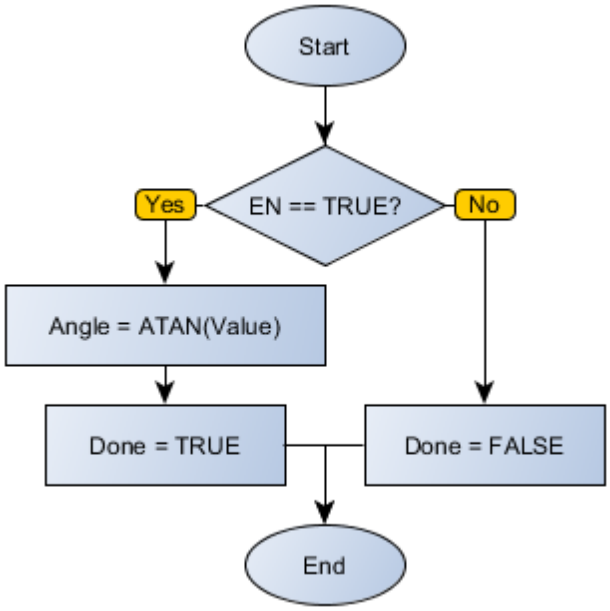
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Value of tangent
VAR_OUTPUT	Done	BOOL	End of operation
	Angle	REAL	Value of the angle whose tangent is equal to Value (in radians)

Operation

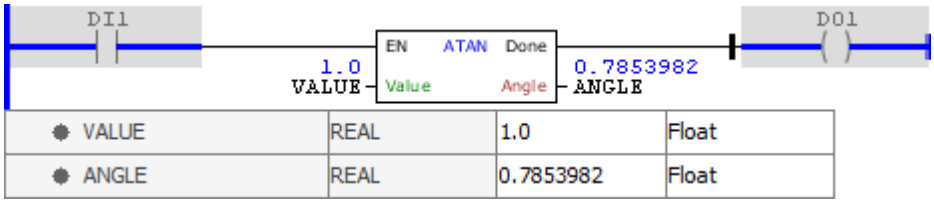
When this block has a TRUE value in EN, it sends to the Angle output the arctangent of Value. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Angle remains with its value unchanged.

When EN has FALSE value, Angle remains unchanged and Done remains in FALSE.

Block Flowchart

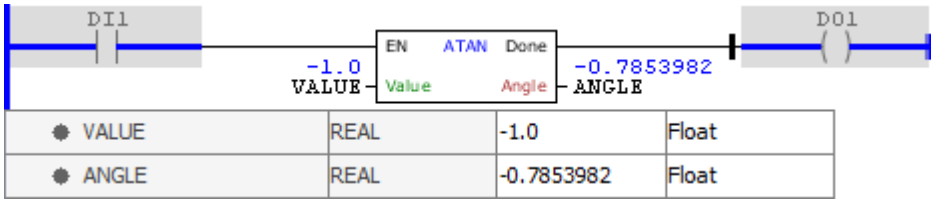


Example



VALUE	REAL	1.0	Float
ANGLE	REAL	0.7853982	Float

The above example calculates the arc, in radians, whose tangent is the VALUE variable, storing the final result in RESULT. The arc, for positive values, is always in the first quadrant. The block ends with success and Done output is activated.



VALUE	REAL	-1.0	Float
ANGLE	REAL	-0.7853982	Float

The above example calculates the arc, in radians, whose tangent is the VALUE variable, storing the final result in RESULT. The arc, for negative values, is always in the fourth quadrant. The block ends with success and Done output is activated.

11.10.7.12.3.4 ATAN2

Block that calculates the arctangent of Y/X, storing the result in Angle.

Ladder Representation

Block Structure

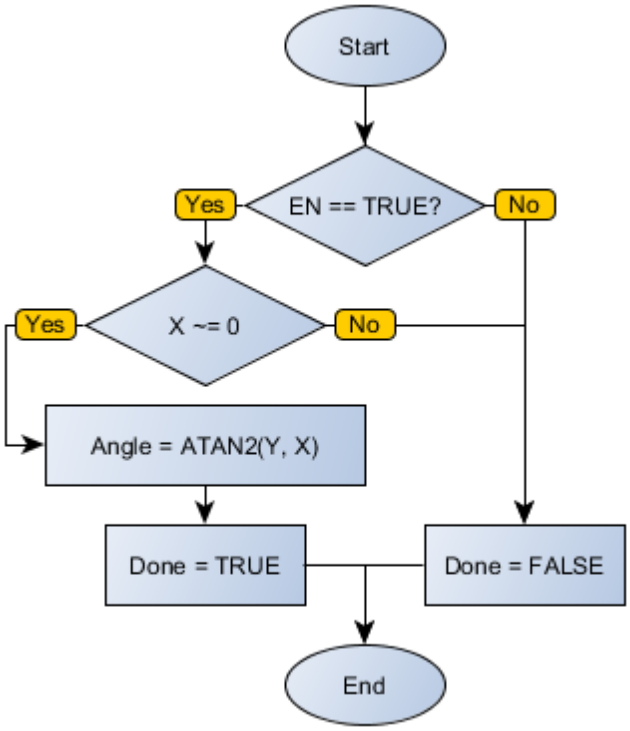
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	X	REAL	Parameter X of the function
	Y	REAL	Parameter Y of the function
VAR_OUTPUT	Done	BOOL	End of operation
	Angle	REAL	Value of the angle whose tangent is equal to (Y/X) (in radians)

Operation

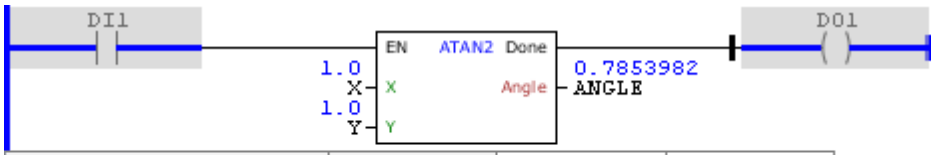
When this block has a TRUE value in EN, it sends to the Angle output the arctangent of Y/X. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Angle remains with its value unchanged.

When EN has FALSE value, Angle remains unchanged and Done remains in FALSE.

Block Flowchart



Example



X	REAL	1.0	Float
Y	REAL	1.0	Float
ANGLE	REAL	0.7853982	Float

The above example calculates the arc, in radians, whose tangent is the Y/X variable, storing the final result in RESULT. The arc, for positive values of X and Y, is always in the first quadrant. The block ends with success and Done output is activated.

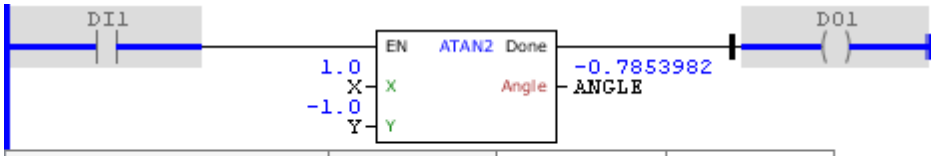
X	REAL	-1.0	Float
Y	REAL	1.0	Float
ANGLE	REAL	2.3561945	Float

The above example calculates the arc, in radians, whose tangent is the Y/X variable, storing the final

result in RESULT. The arc, for negative values of X and positive values of Y, is always in the second quadrant. The block ends with success and Done output is activated.

X	REAL	-1.0	Float
Y	REAL	-1.0	Float
ANGLE	REAL	-2.3561945	Float

The above example calculates the arc, in radians, whose tangent is the Y/X variable, storing the final result in RESULT. The arc, for negative values of X and Y, is always in the third quadrant. The block ends with success and Done output is activated.



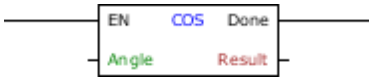
X	REAL	1.0	Float
Y	REAL	-1.0	Float
ANGLE	REAL	-0.7853982	Float

The above example calculates the arc, in radians, whose tangent is the Y/X variable, storing the final result in RESULT. The arc, for positive values of X and negative values of Y, is always in the fourth quadrant. The block ends with success and Done output is activated.

11.10.7.12.3.5 COS

Block that calculates the cosine of Angle, storing the result in Result.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Angle	REAL	Angle (in radians)
VAR_OUTPUT	Done	BOOL	End of operation
	Result	REAL	Variable that stores the result of the operation

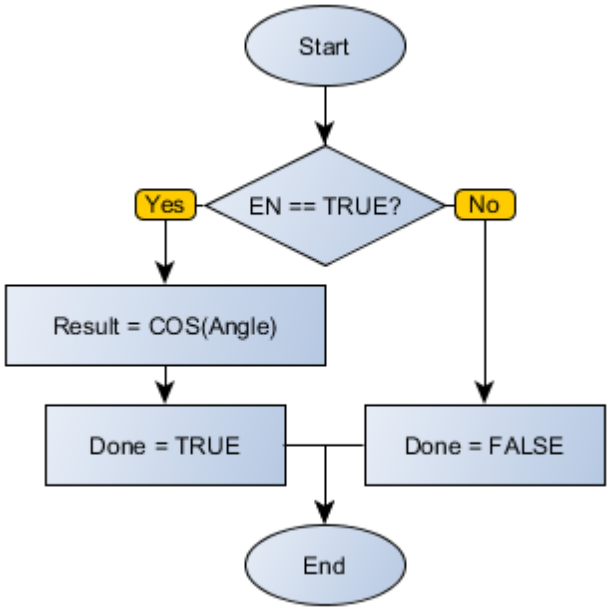
Operation

When this block has a TRUE value in EN, it sends to the Result output the cosine of Angle. If no

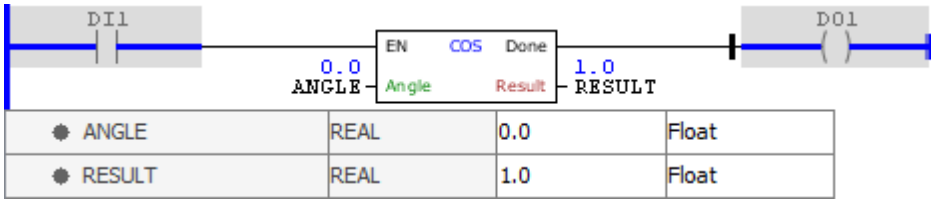
errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

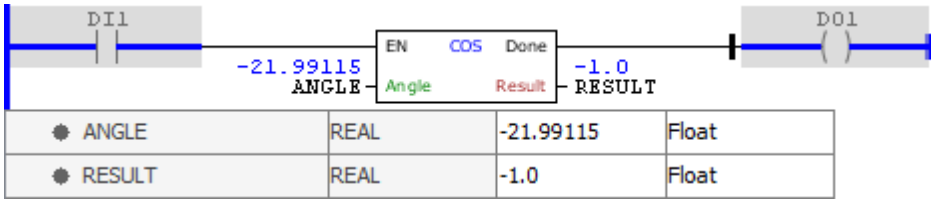


Example



ANGLE	REAL	0.0	Float
RESULT	REAL	1.0	Float

The above example calculates the cosine of the VALUE variable, interpreted in radians, storing the final result in RESULT. The block ends with success and Done output is activated.



ANGLE	REAL	-21.99115	Float
RESULT	REAL	-1.0	Float

The above example calculates the cosine of the VALUE variable, interpreted in radians, storing the final result in RESULT. The block ends with success and Done output is activated. Notice that the block accepts negative input values and greater than one turn.

11.10.7.12.3.6 SIN

Block that calculates the sine of Angle, storing the result in Result.

Ladder Representation

Block Structure

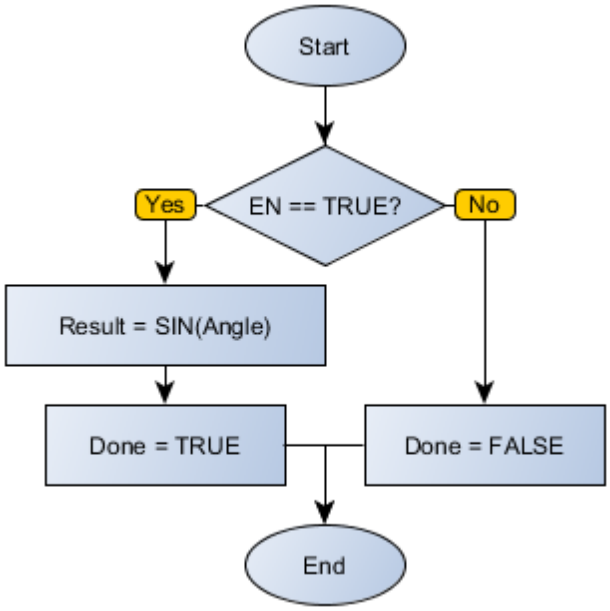
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Angle	REAL	Angle (in radians)
VAR_OUTPUT	Done	BOOL	End of operation
	Result	REAL	Variable that stores the result of the operation

Operation

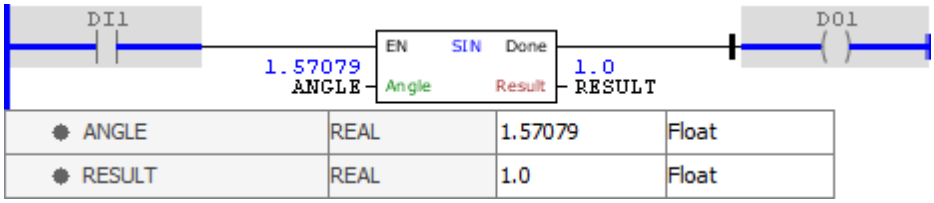
When this block has a TRUE value in EN, it sends to the Result output the sine of Angle. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

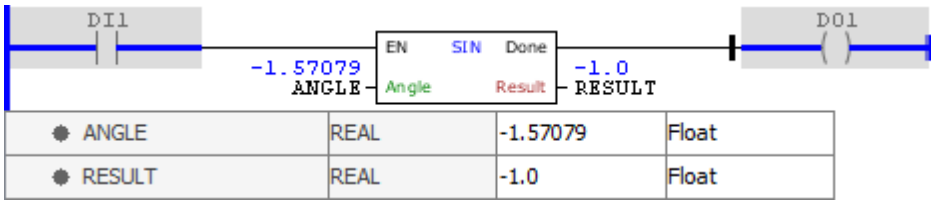


Example



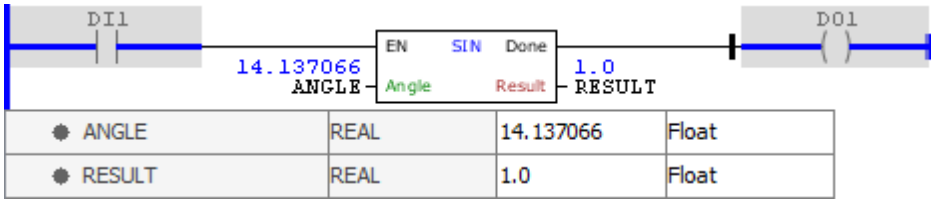
ANGLE	REAL	1.57079	Float
RESULT	REAL	1.0	Float

The above example calculates the sine of the VALUE variable, interpreted in radians, storing the final result in RESULT. The block ends with success and Done output is activated.



ANGLE	REAL	-1.57079	Float
RESULT	REAL	-1.0	Float

The above example calculates the sine of the VALUE variable, interpreted in radians, storing the final result in RESULT. The block ends with success and Done output is activated. Notice that the block accepts negative input values.



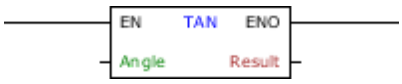
ANGLE	REAL	14.137066	Float
RESULT	REAL	1.0	Float

The above example calculates the sine of the VALUE variable, interpreted in radians, storing the final result in RESULT. The block ends with success and Done output is activated. Notice that the block accepts values greater than one full turn.

11.10.7.12.3.7 TAN

Block that calculates the tangent of Angle, storing the result in Result.

Ladder Representation



Block Structure

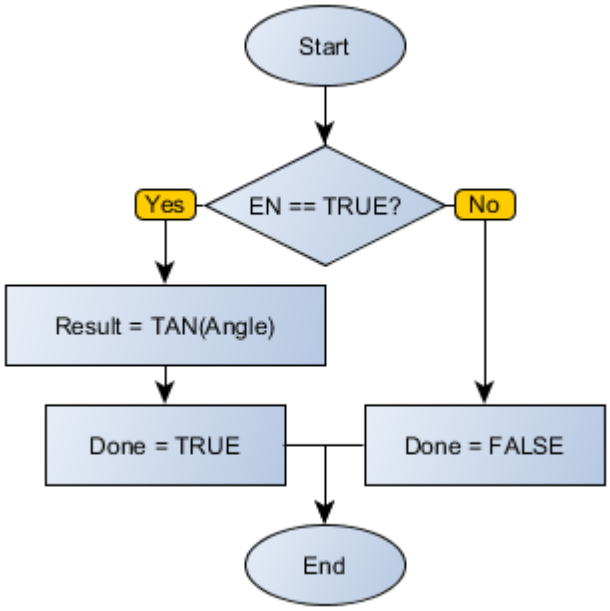
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Angle	REAL	Angle (in radians)
VAR_OUTPUT	Done	BOOL	End of operation
	Result	REAL	Variable that stores the result of the operation

Operation

When this block has a TRUE value in EN, it sends to the Result output the tangent of Angle. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart



Example

ANGLE	REAL	1.0	Float
RESULT	REAL	1.5574077	Float

The above example calculates the tangent of the VALUE variable, interpreted in radians, storing the final result in RESULT. The block ends with success and Done output is activated.

ANGLE	REAL	-30.0	Float
RESULT	REAL	6.405331	Float

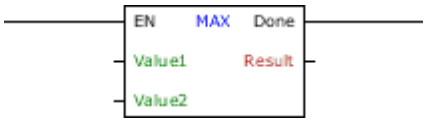
The above example calculates the tangent of the VALUE variable, interpreted in radians, storing the final result in RESULT. The block ends with success and Done output is activated. Notice that the block accepts negative input values and greater than one turn.

11.10.7.12.4 Math Util

11.10.7.12.4.1 MAX

Block that compares the values of Value1 and Value2 and stores the highest of them in Result.

Ladder Representation



Block Structure

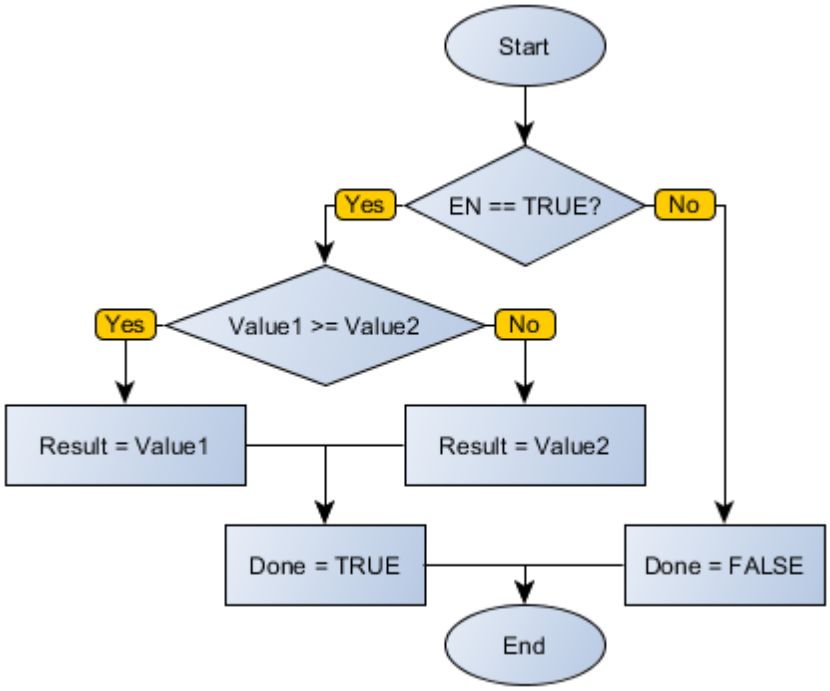
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	First value of comparison
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Second value of comparison
VAR_OUTPUT	Done	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Highest of the values compared

Operation

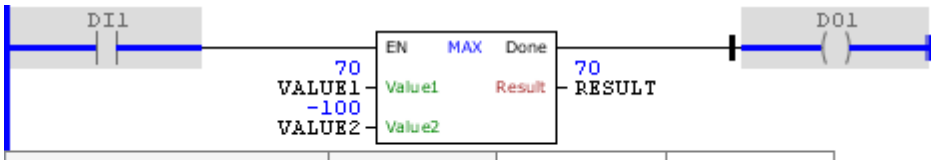
When this block has a TRUE value in EN, it sends to the Result output the highest value in the comparison between Value1 and Value2. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

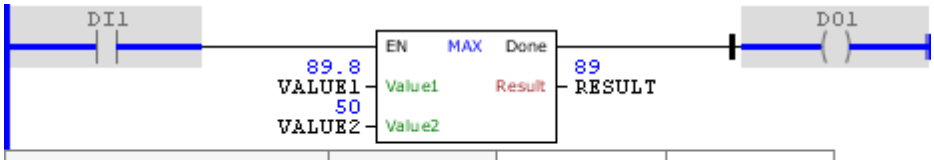


Example



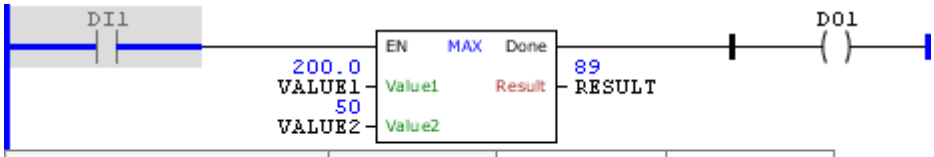
VALUE1	SINT	70	Decimal
VALUE2	SINT	-100	Decimal
RESULT	SINT	70	Decimal

The above example calculates the maximum value between VALUE 1 and VALUE2 variables, storing the final result in RESULT. The block ends with success and Done output is activated.



VALUE1	REAL	89.8	Float
VALUE2	SINT	50	Decimal
RESULT	SINT	89	Decimal

The above example calculates the maximum value between VALUE 1 and VALUE2 variables, storing the final result in RESULT. The block ends with success and Done output is activated. Notice that the types of the input variables can be different without causing execution problems.



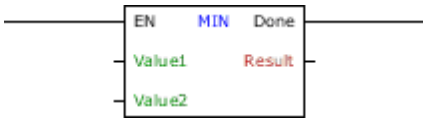
VALUE1	REAL	200.0	Float
VALUE2	SINT	50	Decimal
RESULT	SINT	89	Decimal

The above example calculates the maximum value between VALUE 1 and VALUE2 variables, storing the final result in RESULT. Since the result is higher than the maximum supported by SINT type, the block generates an error and Done output is disabled.

11.10.7.12.4.2 MIN

Block that compares the values of Value1 and Value2 and stores the lowest of them in Result.

Ladder Representation



Block Structure

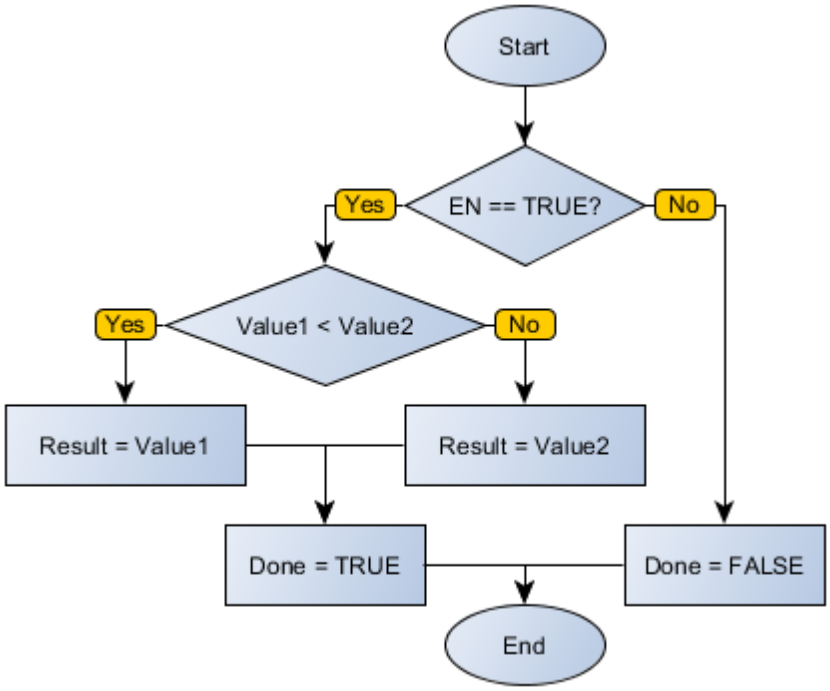
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	First value of comparison
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Second value of comparison
VAR_OUTPUT	Done	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Low est of the values compared

Operation

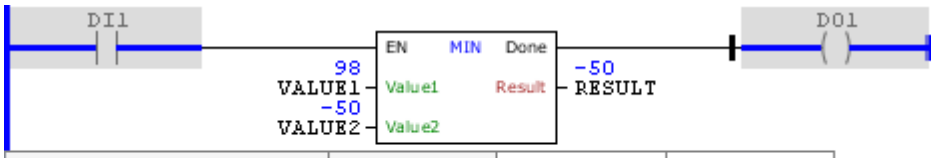
When this block has a TRUE value in EN, it sends to the Result output the lowest value in the comparison between Value1 and Value2. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

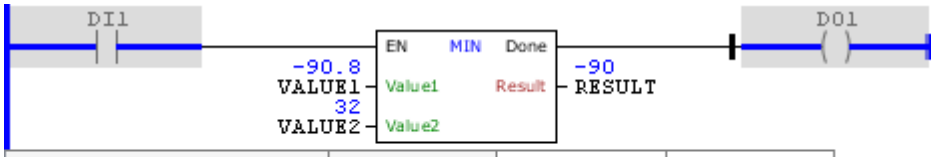


Example



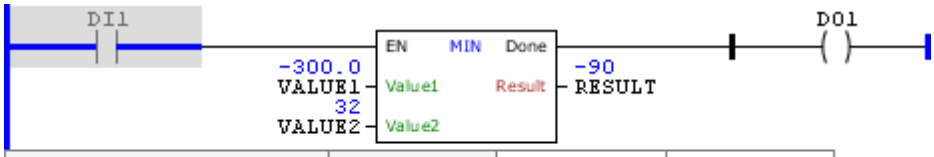
VALUE1	SINT	98	Decimal
VALUE2	SINT	-50	Decimal
RESULT	SINT	-50	Decimal

The above example calculates the minimum value between VALUE 1 and VALUE2 variables, storing the final result in RESULT. The block ends with success and Done output is activated.



VALUE1	REAL	-90.8	Float
VALUE2	SINT	32	Decimal
RESULT	SINT	-90	Decimal

The above example calculates the minimum value between VALUE 1 and VALUE2 variables, storing the final result in RESULT. The block ends with success and Done output is activated. Notice that the types of the input variables can be different without causing execution problems.



VALUE1	REAL	-300.0	Float
VALUE2	SINT	32	Decimal
RESULT	SINT	-90	Decimal

The above example calculates the minimum value between VALUE 1 and VALUE2 variables, storing the final result in RESULT. Since the result is lower than the minimum supported by SINT type, the block generates an error and Done output is disabled.

11.10.7.12.4.3 SAT

Block that performs a routine for saturation of the value found in Value in accordance with the limits for Minimum and Maximum, storing the result in Result.

Ladder Representation

Block Structure

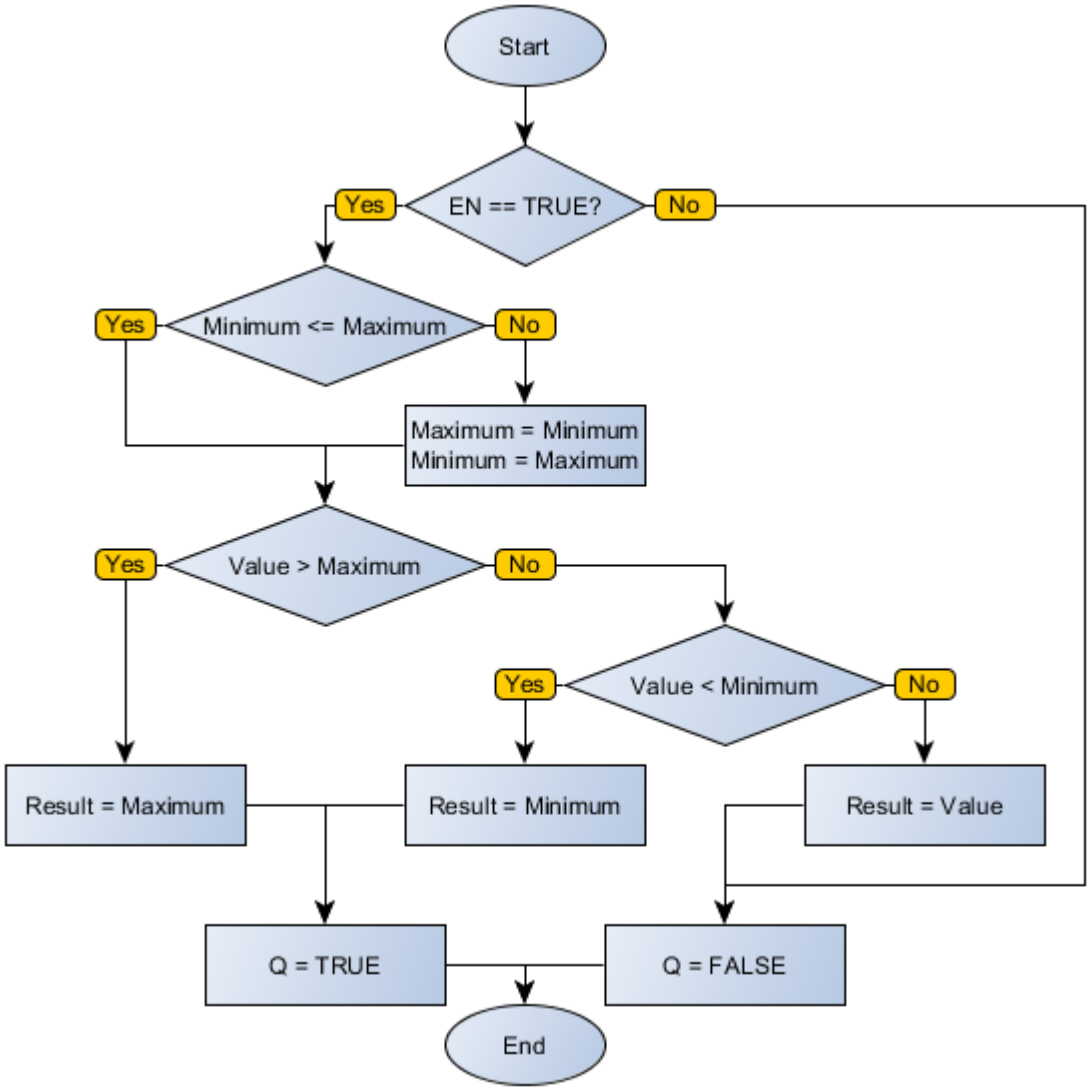
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Reference value
	Minimum	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Inferior saturation value
	Maximum	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Superior saturation value
VAR_OUTPUT	Q	BOOL	Indicator that there w as saturation in the process
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Result of operation

Operation

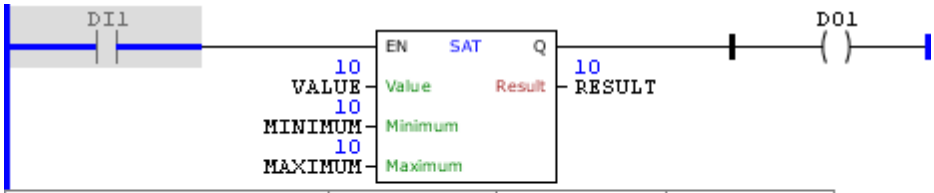
When this block has a TRUE value in EN, it performs a comparison between Value and Minimum and Maximum. If Value is in the range between Minimum and Maximum, Result receives the value of Value and Q remains FALSE. If Value is higher than Maximum, Result receives Maximum and Q receives TRUE. If Value is lower than Minimum, Result receives Minimum and Q receives TRUE. If there is any error in the operation, Q is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Q remains in FALSE.

Block Flowchart

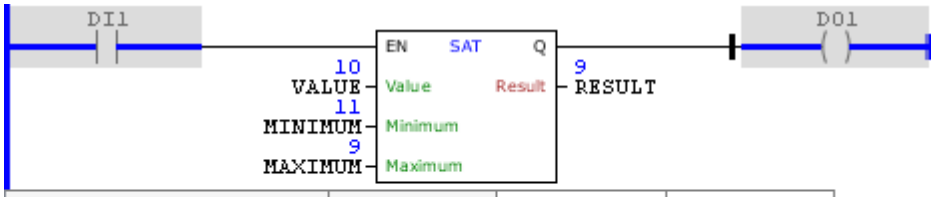


Example



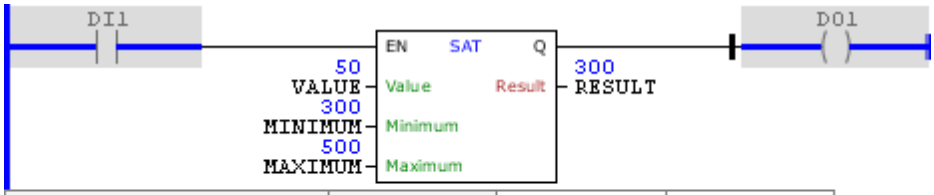
VALUE	INT	10	Decimal
MINIMUM	INT	10	Decimal
MAXIMUM	INT	10	Decimal
RESULT	INT	10	Decimal

The above example passes the VALUE value to RESULT, since it is not lower than MINIMUM or higher than MAXIMUM. The block ends successfully and the Q output is disabled, since there was no saturation.



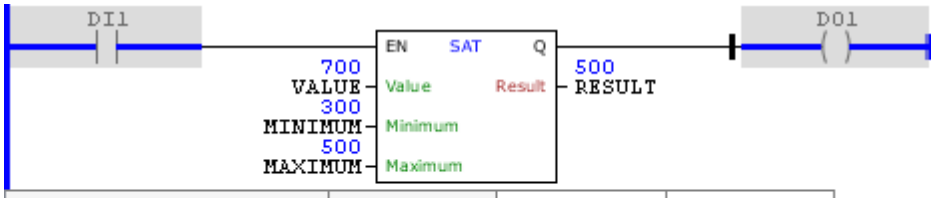
VALUE	INT	10	Decimal
MINIMUM	INT	11	Decimal
MAXIMUM	INT	9	Decimal
RESULT	INT	9	Decimal

The above example passes the MAXIMUM to RESULT, since VALUE is higher than MAXIMUM. The block ends successfully and the Q output is activated, since there was saturation.



VALUE	INT	50	Decimal
MINIMUM	INT	300	Decimal
MAXIMUM	INT	500	Decimal
RESULT	INT	300	Decimal

The above example passes the MINIMUM to RESULT, since VALUE is lower than MINIMUM. The block ends successfully and the Q output is activated, since there was saturation.



VALUE	INT	700	Decimal
MINIMUM	INT	300	Decimal
MAXIMUM	INT	500	Decimal
RESULT	INT	500	Decimal

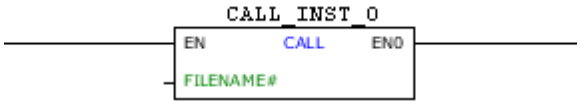
The above example passes the MAXIMUM value to RESULT, since VALUE is higher than MAXIMUM. The block ends successfully and the Q output is activated, since there was saturation.

11.10.7.13 Module

11.10.7.13.1 CALL

Block that loads a file and do a ladder call.

Ladder Representation



Block Structure

Variable Type	Name	Data type	Description
VAR_INPUT	EN	BOOL	Block enabling
	FILENAME#	STRING	Ladder file name (POU) enclosed in gingle quotation marks
VAR_OUTPUT	ENO	BOOL	End of operation
VAR	CALL_INST_0	CALL	Instance of access to block structure

Operation

When this block has a TRUE value in EN, it updates the values of internal fields with the input variables, performs the Ladder routine loading the file and updates the values of the outputs after completing routine.

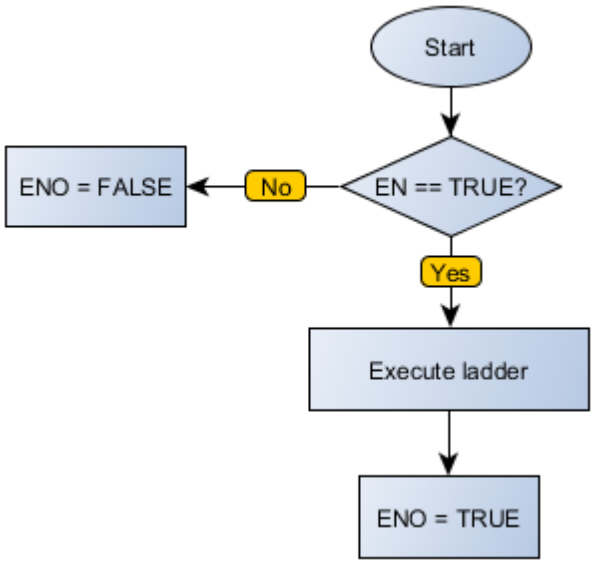
When EN has FALSE value, outputs remain unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Compatibility

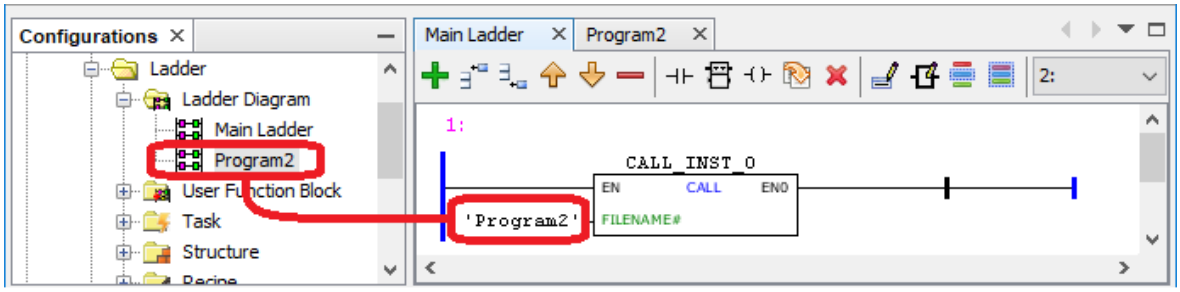
Device	Version
PLC300	4.03 or higher

Block Flowchart



Example

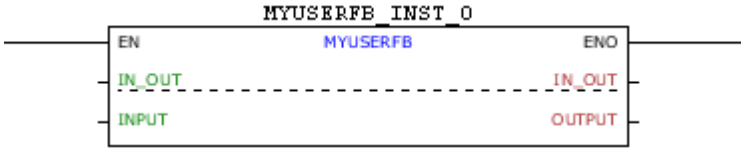
In the example below, the POU 'Program2' will be executed through the 'Main Ladder'.



11.10.7.13.2 USERFB

Block that performs a subroutine programmed by the user.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	INPUT	Conforme programação do usuário	Block inputs
VAR_OUTPUT	ENO	BOOL	End of operation
	OUTPUT	Conforme programação do usuário	Block outputs
VAR_IN_OUT	IN_OUT	Conforme programação do usuário	Block inputs/outputs
VAR	MYUSERFB_INST_0	MYUSERFB	Instance of access to block structure

Operation

When this block has a TRUE value in EN, it updates the values of internal fields with the input variables, performs the Ladder routine programmed by the user and updates the values of the outputs after completing routine.

When EN has FALSE value, outputs remain unchanged.

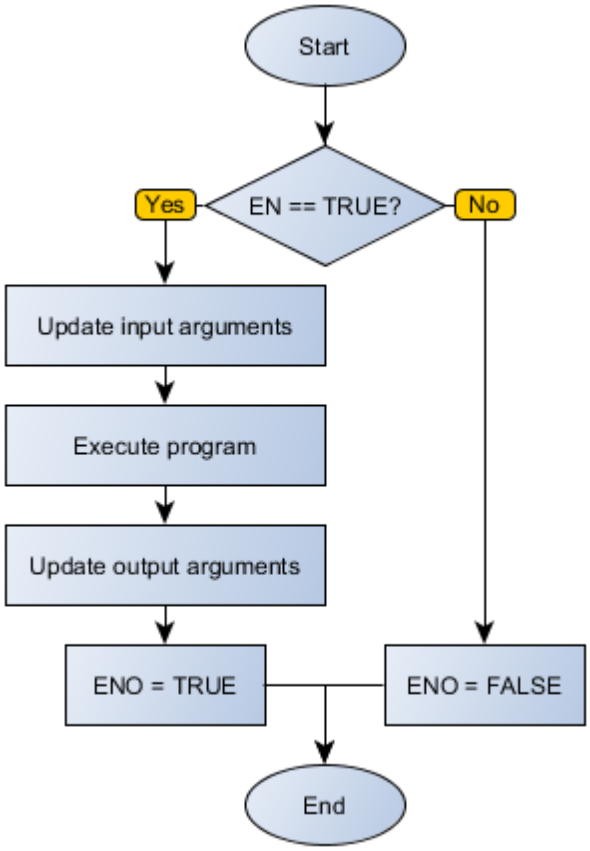
The ENO value forwards to the next Ladder block the EN value after the operation is completed.

NOTE!
Refer to section Working with USERFBs for further information.

Compatibility

Device	Version
PLC300	1.50 or higher
SCA06	2.00 or higher

Block Flowchart



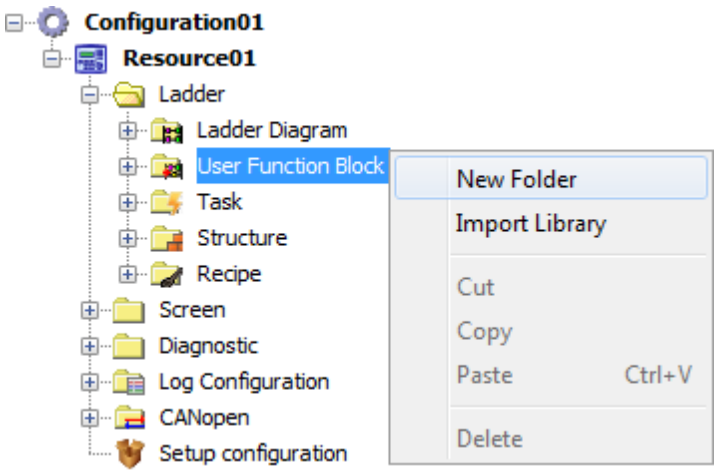
11.10.7.13.3 Working with USERFBs

11.10.7.13.3.1 Creating USERFBs

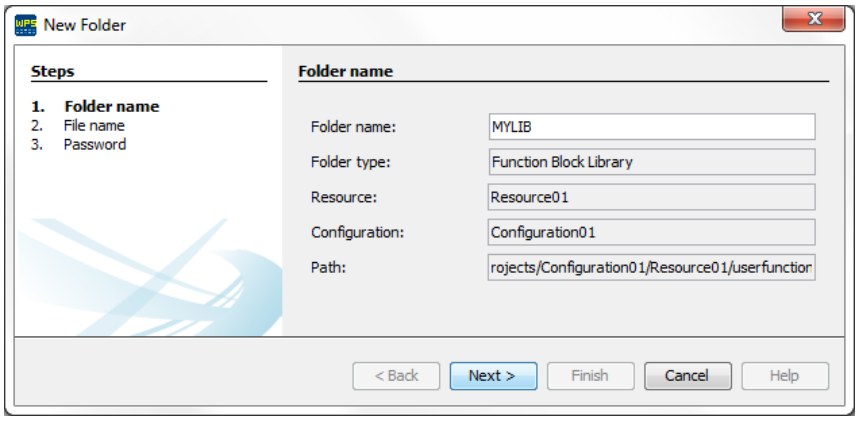
USERFBs are cuser-customizable functional blocks. Its utilization is encouraged to make the Ladder program less bulky and polluted, abstracting information with which one does not want to work often and systematizing complex tasks.

In these blocks, the inputs and outputs are defined by the user, who handles them in the Ladder diagram associated with the block. Here's how to create your USERFB.

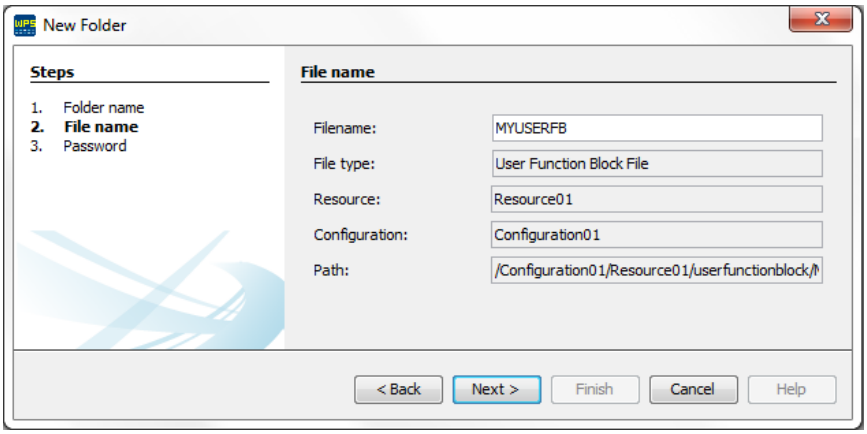
1. In the **Projects** window, locate the resource in which you want to create the USERFB, right-click in **User Function Block** and click in **New Folder**.



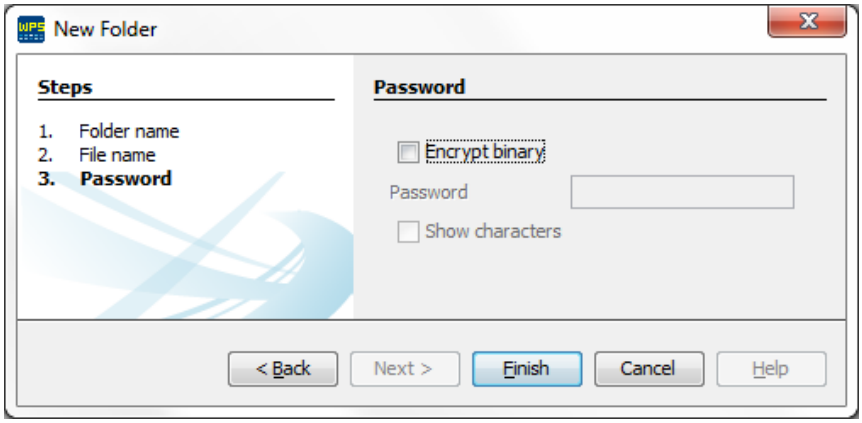
2. In the wizard, insert a name for the library to which the USERFB will belong and click **Next**.



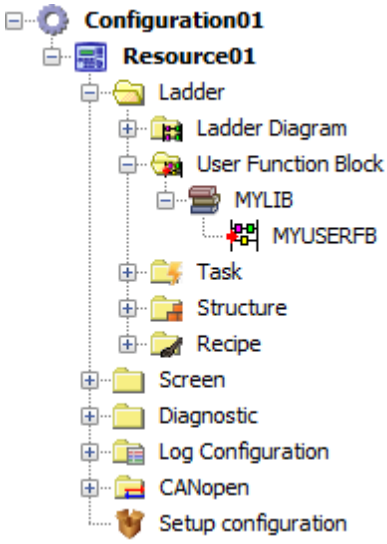
3. Insert a valid name for the USERFB and click **Next**.



4. If you want to insert a password to protect the block code, check the **Encrypt binary** checkbox and type a password. Otherwise, uncheck it. Click **Finish**.



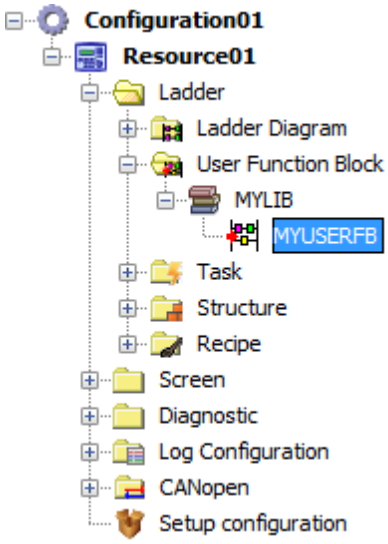
That's it! The USERFB has been succesfully created. You should see the following in the **Projects** window.



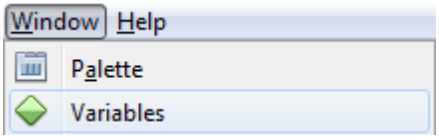
11.10.7.13.3.2 Adding input/output

Now we'll cover how to create inputs and outputs for the USERFB.

1. In the **Projects** window, double click the USERFB file in order to open its Ladder editor.

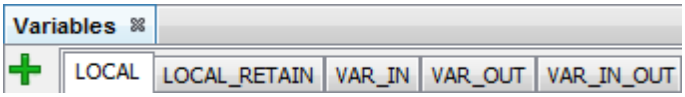


2. In the **Window** menu, click **Variables**.

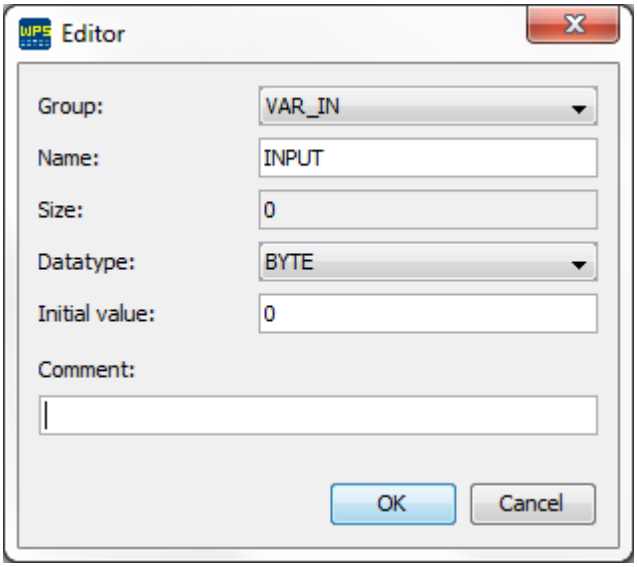


Analysing the following figure, we see that the USERFB **Variables** window is different from other Ladder files. It has only volatile and retain variables in **LOCAL** scope, which are the internal variables of the block, used in its subroutine. Besides these, it has three more groups: **VAR_IN**, **VAR_OUT** and **VAR_IN_OUT**.

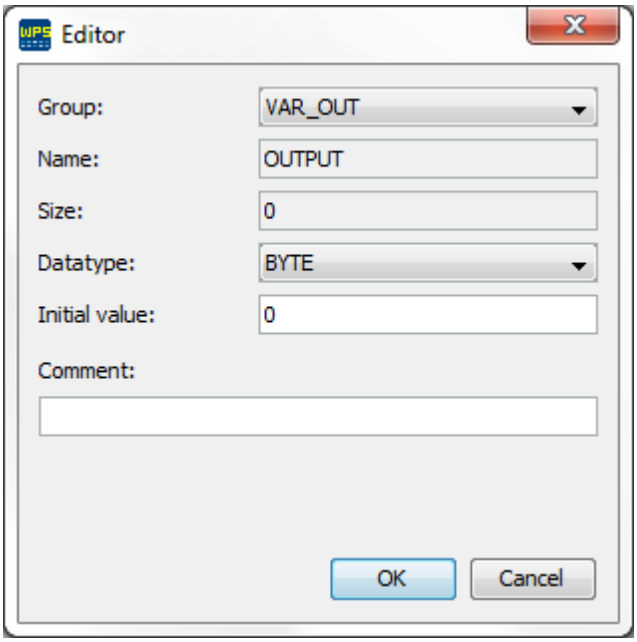
- **VAR_IN**: internal variables that repersent the input arguments for that block.
- **VAR_OUT**: internal variables that repersent the output arguments for that block.
- **VAR_IN_OUT**: internal variables that repersent the input/output arguments for that block.



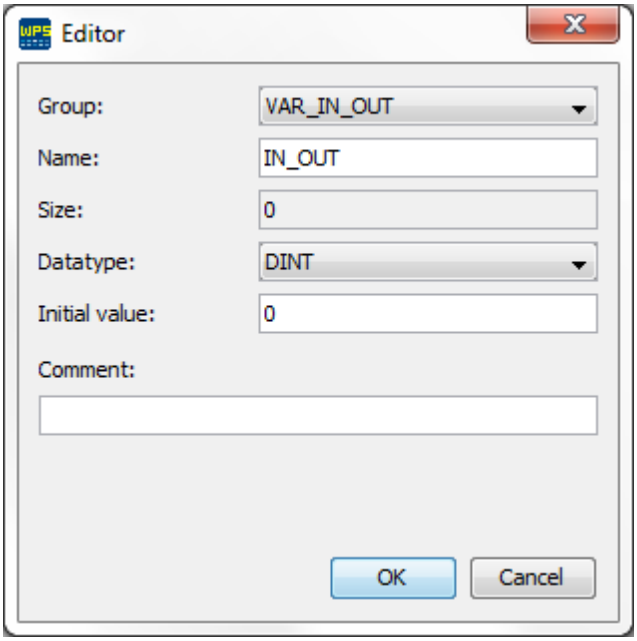
3. In order to create an input, click in the **VAR_IN** tab and click in the symbol. In the window, set a name and a datatype to this variable and click **OK**.



4. In order to create an output, click in the **VAR_OUT** tab and click in the symbol. In the window, set a name and a datatype to this variable and click **OK**.



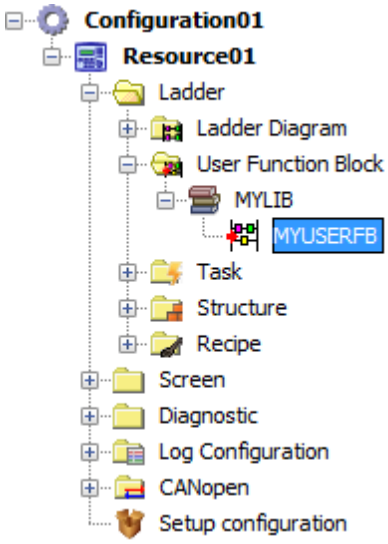
5. In order to create an input/output, click in the **VAR_IN_OUT** tab and click in the symbol. In the window, set a name and a datatype to this variable and click **OK**.



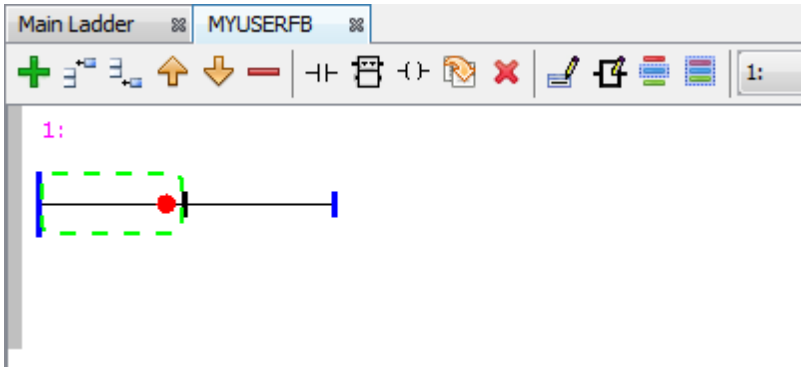
11.10.7.13.3.3 Editing the Ladder

Now we'll cover how to edit the USERFB subroutine.

1. In the **Projects** window, double click the USERFB file in order to open its Ladder editor.



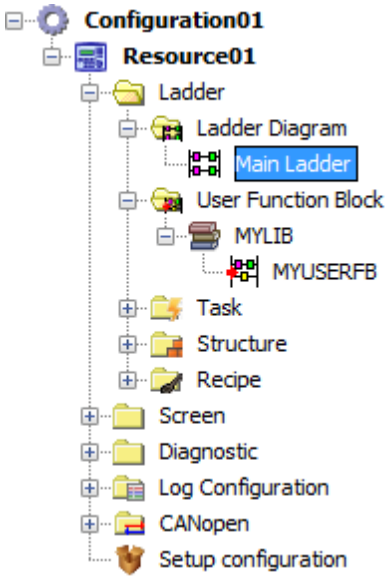
The **Ladder Editor** will open, like any other Ladder diagram. Any block may be inserted in it, including other USERFBs. Remember that only local variables may be used in it.



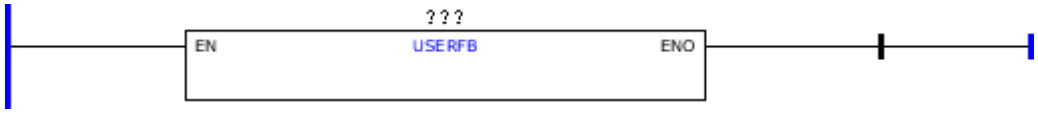
11.10.7.13.3.4 Using USERFBs

Lastly we'll cover how to make use of the USERFB, inserting it in other Ladder diagrams.

1. In the **Projects** window, double click the USERFB file in order to open its Ladder editor.

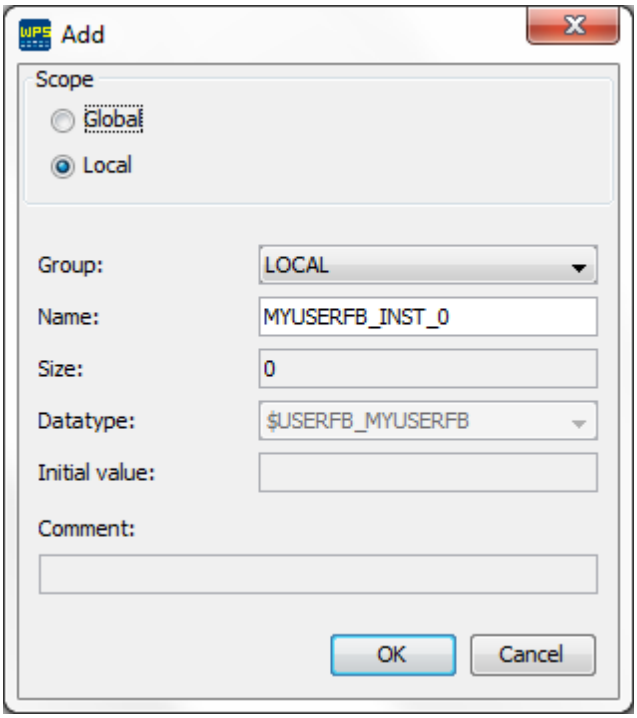


2. In the **Pallete** window, select the USERFB block from the **Module** category and drag it to the position where you want to use it in the Ladder diagram.

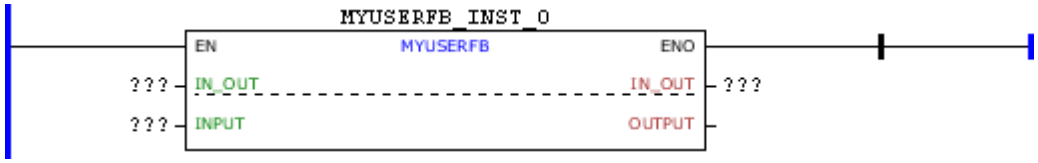


3. Double click the question marks (**???**) above the block in order to insert a instance variable for the USERFB. Type in the variable name and click **Edit**. In the confirmation dialog, click **Yes** to create the new variable.

4. In the Add dialog, type in a name for the variable and select its parameters. In the Datatype field, choose the name of the desired USERFB (if there is only one, the field will not be enabled).
For example, if your USERFB name is MYUSERFB, the correct datatype to be selected is $USERFB_MYUSERFB.



That's it! Your very own USERFB is inserted in the diagram and ready to work!



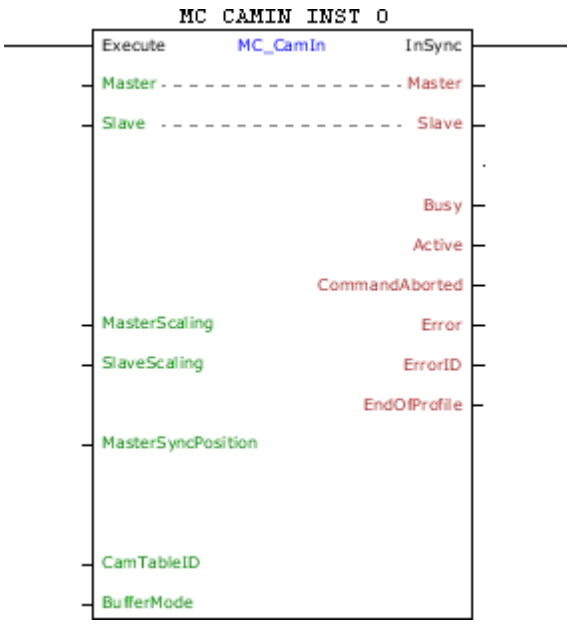
11.10.7.14 Motion Control

11.10.7.14.1 Motion Control Cam

11.10.7.14.1.1 MC_CamIn

Block responsible for the execution of a defined positioning a cam table of a curve CAM.

Ladder Representation



Execution Features

Program Memory Size	92 Bytes
Data Memory Size	52 Bytes

Block Structure

Variable Type	Name	Data Type	Description
VAR_IN_OUT	Master	BYTE	Selection of operation master (0 - Fast digital inputs) (1 - CANopen) (2 - Encoder 1) (3 - Virtual Axis) (4 - Encoder 2)
	Slave	BYTE	Selection of operation slave (0 - Real axis)
VAR_INPUT	Execute	BOOL	Block enabling
	MasterScaling	REAL	Gain in the position values of the master axis.
	SlaveScaling	REAL	Gain in the position values of the slave axis.
	MasterSyncPosition	LREAL	Position of the master axis where the slave will start the CAM curve.
	CamTableID	WORD	Selected CAM table
	BufferMode	BYTE	Execution start mode (0 - Starts block immediately, if there is another block in the execution it will be aborted) (1 - When another block is in execution, the block in execution will continue its motion until the end and this new block will wait to be executed.) (6 – If another block is in execution, this block will go into error 52 and will not be executed. The HMI will show the alarm A00052.)
VAR_OUTPUT	InSync	BOOL	Output enabling
	Busy	BOOL	Flag indicating the block has not yet been ended
	Active	BOOL	Block flag with control on the axis
	CommandAborted	BOOL	Flag of aborted command
	Error	BOOL	Error in the execution flag
	ErrorID	WORD	Identifier of the occurred error
	EndOfProfile	BOOL	End of the CAM profile flag
VAR	MC_CAMIN_INST_0	MC_CAMIN	Instance of access to block structure

Operation

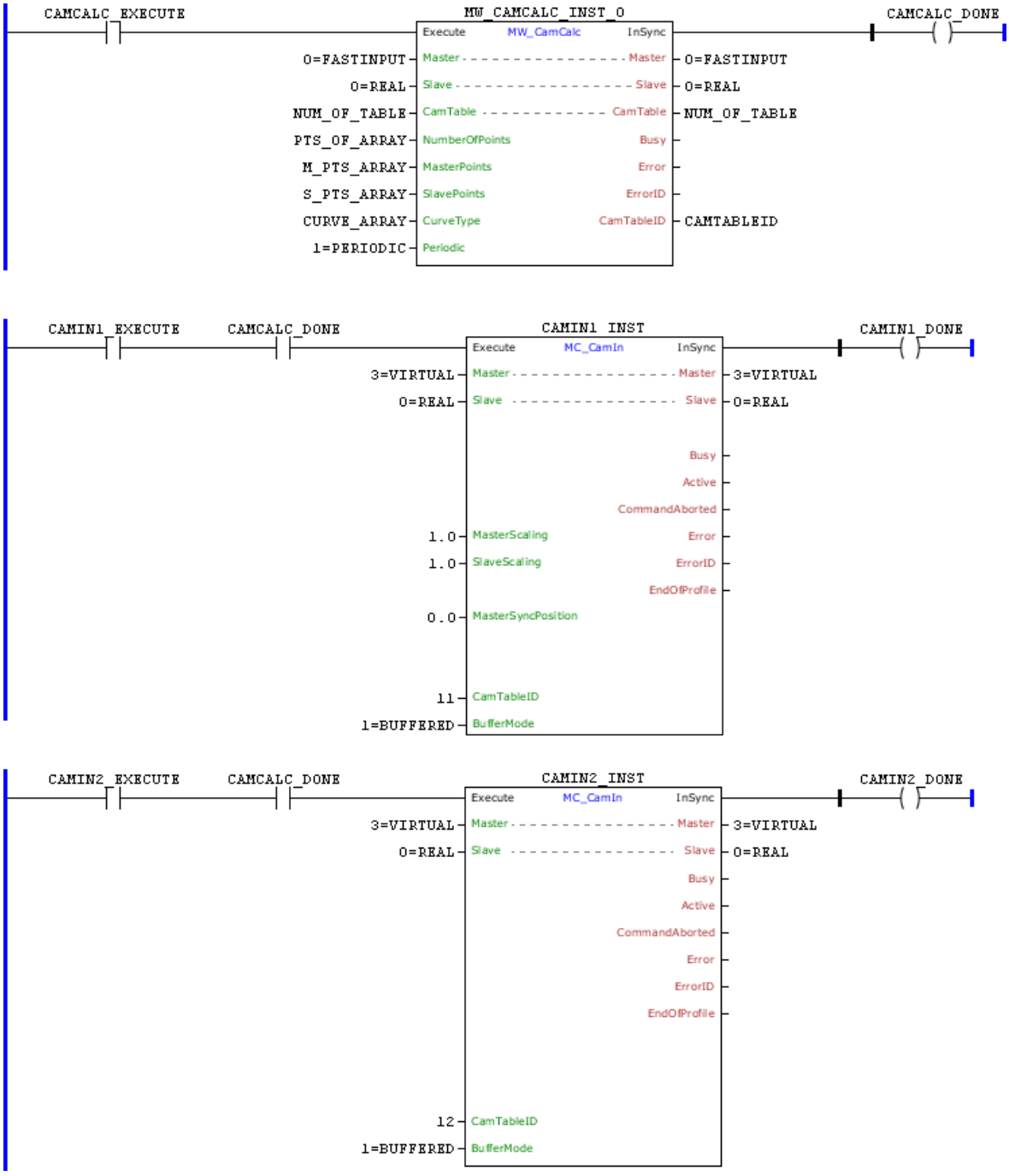
When this block detects a leading edge in Execute, it sends a command for the drive to execute a defined positioning by CamTableID.

When Execute has FALSE value, Done remains FALSE. The Done output is activated when the block finishes the execution successfully, remaining at TRUE level until Execute receives FALSE.

If there is any error in the execution, the Error output is enabled and ErrorID displays an error code according to the table below.

Code	Description
52	Attempt to execute block w ith BufferMode in Single w hen another block is active.
67	Drive in the "Disabled" or “Errorstop” status.
69	Drive in the "Stopping" status.
70	Attempt to execute block w ith BufferMode in Buffered w hen another block is active and another block is w aiting.
71	P202 different from 4.
74	Drive in the "Homing" status.
78	MC block not executed – Internal fault.
85	CamTableID not valid.

Example


CAMCALC_EXECUTE
MW_CAMCALC_INST_0
Execute
MW_CamCalc
InSync
0=FASTINPUT
Master
Master
0=FASTINPUT
0=REAL
Slave
Slave
0=REAL
NUM_OF_TABLE
CamTable
CamTable
NUM_OF_TABLE
PTS_OF_ARRAY
NumberOfPoints
Busy
M_PTS_ARRAY
MasterPoints
Error
S_PTS_ARRAY
SlavePoints
ErrorID
CURVE_ARRAY
CurveType
CamTableID
CAMTABLEID
1=PERIODIC
Periodic
CAMCALC_DONE
CAMIN1_EXECUTE
CAMCALC_DONE
CAMIN1_INST
Execute
MC_CamIn
InSync
3=VIRTUAL
Master
Master
3=VIRTUAL
0=REAL
Slave
Slave
0=REAL
Busy
Active
CommandAborted
1.0
MasterScaling
Error
1.0
SlaveScaling
ErrorID
EndOfProfile
0.0
MasterSyncPosition
11
CamTableID
1=BUFFERED
BufferMode
CAMIN1_DONE
CAMIN2_EXECUTE
CAMCALC_DONE
CAMIN2_INST
Execute
MC_CamIn
InSync
3=VIRTUAL
Master
Master
3=VIRTUAL
0=REAL
Slave
Slave
0=REAL
Busy
Active
CommandAborted
Error
ErrorID
EndOfProfile
12
CamTableID
1=BUFFERED
BufferMode
CAMIN2_DONE

LOCAL | LOCAL_RETAIN

Tag	Tamanho	Tipo de Dado	Valor Inicial	Comentário
CAMIN1_INST	0	MC_CamIn		
CAMIN1_EXECUTE	0	BOOL	0	
CAMIN1_DONE	0	BOOL	0	
CAMIN2_INST	0	MC_CamIn		
CAMIN2_EXECUTE	0	BOOL	0	
CAMIN2_DONE	0	BOOL	0	
MW_CAMCALC_INST_0	0	MW_CamCalc		
NUM_OF_TABLE	0	WORD	11	
PTS_OF_ARRAY	0	WORD	2	
M_PTS_ARRAY	2	LREAL		
M_PTS_ARRAY[0]	0	LREAL	3.0	
M_PTS_ARRAY[1]	0	LREAL	7.0	
S_PTS_ARRAY	2	LREAL		
S_PTS_ARRAY[0]	0	LREAL	10.0	
S_PTS_ARRAY[1]	0	LREAL	-5.0	
CURVE_ARRAY	2	WORD		
CURVE_ARRAY[0]	0	WORD	0	
CURVE_ARRAY[1]	0	WORD	1	
CAMTABLEID	0	WORD	11	

GLOBAL | LOCAL

In the up transition of CAMCALC_EXECUTE, the MW_CamCalc block is executed and the cam table of NUM_OF_TABLE will be calculated according to the block arguments.

When the calculation of cam table 11 is finished, the InSync output is set while the Execute input remains set.

With the CAMCALC_DONE set, the MC_CamIn block can be executed.

In the up transition of CAMIN1_EXECUTE, the first MC_CamIn block is executed.

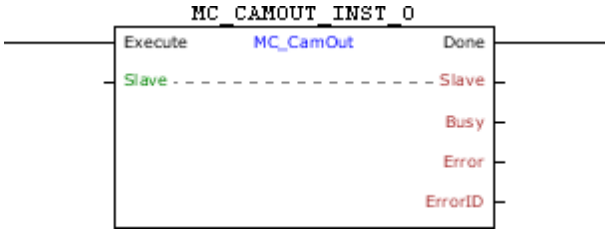
If it is necessary to adjust the cam table of the CAM curve, just make the adjustment in MASTER_PT [0], MASTER_PT[1], SLAVE_PT[0] and SLAVE_PT[1], change the content of NUM_OF_TABLE to 12 and execute the MW_CamCalc block again.

In the transition of CAMIN2_EXECUTE, the second MC_CamIn block will be executed (without losing the position of the master axis) as soon as the first MC_CamIn block finishes executing the curve in execution.

11.10.7.14.1.2 MC_CamOut

Block responsible for the completion of a synchronization established by a block MC_CamIn.

Ladder Representation



Execution Features

Program Memory Size	28 Bytes
Data Memory Size	4 Bytes

Block Structure

Variable Type	Name	Data Type	Description
VAR_IN_OUT	Slave	BYTE	Selection of operation slave (0 - Real axis)
VAR_INPUT	Execute	BOOL	Block enabling
	CamTableID	WORD	Selected CAM table
	BufferMode	BYTE	Execution start mode (0 - Starts block immediately, if there is another block in the execution it will be aborted) (1 - When another block is in execution, the block in execution will continue its motion until the end and this new block will wait to be executed.) (6 – If another block is in execution, this block will go into error 52 and will not be executed. The HMI will show the alarm A00052.)
VAR_OUTPUT	Done	BOOL	Output enabling
	Busy	BOOL	Flag indicating the block has not yet been ended
	Error	BOOL	Error in the execution flag
	ErrorID	WORD	Identifier of the occurred error
VAR	MC_CAMOUT_INST_0	MC_CAMOUT	Instance of access to block structure

Operation

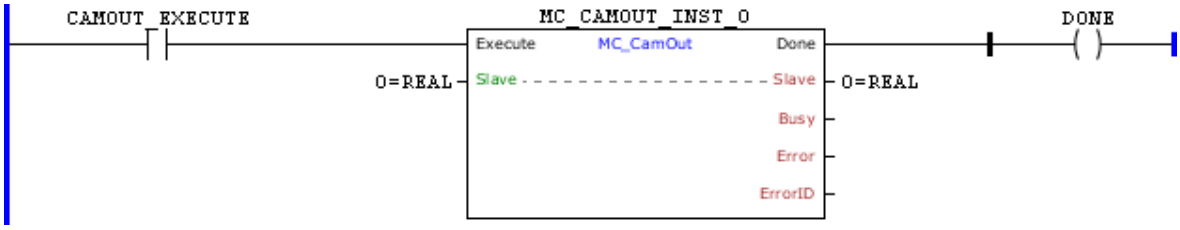
When this block detects a leading edge in Execute it concludes the existing synchronism for the last execution of a block MC_CamIn. The axis will keep the speed of the moment in which the block is executed.

When Execute has FALSE value, Done remains FALSE. The Done output is activated when the block finishes the execution successfully, remaining at TRUE level until Execute receives FALSE.

If there is any error in the execution, the Error output is enabled and ErrorID displays an error code according to the table below.

Code	Description
67	Drive in the "Disabled" or “Errorstop” status.
71	P202 different from 4.
73	Drive is not in the “Synchronized Motion” status
78	MC block not executed – Internal fault.

Example

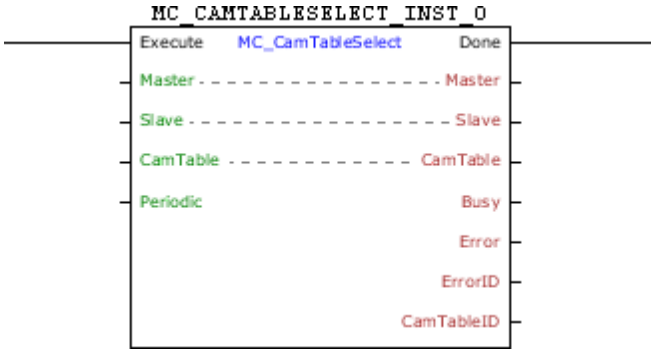


In the up transition of CAMOUT_EXECUTE, the MC_CamOut block is executed. With this, the Busy and Active signals of this block are set and synchronism started by other CAM blocks ends. When the process finishes, the Done output of the block is set and remains TRUE while the Execute input is set.

11.10.7.14.1.3 MC_CamTableSelect

Block that selects a cam table of a CAM curve previously programmed through the tool CAM Profiles.

Ladder Representation



Execution Features

Program Memory Size	44 Bytes
Data Memory Size	16 Bytes

Block Structure

Variable Type	Name	Data Type	Description
VAR_IN_OUT	Master	BYTE	Selection of operation master (0 - Fast digital inputs) (1 - CANopen) (2 - Encoder 1) (3 - Virtual Axis) (4 - Encoder 2)
	Slave	BYTE	Selection of operation slave (0 - Real axis)
	CamTable	WORD	CAM table code
VAR_INPUT	Execute	BOOL	Block enabling
	Periodic	BOOL	Control of the request execution (0 - Single execution) (1 - Periodic execution)
VAR_OUTPUT	Done	BOOL	Output enabling
	Busy	BOOL	Flag indicating the block has not yet been ended
	Error	BOOL	Error in the execution flag
	ErrorID	WORD	Identifier of the occurred error
	CamTableID	WORD	Selected CAM table
VAR	MC_CAMTABLESELECT_INST_0	MC_CAMTABLESELECT	Instance of access to block structure

Operation

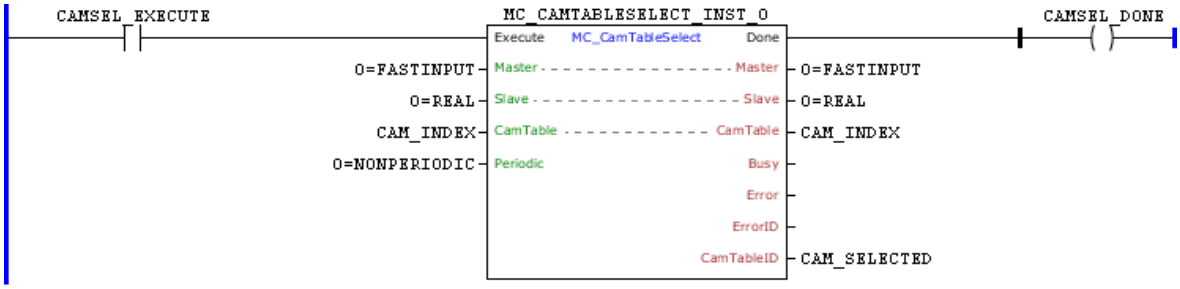
When this block detects a leading edge in Execute, it searches the specified table in CamTable so that it can be used by the MC_CamIn block.

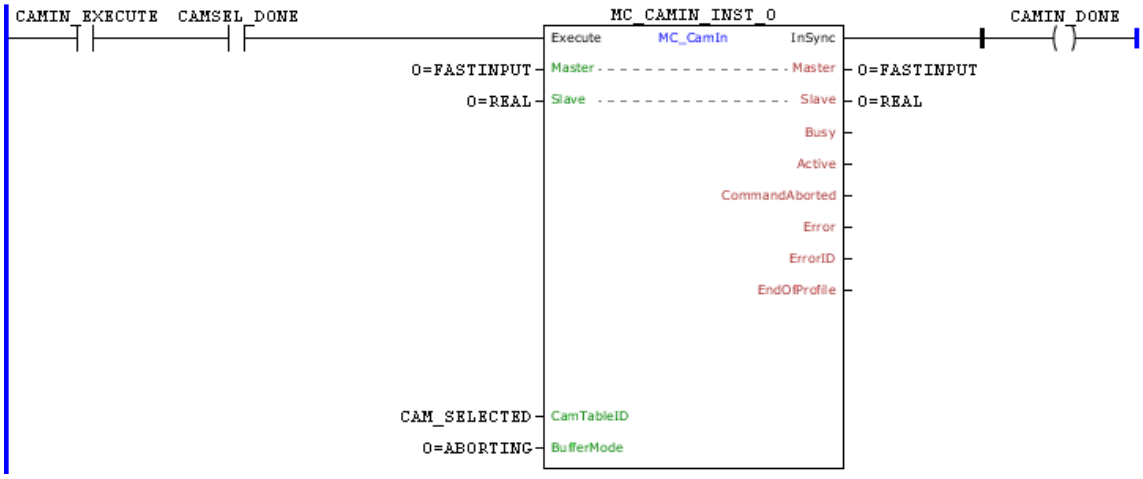
When Execute has FALSE value, Done remains FALSE. The Done output is activated when the block finishes the execution successfully, remaining at TRUE level until Execute receives FALSE.

If there is any error in the execution, the Error output is enabled and ErrorID displays an error code according to the table below.

Code	Description
83	Invalid file of the CAM curve cam table.
84	Invalid Cam Table. Cam Table must be from 1 to 10.

Example





LOCAL | LOCAL_RETAIN

Tag	Tamanho	Tipo de Dado	Valor Inicial	Comentário
MC_CAMTABLESELECT_INST_0	0	MC_CamTableSelect		
CAMSEL_EXECUTE	0	BOOL	0	
CAM_INDEX	0	WORD	3	
CAM_SELECTED	0	WORD	3	
CAMSEL_DONE	0	BOOL	0	
MC_CAMIN_INST_0	0	MC_CamIn		
CAMIN_EXECUTE	0	BOOL	0	
CAMIN_DONE	0	BOOL	0	

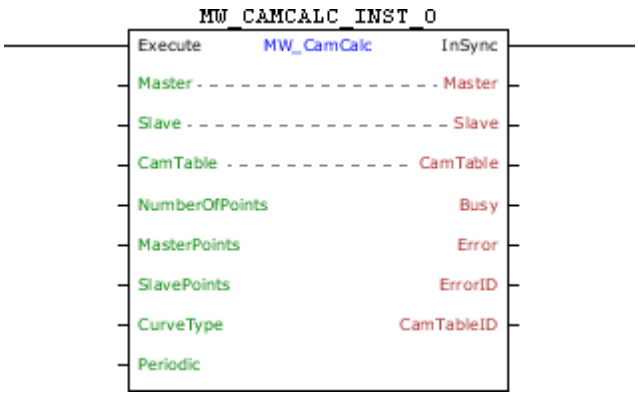
GLOBAL | LOCAL

In the up transition of CAMSEL_EXECUTE the MC_CamTableSelect block is executed and, thus, the cam table selected in CAM_INDEX can be used by the MC_CamIn block.
To execute the block, the Done output and CAMSEL_DONE are set and remain at TRUE level while the Execute input is set.
In this example, CAMSEL_DONE ensures that the MC_CamIn block will not be activated before the MC_CamTableSelect block is executed successfully.

11.10.7.14.1.4 MW_CamCalc

This block calculates a cam table of the CAM curve.

Ladder Representation



Execution Features

Program Memory Size	66 Bytes
Data Memory Size	24 Bytes

Block Structure

Variable Type	Name	Data Type	Description
VAR_IN_OUT	Master	BYTE	Selection of operation master (0 - Fast digital inputs) (1 - CANopen) (2 - Encoder 1) (3 - Virtual Axis) (4 - Encoder 2)
	Slave	BYTE	Selection of operation slave (0 - Real axis)
	CamTable	WORD	CAM table code
VAR_INPUT	Execute	BOOL	Block enabling
	NumberOfPoints	WORD	Number of points of the table
	MasterPoints	LREAL	Points of the master
	SlavePoints	LREAL	Points of the slave
	CurveType	WORD	Type of curve
	Periodic	BOOL	Control of the request execution (0 - Single execution) (1 - Periodic execution)
VAR_OUTPUT	InSync	BOOL	Output enabling
	Busy	BOOL	Flag indicating the block has not yet been ended
	Error	BOOL	Error in the execution flag
	ErrorID	WORD	Identifier of the occurred error
	CamTableID	WORD	Selected CAM table
VAR	MW_CAMCALC_INST_0	MW_CAMCALC	Instance of access to block structure

Operation

When this block detects a leading edge in Execute, it builds the specified table in CamTable with the

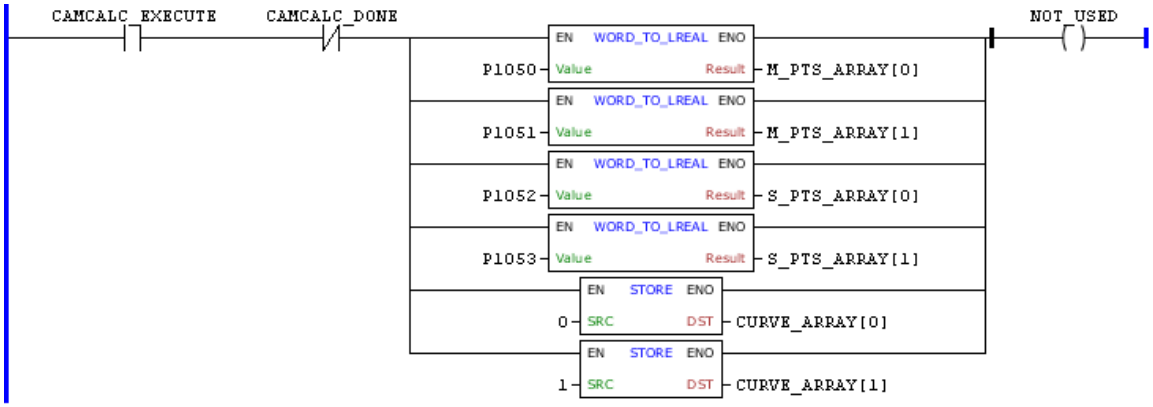
programmed data so that it can be used by the MC_CamIn block.

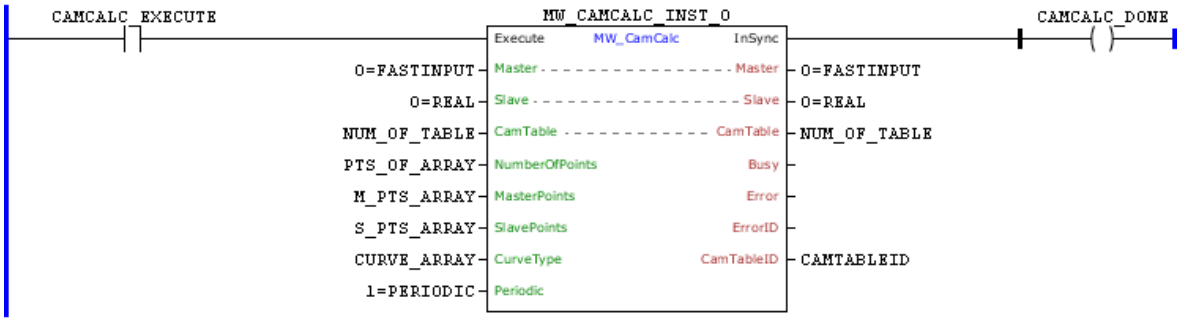
When Execute has FALSE value, Done remains FALSE. The Done output is activated when the block finishes the execution successfully, remaining at TRUE level until Execute receives FALSE.

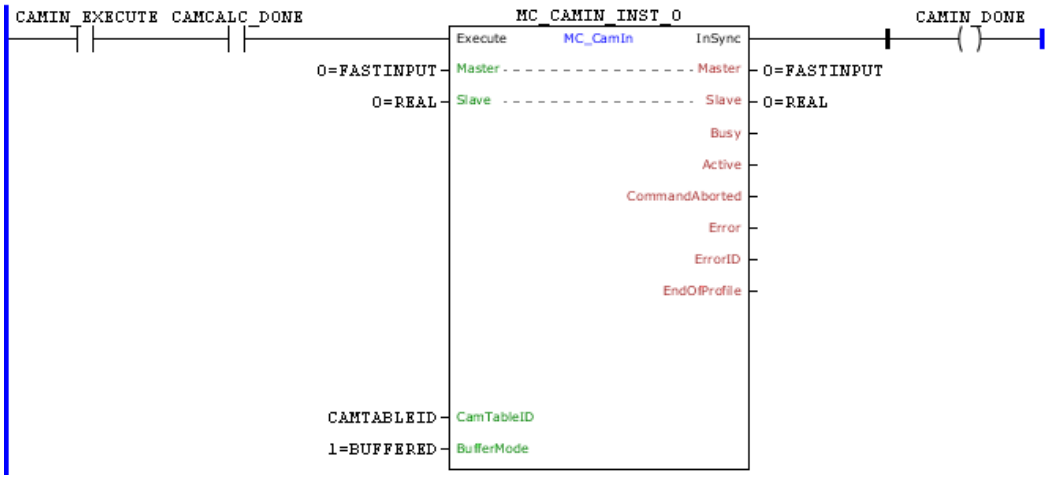
If there is any error in the execution, the Error output is enabled and ErrorID displays an error code according to the table below.

Code	Description
83	Invalid file of the CAM curve cam table.
84	Invalid Cam Table. Cam Table must be from 11 to 20.
86	Number of points above the programmed in the CAM PROFILES configurator.
87	Invalid position of the master axis. The position of the master axis must be greater than the position of the previous point.
88	MW_CamCalc block in execution. Is only allowed the execution of an MW_CamCalc block at a time.
89	Cam table in use by the MC_CamIn block.
90	Double marker with position of the nonexistent master axis.
91	Double marker with position of the nonexistent slave axis.
92	Word marker with nonexistent curve type.

Example


CAMCALC_EXECUTE
CAMCALC_DONE
NOT_USED
WORD_TO_LREAL
P1050
M_PTS_ARRAY[0]
P1051
M_PTS_ARRAY[1]
P1052
S_PTS_ARRAY[0]
P1053
S_PTS_ARRAY[1]
STORE
0
CURVE_ARRAY[0]
1
CURVE_ARRAY[1]


CAMCALC_EXECUTE
MW_CAMCALC_INST_0
CAMCALC_DONE
Execute
MW_CamCalc
InSync
0=FASTINPUT
Master
0=REAL
Slave
NUM_OF_TABLE
CamTable
PTS_OF_ARRAY
NumberOfPoints
Busy
M_PTS_ARRAY
MasterPoints
Error
S_PTS_ARRAY
SlavePoints
ErrorID
CURVE_ARRAY
CurveType
CamTableID
CAMTABLEID
1=PERIODIC
Periodic


CAMIN_EXECUTE
CAMCALC_DONE
MC_CAMIN_INST_0
CAMIN_DONE
Execute
MC_CamIn
InSync
0=FASTINPUT
Master
0=REAL
Slave
Busy
Active
CommandAborted
Error
ErrorID
EndOfProfile
CAMTABLEID
CamTableID
1=BUFFERED
BufferMode

LOCAL | LOCAL_RETAIN

Tag	Tamanho	Tipo de Dado	Valor Inicial	Comentário
NOT_USED	0	BOOL	0	
MW_CAMCALC_INST_0	0	MW_CamCalc		
CAMCALC_EXECUTE	0	BOOL	0	
CAMCALC_DONE	0	BOOL	0	
CAMTABLEID	0	WORD	11	
PTS_OF_ARRAY	0	WORD	2	
M_PTS_ARRAY	2	LREAL		
M_PTS_ARRAY[0]	0	LREAL	3.0	
M_PTS_ARRAY[1]	0	LREAL	7.0	
S_PTS_ARRAY	2	LREAL		
S_PTS_ARRAY[0]	0	LREAL	10.0	
S_PTS_ARRAY[1]	0	LREAL	-5.0	
CURVE_ARRAY	2	WORD		
CURVE_ARRAY[0]	0	WORD	0	
CURVE_ARRAY[1]	0	WORD	1	
NUM_OF_TABLE	0	WORD	11	
MC_CAMIN_INST_0	0	MC_CamIn		
CAMIN_EXECUTE	0	BOOL	0	
CAMIN_DONE	0	BOOL	0	

GLOBAL | LOCAL

CAM Profile List

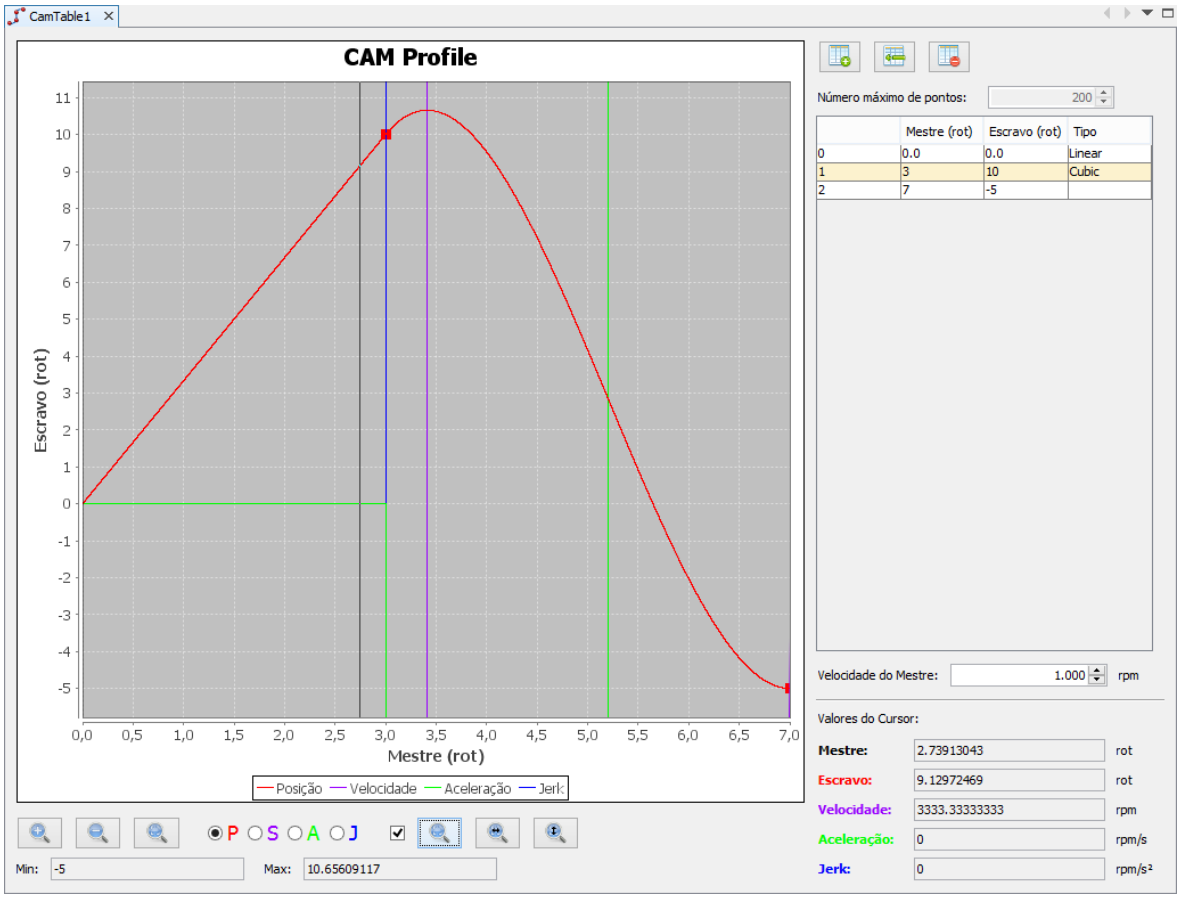
Cam Table	Cam Type	Cam File	Max Points
1	Fixed	---	0
2	Fixed	---	0
3	Fixed	---	0
4	Fixed	---	0
5	Fixed	---	0
6	Fixed	---	0
7	Fixed	---	0
8	Fixed	---	0
9	Fixed	---	0
10	Fixed	---	0
11	Calculable		2
12	Calculable		0
13	Calculable		0
14	Calculable		0
15	Calculable		0
16	Calculable		0
17	Calculable		0
18	Calculable		0
19	Calculable		0
20	Calculable		0

Editar Perfil | Selecionar Perfil | Desconsiderar Perfil

In the transition from 0 to 1 of bit marker CAMCALC_EXECUTE, the MW_CamCalc block is executed and the cam table of TABLE will be calculated according to the block arguments.

In this example, the number of points of the curve will be contents of the NUM_POINTS, the position of the master axis will be in accordance with the contents of MASTER_POINTS [0] and MASTER_POINTS [1], the position of the slave axis will be in accordance with the contents of SLAVE_POINTS [0] and SLAVE_POINTS [1] and the type of curve will be in accordance with the contents of CURVE_TYPE [0] and CURVE_TYPE[1].

Entering the same values in the tool, we can observe the curve below:



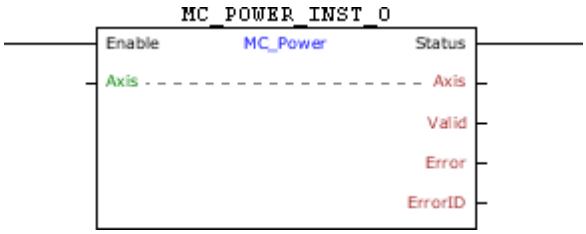
When the calculation of the cam table 11 is finished, the InSync output is set while the Execute input remains set.
With the CAMCALC_DONE set, the MC_CamIn block can be executed.

11.10.7.14.2 Motion Control Command

11.10.7.14.2.1 MC_Power

Block responsible for enabling/disabling the drive axis.

Ladder Representation



Execution Features

Program Memory Size	40 Bytes
Data Memory Size	4 Bytes

Block Structure

Variable Type	Name	Data Type	Description
VAR_IN_OUT	Axis	BYTE	Selection of operation axis (0 - Real axis)
VAR_INPUT	Enable	BOOL	Block enabling
VAR_OUTPUT	Status	BOOL	Output enabling
	Valid	BOOL	Flag indicating validity of the output signals
	Error	BOOL	Error in the execution flag
	ErrorID	WORD	Identifier of the occurred error
VAR	MC_POWER_INST_0	MC_POWER	Instance of access to block structure

Operation

This block performs an Enable/Disable command of the Axis axis according to the Enable input, disabling it Enable is FALSE and enabling if Enable is TRUE.

NOTE!
When the MC_Power block is used to enable/disable the real axis, no digital inputs must be programmed for the Enable function (option 1), or an Alarm A0120 may occur.

When enabling the real axis for the first time, the drive may operate in grid position, depending on the value of parameter P0773. The position proportional gain (P0159) must be set to obtain a better drive performance.

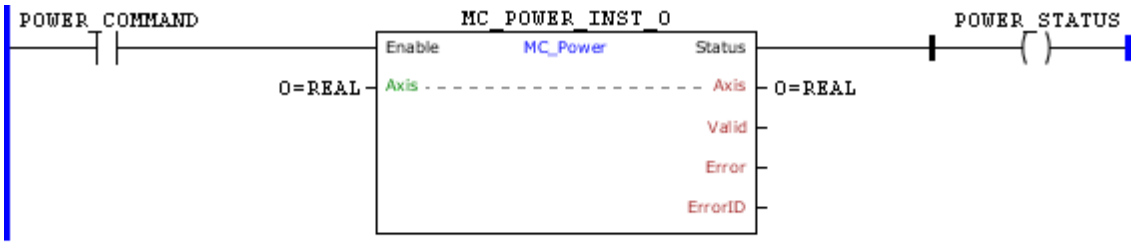
When the real axis is disabled, the axis status will be Disabled. When enabling the real axis, the axis status will change to Standstill.

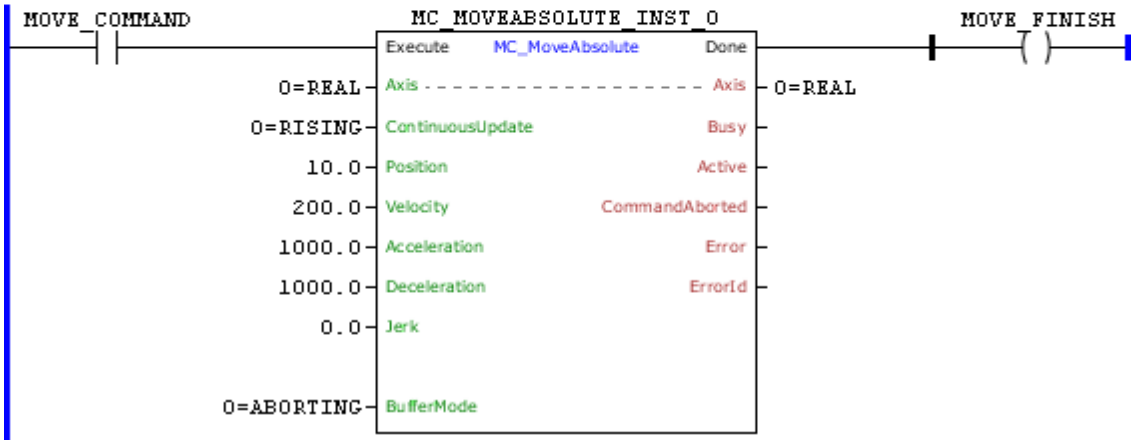
When EN has FALSE value, Status remains FALSE. The Status output is activated when the block finishes the execution successfully, remaining at TRUE level until Enable receives FALSE.

If there is any error in the execution, the Error output is enabled and ErrorID displays an error code according to the table below.

Code	Description
66	Drive in the "ErrorStop" status.
71	P202 different from 4.

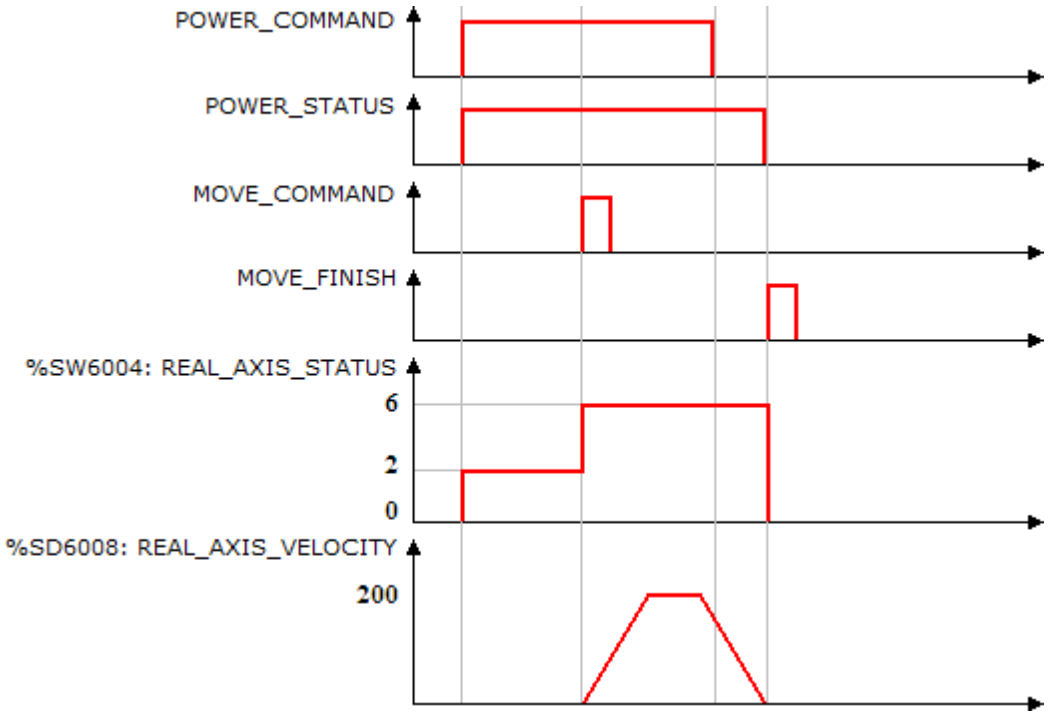
Example





LOCAL | LOCAL_RETAIN

Tag	Tamanho	Tipo de Dado	Valor Inicial	Comentário
MC_POWER_INST_0	0	MC_Power		
POWER_COMMAND	0	BOOL	0	
MC_MOVEABSOLUTE_INST_0	0	MC_MoveAbsolute		
POWER_STATUS	0	BOOL	0	
MOVE_COMMAND	0	BOOL	0	
MOVE_FINISH	0	BOOL	0	



On the leading edge of POWER_COMMAND, the real axis is enabled and its status, the system marker REAL_AXIS_STATUS (%SW6004), is changed to Standstill (%SW6004 = 2). The Status output is set.

After the transition from 0 to 1 of MOTION_COMMAND, the MC_MotionAbsolut block is executed and the positioning for the position 10 revolutions starts. The axis status is changed to “Discrete Motion” (%SW3406 = 6).

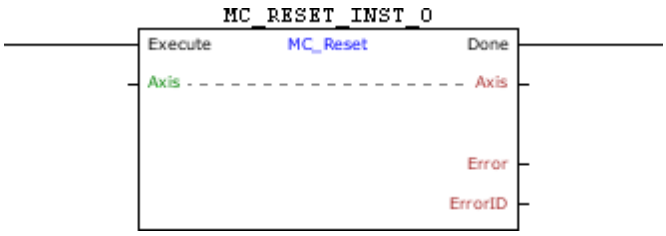
While positioning is executed, POWER_COMMAND is reset. Since the BufferMode of the MC_Power is set to Buffered, the axis will be disabled only in the positioning completion.

When the positioning is finished, the Done output of the MC_MotionAbsolut block is set for 1 scan cycle and the axis is disabled (MC_Power.Enable = 0). The axis status changes to “Disable” (% SW6004 = 0).

11.10.7.14.2.2 MC_Reset

Block responsible for cleaning up the failure status of drive.

Ladder Representation



Execution Features

Program Memory Size	28 Bytes
Data Memory Size	4 Bytes

Block Structure

Variable Type	Name	Data Type	Description
VAR_IN_OUT	Axis	BYTE	Selection of operation axis (0 - Real axis)
VAR_INPUT	Execute	BOOL	Block enabling
VAR_OUTPUT	Done	BOOL	Output enabling
	Error	BOOL	Error in the execution flag
	ErrorID	WORD	Identifier of the occurred error
VAR	MC_RESET_INST_0	MC_RESET	Instance of access to block structure

Operation

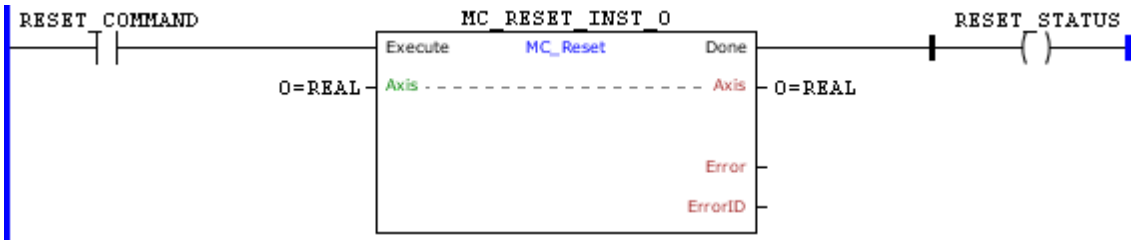
When this block detects a leading edge in Execute, it performs cleanliness in the drive status, changing it from Errorstop to Disabled.

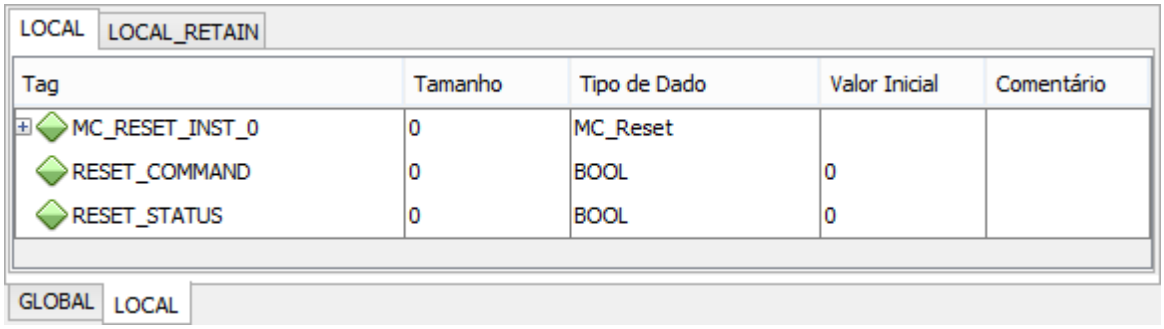
When Execute has FALSE value, Done remains FALSE. The Done output is activated when the block finishes the execution successfully, remaining at TRUE level until Execute receives FALSE.

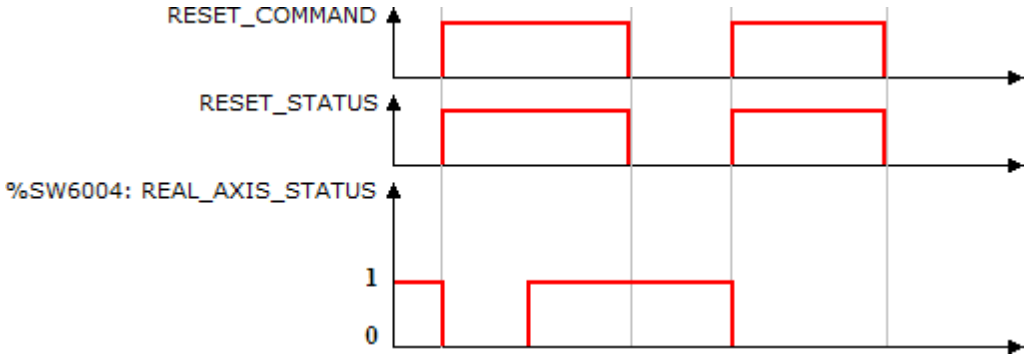
If there is any error in the execution, the Error output is enabled and ErrorID displays an error code according to the table below.

Code	Description
71	P202 different from 4.

Example





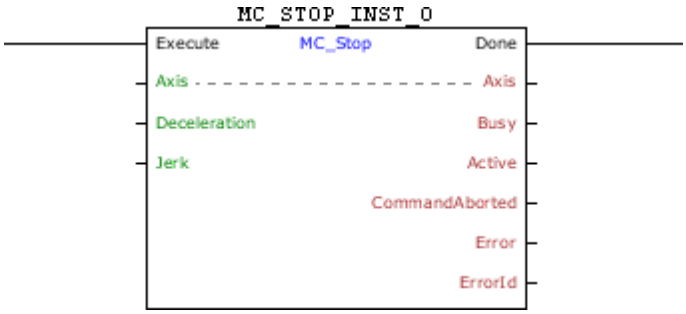


If a fault occurs on the drive, the axis status will change to “Errorstop” (%SW6004 = 1).
When an up transition of RESET_COMMAND occurs, the MC_Reset block will be executed and axis status will change to Disabled (SW6004 = 0%). The Done output will remain set while the Execute input is TRUE level.

11.10.7.14.2.3 MC_Stop

Block responsible for executing a controlled stop.

Ladder Representation



Execution Features

Program Memory Size	52 Bytes
Data Memory Size	12 Bytes

Block Structure

Variable Type	Name	Data Type	Description
VAR_IN_OUT	Axis	BYTE	Selection of operation axis (0 - Real axis)
VAR_INPUT	Execute	BOOL	Block enabling
	Deceleration	REAL	Deceleration of the stop
	Jerk	REAL	Jerk
VAR_OUTPUT	Done	BOOL	Output enabling
	Busy	BOOL	Flag indicating the block has not yet been ended
	Active	BOOL	Block flag with control on the axis
	CommandAborted	BOOL	Flag of aborted command
	Error	BOOL	Error in the execution flag
	ErrorID	WORD	Identifier of the occurred error
VAR	MC_STOP_INST_0	MC_STOP	Instance of access to block structure

Operation

When this block detects a leading edge in Execute, it sends a command for a controlled stop of the axis. While Execute has TRUE level, no other MC block is executed.

When the MC_Stop block is executed, the drive will star operating in grid position and remain this way even after the conclusion of the block. The position proportional gain must be set (P0159) so as to obtain a better drive performance.

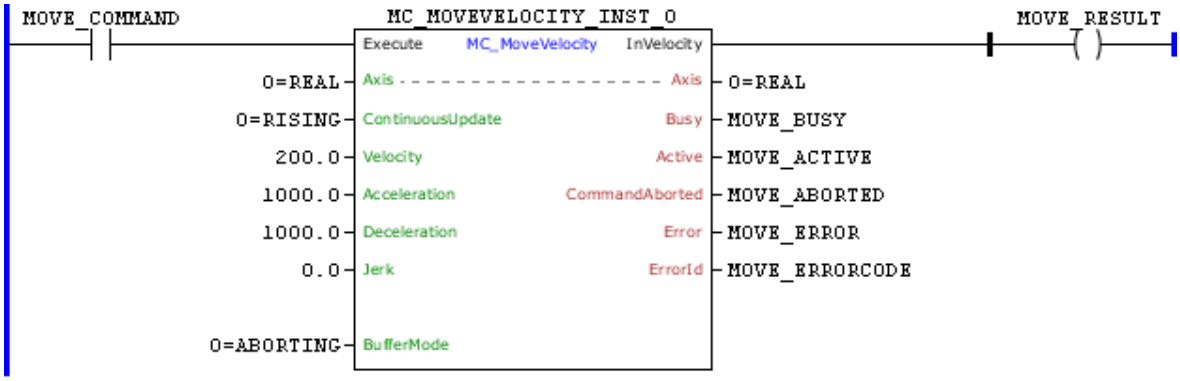
In the execution of the block, the axis status will change to Stopping. When the stop is ended and the block is no longer active, the axis status will change to Standstill.

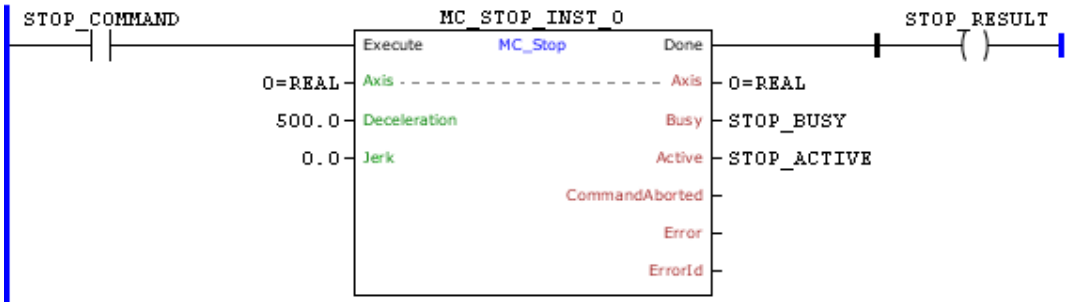
When Execute has FALSE value, Done remains FALSE. The Done output is activated when the block finishes the execution successfully, remaining at TRUE level until Execute receives FALSE.

If there is any error in the execution, the Error output is enabled and ErrorID displays an error code according to the table below.

Code	Description
64	Deceleration programmed below the minimum allowed.
65	Deceleration programmed above the maximum allowed.
67	Drive in the "Disabled" or “Errorstop” status.
71	P202 different from 4.
78	MC block not executed – Internal fault.
93	Jerk programmed below the minimum allowed.
94	Jerk programmed above the maximum allowed.

Example

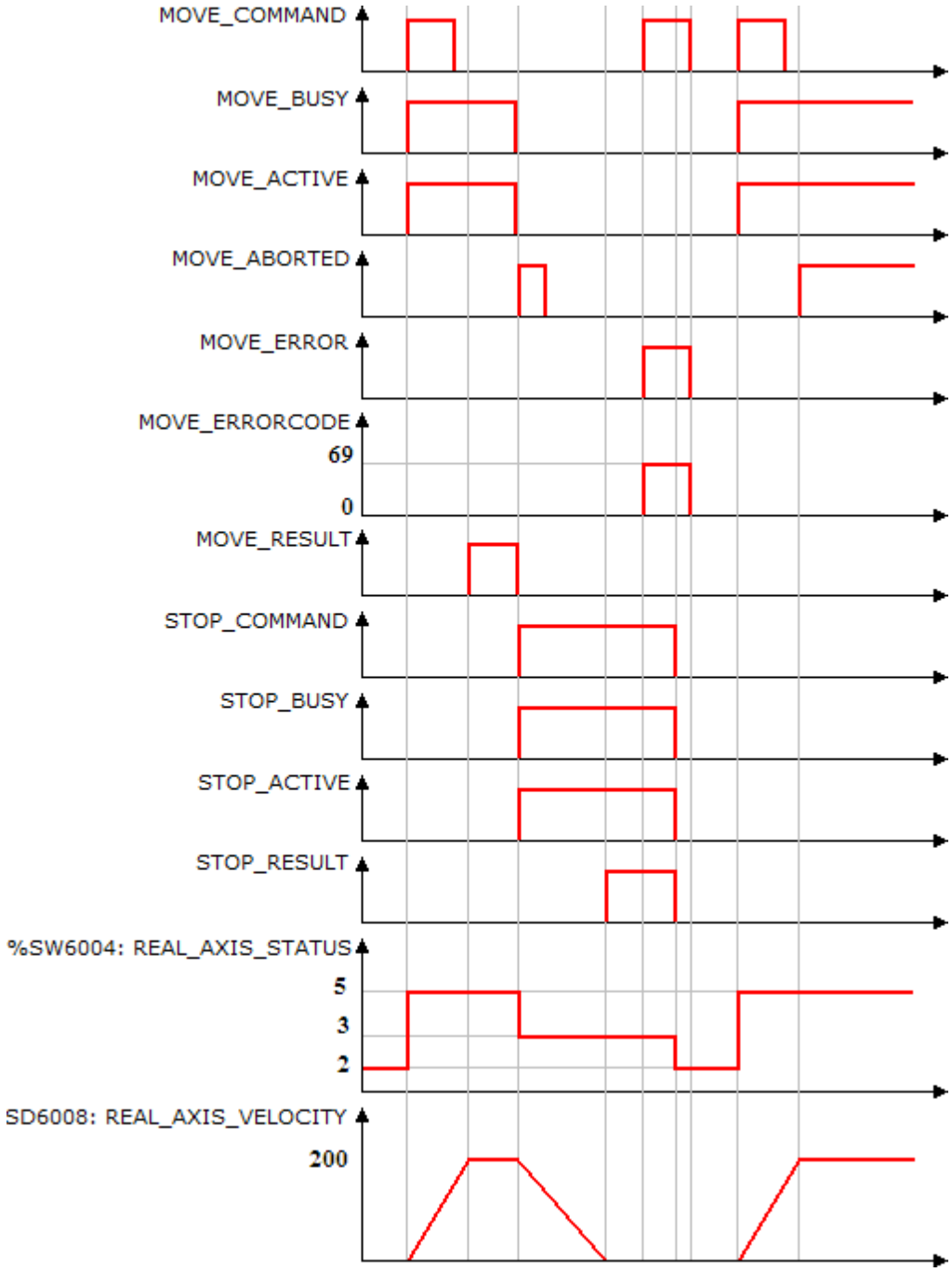




LOCAL | LOCAL_RETAIN

Tag	Tamanho	Tipo de Dado	Valor Inicial	Comentário
MC_MOVEVELOCITY_INST_0	0	MC_MoveVelocity		
MC_STOP_INST_0	0	MC_Stop		
MOVE_COMMAND	0	BOOL	0	
MOVE_RESULT	0	BOOL	0	
MOVE_BUSY	0	BOOL	0	
MOVE_ACTIVE	0	BOOL	0	
MOVE_ABORTED	0	BOOL	0	
MOVE_ERROR	0	BOOL	0	
MOVE_ERRORCODE	0	WORD	0	
STOP_COMMAND	0	BOOL	0	
STOP_RESULT	0	BOOL	0	
STOP_BUSY	0	BOOL	0	
STOP_ACTIVE	0	BOOL	0	

GLOBAL | LOCAL



In the up transition of MOTION_COMMAND, the MC_MotionVelocity block is executed. With this, the Busy and Active signals of this block are set and the motion to reach the speed of 200 RPM starts. The axis status (%SW6004) changes from Standstill (2) to Continuous Motion (5).
At the moment the speed reaches 200 rpm, MOTION_RESULT is set.

With the up transition of STOP_COMMAND, the MC_Stop block is instantly executed; thus, the Busy and Active signals of this block are set and the stop starts. At the same time, the Busy, Active and InVelocity signals of the MC_MotionVelocity block are reset and the CommandAborted signal is set for 1 scan. The axis status (%SW6004) changes from Continuous Motion (5) to Stopping (3).

At the end of the stop, the Done output MC_Stop block is set and remains until Execute input is reset. The axis status (%SW6004) remains equal to Stopping (3) and no other MC block will be executed.

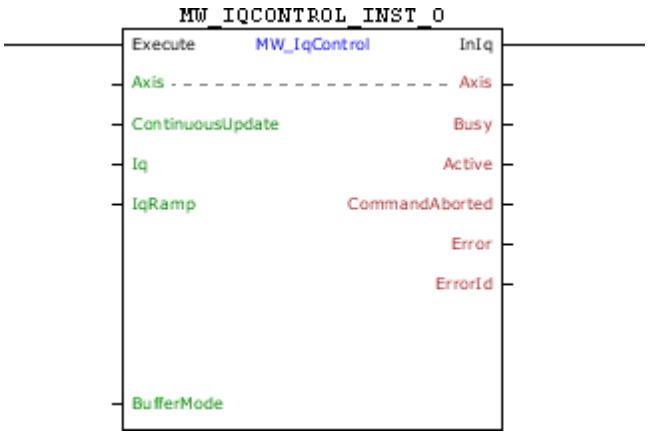
With the transition of rising of MOTION_COMMAND, the MC_MotionVelocity block is started, but as the MC_Stop block is active, an error occurs and the Error signal will be set by entering the value 69 in ErrorID.

When the Execute input of the MC_Stop block is reset, the Busy, Active and Done signal are reset. The axis status (%SW6004) changes from Stopping (3) to Standstill (2) and other MCs blocks can be executed.

11.10.7.14.2.4 MW_IqControl

Block conducting the Iq control programmed.

Ladder Representation



Execution Features

Program Memory Size	78 Bytes
Data Memory Size	32 Bytes

Block Structure

Variable Type	Name	Data Type	Description
VAR_IN_OUT	Axis	BYTE	Selection of operation axis (0 - Real axis)
VAR_INPUT	Execute	BOOL	Block enabling
	ContinuousUpdate	BYTE	Automatic update mode (0 – Leading edge) (1 – Real time)
	Iq	REAL	Iq Value
	IqRamp	REAL	Iq ramp value
	BufferMode	BYTE	Execution start mode (0 - Starts block immediately, if there is another block in the execution it will be aborted) (1 - When another block is in execution, the block in execution will continue its motion until the end and this new block will wait to be executed.) (6 – If another block is in execution, this block will go into error 52 and will not be executed. The HMI will show the alarm A00052.)
VAR_OUTPUT	InIq	BOOL	Output enabling
	Busy	BOOL	Flag indicating the block has not yet been ended
	Active	BOOL	Block flag with control on the axis
	CommandAborted	BOOL	Flag of aborted command
	Error	BOOL	Error in the execution flag
	ErrorID	WORD	Identifier of the occurred error
VAR	MW_IQCONTROL_INST_0	MW_IQCONTROL	Instance of access to block structure

Operation

When this block detects a leading edge on Execute, it sends an execution command of the Iq control according to the programmed parameters.

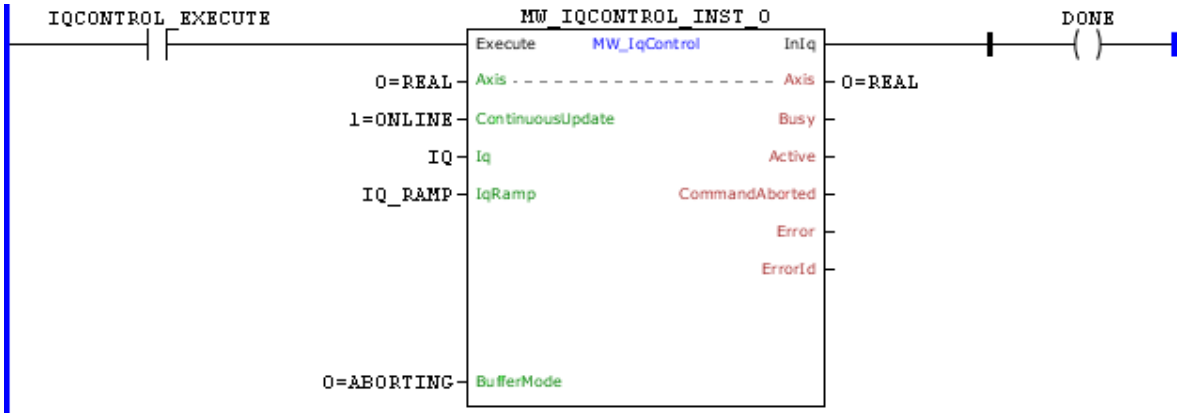
In the execution of the block the axis status will change to Continuous Motion. It order to finish the block, it is necessary to execute another block or the changing of the drive to the Disable or Errorstop status.

When Execute has FALSE value, Done remains FALSE. The Done output is activated when the block finishes the execution successfully, remaining at TRUE level until Execute receives FALSE.

If there is any error in the execution, the Error output is enabled and ErrorID displays an error code according to the table below.

Code	Description
52	Attempt to execute block with BufferMode in Single when another block is active.
67	Drive in the "Disabled" or “Errorstop” status.
69	Drive in the "Stopping" status.
70	Attempt to execute block with BufferMode in Buffered when another block is active and another block is waiting.
71	P202 different from 4.
74	Drive in the "Homing" status.
78	MC block not executed – Internal fault.
80	Iq programmed above the maximum allowed.
81	IqRamp programmed below the minimum allowed.
82	IqRamp programmed above the maximum allowed.

Example



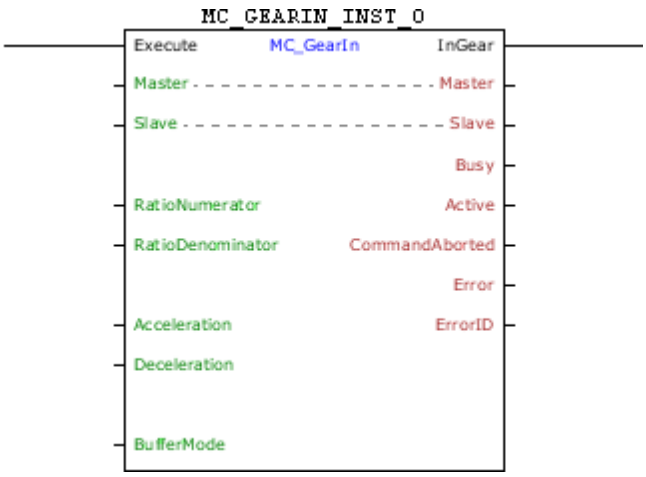
In the up transition of IQCONTROL_EXECUTE, the MW_IqControl block is executed. With this, the Busy and Active signals of this block are set and starts the execution of Iq control according to parameters set. When the process finishes, the Done output of the block is set and remains TRUE while the Execute input is set.

11.10.7.14.3 Motion Control Gear

11.10.7.14.3.1 MC_GearIn

Block responsible for execution of the synchronism in speed between the programmed axes.

Ladder Representation



Execution Features

Program Memory Size	74 Bytes
Data Memory Size	28 Bytes

Block Structure

Variable Type	Name	Data Type	Description
VAR_IN_OUT	Master	BYTE	Selection of operation master (0 - Fast digital inputs) (1 - CANopen) (2 - Encoder 1) (3 - Virtual Axis) (4 - Encoder 2)
	Slave	BYTE	Selection of operation slave (0 - Real axis)
VAR_INPUT	Execute	BOOL	Block enabling
	RatioNumerator	INT	Numerator of the synchronism ratio
	RatioDenominator	WORD	Denominator of the synchronism ratio
	Acceleration	REAL	Acceleration
	Deceleration	REAL	Deceleration
	BufferMode	BYTE	Execution start mode (0 - Starts block immediately, if there is another block in the execution it will be aborted) (1 - When another block is in execution, the block in execution will continue its motion until the end and this new block will wait to be executed.) (6 – If another block is in execution, this block will go into error 52 and will not be executed. The HMI will show the alarm A00052.)
VAR_OUTPUT	InGear	BOOL	Output enabling
	Busy	BOOL	Flag indicating the block has not yet been ended
	Active	BOOL	Block flag with control on the axis
	CommandAborted	BOOL	Flag of aborted command
	Error	BOOL	Error in the execution flag
	ErrorID	WORD	Identifier of the occurred error
VAR	MC_GEARIN_INST_0	MC_GEARIN	Instance of access to block structure

Operation

When this block detects a leading edge on Execute, it sends a command for synchronism in speed between the programmed axes.

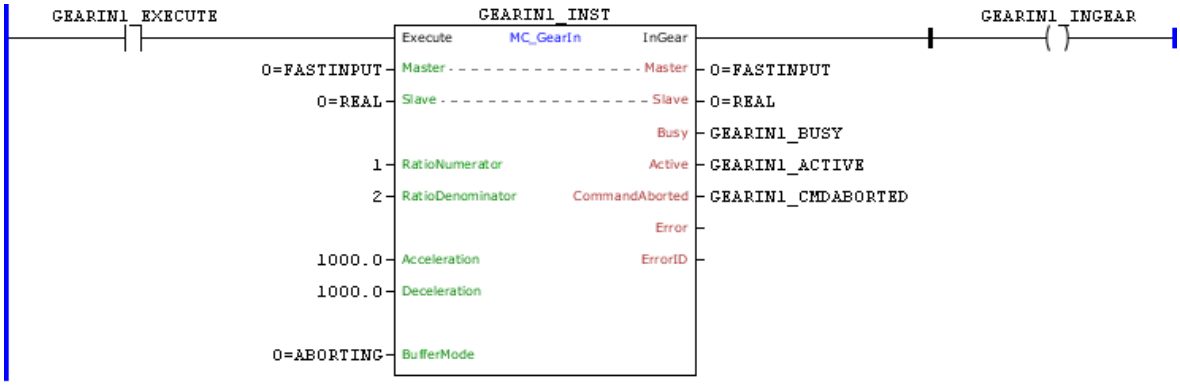
For the slave axis to reach the speed of the master axis, a motion will be performed with an acceleration/deceleration configured in the “Acceleration” and “Deceleration” arguments. The motion direction will depend on the signal of the RatioNumerator. If RatioNumerator is greater than zero, the motion will be in the same direction as the master axis, and if the RatioNumerator is smaller than zero, the motion will be in the opposite the direction of the master axis.

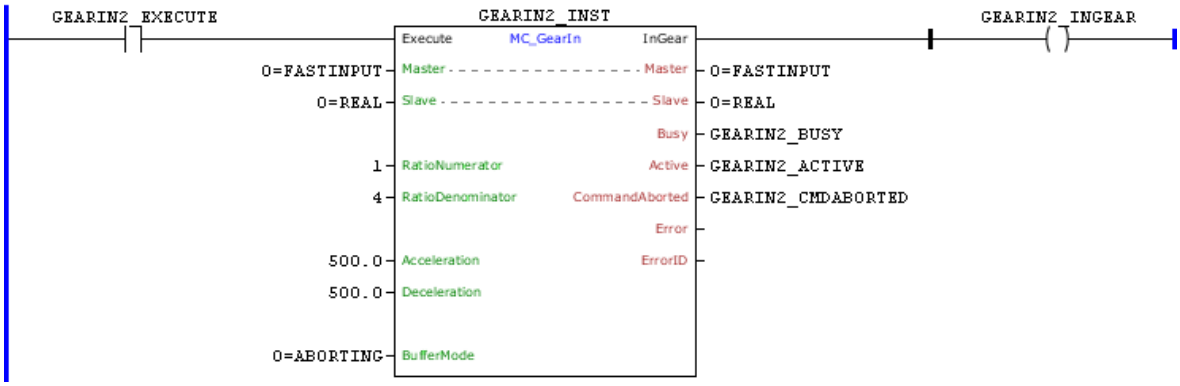
The InGear output is activated when the sync is achieved. It order to finish the block, it is necessary to execute another block or the changing of the drive to the Disabled or Errorstop status.

If there is any error in the execution, the Error output is enabled and ErrorID displays an error code according to the table below.

Code	Description
52	Attempt to execute block w ith BufferMode in Single w hen another block is active.
62	Acceleration programmed below the minimum allow ed.
63	Acceleration programmed above the maximum allow ed.
64	Deceleration programmed below the minimum allow ed.
65	Deceleration programmed above the maximum allow ed.
67	Drive in the "Disabled" or “Errorstop” status.
69	Drive in the "Stopping" status.
70	Attempt to execute block w ith BufferMode in Buffered w hen another block is active and another block is w aiting.
71	P202 different from 4.
72	Invalid synchronism ratio.
74	Drive in the "Homing" status.
78	MC block not executed – Internal fault.

Example

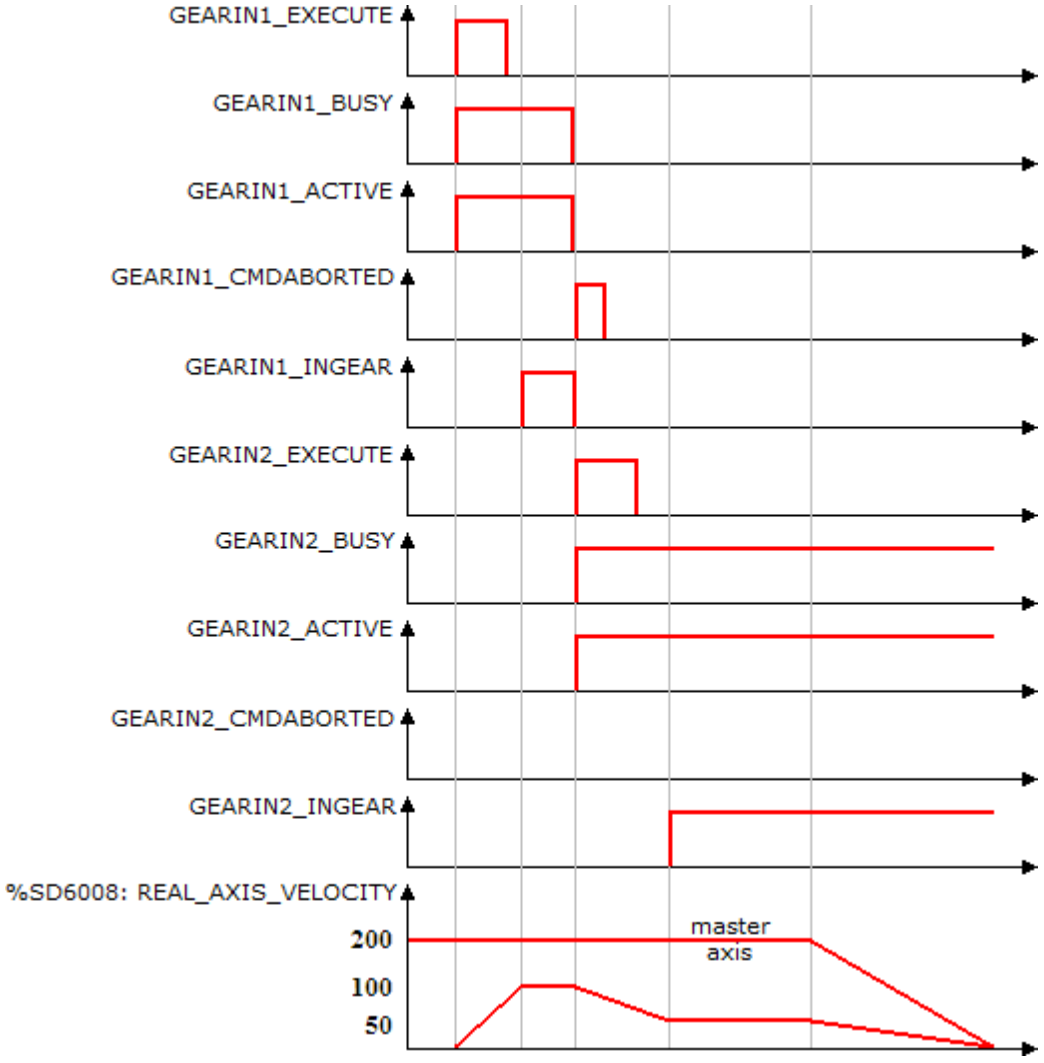




LOCAL | LOCAL_RETAIN

Tag	Tamanho	Tipo de Dado	Valor Inicial	Comentário
GEARIN1_INST	0	MC_GearIn		
GEARIN1_EXECUTE	0	BOOL	0	
GEARIN1_INGEAR	0	BOOL	0	
GEARIN1_BUSY	0	BOOL	0	
GEARIN1_ACTIVE	0	BOOL	0	
GEARIN1_CMDABORTED	0	BOOL	0	
GEARIN2_INST	0	MC_GearIn		
GEARIN2_EXECUTE	0	BOOL	0	
GEARIN2_INGEAR	0	BOOL	0	
GEARIN2_BUSY	0	BOOL	0	
GEARIN2_ACTIVE	0	BOOL	0	
GEARIN2_CMDABORTED	0	BOOL	0	

GLOBAL | LOCAL



In the up transition of GEARIN1_EXECUTE, the first MC_GearIn block is executed. With this the Busy Active and this block signals are set and the search of synchronization with the configured

acceleration begins. As the ratio configured is 1:2 and the master axis is at 200 RPM, the slave axis must reach 100 RPM to establish the synchronism.
At the moment in which the speed reaches 100 RPM, the InGear output is set.

With the transition of rising GEARIN2_EXECUTE, the second MC_GearIn block is instantly executed. With this the Busy Active signals of this block are set and the search of synchronization with the configured acceleration begins. As the ratio configured is 1:4 and the master axis is at 200 RPM, the slave axis must reach 50 RPM to establish the synchronism. At the same time, the Busy, Active and InGear signals of the first block are reset and the CommandAborted signal is set for 1 scan.

When the speed of 50 RPM is reached, the InGear output of the second block is set and remains until the execution of other block.

11.10.7.14.3.2 MC_GearInPos

Block responsible for execution of the synchronism in position between the programmed axes.

Ladder Representation

Execution Features

Program Memory Size	104 Bytes
Data Memory Size	56 Bytes

Block Structure

Variable Type	Name	Data Type	Description
VAR_IN_OUT	Master	BYTE	Selection of operation master (0 - Fast digital inputs) (1 - CANopen) (2 - Encoder 1) (3 - Virtual Axis) (4 - Encoder 2)
	Slave	BYTE	Selection of operation slave (0 - Real axis)
VAR_INPUT	Execute	BOOL	Block enabling
	RatioNumerator	INT	Numerator of the synchronism ratio
	RatioDenominator	WORD	Denominator of the synchronism ratio
	Acceleration	REAL	Acceleration
	Deceleration	REAL	Deceleration
	BufferMode	BYTE	Execution start mode (0 - Starts block immediately, if there is another block in the execution it will be aborted) (1 - When another block is in execution, the block in execution will continue its motion until the end and this new block will wait to be executed.) (6 – If another block is in execution, this block will go into error 52 and will not be executed. The HMI will show the alarm A00052.)
VAR_OUTPUT	InSync	BOOL	Output enabling
	Busy	BOOL	Flag indicating the block has not yet been ended
	Active	BOOL	Block flag with control on the axis
	CommandAborted	BOOL	Flag of aborted command
	Error	BOOL	Error in the execution flag
	ErrorID	WORD	Identifier of the occurred error
VAR	MC_GEARINPOS_INST_0	MC_GEARINPOS	Instance of access to block structure

Operation

When this block detects a leading edge on Execute, it sends a command for synchronism in position between the programmed axes.

For the slave axis to reach the speed of the master axis, a motion will be performed with an acceleration/deceleration configured in the “Acceleration” and “Deceleration” arguments. The motion direction will depend on the signal of the RatioNumerator. If RatioNumerator is greater than zero, the motion will be in the same direction as the master axis, and if the RatioNumerator is smaller than zero, the motion will be in the opposite the direction of the master axis.

When the MC_GearInPos block is executed, the drive will start operating in grid position and remain this way even after the conclusion of the block. The position proportional gain must be set (P0159) so as to obtain a better drive performance.

In the execution of the block, the axis status will change to Synchronized Motion.

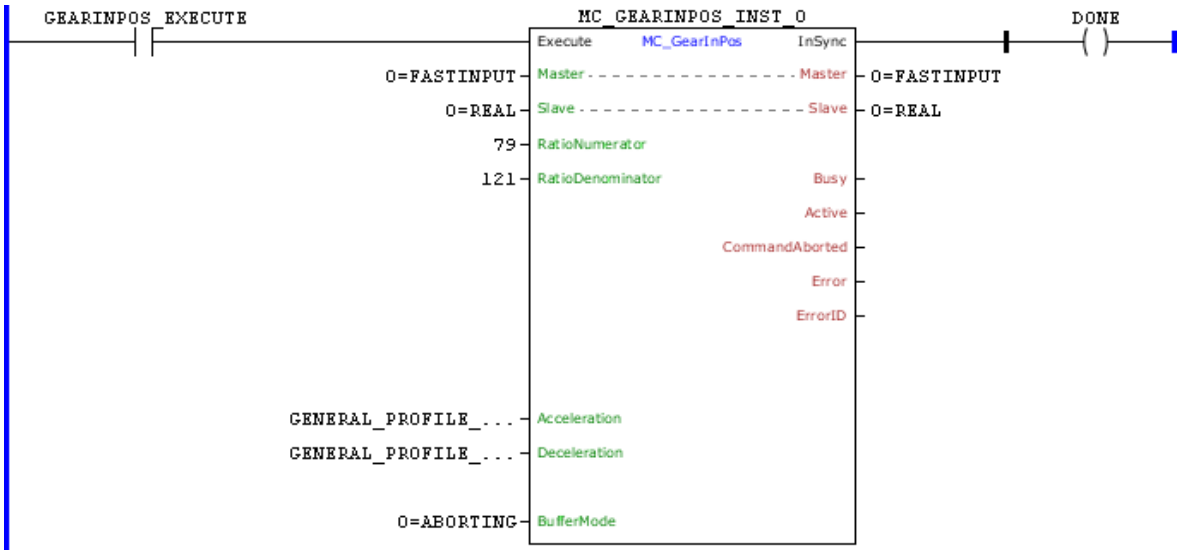
The InSync output is activated when the sync is achieved. It order to finish the block, it is necessary

to execute another block or the changing of the drive to the Disabled or Errorstop status.

If there is any error in the execution, the Error output is enabled and ErrorID displays an error code according to the table below.

Code	Description
52	Attempt to execute block w ith BufferMode in Single w hen another block is active.
62	Acceleration programmed below the minimum allow ed.
63	Acceleration programmed above the maximum allow ed.
64	Deceleration programmed below the minimum allow ed.
65	Deceleration programmed above the maximum allow ed.
67	Drive in the "Disabled" or “Errorstop” status.
69	Drive in the "Stopping" status.
70	Attempt to execute block w ith BufferMode in Buffered w hen another block is active and another block is w aiting.
71	P202 different from 4.
72	Invalid synchronism ratio.
74	Drive in the "Homing" status.
78	MC block not executed – Internal fault.

Example



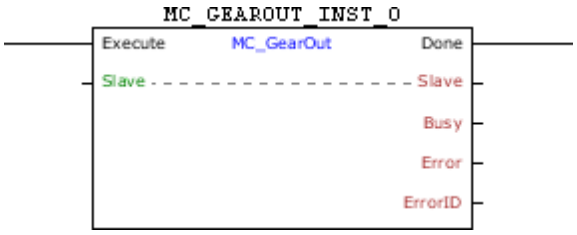
In the up transition of GEARINPOS_EXECUTE, the MC_GearInPos block is executed. With this the Busy Active and this block signals are set and the search of synchronization with the configured acceleration begins. As the ratio configured is 79:121 and the master axis is at 200 RPM, the slave axis must reach 131 RPM to establish the synchronism.

At the moment in which the speed reaches 131 RPM, the InSync output is set.

11.10.7.14.3.3 MC_GearOut

Block responsible for finalizing of the synchronism in position between the programmed axes.

Ladder Representation



Execution Features

Program Memory Size	28 Bytes
Data Memory Size	4 Bytes

Block Structure

Variable Type	Name	Data Type	Description
VAR_IN_OUT	Slave	BYTE	Selection of operation slave (0 - Real axis)
VAR_INPUT	Execute	BOOL	Block enabling
VAR_OUTPUT	Done	BOOL	Output enabling
	Busy	BOOL	Flag indicating the block has not yet been ended
	Error	BOOL	Error in the execution flag
	ErrorID	WORD	Identifier of the occurred error
VAR	MC_GEAROUT_INST_0	MC_GEAROUT	Instance of access to block structure

Operation

When this block detects a leading edge on Execute it concludes the synchronism of MC_GearIn MC_GearInPos or blocks the programmed axis. The axis will keep the speed of the moment in which the block is executed.

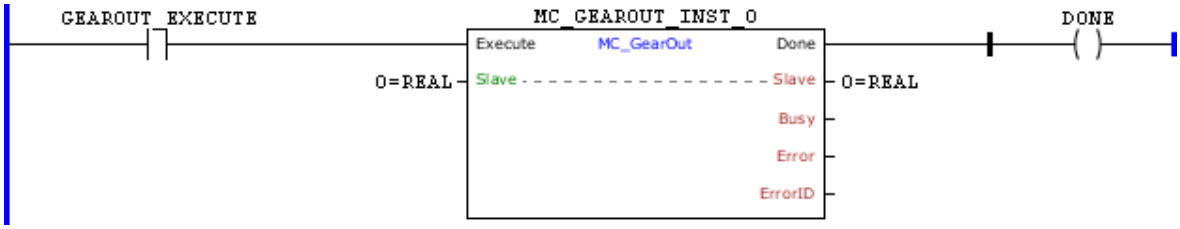
When the MC_GearOut block is executed, the drive does not operate in grid position. In the execution of the block the axis status will change to ContinuousMotion.

When Execute has FALSE value, Done remains FALSE. The Done output is activated when the block finishes the execution successfully, remaining at TRUE level until Execute receives FALSE.

If there is any error in the execution, the Error output is enabled and ErrorID displays an error code according to the table below.

Code	Description
67	Drive in the "Disabled" or “Errorstop” status.
71	P202 different from 4.
73	Drive is not in the “Synchronized Motion” status.
78	MC block not executed – Internal fault.

Example

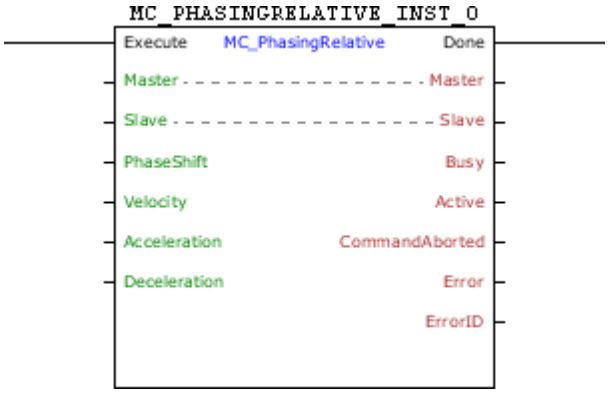


In the up transition of GEAROUT_EXECUTE, the MC_GearOut block is executed. With this, the Busy and Active signals of this block are set and synchronism concluded by other blocks of synchronism between master and slave. When the process finishes, the Done output of the block is set and remains TRUE while the Execute input is set.

11.10.7.14.3.4 MC_PhasingRelative

Block responsible for execution of a phase difference in position between the programmed axes.

Ladder Representation



Execution Features

Program Memory Size	78 Bytes
Data Memory Size	32 Bytes

Block Structure

Variable Type	Name	Data Type	Description
VAR_IN_OUT	Master	BYTE	Selection of operation master (0 - Fast digital inputs) (1 - CANopen) (2 - Encoder 1) (3 - Virtual Axis) (4 - Encoder 2)
	Slave	BYTE	Selection of operation slave (0 - Real axis)
VAR_INPUT	Execute	BOOL	Block enabling
	PhaseShift	REAL	Phase shift between master and slave
	Velocity	REAL	Speed
	Acceleration	REAL	Acceleration
	Deceleration	REAL	Deceleration
VAR_OUTPUT	Done	BOOL	Output enabling
	Busy	BOOL	Flag indicating the block has not yet been ended
	Active	BOOL	Block flag with control on the axis
	CommandAborted	BOOL	Flag of aborted command
	Error	BOOL	Error in the execution flag
	ErrorID	WORD	Identifier of the occurred error
VAR	MC_GEARIN_INST_0	MC_GEARIN	Instance of access to block structure

Operation

When this block detects a leading edge on Execute, it checks the synchronism between master and slave and sends a command to motion the master axis in order to let it out-of-phase of the slave axis in the magnitude of PhaseShift.

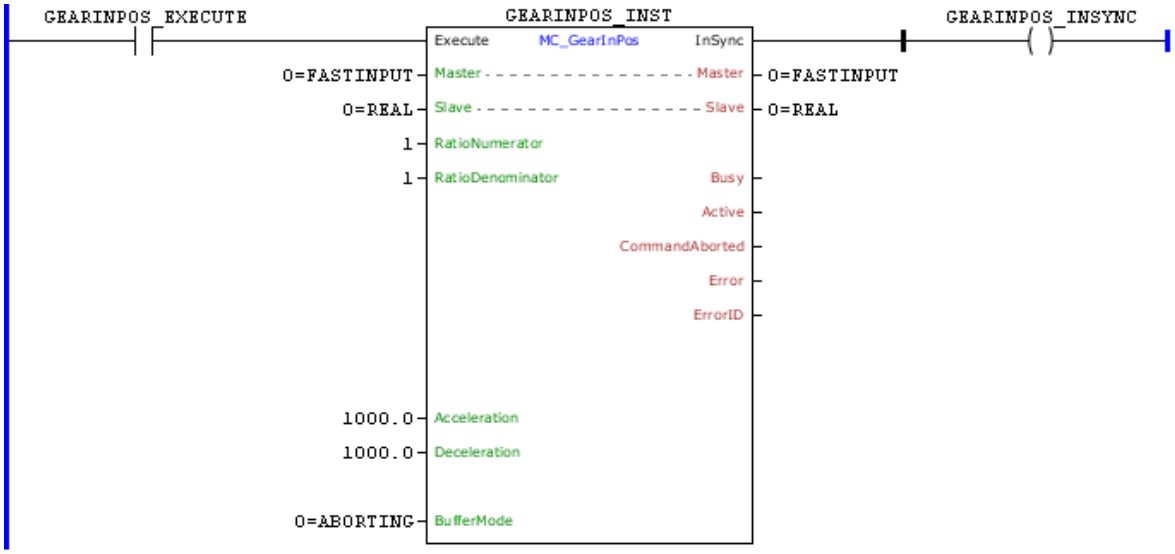
When the MC_PhasingRelative block is executed, the drive does not change the current operating mode. In the execution of the block, the axis status will not change.

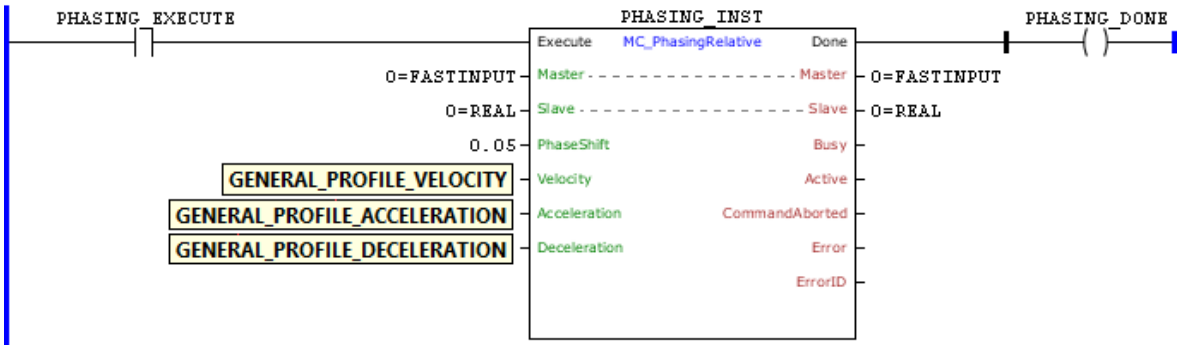
When Execute has FALSE value, Done remains FALSE. The Done output is activated when the block finishes the execution successfully, remaining at TRUE level until Execute receives FALSE.

If there is any error in the execution, the Error output is enabled and ErrorID displays an error code according to the table below.

Code	Description
67	Drive in the "Disabled" or “Errorstop” status.
71	P202 different from 4.
73	Drive is not in the “Synchronized Motion” status.
78	MC block not executed – Internal fault.
79	Master axis is not in synchronism.
96	MC_PhasingRelative block in execution.

Example

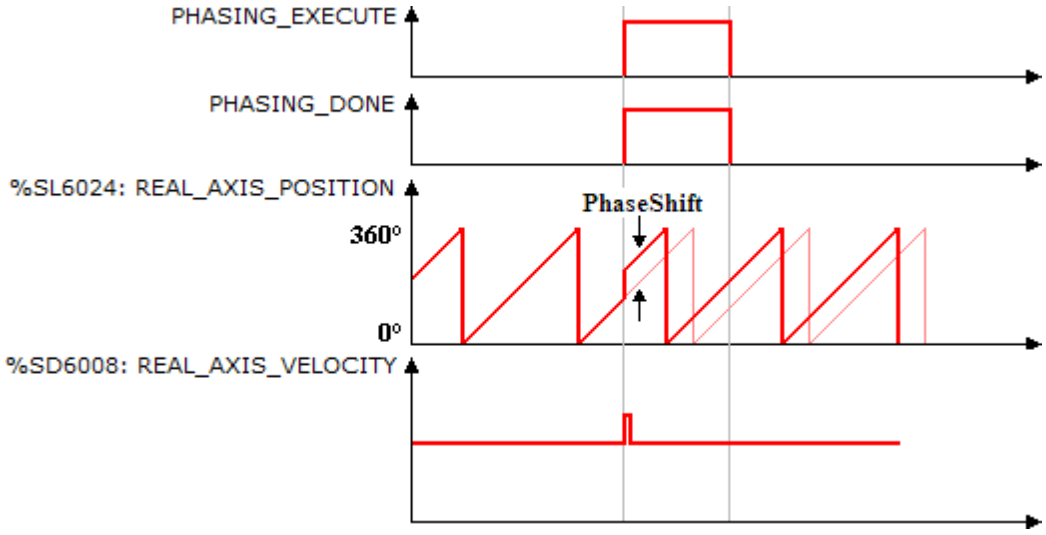




LOCAL | LOCAL_RETAIN

Tag	Tamanho	Tipo de Dado	Valor Inicial	Comentário
GEARINPOS_INST	0	MC_GearInPos		
GEARINPOS_EXECUTE	0	BOOL	0	
GEARINPOS_INSYNC	0	BOOL	0	
PHASING_INST	0	MC_PhasingRelative		
PHASING_EXECUTE	0	BOOL	0	
PHASING_DONE	0	BOOL	0	

GLOBAL | LOCAL



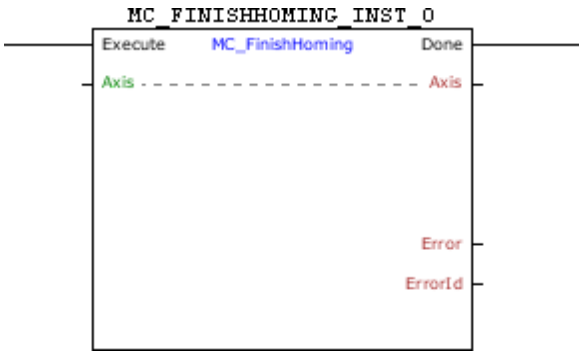
With the position synchronism of the Real Axis with the Quick Counter through the MC_GearInPos block, and with the occurrence of an up transition in PHASING_EXECUTE, the MC_PhasingRelative block is executed and a shift of 0.05 turn is applied to the master axis, resulting in a pulse in the speed. The Done output is set while the Execute input is set.

11.10.7.14.4 Motion Control Homing

11.10.7.14.4.1 MC_FinishHoming

Block responsible for changing the axis status from Homing to Standstill.

Ladder Representation



Execution Features

Program Memory Size	70 Bytes
Data Memory Size	8 Bytes

Block Structure

Variable Type	Name	Data Type	Description
VAR_IN_OUT	Axis	BYTE	Selection of operation axis (0 - Real axis)
VAR_INPUT	Execute	BOOL	Block enabling
VAR_OUTPUT	Done	BOOL	Output enabling
	Error	BOOL	Error in the execution flag
	ErrorID	WORD	Identifier of the occurred error
VAR	MC_HOMEDIRECT_INST_0	MC_HOMEDIRECT	Instance of access to block structure

Operation

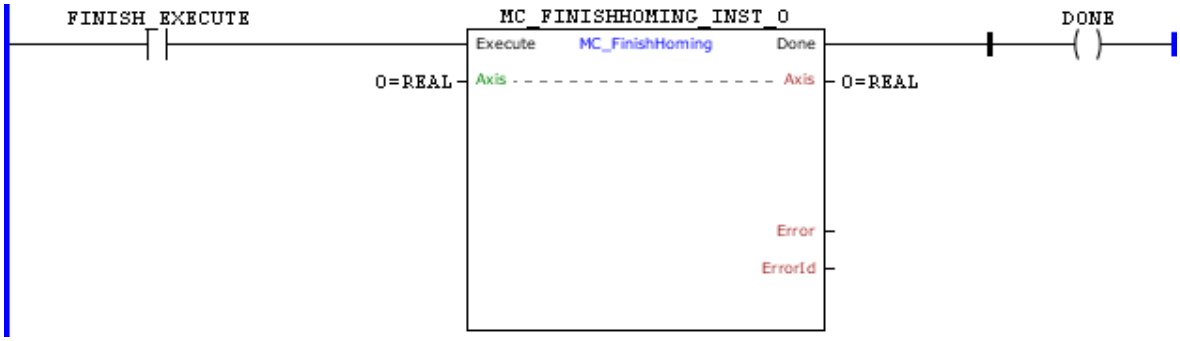
When this block detects a leading edge in Execute, if the axis status is Homing, the axis status changes to the Standstill.

When Execute has FALSE value, Done remains FALSE. The Done output is activated when the block finishes the execution successfully, remaining at TRUE level until Execute receives FALSE.

If there is any error in the execution, the Error output is enabled and ErrorID displays an error code according to the table below.

Code	Description
71	P202 different from 4.
75	Status of the Drive different from “Homing”.

Example

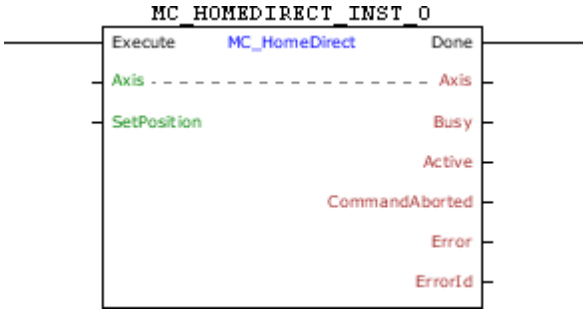


In the up transition of FINISH_EXECUTE, the MC_FinishHoming block is executed. With this, the Busy and Active signals of this block are set and the status of the selected axis changes to Standstill. When the process finishes, the Done output of the block is set and remains TRUE while the Execute input is set.

11.10.7.14.4.2 MC_HomeDirect

Block responsible for changing the reference position of the user.

Ladder Representation



Execution Features

Program Memory Size	46 Bytes
Data Memory Size	16 Bytes

Block Structure

Variable Type	Name	Data Type	Description
VAR_IN_OUT	Axis	BYTE	Selection of operation axis (0 - Real axis)
VAR_INPUT	Execute	BOOL	Block enabling
	SetPosition	LREAL	New reference position for user
VAR_OUTPUT	Done	BOOL	Output enabling
	Busy	BOOL	Flag indicating the block has not yet been ended
	Active	BOOL	Block flag with control on the axis
	CommandAborted	BOOL	Flag of aborted command
	Error	BOOL	Error in the execution flag
	ErrorID	WORD	Identifier of the occurred error
VAR	MC_HOMEDIRECT_INST_0	MC_HOMEDIRECT	Instance of access to block structure

Operation

When this block detects a leading edge on Execute, it makes the reference position of the user (P0051, P0052 and P0053) to be changed to the value of the SetPosition argument.

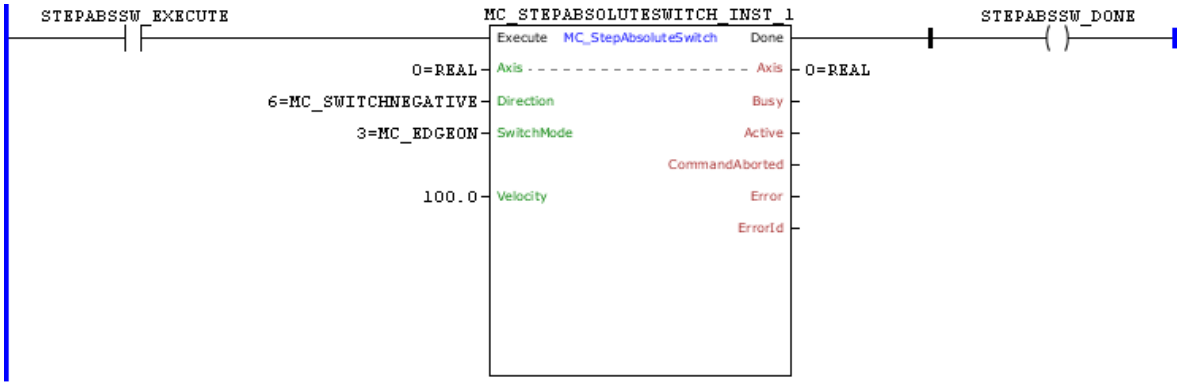
In the execution of the block, if the axis status is Homing, the axis status changes to the Standstill. Otherwise, it will remain in its current status.

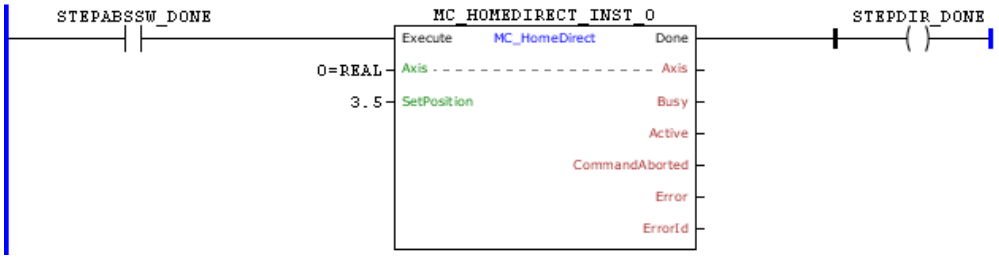
When Execute has FALSE value, Done remains FALSE. The Done output is activated when the block finishes the execution successfully, remaining at TRUE level until Execute receives FALSE.

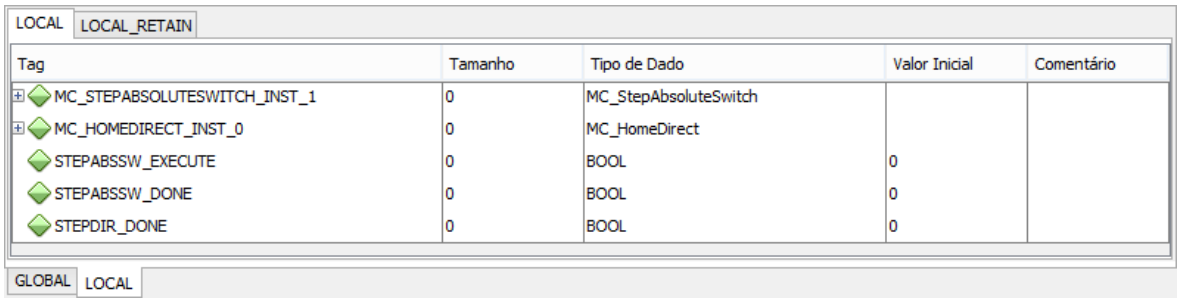
If there is any error in the execution, the Error output is enabled and ErrorID displays an error code according to the table below.

Code	Description
71	P202 different from 4.
76	Status of the Drive different from “Standstill” or “Homing”.

Example



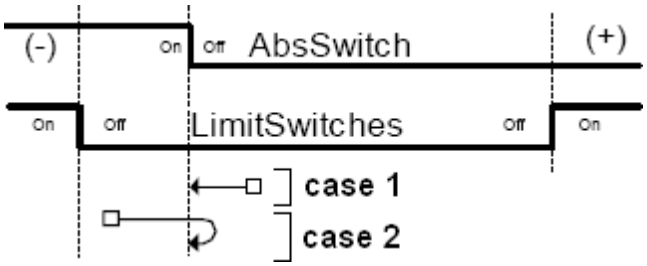




LOCAL | LOCAL_RETAIN

Tag	Tamanho	Tipo de Dado	Valor Inicial	Comentário
MC_STEPABSOLUTESWITCH_INST_1	0	MC_StepAbsoluteSwitch		
MC_HOMEDIRECT_INST_0	0	MC_HomeDirect		
STEPABSSW_EXECUTE	0	BOOL	0	
STEPABSSW_DONE	0	BOOL	0	
STEPDIR_DONE	0	BOOL	0	

GLOBAL | LOCAL



In the up transition of STEPABSSW_EXECUTE, the MC_StepAbsoluteSwitch block is executed and the search for the AbsoluteSwitch starts. The axis status is changed to Homing.

In case 1, when executing the block, the AbsoluteSwitch is not activated. Since the Direction argument is set to MC_SwitchNegative, the motion is in the negative direction. When a leading edge occurs in AbsoluteSwitch (SwitchMode = MC_EdgeOn), the motor stops and revolutions to the position in which the edge occurred.

In case 2, when executing the block, the AbsoluteSwitch is activated. Since the argument Direction is set to MC_SwitchPositive, the motion is in the positive direction, when exiting the AbsoluteSwitch, the motor stops and changes to motion in the negative direction. When a leading edge occurs in AbsoluteSwitch (SwitchMode = MC_EdgeOn), the motor stops and revolutions to the position in which the edge occurred.

All the motions are performed with an acceleration/deceleration programmed in P0100 and P0101.

When returning to the leading edge position of AbsoluteSwitch, the Done output of the block is set and remains TRUE while the Execute input is set.

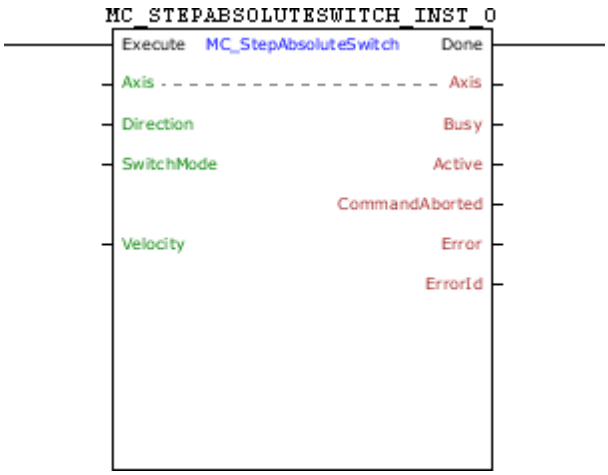
In the up transition of STEPABSSW_DONE, the MC_StepDirect block is executed and the user's reference position (P0051, P0052 and P0053) is changed to 3.5 revolutions (P0051 = 8192, P0052 = 3 and P0053 = 0). The axis status is changed to Standstill.

When STEPABSSW_EXECUTE is reset, STEPABSSW_DONE and STEPDIR_DONE are also reset.

11.10.7.14.4.3 MC_StepAbsoluteSwitch

Block responsible for searching the position of AbsoluteSwitch.

Ladder Representation



Execution Features

Program Memory Size	74 Bytes
Data Memory Size	40 Bytes

Block Structure

Variable Type	Name	Data Type	Description
VAR_IN_OUT	Axis	BYTE	Selection of operation axis (0 - Real axis)
VAR_INPUT	Execute	BOOL	Block enabling
	Direction	BYTE	Search direction (0 – Positive direction) (1 – Negative direction) (5 - MC_SwitchPositive: Positive direction, if AbsoluteSwitch not activated) (6 - MC_SwitchNegative: Negative direction, if AbsoluteSwitch not activated)
	SwitchMode	BYTE	Search mode (3 - MC_EdgeOn) (4 - MC_EdgeOff)
	Velocity	REAL	Speed
VAR_OUTPUT	Done	BOOL	Output enabling
	Busy	BOOL	Flag indicating the block has not yet been ended
	Active	BOOL	Block flag with control on the axis
	CommandAborted	BOOL	Flag of aborted command
	Error	BOOL	Error in the execution flag
	ErrorID	WORD	Identifier of the occurred error
VAR	MC_STEPABSOLUTESWITCH_INST_0	MC_STEPABSOLUTESWITCH	Instance of access to block structure

Operation

When this block detects a leading edge in Execute, it sends a command to search the position of AbsoluteSwitch.

The AbsoluteSwitch can only be connected to digital inputs 1, 2 or 3, seeing that the programmed function of the digital input must be in accordance with the SwitchMode argument. If SwitchMode is configured as MC_EdgeOn (leading edge), the function of the digital input (P0300, P0301 or P0302) must be “store position - leading edge" (option 8). If the SwitchMode is configured as MC_EdgeOff (falling edge), the function of the digital input (P0300, P0301 or P0302) must be “store position - falling edge” (option 9). AbsoluteSwitch will be considered the first digital input configured according to SwitchMode from digital input 1. If no digital inputs are configured according to SwitchMode, error 77 will occur in the block and it will not be executed.

If when searching the AbsoluteSwitch position, the LimitSwitch position is reached, the motion will change direction up to the AbsoluteSwitch position.

The search will be executed with the speed configured in the “Velocity” argument and an acceleration/ deceleration configured in the General Profile.

With the execution of the MC_StepAbsoluteSwitch block, the user’s reference position (P0051, P0052 and P0053) is not changed. The drive will start operating in grid position and remain this way even after the conclusion of the block. The position proportional gain must be set (P0159) so as to obtain a better drive performance.

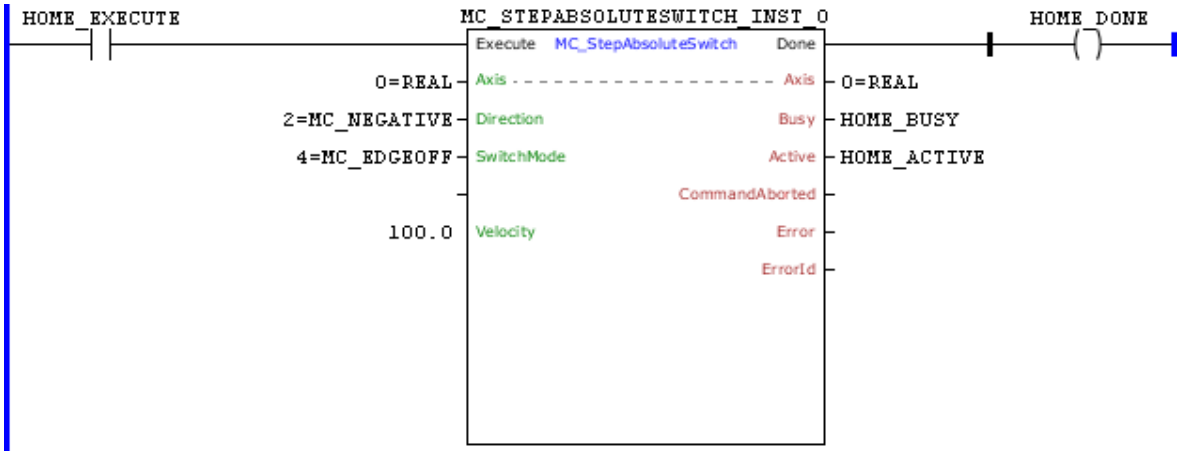
In the execution of the block, the will change to Homing and will remain this way until the execution of the MC_StepRefPulse, MC_StepDirect or MC_FinishHoming blocks.

When Execute has FALSE value, Done remains FALSE. The Done output is activated when the block finishes the execution successfully, remaining at TRUE level until Execute receives FALSE.

If there is any error in the execution, the Error output is enabled and ErrorID displays an error code according to the table below.

Code	Description
60	Speed programmed below the minimum allowed.
61	Speed programmed above the maximum allowed.
67	Drive in the "Disabled" or “Errorstop” status.
69	Drive in the "Stopping" status.
71	P202 different from 4.
76	Status of the Drive different from “Standstill” or “Homing”.
77	Digital inputs 1, 2 and 3 not configured according to “SiwitchMode”.

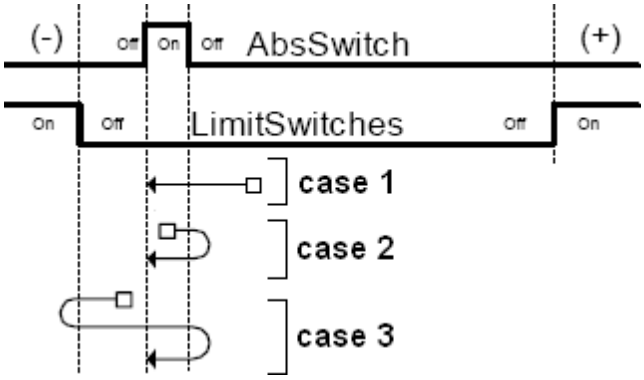
Example

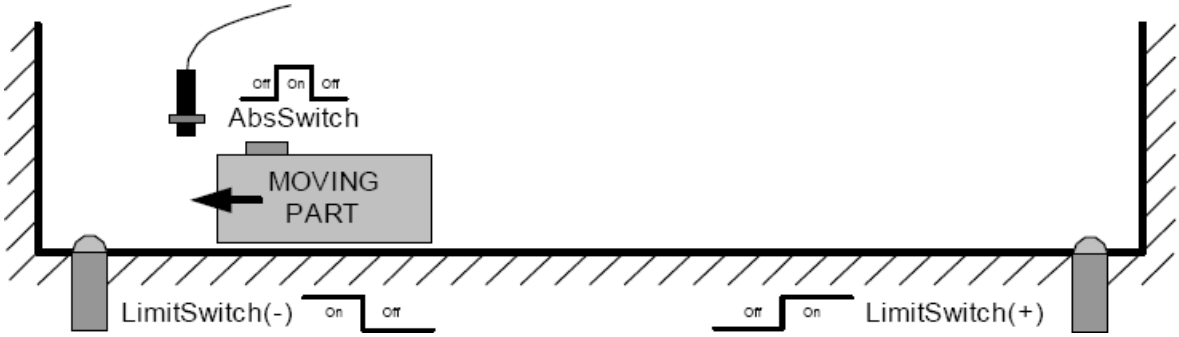


LOCAL | LOCAL_RETAIN

Tag	Tamanho	Tipo de Dado	Valor Inicial	Comentário
MC_STEPABSOLUTESWITCH_INST_0	0	MC_StepAbsoluteSwitch		
HOME_EXECUTE	0	BOOL	0	
HOME_BUSY	0	BOOL	0	
HOME_ACTIVE	0	BOOL	0	
HOME_DONE	0	BOOL	0	

GLOBAL | LOCAL





In the up transition of HOME_EXECUTE the MC_StepAbsoluteSwitch block is executed. With this, the Busy and Active signals of this block are set and the search for AbsoluteSwitch begins.

In case 1, when executing the block, the AbsoluteSwitch is not activated. Since the Direction argument is set to MC_SwitchNegative, the motion is in the negative direction. When a falling edge in AbsoluteSwitch (SwitchMode = MC_EdgeOff) occurs, the motor stops and revolutions to the position in which the edge occurred.

In case 2, when executing the block, the AbsoluteSwitch is activated. Since the argument Direction is set to MC_SwitchPositive, the motion is in the positive direction, when exiting the AbsoluteSwitch, the motor stops and changes to motion in the negative direction. When a falling edge in AbsoluteSwitch (SwitchMode = MC_EdgeOff) occurs, the motor stops and revolutions to the position in which the edge occurred.

In case 3, when executing the block the AbsoluteSwitch is not activated. Since the Direction argument is set to MC_SwitchNegative, the motion is in the negative direction. But when the LimitSwitch is found, the motor stops and changes the motion to the positive direction. When leaving the AbsoluteSwitch, the motor stops again and changes the motion to the negative direction. When a falling edge in AbsoluteSwitch (SwitchMode = MC_EdgeOff) occurs, the motor stops and revolutions to the position in which the edge occurred.

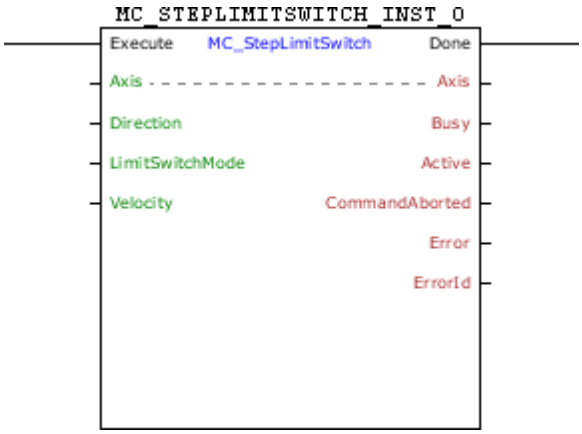
All the motions are performed with an acceleration/deceleration programmed in P0100 e P0101, except when the LimitSwitch is found, case in which the motor stops instantly.

When returning to the falling edge position of the AbsoluteSwitch, the Done output of the block is set and the Busy and Active signals of this block are reset. The Done output remains TRUE while the Execute input is set.

11.10.7.14.4.4 MC_StepLimitSwitch

Block responsible for searching the position of LimitSwitch.

Ladder Representation



Execution Features

Program Memory Size	72 Bytes
Data Memory Size	40 Bytes

Block Structure

Variable Type	Name	Data Type	Description
VAR_IN_OUT	Axis	BYTE	Selection of operation axis (0 - Real axis)
VAR_INPUT	Execute	BOOL	Block enabling
	Direction	BYTE	Search direction (0 – Positive direction) (1 – Negative direction)
	LimitSwitchMode	BYTE	Search mode (3 - MC_EdgeOn) (4 - MC_EdgeOff)
	Velocity	REAL	Speed
VAR_OUTPUT	Done	BOOL	Output enabling
	Busy	BOOL	Flag indicating the block has not yet been ended
	Active	BOOL	Block flag w ith control on the axis
	CommandAborted	BOOL	Flag of aborted command
	Error	BOOL	Error in the execution flag
	ErrorID	WORD	Identifier of the occurred error
VAR	MC_STEPLIMITSWITCH_INST_0	MC_STEPLIMITSWITCH	Instance of access to block structure

Operation

When this block detects a rising edge in Execute, it sends a command to search the position of LimitSwitch.

The LimitSwitch can only be connected to digital inputs 1, 2 or 3, seeing that the programmed function of the digital input must be in accordance with the "LimitSwitchMode" argument and the “Direction” argument. It will be considered LimitSwitch the first digital input configured according to the table, from digital input 1. If no digital inputs are configured according to LimitSwitchMode and Direction, error 77 in the block will occur and it will not be executed:

Direction	Limit Switch Mode	Digital Input Function
MC_Positive	MC_EdgeOn	Limit switch clockwise active high (option 12)
MC_Positive	MC_EdgeOff	Limit switch clockwise active low (option 13)
MC_Negative	MC_EdgeOn	Limit switch counterclockwise active high (option 14)
MC_Negative	MC_EdgeOff	Limit switch counterclockwise active low (option 15)

The search will be executed with the speed configured in the “Velocity” argument and an acceleration/ deceleration configured in the General Profile.

With the execution of the MC_StepLimitSwitch block, the user’s reference position (P0051, P0052 and P0053) is not changed. The drive will start operating in grid position and remain this way even after the conclusion of the block. The position proportional gain must be set (P0159) so as to obtain a better drive performance.

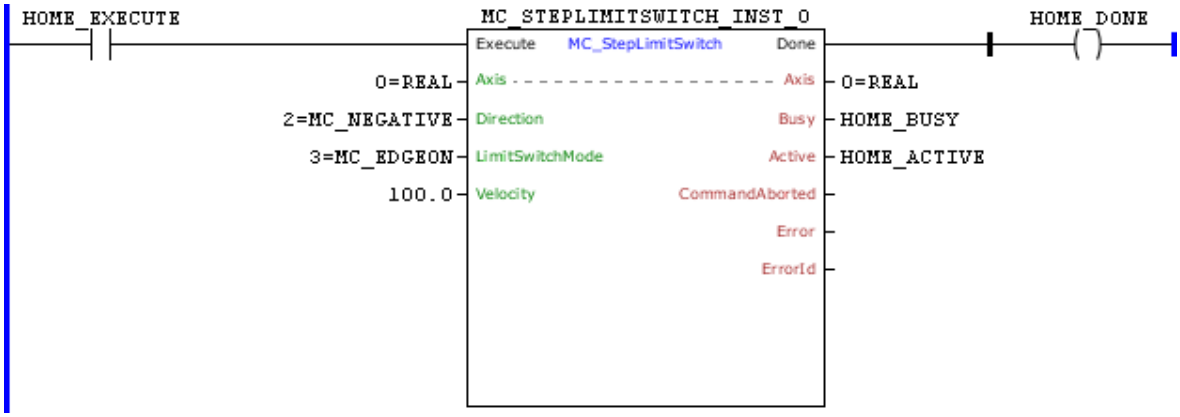
In the execution of the block, the will change to Homing and will remain this way until the execution of the MC_StepRefPulse, MC_StepDirect or MC_FinishHoming blocks.

When Execute has FALSE value, Done remains FALSE. The Done output is activated when the block finishes the execution successfully, remaining at TRUE level until Execute receives FALSE.

If there is any error in the execution, the Error output is enabled and ErrorID displays an error code according to the table below.

Code	Description
60	Speed programmed below the minimum allowed.
61	Speed programmed above the maximum allowed.
67	Drive in the "Disabled" or “Errorstop” status.
69	Drive in the "Stopping" status.
71	P202 different from 4.
76	Status of the Drive different from “Standstill” or “Homing”.
77	Digital inputs 1, 2 and 3 not configured according to “SiwitchMode”.
97	Position feedback not allowed. Please check P290 and P360.

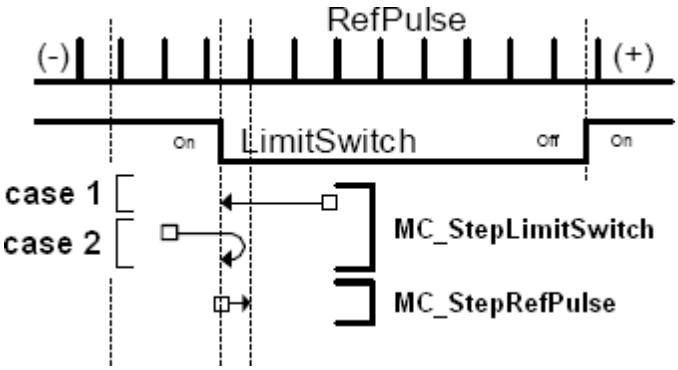
Example



LOCAL | LOCAL_RETAIN

Tag	Tamanho	Tipo de Dado	Valor Inicial	Comentário
MC_STEPLIMITSWITCH_INST_0	0	MC_StepLimitSwitch		
HOME_EXECUTE	0	BOOL	0	
HOME_BUSY	0	BOOL	0	
HOME_ACTIVE	0	BOOL	0	
HOME_DONE	0	BOOL	0	

GLOBAL | LOCAL



In the up transition of HOME_EXECUTE the MC_StepLimitSwitch block is executed. With this, the Busy and Active signals of this block are set and the search for LimitSwitch begins.

In case 1, when executing the block, the LimitSwitch is not activated. Since the Direction argument is set to MC_Negative, the motion is in the negative direction. When the leading edge in LimitSwitch (LimitSwitchMode = MC_EdgeOn) occurs, the motor stops and revolutions to the position in which the edge occurred.

In case 2, when executing the block, the LimitSwitch is activated. Even with the "Direction" argument configured as MC_Negative, the motion will be in the positive direction and, when leaving the LimitSwitch, the motor stops and changes the motion to the negative direction. When the leading edge in LimitSwitch (SwitchMode = MC_EdgeOn) occurs, the motor stops and revolutions to the position in which the edge occurred.

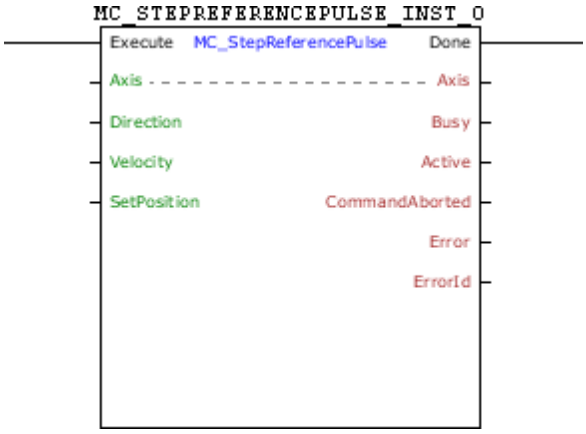
All the motions are performed with an acceleration/deceleration programmed in P0100 e P0101, except when the LimitSwitch is found, case in which the motor stops instantly.

When returning to the leading edge position of the LimitSwitch, the Done output of the block is set and the Busy and Active signals of this block are reset. The Done output remains TRUE while the Execute input is set.

11.10.7.14.4.5 MC_StepReferencePulse

Block responsible for searching the position of the zero pulse.

Ladder Representation



Execution Features

Program Memory Size	72 Bytes
Data Memory Size	40 Bytes

Block Structure

Variable Type	Name	Data Type	Description
VAR_IN_OUT	Axis	BYTE	Selection of operation axis (0 - Real axis)
VAR_INPUT	Execute	BOOL	Block enabling
	Direction	BYTE	Search direction (0 – Positive direction) (1 – Negative direction) (5 - MC_SwitchPositive: Positive direction, if AbsoluteSwitch not activated) (6 - MC_SwitchNegative: Negative direction, if AbsoluteSwitch not activated)
	Velocity	REAL	Speed
	SetPosition	LREAL	New reference position for user
VAR_OUTPUT	Done	BOOL	Output enabling
	Busy	BOOL	Flag indicating the block has not yet been ended
	Active	BOOL	Block flag with control on the axis
	CommandAborted	BOOL	Flag of aborted command
	Error	BOOL	Error in the execution flag
	ErrorID	WORD	Identifier of the occurred error
VAR	MC_STEPREFERENCEPULSE_INST_0	MC_STEPREFERENCEPULSE	Instance of access to block structure

Operation

When this block detects a rising edge in Execute, it sends a command to search the position of zero pulse.

The search will be executed with the speed configured in the “Velocity” argument and an acceleration/ deceleration configured in the General Profile.

With the execution of MC_StepReferencePulse block, the reference position of the user (P0051, P0052 and P0053) is changed to the value of the SetPosition argument. The drive will start operating in grid position and remain this way even after the conclusion of the block. The position proportional gain must be set (P0159) so as to obtain a better drive performance.

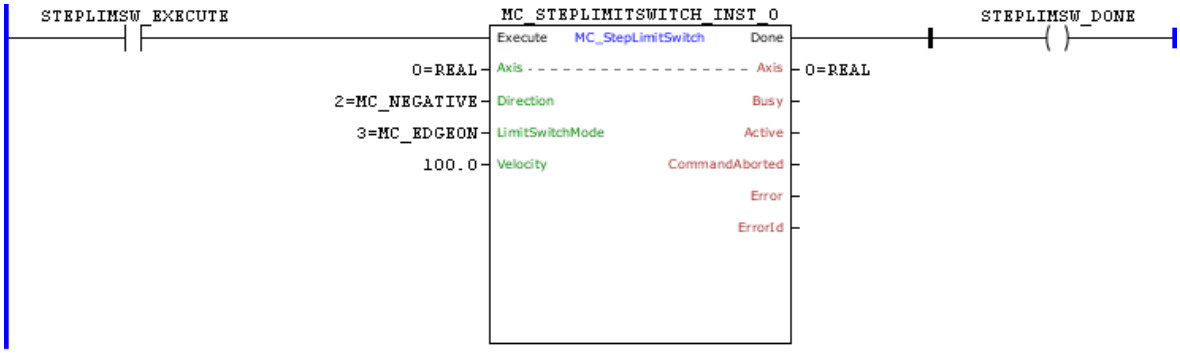
In the execution of the block, the axis status will change to Homing. When concluding the search, the axis status will change to Standstill.

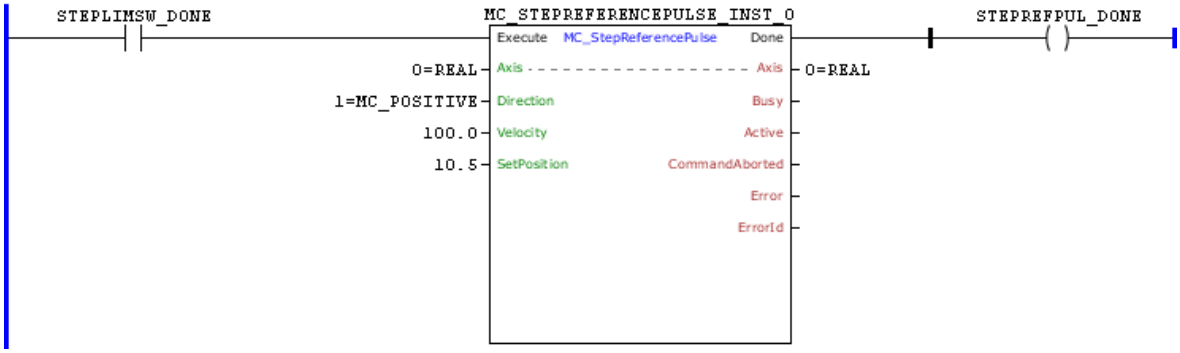
When Execute has FALSE value, Done remains FALSE. The Done output is activated when the block finishes the execution successfully, remaining at TRUE level until Execute receives FALSE.

If there is any error in the execution, the Error output is enabled and ErrorID displays an error code according to the table below.

Code	Description
60	Speed programmed below the minimum allowed.
61	Speed programmed above the maximum allowed.
67	Drive in the "Disabled" or “Errorstop” status.
69	Drive in the "Stopping" status.
71	P202 different from 4.
76	Status of the Drive different from “Standstill” or “Homing”.

Example

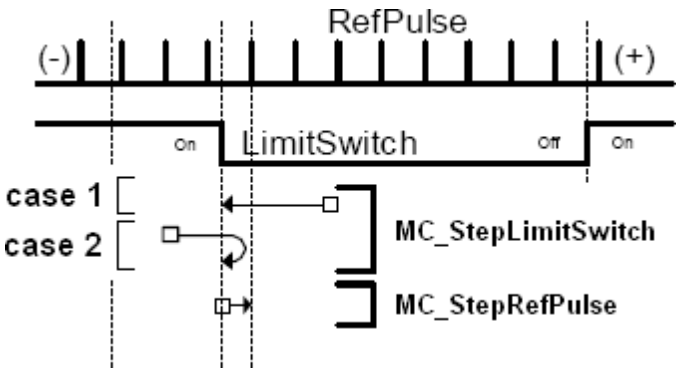




LOCAL | LOCAL_RETAIN

Tag	Tamanho	Tipo de Dado	Valor Inicial	Comentário
MC_STEPLIMITSWITCH_INST_0	0	MC_StepLimitSwitch		
STEPLIMSW_EXECUTE	0	BOOL	0	
STEPLIMSW_DONE	0	BOOL	0	
MC_STEPREFERENCEPULSE_INST_0	0	MC_StepReferencePulse		
STEPREFPUL_DONE	0	BOOL	0	

GLOBAL | LOCAL



In the up transition of STEPLIMSW_EXECUTE the MC_StepLimitSwitch block is executed. With this, the Busy and Active signals of this block are set and the search for LimitSwitch begins.

In case 1, when executing the block, the LimitSwitch is not activated. Since the Direction argument is set to MC_Negative, the motion is in the negative direction. When the leading edge in LimitSwitch (LimitSwitchMode = MC_EdgeOn) occurs, the motor stops and revolutions to the position in which the edge occurred.

In case 2, when executing the block, the LimitSwitch is activated. Even with the "Direction" argument configured as MC_Negative, the motion will be in the positive direction and, when leaving the LimitSwitch, the motor stops and changes the motion to the negative direction. When the leading edge in LimitSwitch (SwitchMode = MC_EdgeOn) occurs, the motor stops and revolutions to the position in which the edge occurred.

All the motions are performed with an acceleration/deceleration programmed in P0100 e P0101, except when the LimitSwitch is found, case in which the motor stops instantly.

When returning to the leading edge position of the LimitSwitch, the Done output of the block is set and the Busy and Active signals of this block are reset. The Done output remains TRUE while the Execute input is set.

In the up transition of STEPLIMSW_DONE, the MC_SteoRefPulse block is executed and the search of the null pulse starts.

The motion will be in the positive direction and when the null pulse is found, the motor stops and revolutions to the null pulse position.

All the motions are performed with an acceleration/deceleration programmed in P0100 and P0101.

When returning to the null pulse position, the Done output of the block is set and remains in TRUE while the Execute input is set. The user's reference position (P0051, P0052 and P0053) is changed to 10.5 revolutions (P0051 = 8192, P0052 = 10 and P0053 = 10).

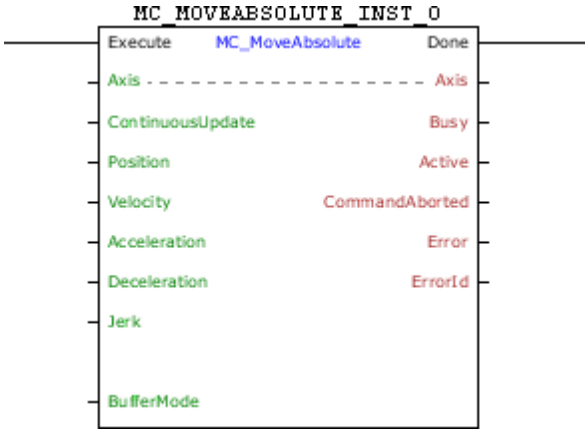
When STEPLIMSW_EXECUTE is reset, STEPLIMSW_DONE and STEPREFPUL_DONE are also reset.

11.10.7.14.5 Motion Control Move

11.10.7.14.5.1 MC_MoveAbsolute

Block responsible for performing one positioning to the absolute position programmed.

Ladder Representation



Execution Features

Program Memory Size	72 Bytes
Data Memory Size	32 Bytes

Block Structure

Variable Type	Name	Data Type	Description
VAR_IN_OUT	Axis	BYTE	Selection of operation axis (0 - Real axis)
VAR_INPUT	Execute	BOOL	Block enabling
	ContinuousUpdate	BYTE	Automatic update mode (0 - Rising edge) (1 - Real time)
	Position	LREAL	Position
	Velocity	REAL	Speed
	Acceleration	REAL	Acceleration
	Deceleration	REAL	Deceleration
	Jerk	REAL	Jerk
	BufferMode	REAL	Execution start mode (0 - Starts block immediately, if there is another block in the execution it will be aborted) (1 - When another block is in execution, the block in execution will continue its motion until the end and this new block will wait to be executed.) (6 – If another block is in execution, this block will go into error 52 and will not be executed. The HMI will show the alarm A00052.)
VAR_OUTPUT	Done	BOOL	Output enabling
	Busy	BOOL	Flag indicating the block has not yet been ended
	Active	BOOL	Block flag with control on the axis
	CommandAborted	BOOL	Flag of aborted command
	Error	BOOL	Error in the execution flag
	ErrorID	WORD	Identifier of the occurred error
VAR	MC_MOVEABSOLUTE_INST_0	MC_MOVEABSOLUTE	Instance of access to block structure

Operation

When this block detects a leading edge on Execute, it will perform a positioning into the absolute position configured in the Position argument with a maximum speed configured in the Velocity argument and acceleration/deceleration set in Acceleration and Deceleration arguments. Depending on the distance of the positioning and the acceleration and deceleration values, the motor speed will not reach the maximum configured speed.

NOTE!
If the Jerk argument value is not zero:
1) the value of the deceleration will be the same as the configured value in the acceleration;
2) the ContinuousUpdate On-line argument will have no effect and the argument values are considered at the time of positive transition Execute;
3) it is not allowed to perform positioning with another active block, occurring ErrorID 95.

When the MC_MotionAbsolute block is executed, the drive will start operating in grid position and

remain this way even after the conclusion of the block. The position proportional gain must be set (P0159) so as to obtain a better drive performance.

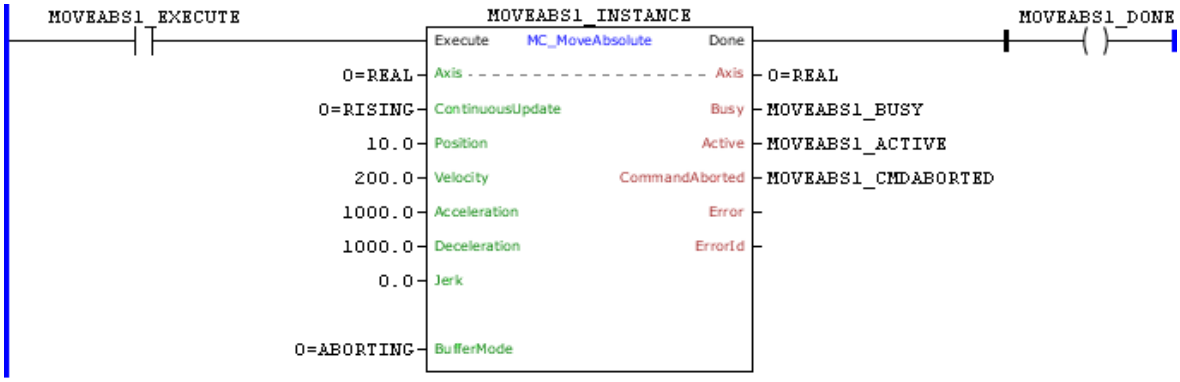
In the execution of the positioning, the axis status will change to Discrete Motion. When the positioning is concluded, the axis status will change to Standstill.

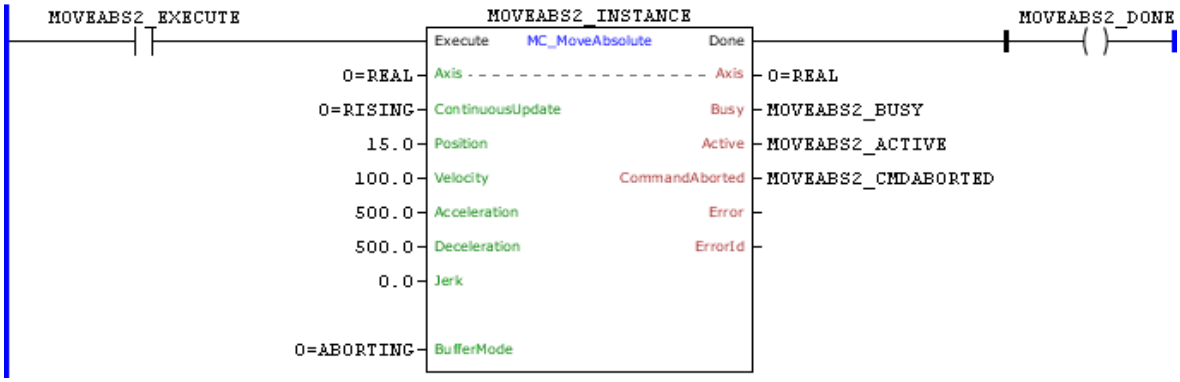
When Execute has FALSE value, Done remains FALSE. The Done output is activated when the block finishes the execution successfully, remaining at TRUE level until Execute receives FALSE.

If there is any error in the execution, the Error output is enabled and ErrorID displays an error code according to the table below.

Code	Description
52	Attempt to execute block w ith BufferMode in Single w hen another block is active.
60	Speed programmed below the minimum allow ed.
61	Speed programmed above the maximum allow ed.
62	Acceleration programmed below the minimum allow ed.
63	Acceleration programmed above the maximum allow ed.
64	Deceleration programmed below the minimum allow ed.
65	Deceleration programmed above the maximum allow ed.
67	Drive in the "Disabled" or “Errorstop” status.
69	Drive in the "Stopping" status.
70	Attempt to execute block w ith BufferMode in Buffered w hen another block is active and another block is w aiting.
71	P202 different from 4.
74	Drive in the "Homing" status.
78	MC block not executed – Internal fault.
93	Jerk programmed below the minimum allow ed.
94	Jerk programmed above the maximum allow ed.
95	It is not allow ed to execute positioning w ith Jerk w hen another block is active.

Example

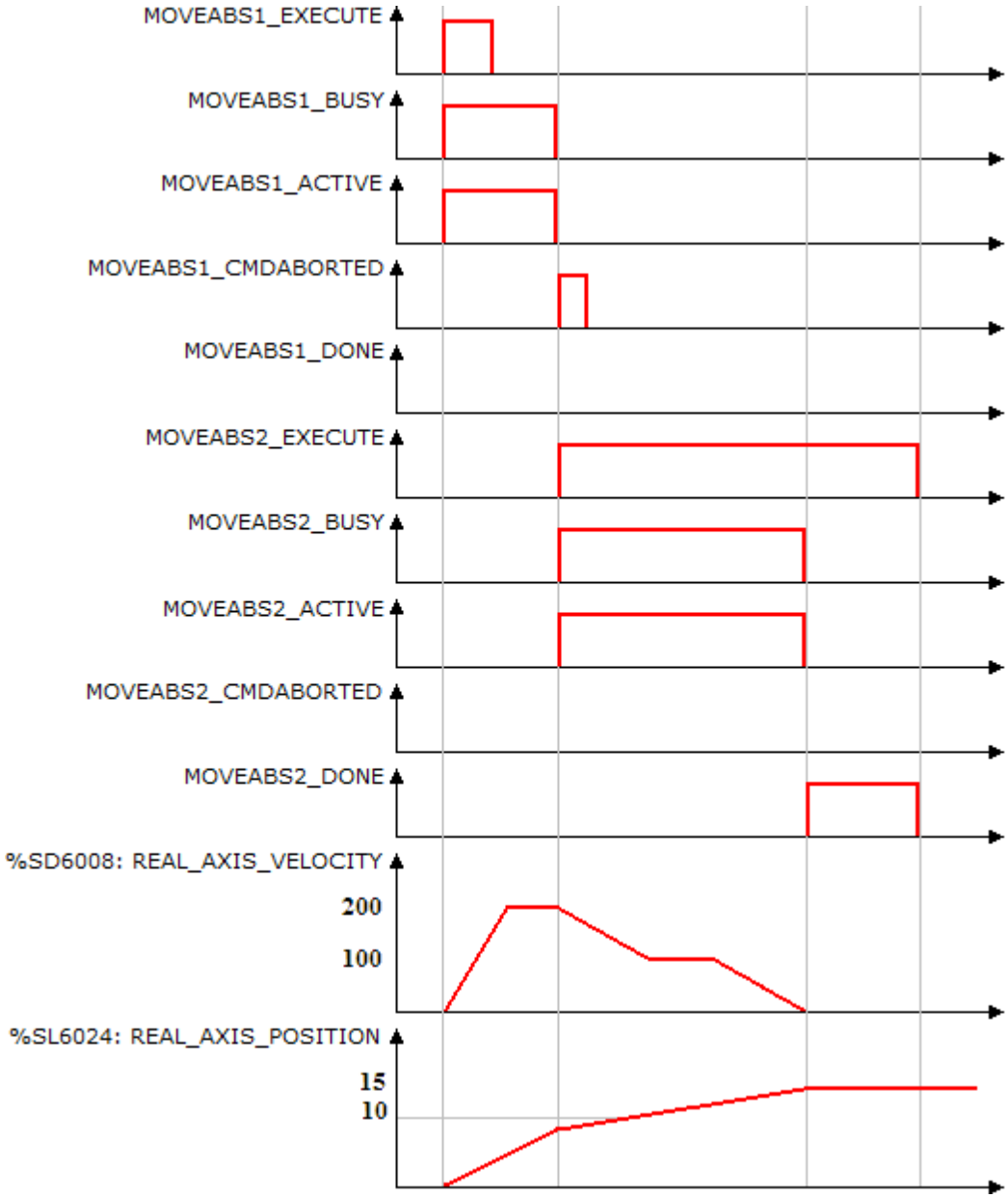




LOCAL | LOCAL_RETAIN

Tag	Tamanho	Tipo de Dado	Valor Inicial	Comentário
MOVEABS1_INSTANCE	0	MC_MoveAbsolute		
MOVEABS2_INSTANCE	0	MC_MoveAbsolute		
MOVEABS1_EXECUTE	0	BOOL	0	
MOVEABS1_DONE	0	BOOL	0	
MOVEABS1_BUSY	0	BOOL	0	
MOVEABS1_ACTIVE	0	BOOL	0	
MOVEABS1_CMDABORTED	0	BOOL	0	
MOVEABS2_EXECUTE	0	BOOL	0	
MOVEABS2_DONE	0	BOOL	0	
MOVEABS2_BUSY	0	BOOL	0	
MOVEABS2_ACTIVE	0	BOOL	0	
MOVEABS2_CMDABORTED	0	BOOL	0	

GLOBAL | LOCAL



In the up transition of MOTIONABS1_EXECUTE, the first MC_MotionAbsolute block is executed. With this, the Busy and Active signals of this block are set and positioning to absolute position "revolutions 10" starts.

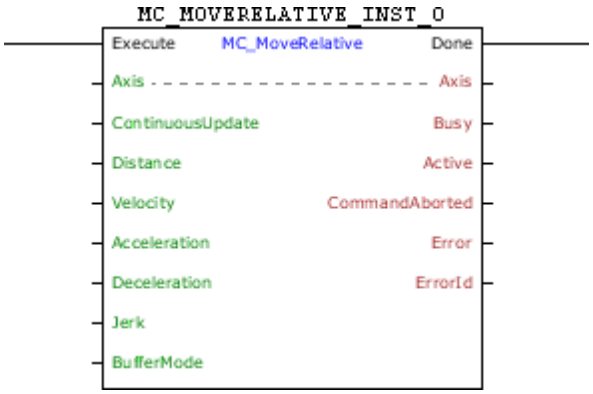
With the up transition of MOTIONABS2_EXECUTE, the second MC_GearAbsolute block is instantly executed (BufferMode - Aborting). With this, the Busy and Active signals of this block are set and positioning to absolute position 15 revolutions starts. At the same time the Busy and Active signals of the first block are reset and the CommandAborted signal is set for 1 scan cycle.

When the position 15 revolutions is reached, the Done output of the second block is set and the Busy and Active signals of this block are reset. The Done output remains TRUE while the Execute input is set.

11.10.7.14.5.2 MC_MoveRelative

Block responsible for performing one positioning to the relative position programmed.

Ladder Representation



Execution Features

Program Memory Size	72 Bytes
Data Memory Size	32 Bytes

Block Structure

Variable Type	Name	Data Type	Description
VAR_IN_OUT	Axis	BYTE	Selection of operation axis (0 - Real axis)
VAR_INPUT	Execute	BOOL	Block enabling
	ContinuousUpdate	BYTE	Automatic update mode (0 - Rising edge) (1 - Real time)
	Distance	LREAL	Shift distance
	Velocity	REAL	Speed
	Acceleration	REAL	Acceleration
	Deceleration	REAL	Deceleration
	Jerk	REAL	Jerk
	BufferMode	REAL	Execution start mode (0 - Starts block immediately, if there is another block in the execution it will be aborted) (1 - When another block is in execution, the block in execution will continue its motion until the end and this new block will wait to be executed.) (6 – If another block is in execution, this block will go into error 52 and will not be executed. The HMI will show the alarm A00052.)
VAR_OUTPUT	Done	BOOL	Output enabling
	Busy	BOOL	Flag indicating the block has not yet been ended
	Active	BOOL	Block flag with control on the axis
	CommandAborted	BOOL	Flag of aborted command
	Error	BOOL	Error in the execution flag
	ErrorID	WORD	Identifier of the occurred error
VAR	MC_MOVERELATIVE_INST_0	MC_MOVERELATIVE	Instance of access to block structure

Operation

When this block detects a leading edge in Execute, it will perform a positioning according to the relative distance to the current position, configured in the Distance argument, with a maximum speed configured in the Velocity argument and acceleration/deceleration configured in Acceleration and Deceleration arguments. Depending on the distance of the positioning and the acceleration and deceleration values, the motor speed will not reach the maximum configured speed.

NOTE!
If the Jerk argument value is not zero:
1) the value of the deceleration will be the same as the configured value in the acceleration;
2) the ContinuousUpdate On-line argument will have no effect and the argument values are considered at the time of positive transition Execute;
3) it is not allowed to perform positioning with another active block, occurring ErrorID 95.

When the MC_MotionRelative block is executed, the drive will start operating in grid position and

remains this way after the conclusion of the block. The position proportional gain must be set (P0159) so as to obtain a better drive performance.

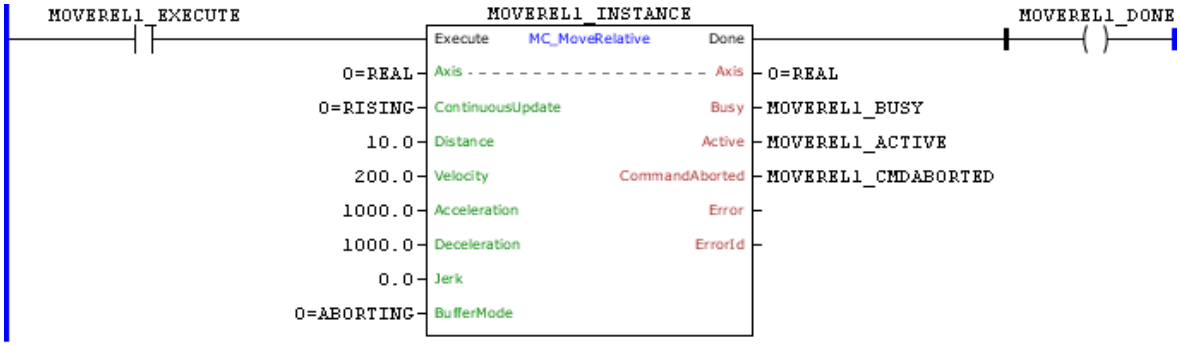
In the execution of the positioning, the axis status will change to Discrete Motion. When the positioning is concluded, the axis status will change to Standstill.

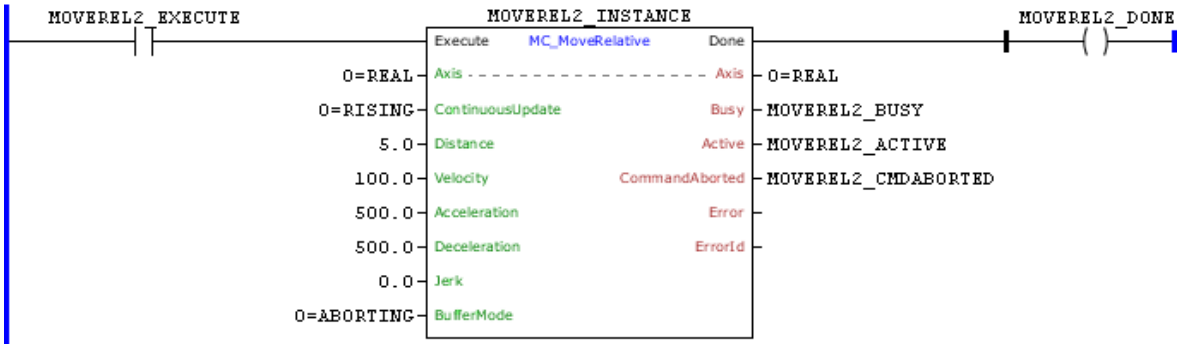
When Execute has FALSE value, Done remains FALSE. The Done output is activated when the block finishes the execution successfully, remaining at TRUE level until Execute receives FALSE.

If there is any error in the execution, the Error output is enabled and ErrorID displays an error code according to the table below.

Code	Description
52	Attempt to execute block w ith BufferMode in Single w hen another block is active.
60	Speed programmed below the minimum allow ed.
61	Speed programmed above the maximum allow ed.
62	Acceleration programmed below the minimum allow ed.
63	Acceleration programmed above the maximum allow ed.
64	Deceleration programmed below the minimum allow ed.
65	Deceleration programmed above the maximum allow ed.
67	Drive in the "Disabled" or “Errorstop” status.
69	Drive in the "Stopping" status.
70	Attempt to execute block w ith BufferMode in Buffered w hen another block is active and another block is w aiting.
71	P202 different from 4.
74	Drive in the "Homing" status.
78	MC block not executed – Internal fault.
93	Jerk programmed below the minimum allow ed.
94	Jerk programmed above the maximum allow ed.
95	It is not allow ed to execute positioning w ith Jerk w hen another block is active.

Example

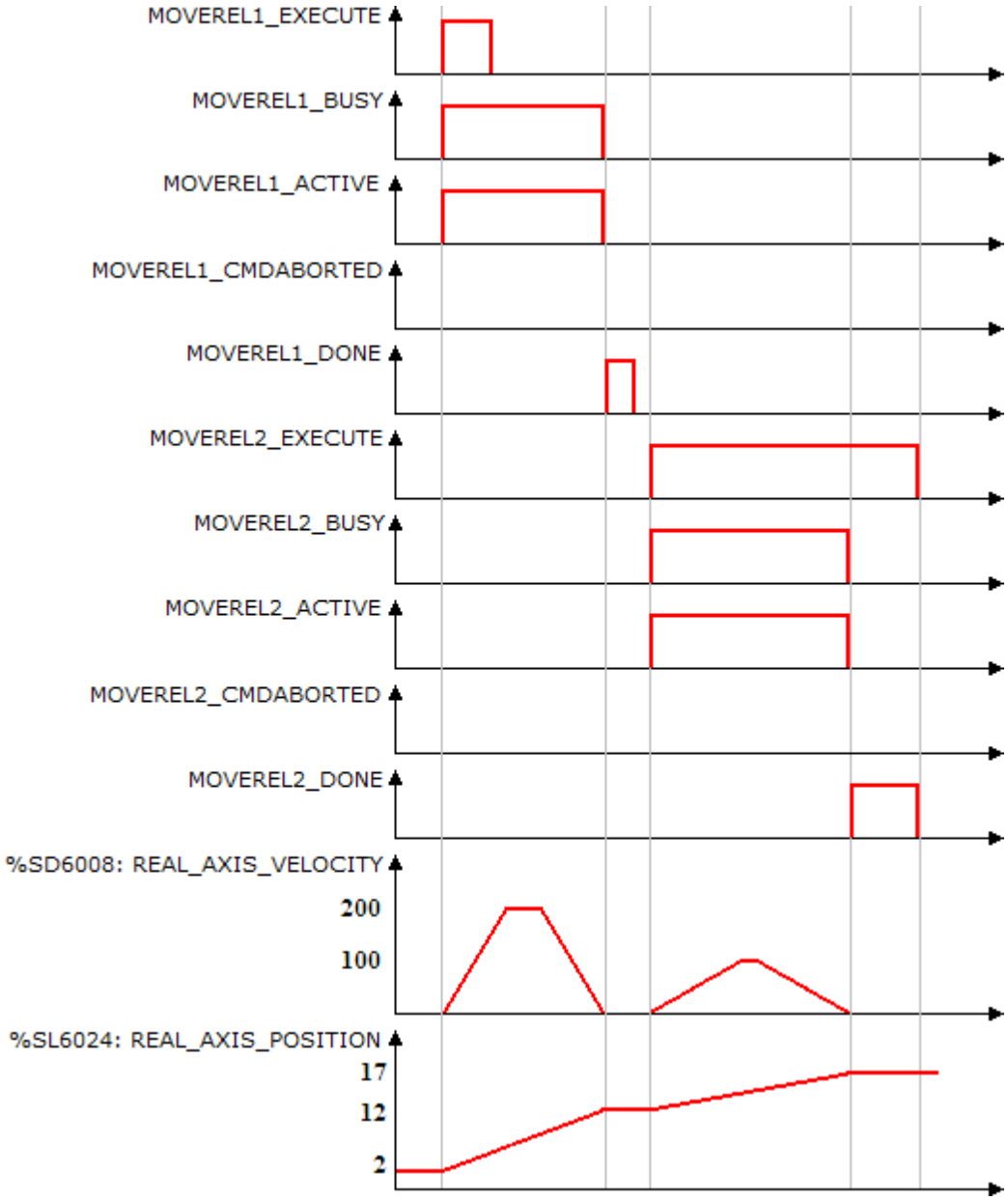




LOCAL | LOCAL_RETAIN

Tag	Tamanho	Tipo de Dado	Valor Inicial	Comentário
MOVEREL1_INSTANCE	0	MC_MoveRelative		
MOVEREL2_INSTANCE	0	MC_MoveRelative		
MOVEREL1_EXECUTE	0	BOOL	0	
MOVEREL1_DONE	0	BOOL	0	
MOVEREL1_BUSY	0	BOOL	0	
MOVEREL1_ACTIVE	0	BOOL	0	
MOVEREL1_CMDABORTED	0	BOOL	0	
MOVEREL2_EXECUTE	0	BOOL	0	
MOVEREL2_DONE	0	BOOL	0	
MOVEREL2_BUSY	0	BOOL	0	
MOVEREL2_ACTIVE	0	BOOL	0	
MOVEREL2_CMDABORTED	0	BOOL	0	

GLOBAL | LOCAL

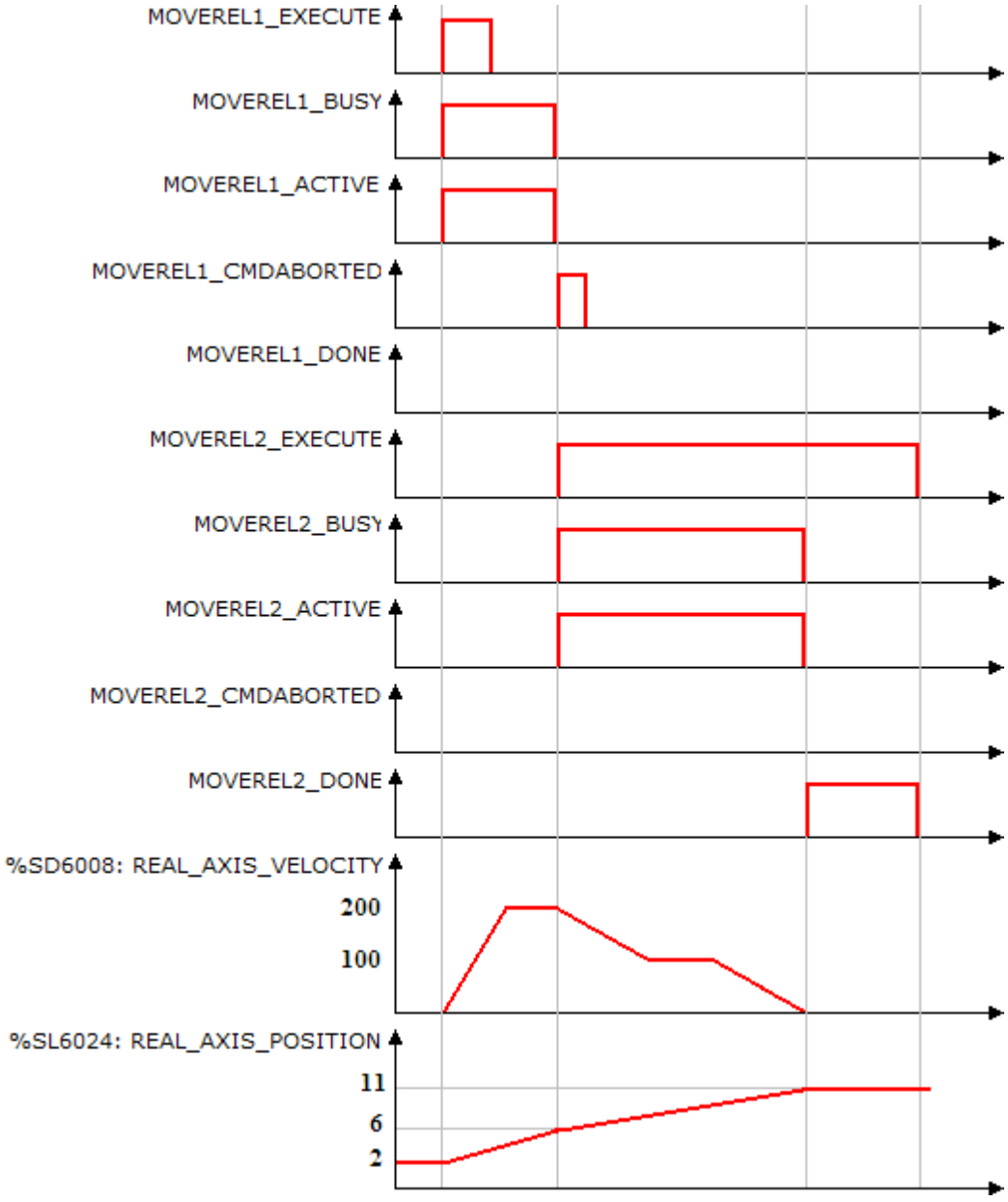


In the up transition of MOTIONREL1_EXECUTE, the first MC_MotionRelative block is executed. With this, the Busy and Active signals of this block are set and positioning of 10 positive revolutions from the current position starts.

When the positioning is finished, the first block is completed. With this, the Busy and Active signals of this block are reset and Done output is set for 1 scan cycle.

In the up transition of MOTIONREL2_EXECUTE, the second MC_MotionRelative block is executed. With this, the Busy and Active signals of this block are set and positioning of 5 positive revolutions from the current position starts.

When the positioning of 5 revolutions finishes, the Done output of the second block is set and the Busy and Active signals of this block are reset. The Done output remains TRUE while the Execute input is set.



In the up transition of MOTIONREL1_EXECUTE, the first MC_MotionRelative block is executed. With this, the Busy and Active signals of this block are set and positioning of 10 positive revolutions from the current position starts.

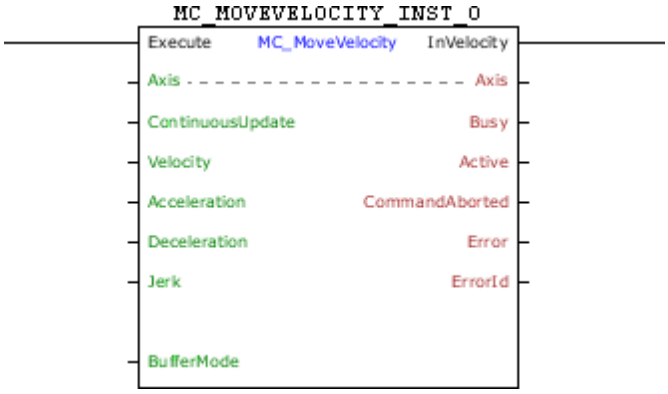
With the up transition of MOTIONREL2_EXECUTE, the second MC_MotionRelative block is instantly executed (BufferMode - Aborting). With this, the Busy and Active signals of this block are set and positioning of 5 positive revolutions from the current position starts. At the same time, the Busy and Active signals of the first block are reset and the CommandAborted signal is set for 1 scan cycle.

When the positioning of 5 revolutions finishes, the Done output of the second block is set and the Busy and Active signals of this block are reset. The Done output remains TRUE while the Execute input is set.

11.10.7.14.5.3 MC_MoveVelocity

Block responsible for making a motion to the programmed speed.

Ladder Representation



Execution Features

Program Memory Size	66 Bytes
Data Memory Size	24 Bytes

Block Structure

Variable Type	Name	Data Type	Description
VAR_IN_OUT	Axis	BYTE	Selection of operation axis (0 - Real axis)
VAR_INPUT	Execute	BOOL	Block enabling
	ContinuousUpdate	BYTE	Automatic update mode (0 - Rising edge) (1 - Real time)
	Velocity	REAL	Speed
	Acceleration	REAL	Acceleration
	Deceleration	REAL	Deceleration
	Jerk	REAL	Jerk
	BufferMode	REAL	Execution start mode (0 - Starts block immediately, if there is another block in the execution it will be aborted) (1 - When another block is in execution, the block in execution will continue its motion until the end and this new block will wait to be executed.) (6 – If another block is in execution, this block will go into error 52 and will not be executed. The HMI will show the alarm A00052.)
VAR_OUTPUT	InVelocity	BOOL	Output enabling
	Busy	BOOL	Flag indicating the block has not yet been ended
	Active	BOOL	Block flag with control on the axis
	CommandAborted	BOOL	Flag of aborted command
	Error	BOOL	Error in the execution flag
	ErrorID	WORD	Identifier of the occurred error
VAR	MC_MOVEVELOCITY_INST_0	MC_MOVEVELOCITY	Instance of access to block structure

Operation

When this block detects a leading edge in Execute, it will perform a motion into the configured speed in the Velocity argument with acceleration/deceleration configured in Acceleration and Deceleration arguments. The motion direction will depend on the speed signal: If the speed is greater than zero, the motion will be in the positive direction (clockwise) and if the speed is less than zero, the motion will be in the negative direction (counterclockwise).

NOTE!
If the Jerk argument is different from zero, the ContinuousUpdate On-line argument will have no effect and the argument values are considered at the time of positive transition Execute.

When the MC_MotionVelocity block is executed, the drive does not operate in grid position.

In the execution of the motion the axis status will change to Continuous Motion.

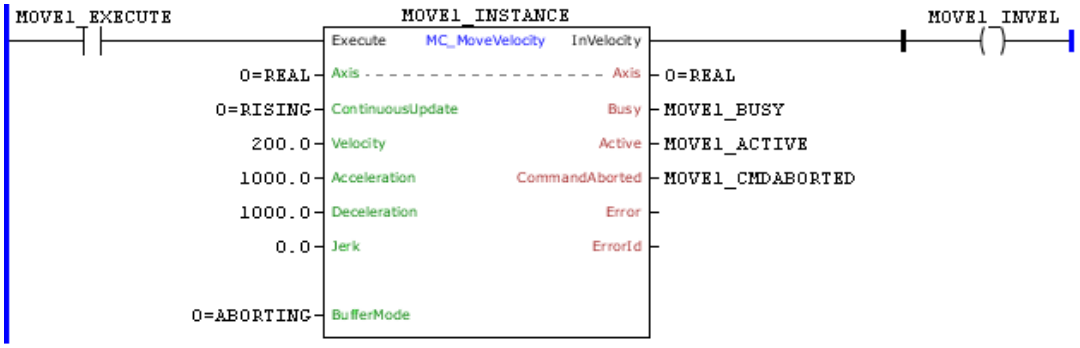
When Execute has FALSE value, InVelocity remains FALSE. The InVelocity output is activated when

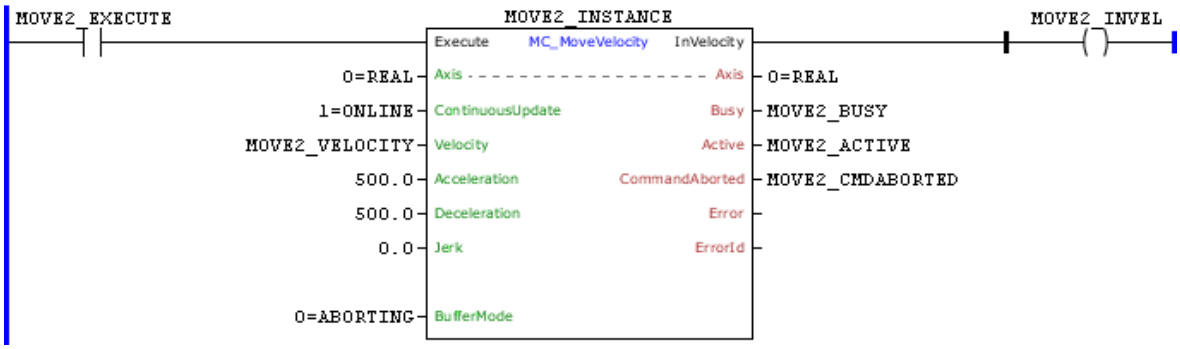
the programmed speed is achieved successfully, remaining TRUE level while the block is active. It order to conclude the block, it is necessary the execution of another block or the changing of the drive to the Disable or Errorstop status.

If there is any error in the execution, the Error output is enabled and ErrorID displays an error code according to the table below.

Code	Description
52	Attempt to execute block w ith BufferMode in Single w hen another block is active.
60	Speed programmed below the minimum allow ed.
61	Speed programmed above the maximum allow ed.
62	Acceleration programmed below the minimum allow ed.
63	Acceleration programmed above the maximum allow ed.
64	Deceleration programmed below the minimum allow ed.
65	Deceleration programmed above the maximum allow ed.
67	Drive in the "Disabled" or “Errorstop” status.
69	Drive in the "Stopping" status.
70	Attempt to execute block w ith BufferMode in Buffered w hen another block is active and another block is w aiting.
71	P202 different from 4.
74	Drive in the "Homing" status.
78	MC block not executed – Internal fault.
93	Jerk programmed below the minimum allow ed.
94	Jerk programmed above the maximum allow ed.
95	It is not allow ed to execute positioning w ith Jerk w hen another block is active.

Example

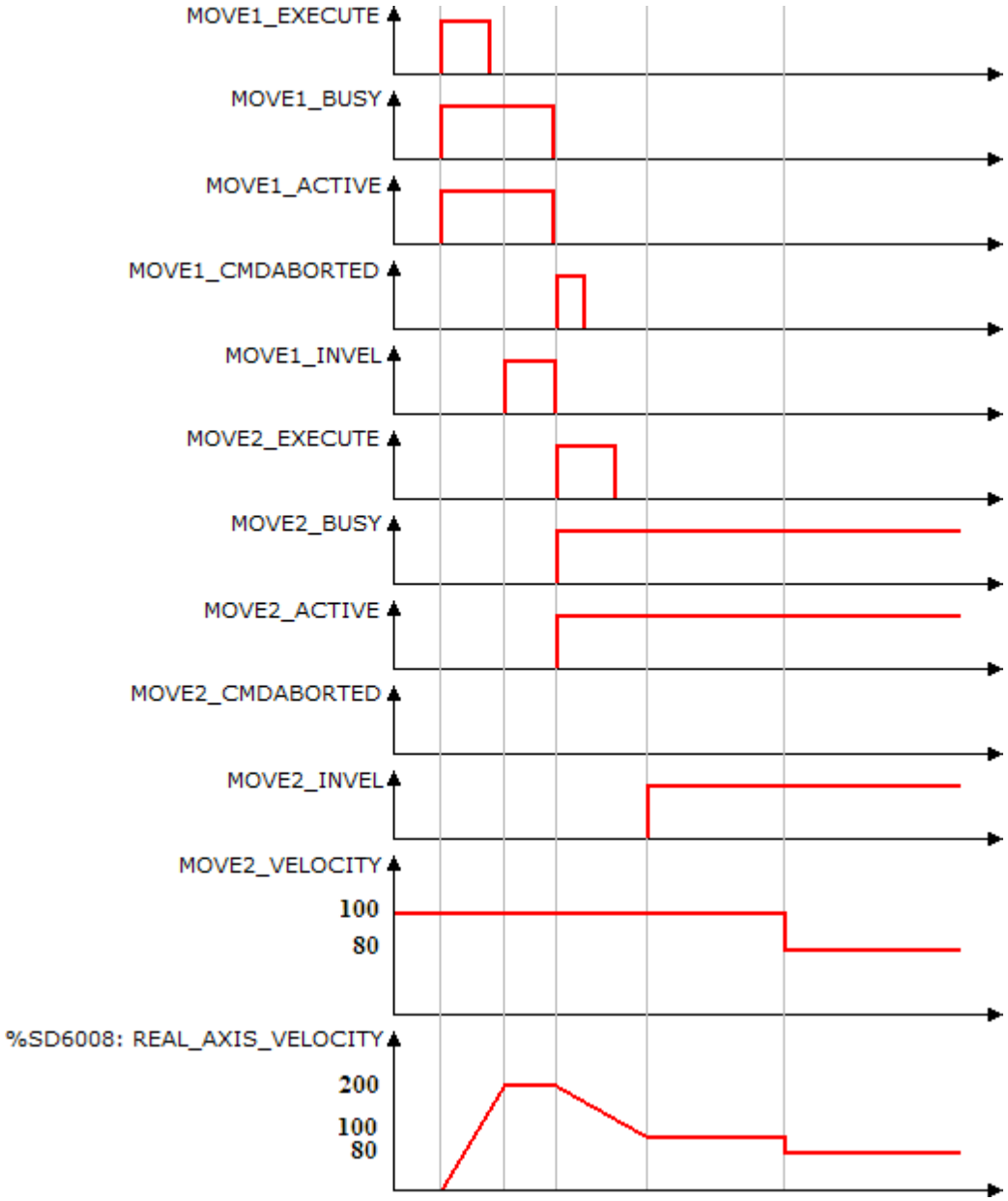




LOCAL | LOCAL_RETAIN

Tag	Tamanho	Tipo de Dado	Valor Inicial	Comentário
MOVE1_INSTANCE	0	MC_MoveVelocity		
MOVE1_EXECUTE	0	BOOL	0	
MOVE1_INVEL	0	BOOL	0	
MOVE1_BUSY	0	BOOL	0	
MOVE1_ACTIVE	0	BOOL	0	
MOVE1_CMDABORTED	0	BOOL	0	
MOVE2_INSTANCE	0	MC_MoveVelocity		
MOVE2_EXECUTE	0	BOOL	0	
MOVE2_INVEL	0	BOOL	0	
MOVE2_VELOCITY	0	REAL	200.0	
MOVE2_BUSY	0	BOOL	0	
MOVE2_ACTIVE	0	BOOL	0	
MOVE2_CMDABORTED	0	BOOL	0	

GLOBAL | LOCAL



In the up transition of MOTION1_EXECUTE, the first MC_MotionVelocity block is executed. With this, the Busy and Active signals of this block are set and the motion to reach the speed of 200 RPM starts.
At the moment in which the speed reaches 200 RPM, the InVelocity output is set.

With the up transition of MOTION2_EXECUTE, the second MC_MotionVelocity block is instantly executed (BufferMode - Aborting). With this, the Busy e Active signals of this block are set and the motion to the speed of 100 RPM (MOTION2_VELOCITY this time contains the value 100) starts. At the same time the Busy, Active and InVelocity signals of the first block are reset and the signal CommandAborted is set for 1 scan cycle.

When the speed of 100 RPM is reached, the InVelocity output of the second block is set and remains until the execution of another block.

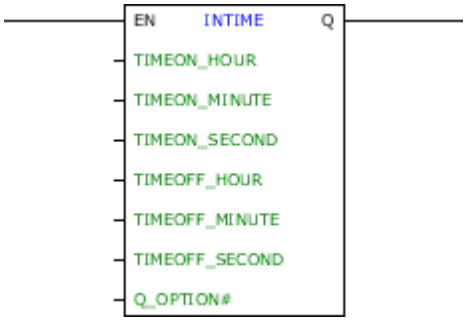
Since the ContinuousUpdate argument is configured as Online, with the change of the MOTION2_VELOCITY value to 80, the speed immediately changes to 80 RPM, without executing an acceleration/deceleration ramp.

11.10.7.15RTC

11.10.7.15.1 INTIME

Block that performs a programmed enabling for a time based on RTC (Real Time Clock).

Ladder Representation



Block Structure

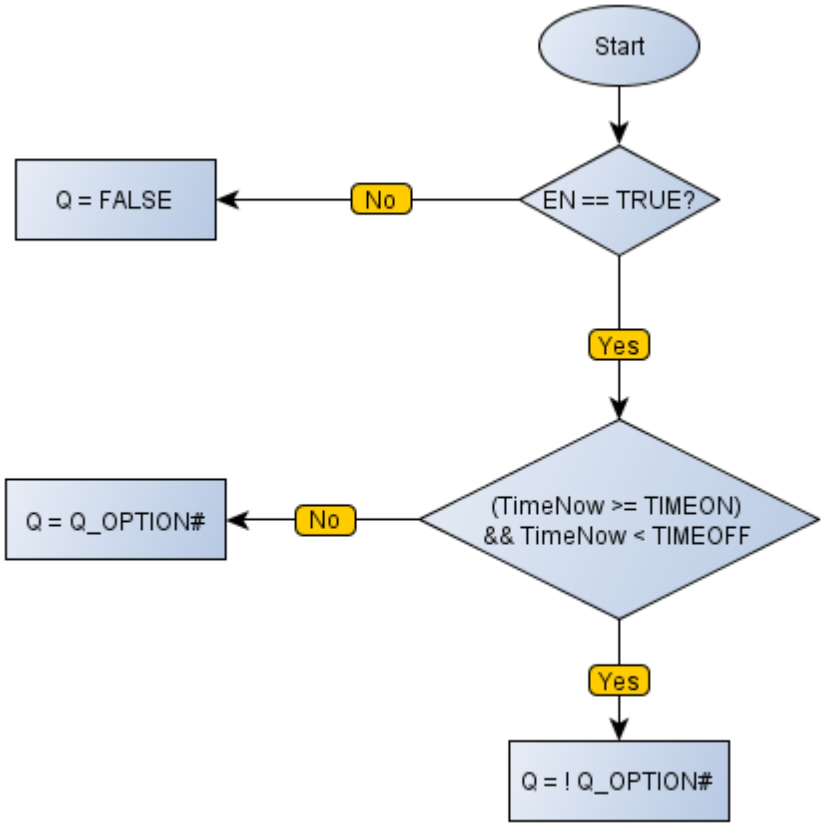
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	TIMEON_HOUR	WORD UINT	Enabling hour
	TIMEON_MINUTE	WORD UINT	Enabling minute
	TIMEON_SECOND	WORD UINT	Enabling second
	TIMEOFF_HOUR	WORD UINT	Disabling hour
	TIMEOFF_MINUTE	WORD UINT	Disabling minute
	TIMEOFF_SECOND	WORD UINT	Disabling second
	Q_OPTION#	BYTE	Output operation
VAR_OUTPUT	Q	BOOL	Block output

Operation

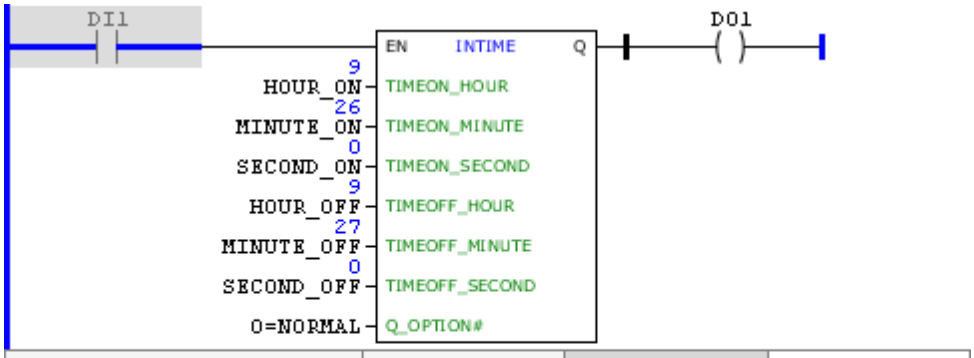
When this block has a TRUE value in EN, it has two modes of operation. If Q_OPTION# is Normal, Q is enabled when the internal clock's time is equal to that defined by the parameters TIMEON and disabled when the internal clock's time is equal to the parameters set by TIMEOFF. If Q_OPTION# is Inverted, Q is disabled when the internal clock's time is equal to that defined by the parameters TIMEON and enabled when the internal clock's time is equal to the parameters set by TIMEOFF.

When EN has FALSE value, Q remains FALSE.

Block Flowchart

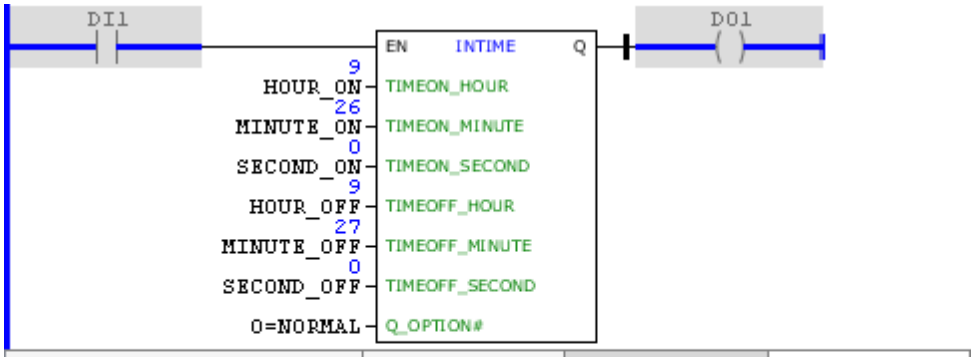


Example



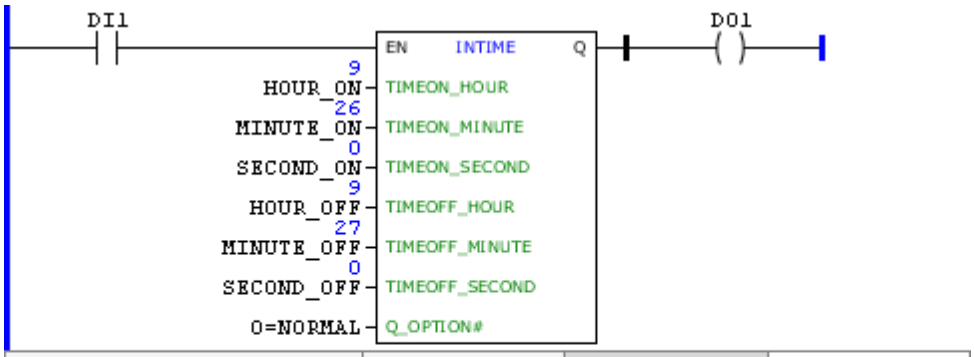
HOUR	UINT	9	Decimal
MINUTE	UINT	25	Decimal
SECOND	UINT	56	Decimal

In the example above, the INTIME block is enabled, the Q_OPTION# input is enabled for NORMAL operation and the current time of the internal clock of the device is lower than the registered enabling inputs of the block (HOUR_ON, MINUTE_ON and SECOND_ON). This way, the Q output is disabled.



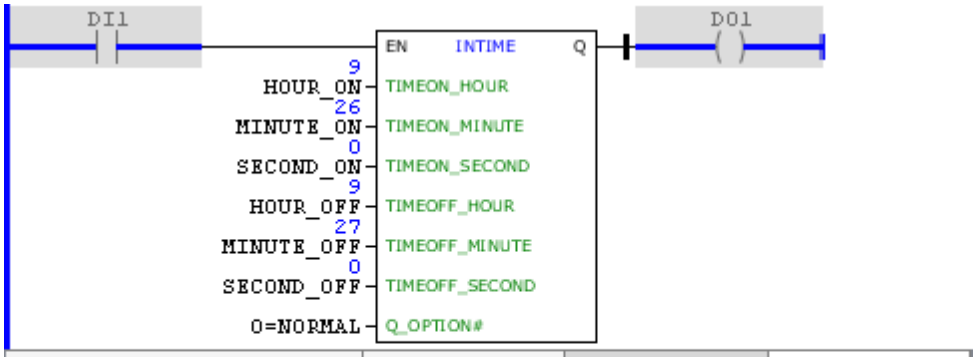
HOUR	UINT	9	Decimal
MINUTE	UINT	26	Decimal
SECOND	UINT	0	Decimal

In the example above, the INTIME block is enabled, the Q_OPTION# input is enabled for NORMAL operation and the current time of the internal clock of the device is equal to the registered in the enabling inputs of the block (HOUR_ON, MINUTE_ON and SECOND_ON). This way, the Q output is disabled.



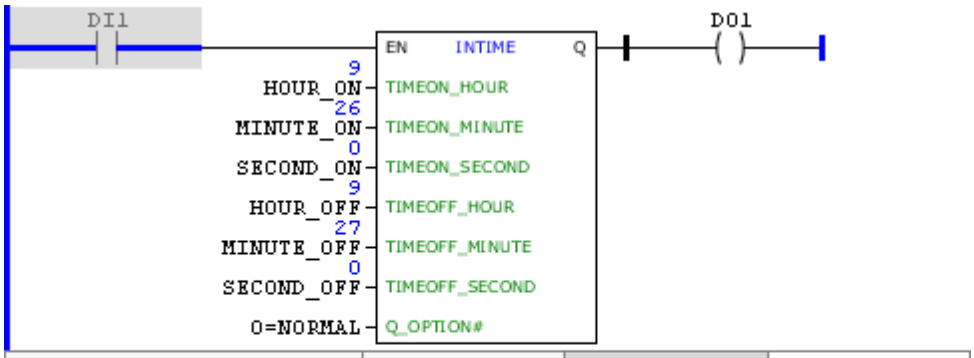
HOUR	UINT	9	Decimal
MINUTE	UINT	26	Decimal
SECOND	UINT	28	Decimal

In the above example, the INTIME block is disabled. This way, regardless of the input, the Q output is disabled.



HOUR	UINT	9	Decimal
MINUTE	UINT	27	Decimal
SECOND	UINT	0	Decimal

In the example above, the INTIME block is enabled, the Q_OPTION# input is enabled for NORMAL operation and the current time of the internal clock of the device is equal to the registered in the disabling inputs of the block (HOUR_OFF, MINUTE_OFF and SECOND_OFF). This way, the Q output is enabled.



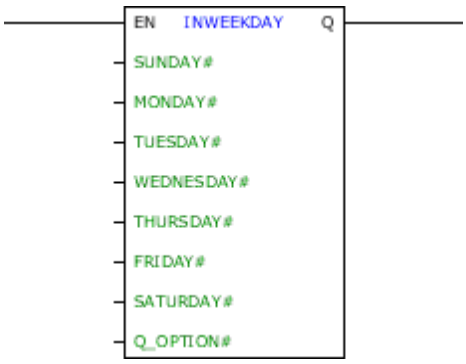
HOUR	UINT	9	Decimal
MINUTE	UINT	27	Decimal
SECOND	UINT	1	Decimal

In the example above, the INTIME block is enabled, the Q_OPTION# input is enabled for NORMAL operation and the current time of the internal clock of the device is superior to the registered in the disabling inputs of the block (HOUR_OFF, MINUTE_OFF and SECOND_OFF). Thus, the Q output is disabled.

11.10.7.15.2 INWEEKDAY

Block that performs a programmed enabling for weekdays based on RTC (Real Time Clock).

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	SUNDAY#	BOOL	Enabled on Sundays
	MONDAY#	BOOL	Enabled on Mondays
	TUESDAY#	BOOL	Enabled on Tuesdays
	WEDNESDAY#	BOOL	Enabled on Wednesdays
	THURSDAY#	BOOL	Enabled on Thursdays
	FRIDAY#	BOOL	Enabled on Fridays
	SATURDAY#	BOOL	Enabled on Saturdays
	Q_OPTION#	BYTE	Output operation
VAR_OUTPUT	Q	BOOL	Block output

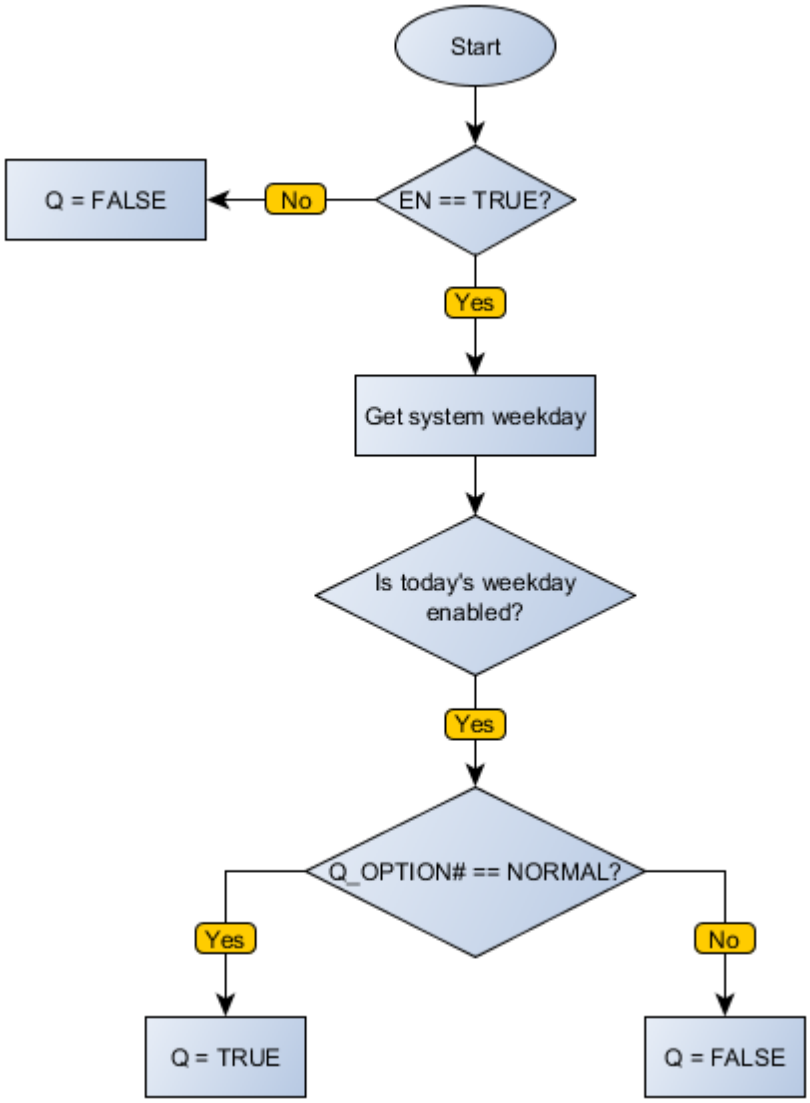
Operation

When this block has a TRUE value in EN, it has two modes of operation. If Q_OPTION# is Normal, Q is enabled if the day of week of the internal clock has Enabled parameter in the block. If Q_OPTION# is Inverted, Q is disabled if the day of week of the internal clock has Enabled parameter in the block.

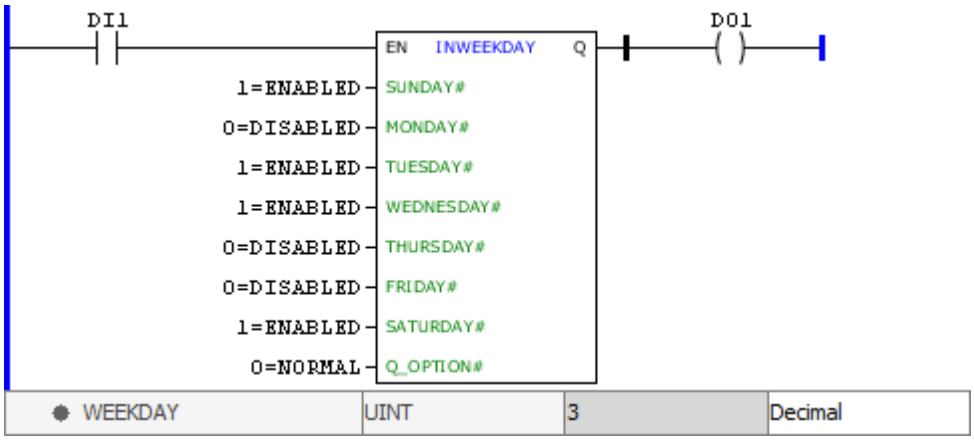
When EN has FALSE value, Q remains FALSE.

NOTE!
The weekdays are identified by numbers, with Sunday being day 0 and Saturday day 6.

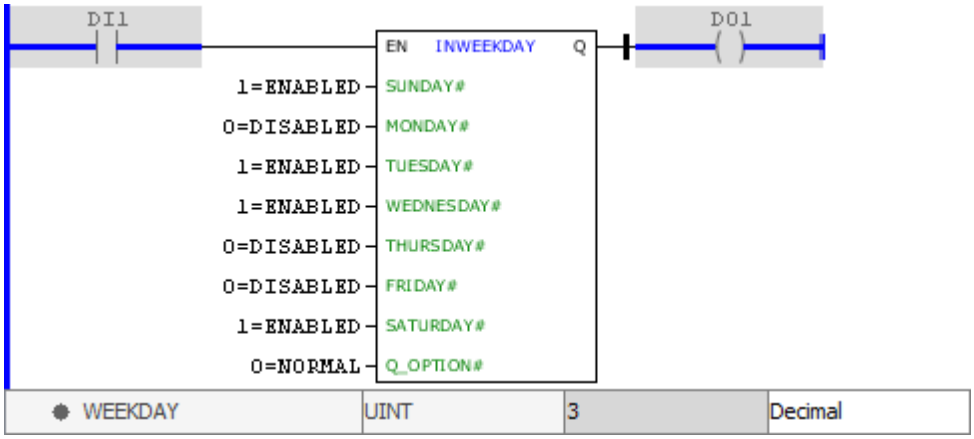
Block Flowchart


Start
EN == TRUE?
No
Q = FALSE
Yes
Get system weekday
Is today's weekday enabled?
Yes
Q_OPTION# == NORMAL?
Yes
No
Q = TRUE
Q = FALSE

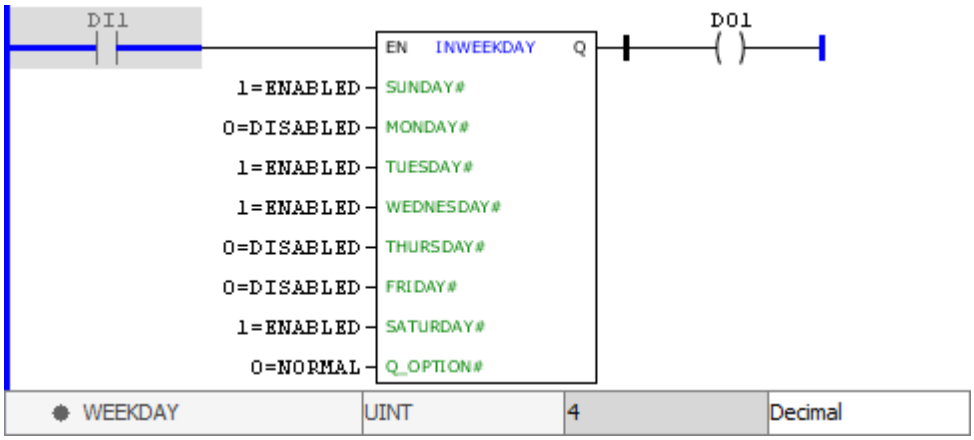
Example


DI1
EN
INWEEKDAY
Q
DO1
1=ENABLED
SUNDAY#
0=DISABLED
MONDAY#
1=ENABLED
TUESDAY#
1=ENABLED
WEDNESDAY#
0=DISABLED
THURSDAY#
0=DISABLED
FRIDAY#
1=ENABLED
SATURDAY#
0=NORMAL
Q_OPTION#
WEEKDAY
UINT
3
Decimal

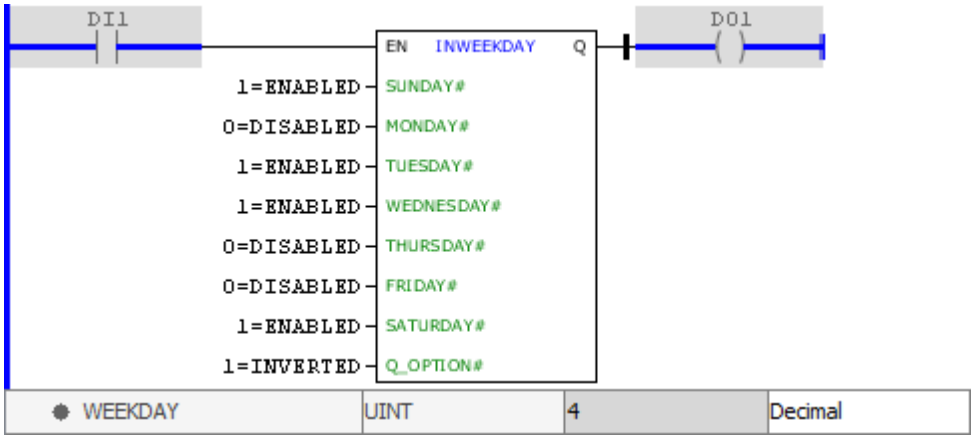
In the above example, the INWEEKDAY block is disabled. This way, regardless of the input, the Q output is disabled.



In the example above, the INWEEKDAY block is enabled and Q_OPTION# input is enabled for NORMAL operation. The current day of the week of the device's internal clock is Wednesday (value 3), which has ENABLED status in the programming. This way, the Q output is enabled.



In the example above, the INWEEKDAY block is enabled and Q_OPTION# input is enabled for NORMAL operation. The current day of the week of the device's internal clock is Thursday (value 4), which has DISABLED status in the programming. Thus, the Q output is disabled.



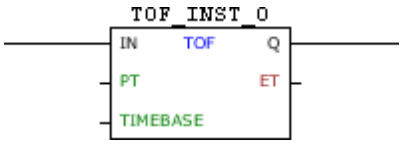
In the example above, the INWEEKDAY block is enabled and Q_OPTION# input is enabled for INVERTED operation. The current day of the week of the device's internal clock is Thursday (value 4), which has DISABLED status in the programming. This way, the Q output is enabled.

11.10.7.16 Timer

11.10.7.16.1 TOF

Timer block that, when energized, disables the output after a delay set by PT.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	IN	BOOL	Block enabling
	PT	WORD UINT DWORD UDINT	Delay of output deactivating
	TIMEBASE	WORD	Time base for PT and ET
VAR_OUTPUT	Q	BOOL	Block output
	ET	WORD UINT DWORD UDINT	Counter elapsed time
VAR	TOF_INST_0	TOF	Instance of access to block structure

NOTE!
In CFW300, the PT e ET fields can only be WORD ou UINT type.

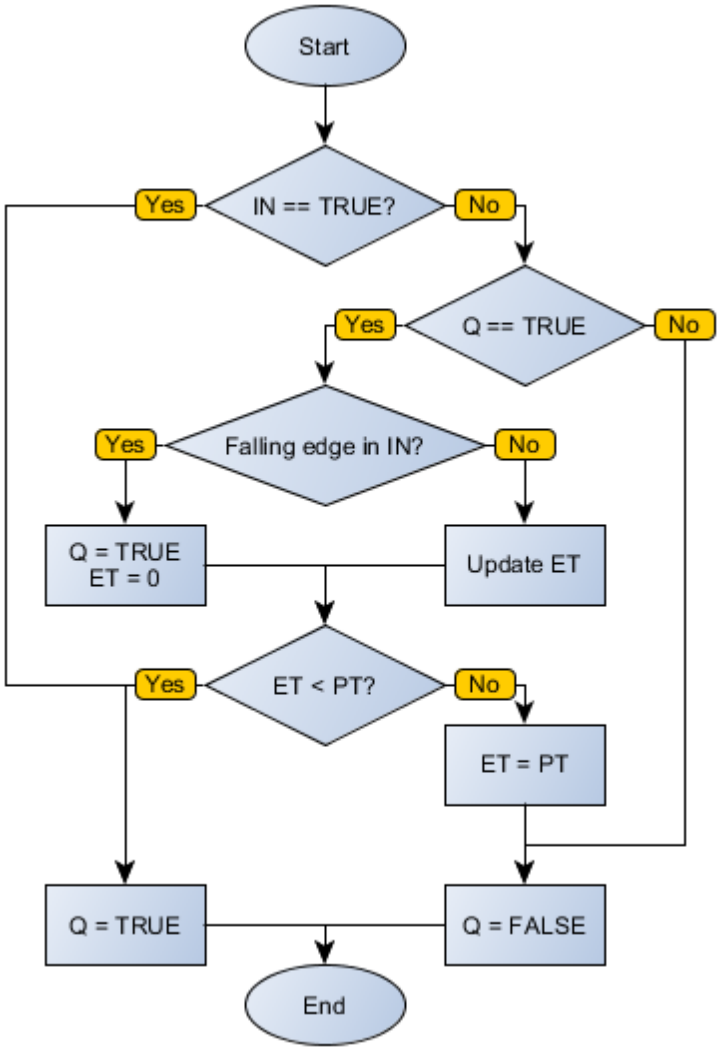
Operation

While the IN input is TRUE, the Q output is also TRUE and ET also receives the value zero.
On the negative transition edge in IN, counting is triggered and ET is incremented according to TIMEBASE. When ET equals PT, the Q output goes to state FALSE until IN revolutions to FALSE.

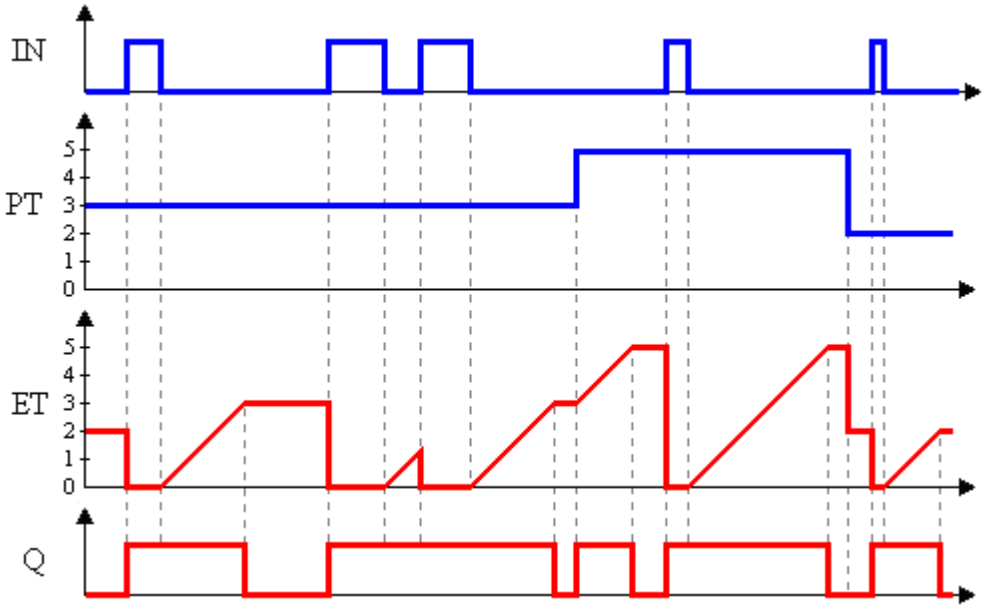
Compatibility

Device	Version
PLC300	1.50 or higher
SCA06	2.00 or higher

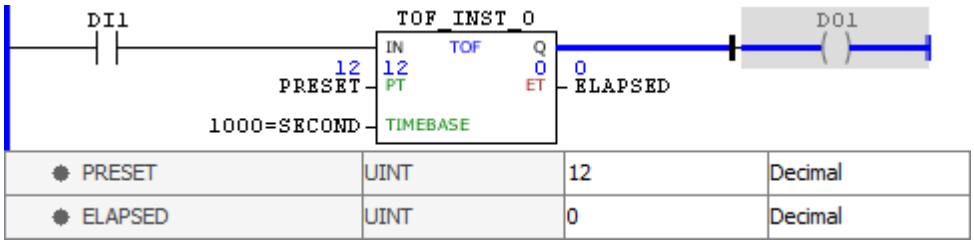
Block Flowchart



Operation Diagram



Example



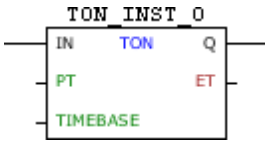
PRESET	UINT	12	Decimal
ELAPSED	UINT	0	Decimal

The above example disables the DO1 output to identify a low level in DI1 for 12 seconds, remaining disabled until DI1 again be TRUE.

11.10.7.16.2 TON

Timer block that, when energized, enables the output after a delay set by PT.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	IN	BOOL	Block enabling
	PT	WORD UINT DWORD UDINT	Delay of output drive
	TIMEBASE	WORD	Time base for PT and ET
VAR_OUTPUT	Q	BOOL	Block output
	ET	WORD UINT DWORD UDINT	Counter elapsed time
VAR	TON_INST_0	TON	Instance of access to block structure

NOTE!
In CFW300, the PT e ET fields can only be WORD ou UINT type.

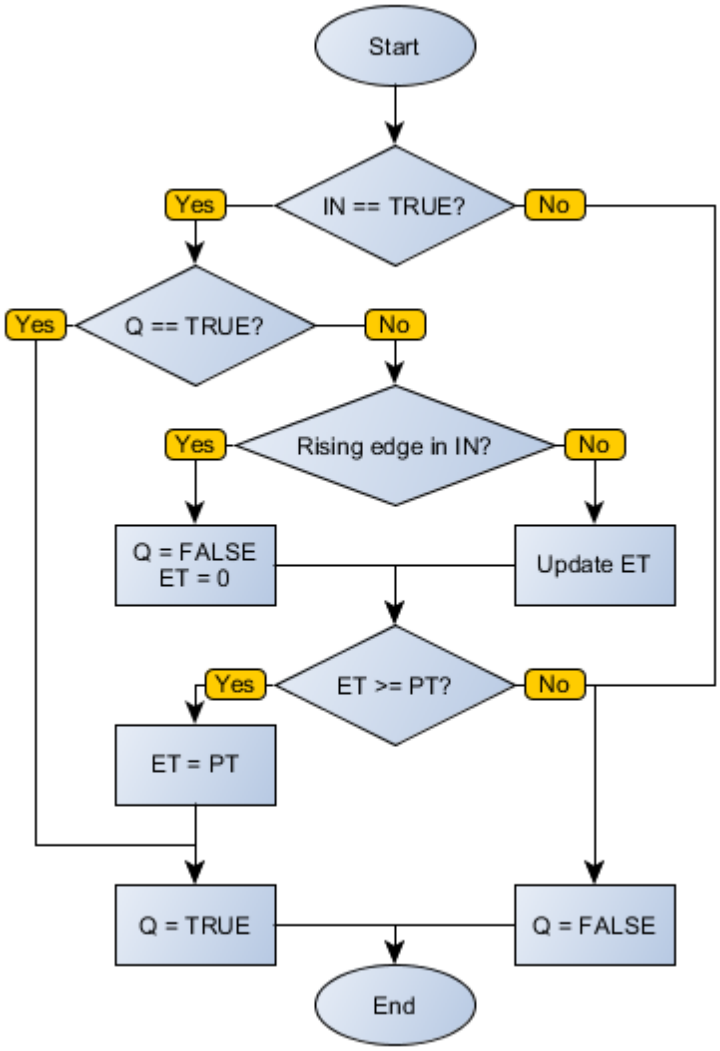
Operation

While the IN input is FALSE, the Q output is FALSE and ET also receives the value zero.
On the edge positive transition in IN, counting is triggered and ET is incremented according to TIMEBASE. When ET equals PT, the Q output goes to state TRUE until IN revolutions to FALSE.

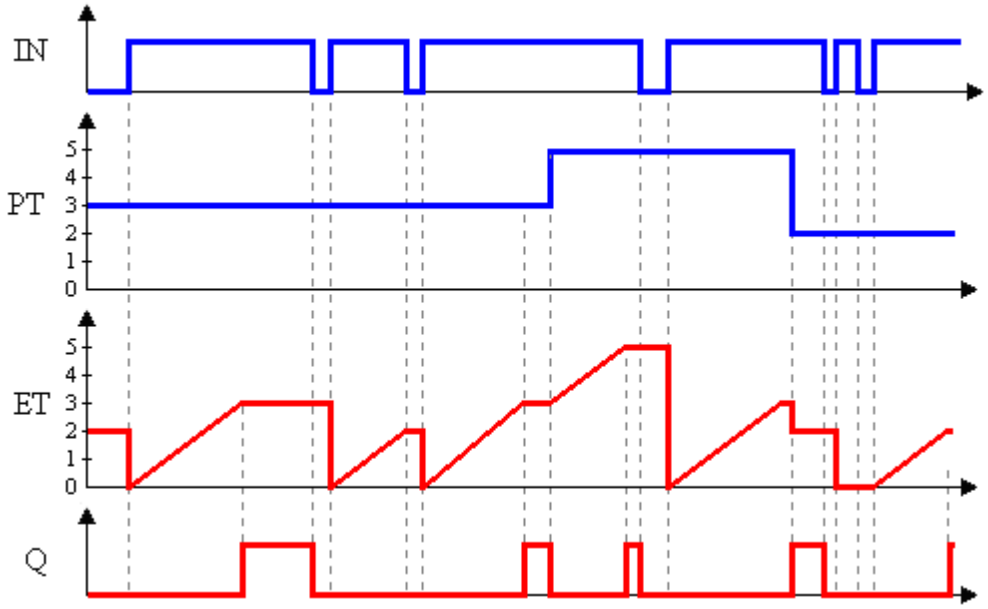
Compatibility

Device	Version
PLC300	1.50 or higher
SCA06	2.00 or higher

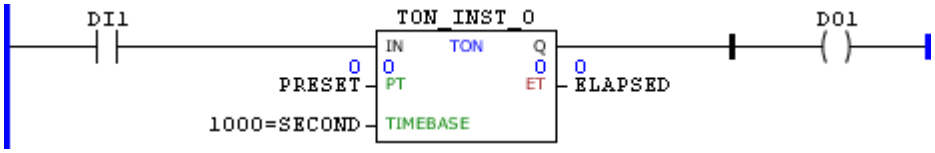
Block Flowchart



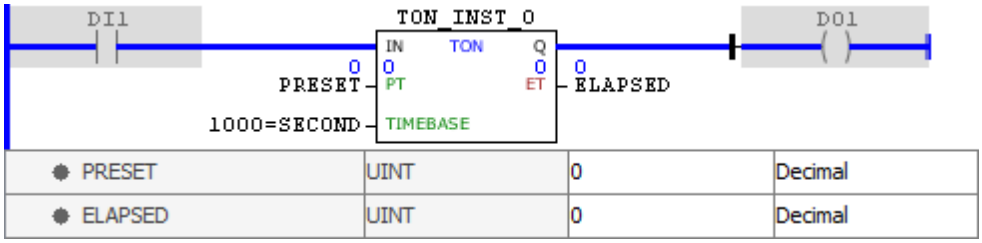
Operation Diagram



Example

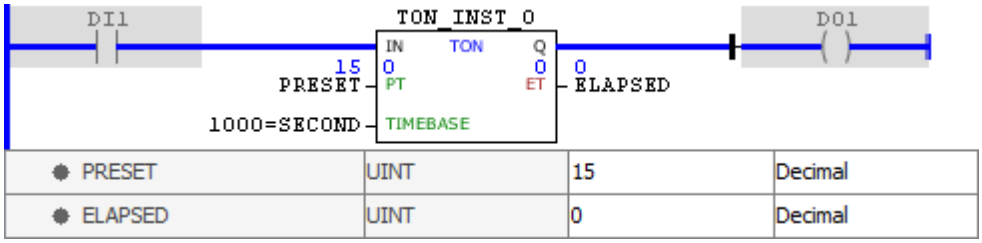


The above example shows the initial conditions of the block and of the routine variables.



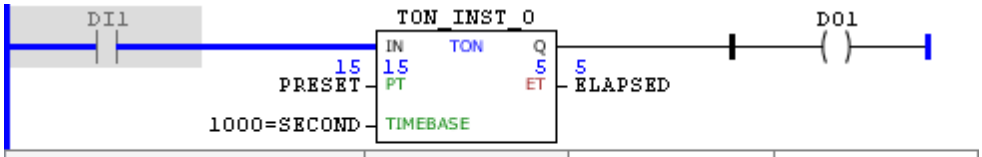
PRESET	UINT	0	Decimal
ELAPSED	UINT	0	Decimal

When activated the IN input, counting is triggered. Since ET equals PT, the Q output is enabled.



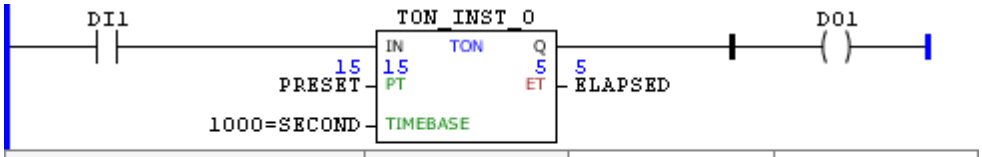
PRESET	UINT	15	Decimal
ELAPSED	UINT	0	Decimal

Note that a change in PRESET variable is not forwarded to the PT field while the IN entry remains enabled.



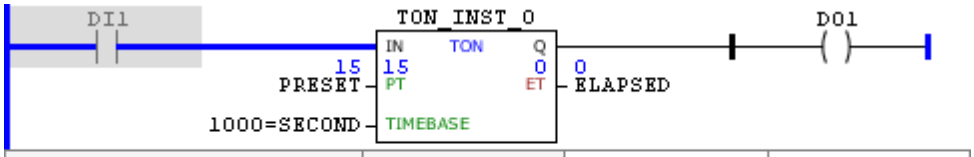
PRESET	UINT	15	Decimal
ELAPSED	UINT	5	Decimal

Disabling the IN input, the value of PT is updated and the Q output is disabled. When activating it again, counting is triggered.



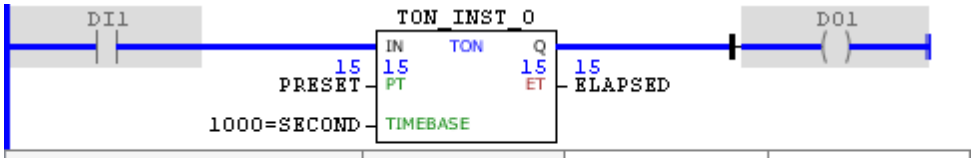
PRESET	UINT	15	Decimal
ELAPSED	UINT	5	Decimal

Disabling the IN input, the value of ET remains saved.



PRESET	UINT	15	Decimal
ELAPSED	UINT	0	Decimal

Enabling the IN input, the value of ET is reset and counting is triggered.



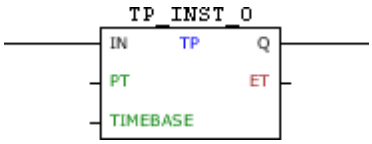
PRESET	UINT	15	Decimal
ELAPSED	UINT	15	Decimal

When ET reaches the value PT, the Q is output enabled and remains so while IN is at TRUE level.

11.10.7.16.3 TP

Timer block that, when identifies it is energized, enables the output after a delay set by PT.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	IN	BOOL	Block enabling
	PT	WORD UINT DWORD UDINT	Time w hile the output is enabled
	TIMEBASE	WORD	Time base for PT and ET
VAR_OUTPUT	Q	BOOL	Block output
	ET	WORD UINT DWORD UDINT	Counter elapsed time
VAR	TP_INST_0	TP	Instance of access to block structure

NOTE!
In CFW300, the PT e ET fields can only be WORD ou UINT type.

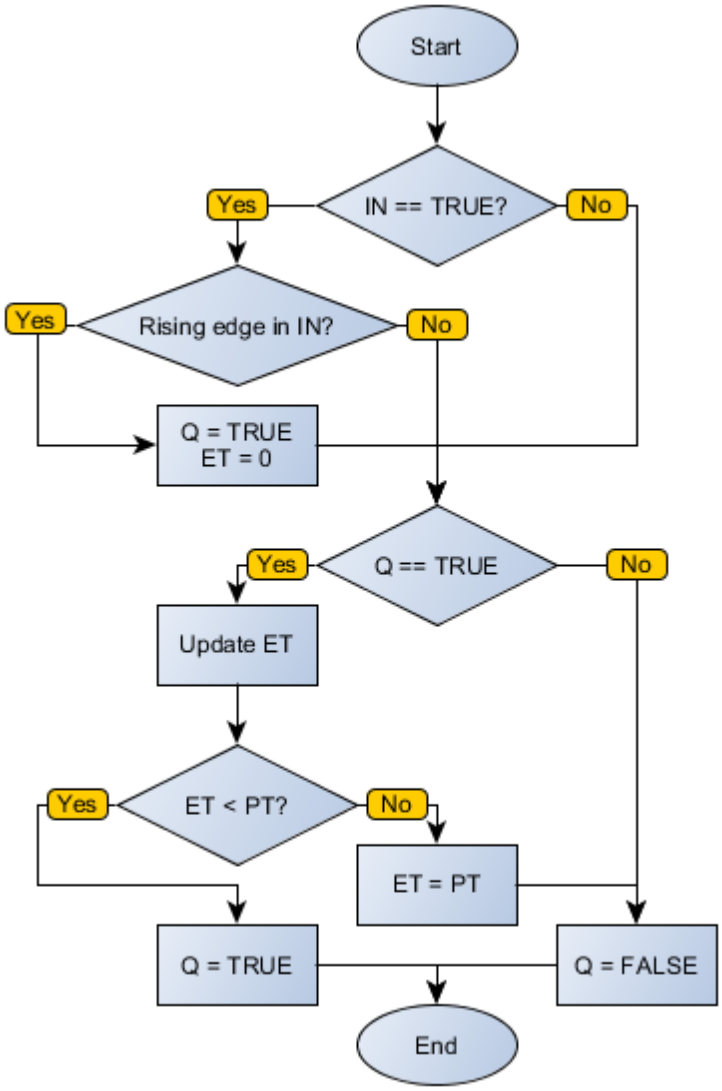
Operation

On the edge positive transition in IN, Q receives TRUE value, counting is triggered and ET is incremented according to TIMEBASE. When ET equals PT, the Q output goes to state FALSE until IN revolutions to FALSE. At that moment, if IN is at TRUE level, nothing happens. On the edge positive transition in IN, ET is automatically reset.

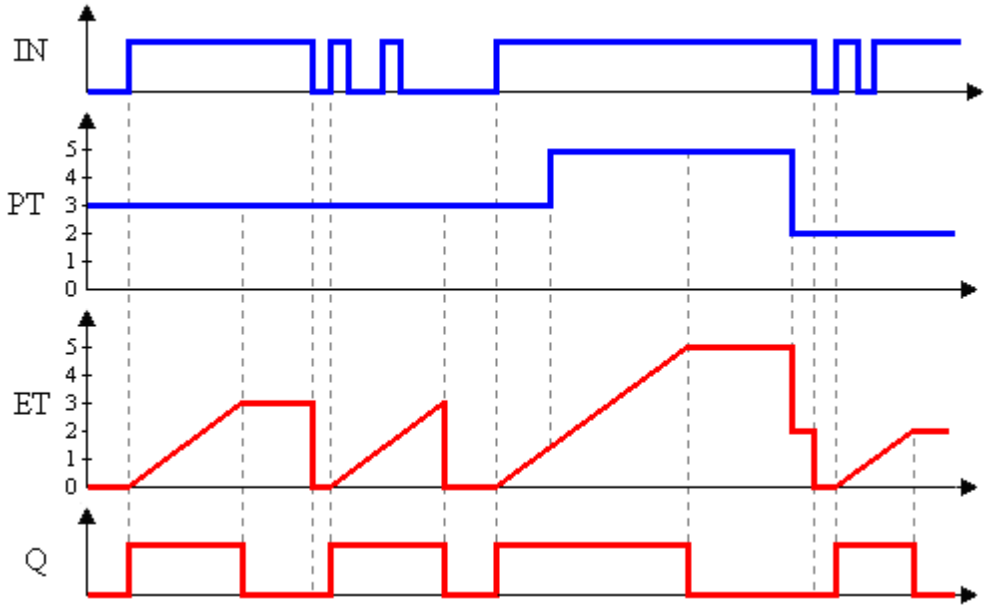
Compatibility

Device	Version
PLC300	1.50 or higher
SCA06	2.00 or higher

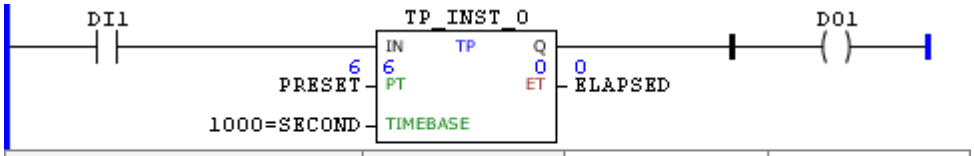
Block Flowchart



Operation Diagram



Example

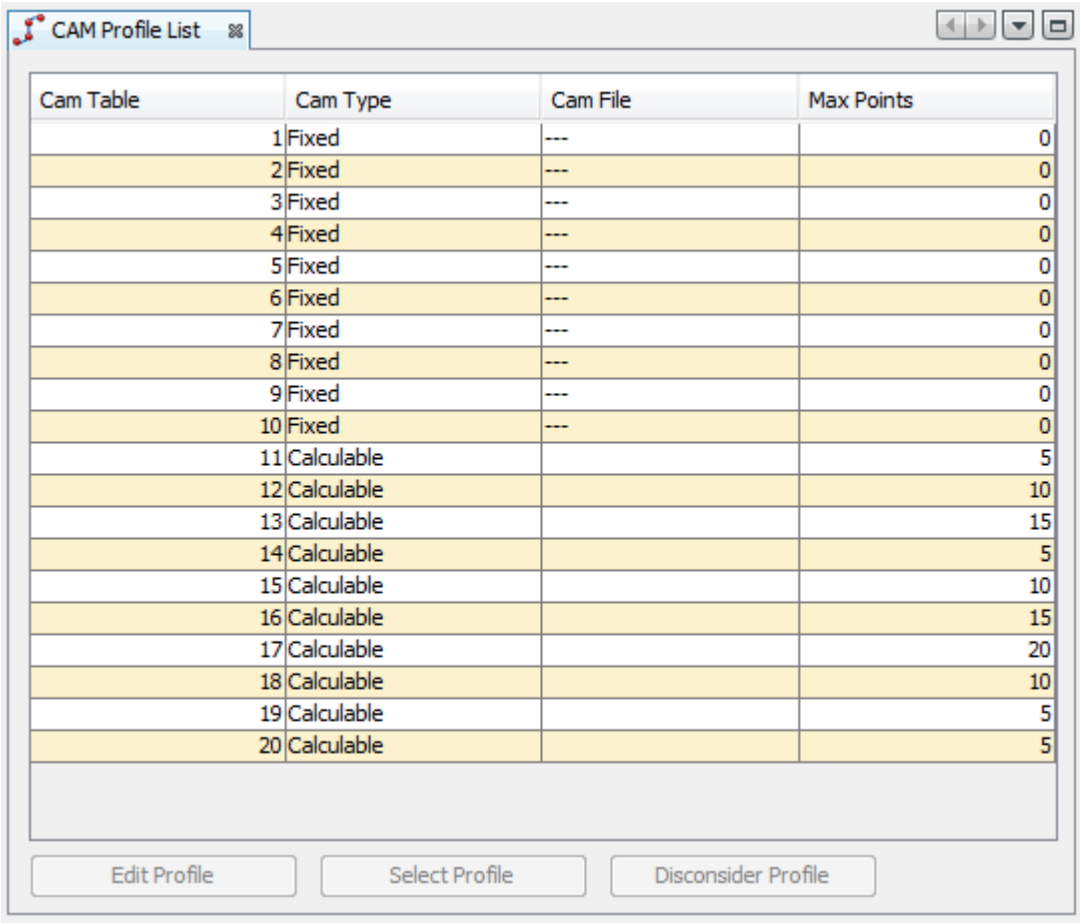


PRESET	UINT	6	Decimal
ELAPSED	UINT	0	Decimal

The above example enables the DO1 output for six seconds at each DI1 positive transition.

11.10.7.17 Cam Profiles

It allows loading and editing the cam table of the CAM curves.
Accessed by the **CAM Profile List** command with the right button of the mouse in the **CAM Profiles** folder of the resource.

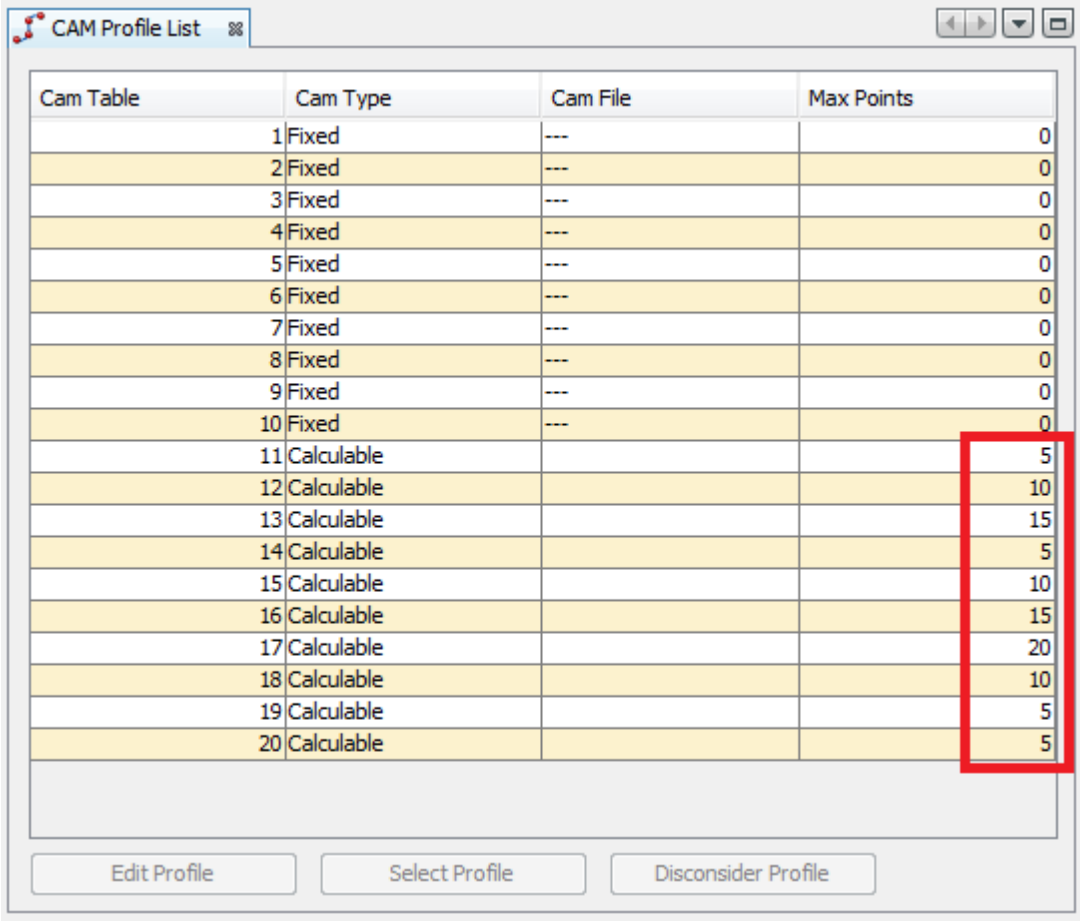


Cam Table	Cam Type	Cam File	Max Points
1	Fixed	---	0
2	Fixed	---	0
3	Fixed	---	0
4	Fixed	---	0
5	Fixed	---	0
6	Fixed	---	0
7	Fixed	---	0
8	Fixed	---	0
9	Fixed	---	0
10	Fixed	---	0
11	Calculable		5
12	Calculable		10
13	Calculable		15
14	Calculable		5
15	Calculable		10
16	Calculable		15
17	Calculable		20
18	Calculable		10
19	Calculable		5
20	Calculable		5

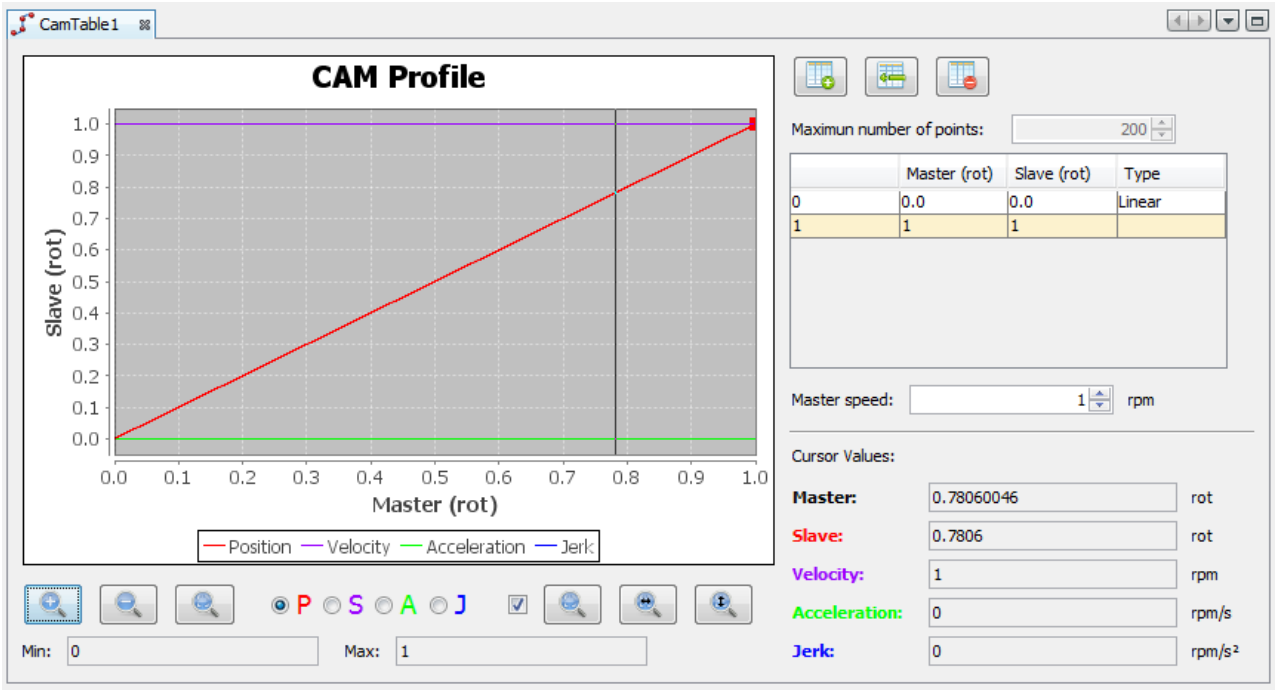
Description

The cam tables from 1 to 10 are tables of fixed points, which are transmitted at the moment of the download of the application. In order to use the tables 1 to 10, first the MC_CamTableSelect block must be executed with the desired table and then the MC_CamIn block.
The cam tables 11 to 20 are tables of variable points. In order to use the tables 11 to 20, first the MC_CamCalc block must be executed with the desired table and then the MC_CamIn block.

For the SCA06 equipment, it is allowed programming at most 200 fixed points and 100 variable points, seeing that the maximum number of variable points of each table must be configured in the Max Points column, as shown below:

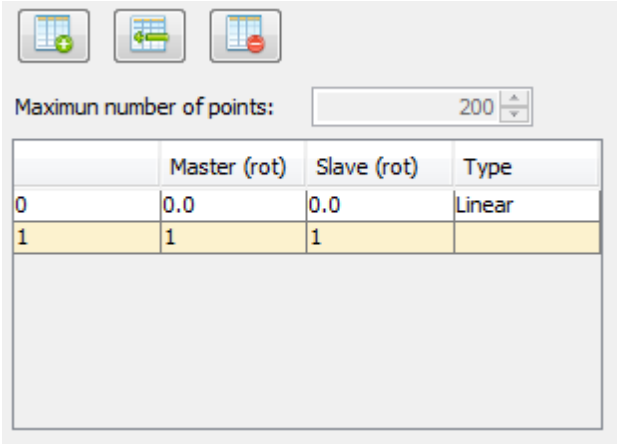


In order to edit the cam table, click on the **Edit** button, and the cam profile editor will open, as in the figure below:



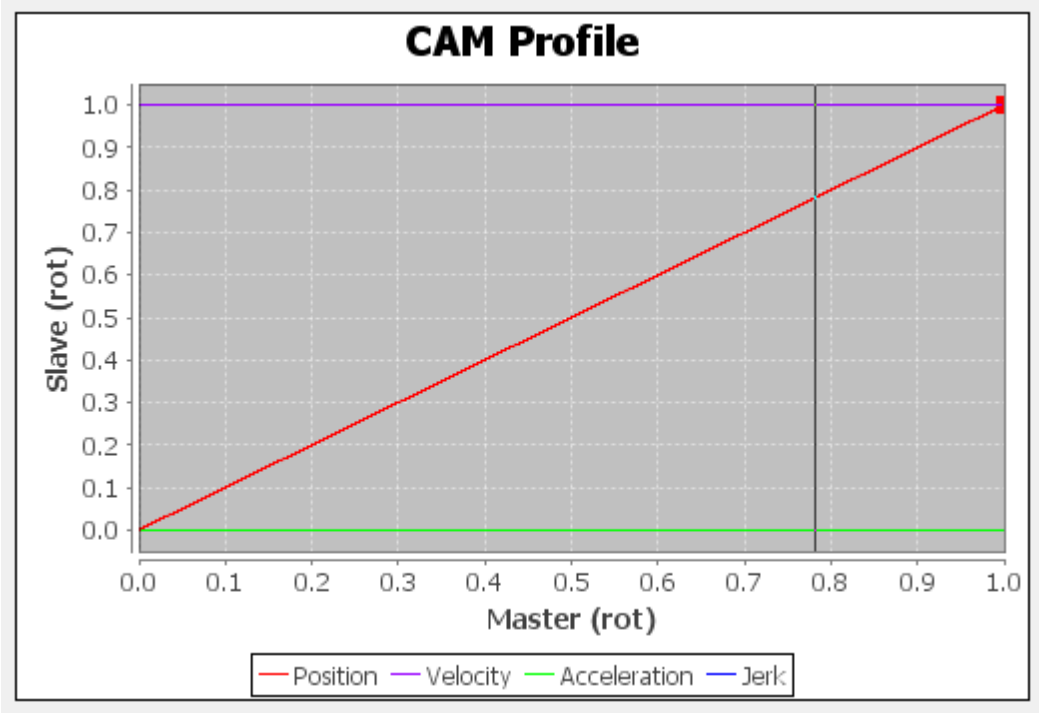
This window has the following controls:

Cam table:

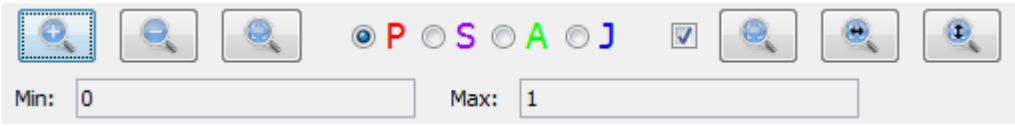


NOTE!
The CAM block is always relative, so the first point of the cam table will always be master= 0 and slave = 0.

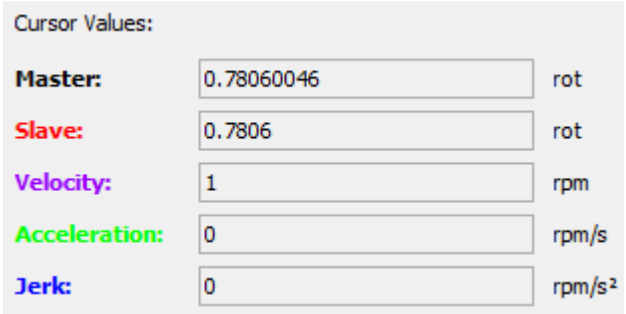
Graphic of the profile:



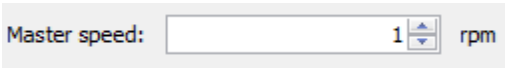
Graphic control tools:



Values of the cursor:
Values relative to the selected point of the cursor.



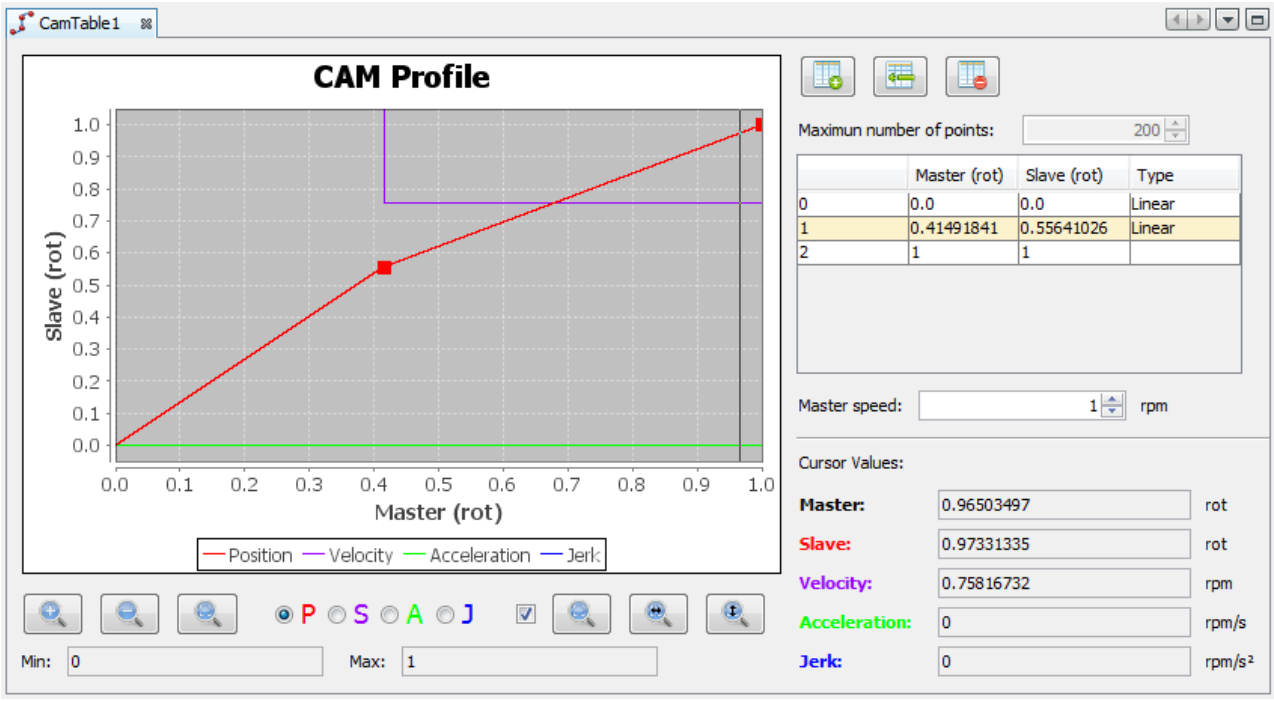
Master speed:
Speed used to calculate the speed, acceleration and jerk of the slave.



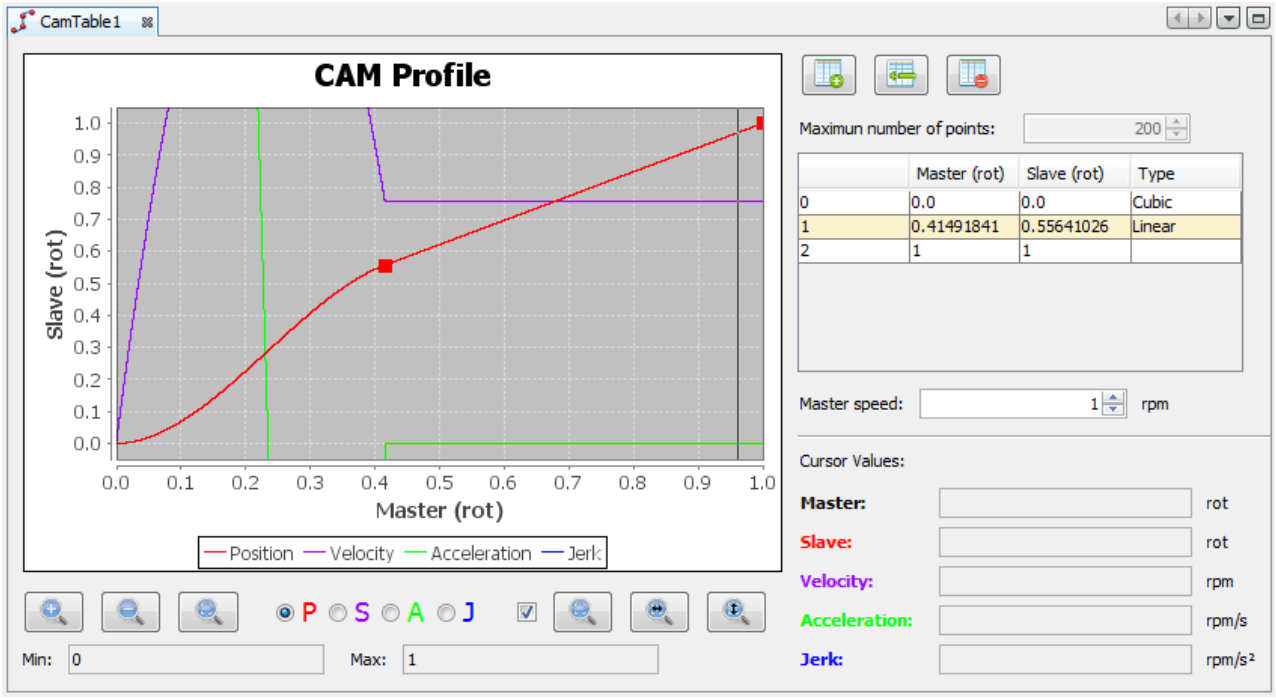
NOTE!
The speed, acceleration and jerk of the slave must be used as reference to develop the cam profile, where they are calculated numerically, not taking into account load, inertia, torque and dynamics of the drive.

Adding a new point to the cam profile

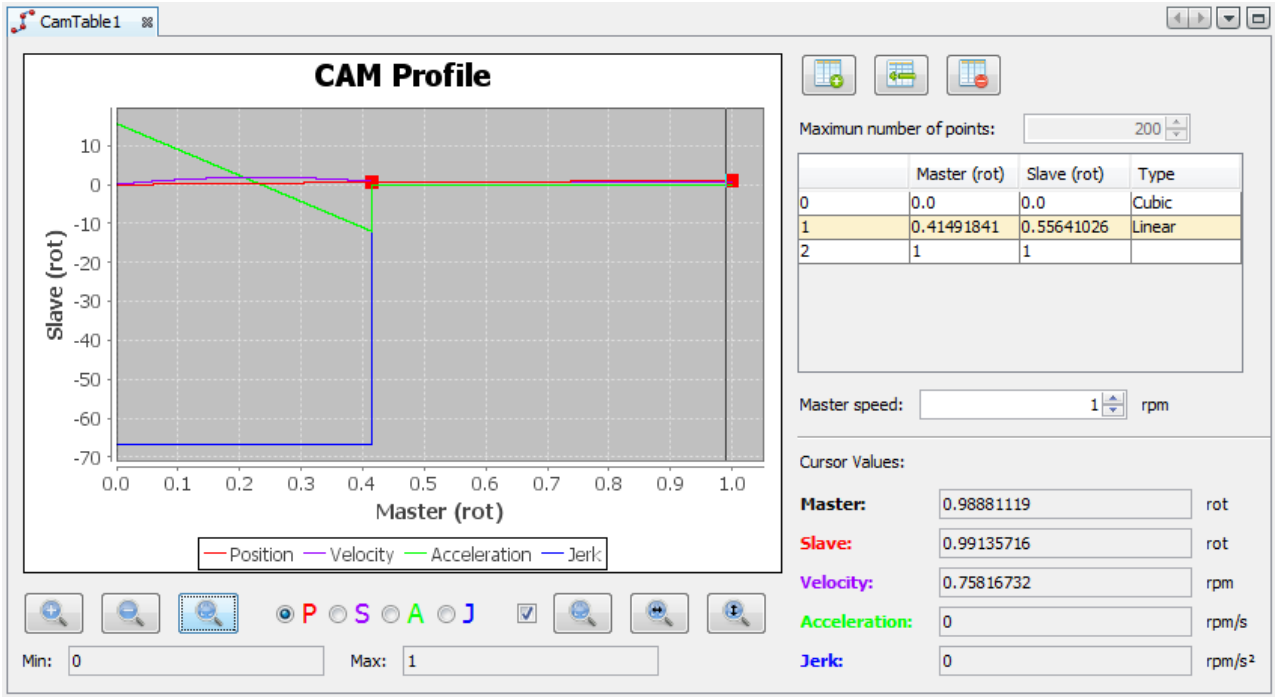
A point can be added by means of the add or insert point buttons or by double clicking the graphic in the position where you wish to add the point. You can double click any region of the graphic. In case an interpolation already exists in this region, the editor will insert this point between the two points of the interpolation.
The point is always added as linear interpolation.
When a point is added or inserted by means of the respective buttons, the master and slave values come zeroed. In case of point insertion, that may cause an interruption of the profile, because the master position must always grow in relation to the origin; therefore, the value of the master and slave must be edited by clicking on their cells in the cam table.
On the figure below, a point was inserted by double clicking:



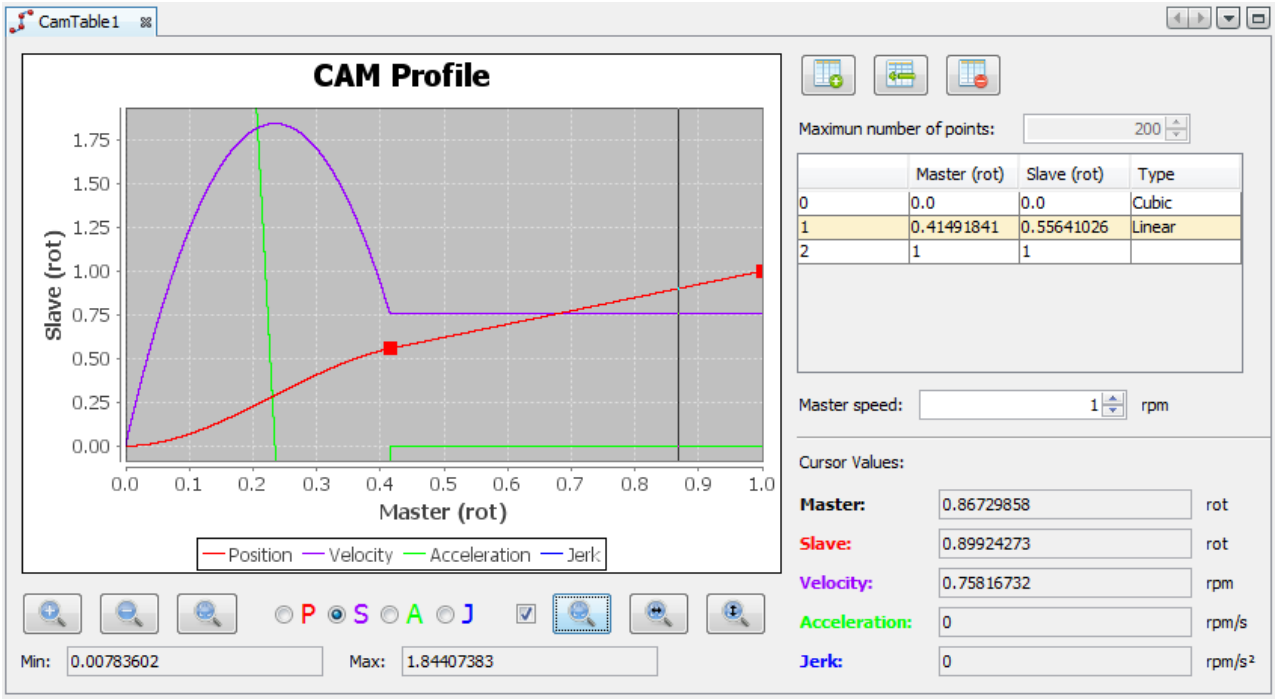
In order to change the type of interpolation, click on the type cell in the line corresponding to the origin of the interpolation and select the desired type.
In the figure below, the point was changed for cubic interpolation.



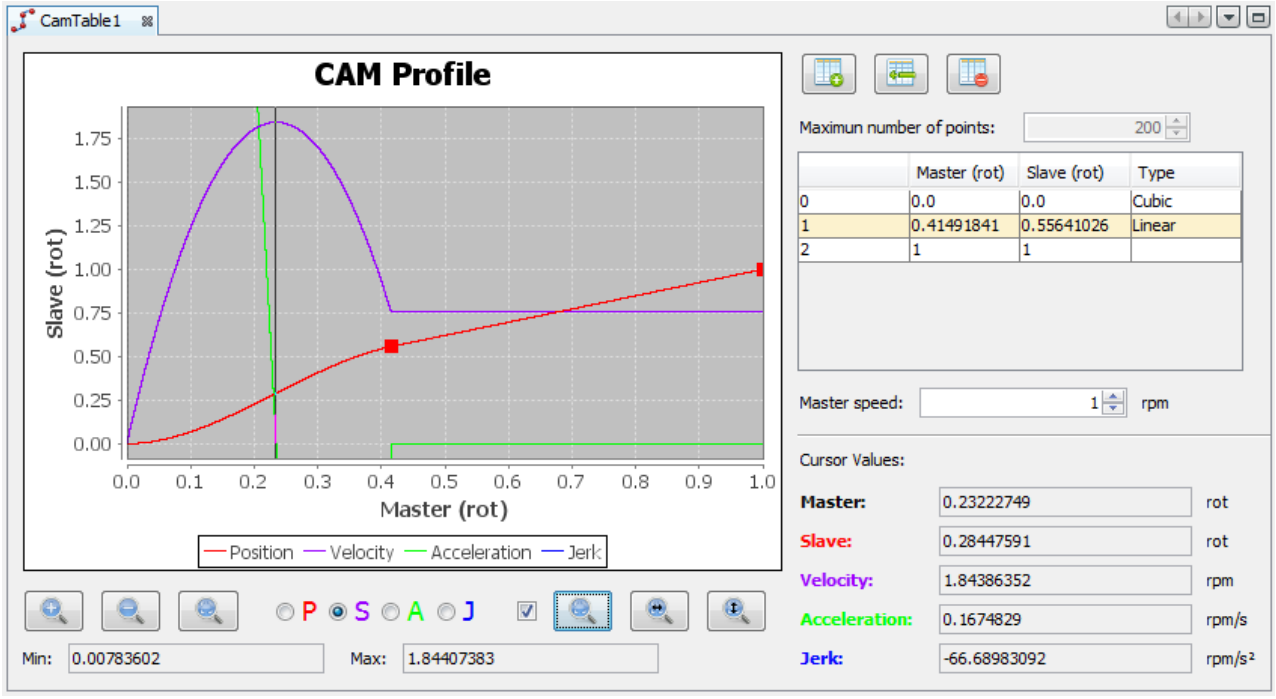
Now, in this curve, it is already possible to see other magnitudes besides the position, such as speed, acceleration and jerk. For a better view of all magnitudes, we can use the **Set Zoom All** button according to the figure below.



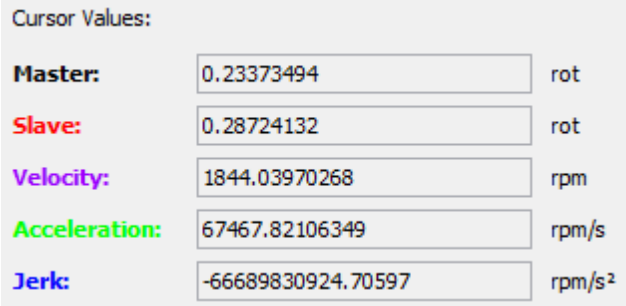
The same way, we can choose one of the magnitudes and use the **Apply Selected Zoom** button. In the example below, a zoom was applied to the speed.



Another interesting tool is the cursor. In the example below, we will place the cursor in point of maximum speed.

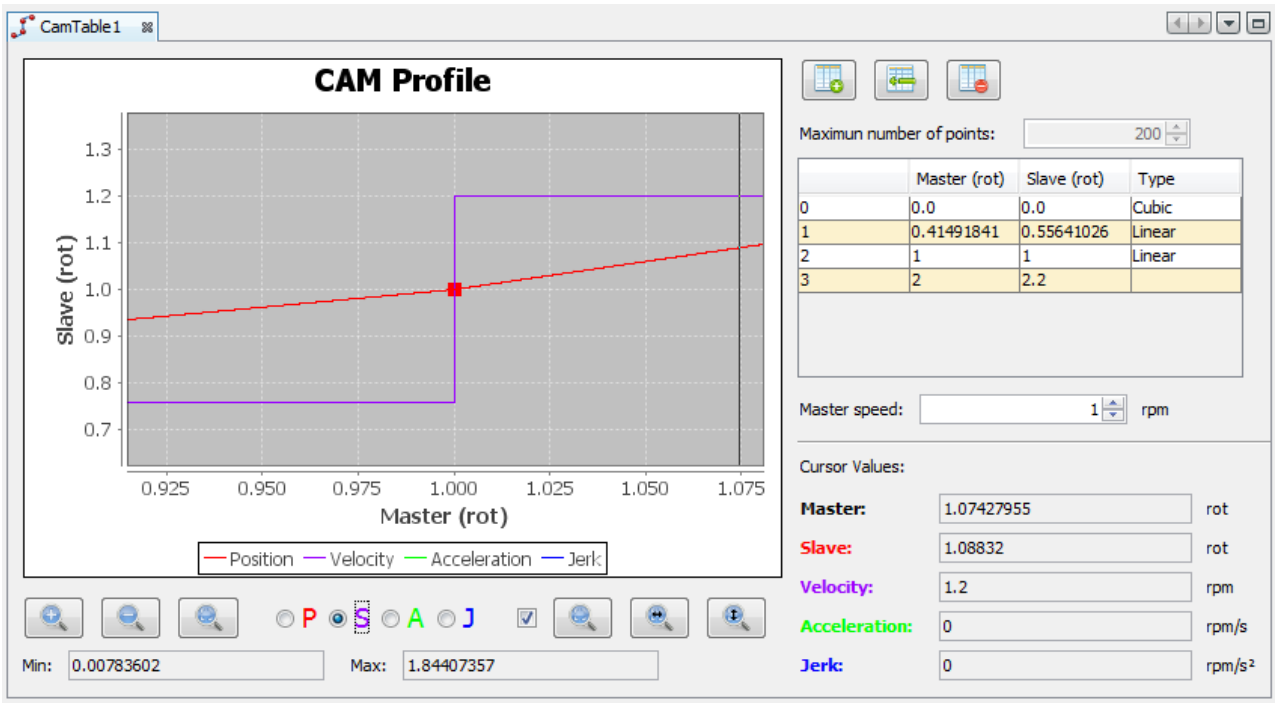


You must bear in mind that the speed, acceleration and jerk of the slave depend on the master speed; therefore, it is interesting to change them so as to simulate something really close to the effective values. In the figure below, the master speed will be changed to 1000 rpm and we will analyze the same position of the cursor.



During the project of the cam profile, all those magnitudes must be observed, because they may be accomplished or not due to mechanical, electrical and electronic limitations of the involved equipment.

Since the acceleration and jerk graphics are calculated taking into account the interpolation between two points, the acceleration and jerk will be shown as equal to zero in the junctions between linear interpolations. Although in theory we know that in a speed step the acceleration and jerk are infinite, in practice the acceleration and jerk at this moment will also depend on the mechanical, electrical and electronic limitations of the involved equipment. Those speed steps must be observed and considered in the project of the cam profile. The figure below shows an example of this situation.



The CAM block offers two types of interpolation: linear and cubic. The following equations are used:

Linear:

$$pe = pie * \left(\frac{pfm - pm}{pfm - pim} \right) + pfe * \left(\frac{pm - pim}{pfm - pim} \right)$$

$$ve = \left(\frac{-pie}{pfm - pim} + \frac{pfe}{pfm - pim} \right) * vm$$

$$ae = 0$$
$$je = 0$$

Cubic:

$$pe = a*(pm - pim)^3 + b*(pm - pim)^2 + c*(pm - pim) + pie$$
$$ve = (3*a*(pm - pim)^2 + 2*b*(pm - pim) + c)*vm$$
$$ae = (6*a*(pm - pim) + 2*b)*vm^2$$
$$je = 6*a*vm^3$$

where:

pe = slave position
ve = slave speed
ae = slave acceleration
je = slave jerk
pm = master position
vm = master speed
pim = master initial position
pfm = master final position
pie = slave initial position
pfe = slave final position
a = coefficient calculated by the CAM editor
b = coefficient calculated by the CAM editor
c = coefficient calculated by the CAM editor

Changing a point in the cam profile

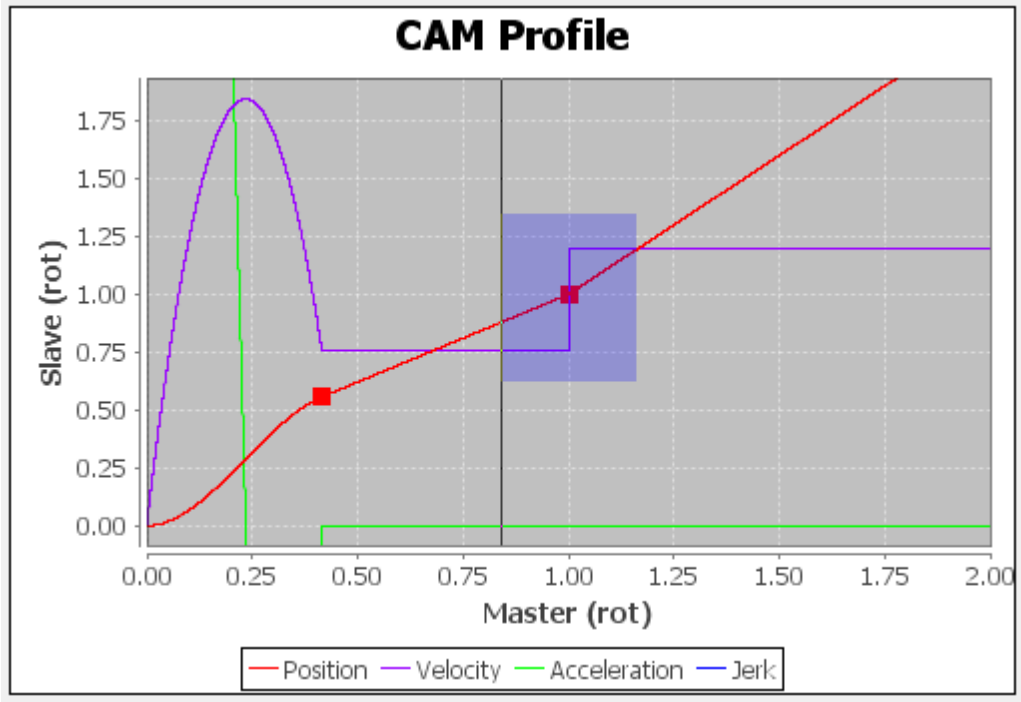
A point can be changed by means of the cam table by using direct edition or moving the point in the graphic. In order to move the point in the graphic, place the cursor on the point, which is marked with a red square, click and hold it, and drag it to the new position.
When you click on the point, the cam table will move to this point, selecting the related cell.
The operation of moving the point in the graphic is interactive and calculates all the profile each time the point is changed. The new point can be seen in the cam table.

Removing a point from the cam profile

The point is removed directly in the cam table. In order to do so, select one of the cells referring to the point and click on the **Remove Point** button.

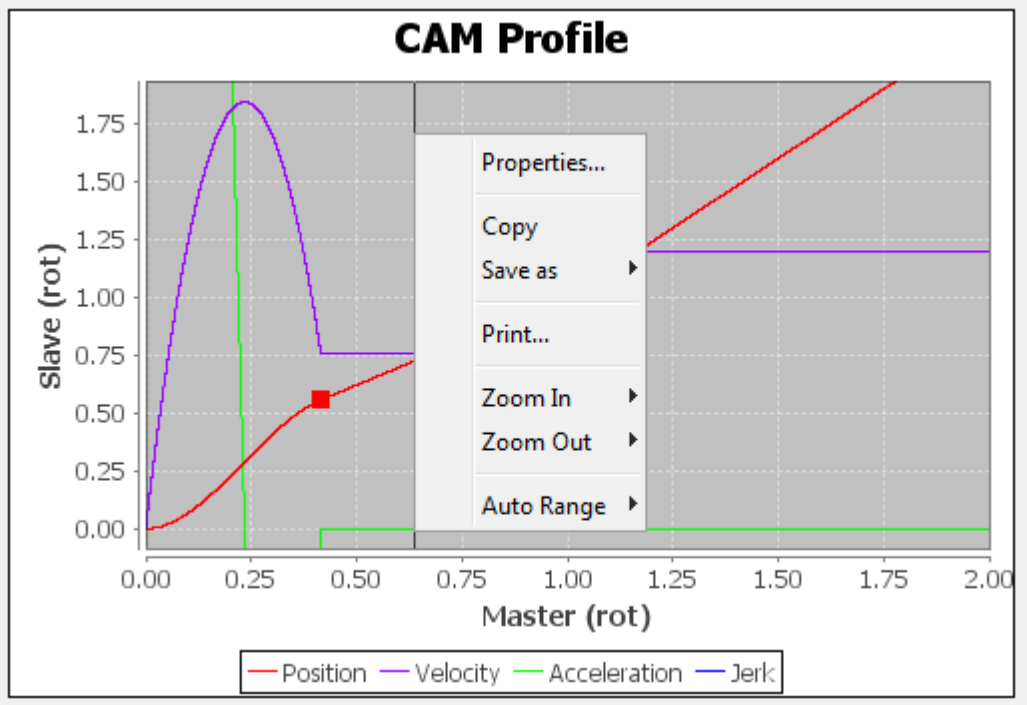
Zoom of a certain area of the graphic

Click on one of the corners of the region you wish to zoom and hold it, and move the mouse so as to mark a region. Then a rectangle will show on the graphic; release the button. The figure below shows an example of this zoom.

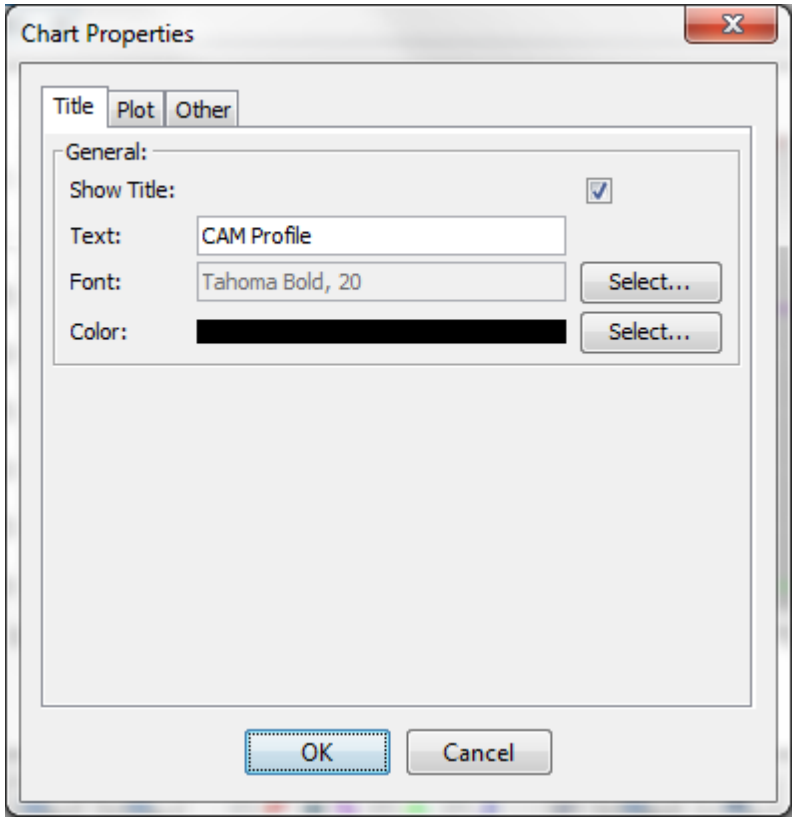


Graphic menu

In order to access the graphic menu, right click on the graphic area, and the following menu will show.



The figure below shows the graphic property box.



11.10.7.18 Structures

Structure is a data grouping used to define a recipe or an object.

In the Ladder program, it is possible to create variables of the structure type and use them in the blocks. To access the internal members of the structure, the '.' is used followed by its respective member.

Creating a structure

1. With the right button of the mouse on the folder **Structure**, click on **New file**.

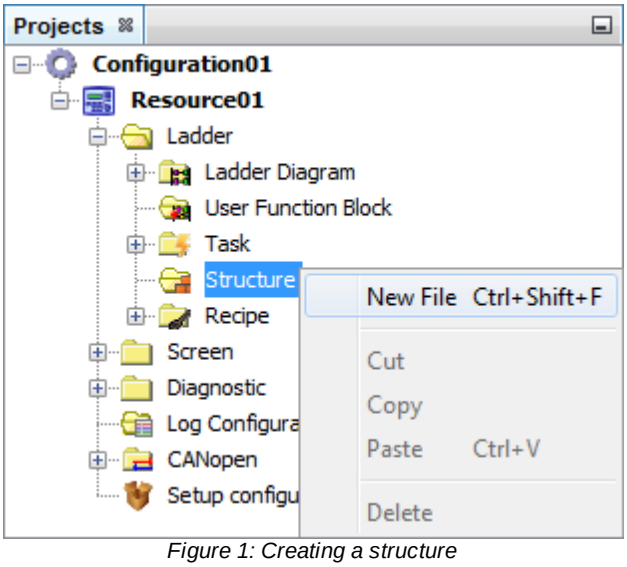


Figure 1: Creating a structure

2. Define the file name and press the **Next** button.

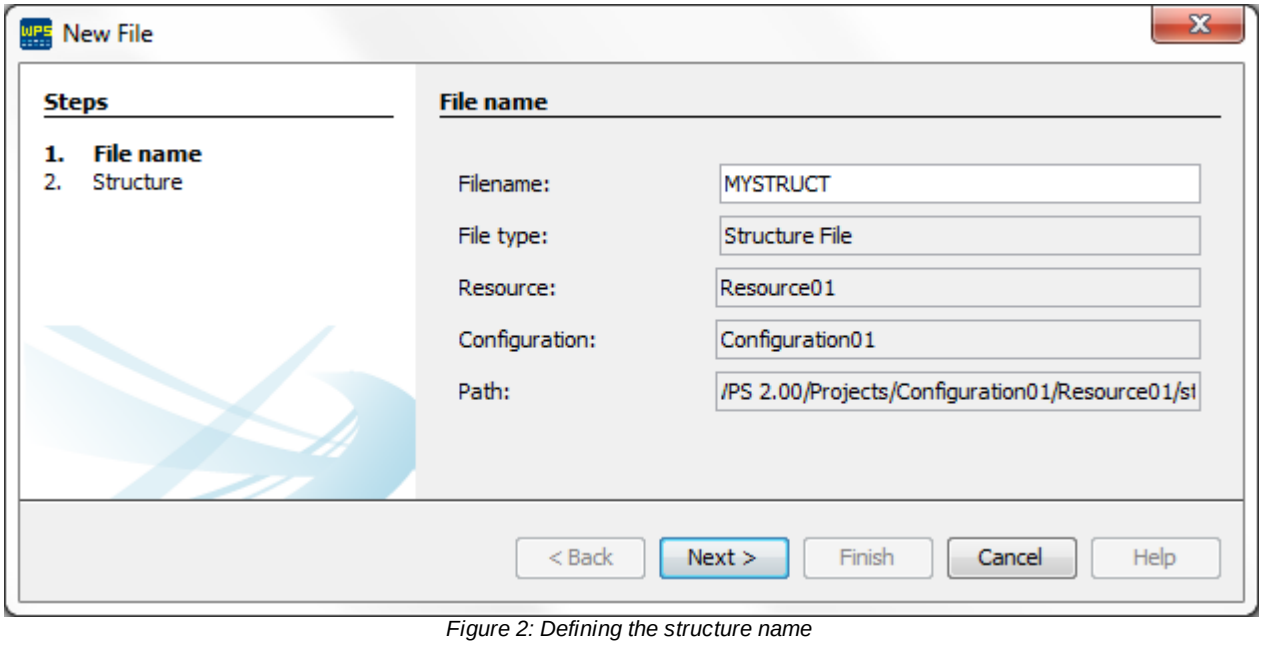


Figure 2: Defining the structure name

3. Configure the structure using the buttons presented in the figure below.

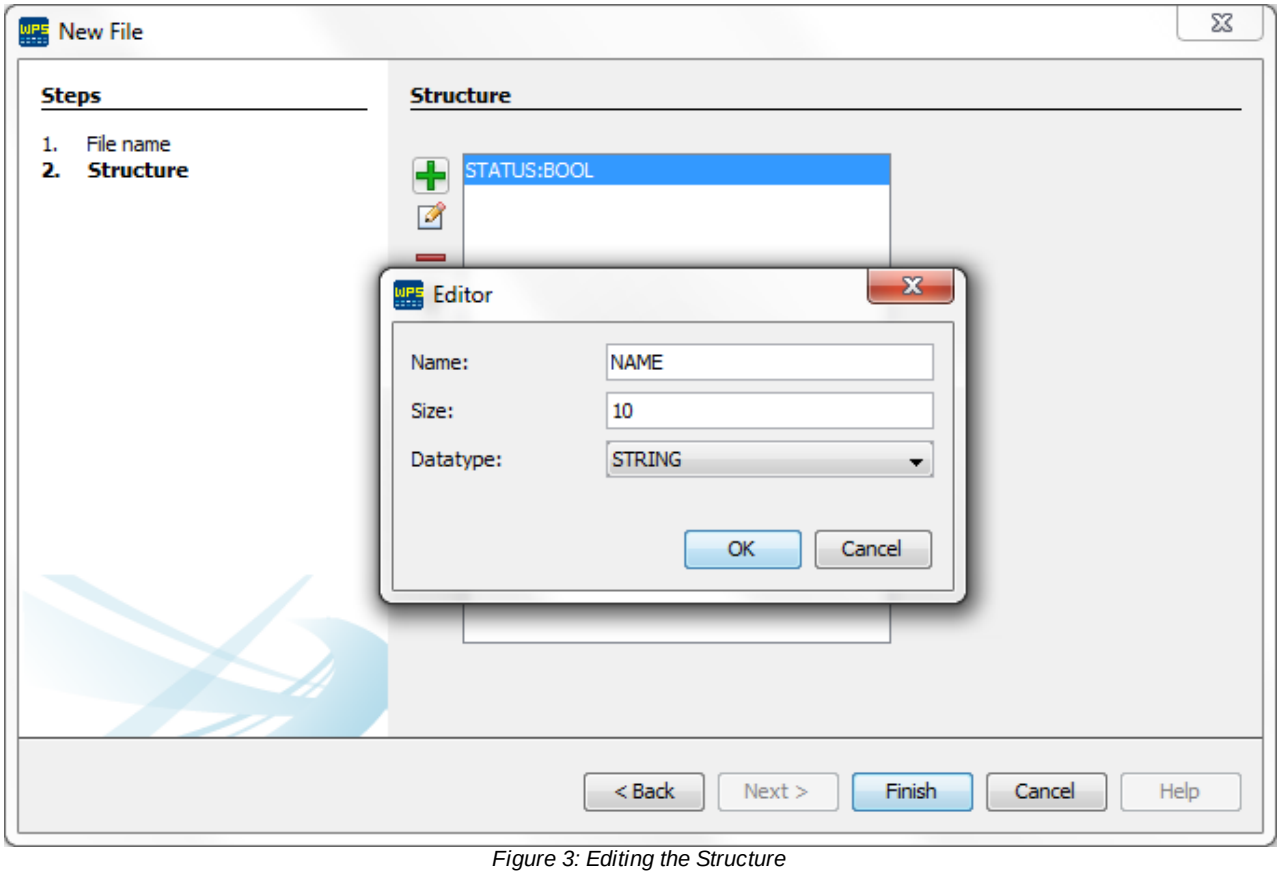


Figure 3: Editing the Structure

4. After finishing the edition of the structure, click on the button **Finish**.

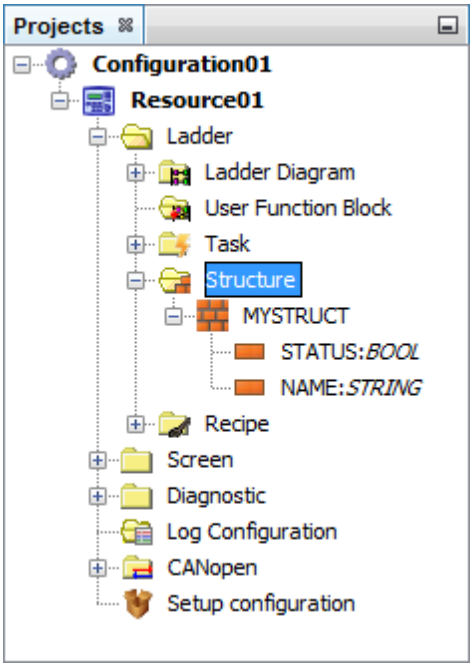


Figure 4: Structure created in the project

Editing a structure

Just double click on the desired structure, as shown in figure 4, and a window will open as shown in figure 3, allowing to insert new data, erase or move the position of the data.

11.10.8Diagnostic

11.10.8.1 Monitoring Panel

Main Signals

11.10.8.1.1 Main Signals

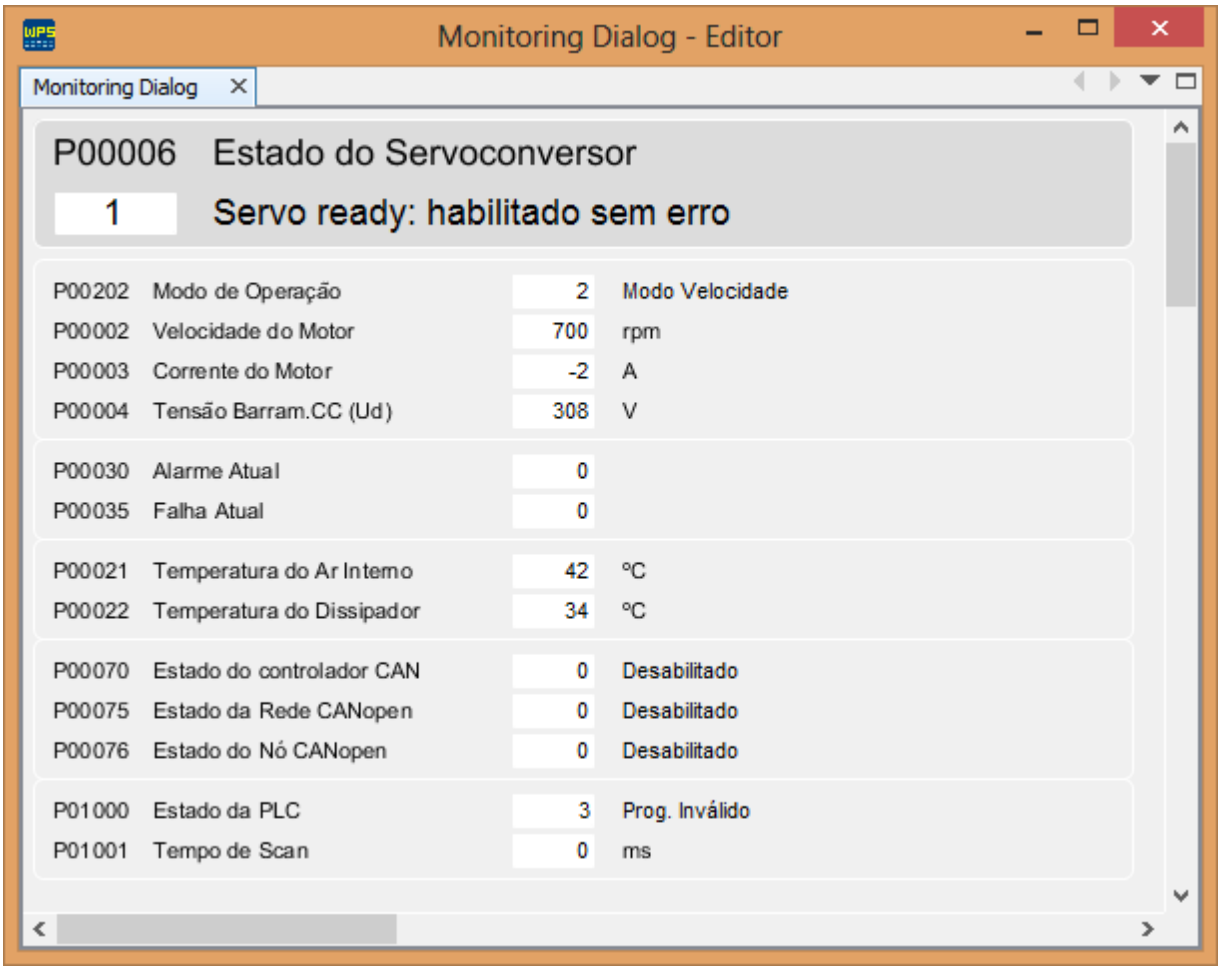
The Main Signals window provides a general view of the main signals of the equipment:

- Drive status (P00006),
- General indications (P00202, P00002, P00003 and P00004),
- Alarm (P00030) and fault (P00035),
- Temperatures (P00021 and P00022),
- Network status (P00070, P00075 and P00076), and
- PLC information (P01000 and P01001).

The main signals are detailed below.

Parameter	Function	Description
P00006	Servodrive Status	It indicates the current status of the servodrive
P00202	Operation Mode	It defines the operation mode of the servodrive, that is, which variable will be controlled: Torque, Speed or if the control will be done via Ladder, CANopen or Profibus
P00002	Motor Speed	It indicates the effective speed value in rpm, except when programmed to receive external position / speed feedback
P00003	Motor Current	It indicates the output Iq current, in amperes rms, of the servodrive
P00004	DC Link voltage	It indicates the present voltage on the DC Link in volts (V)
P00030	Present Alarm	It indicates the number of the alarm which may be present on the servodrive
P00035	Present Fault	It indicates number of the fault which may be present on the servodrive.
P00021	Internal Air Temperature	This parameter presents, in Celsius degrees, the internal air temperature
P00022	Heatsink Temperature	This parameter presents, in Celsius degrees, the heatsink temperature
P00070	CAN Controller Status	It indicates the CAN controller status, responsible for sending and receiving CAN telegrams
P00075	CANopen Network Status	It indicates the CANopen communication status, informing if the protocol was initialized correctly and the status of the slave guarding service
P00076	CANopen Node Status	Each device in the CANopen network has an associated status. It is possible to see the present status of the servodrive through this parameter
P01000	PLC Status	It allows the user to view the program status
P01001	Scan Time	It allows the user to monitor the scan cycle time of the program in milliseconds

The window may be viewed below.



11.10.8.2 Log

Overview

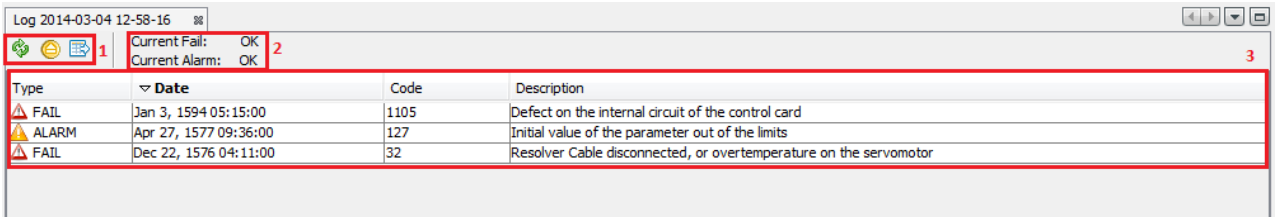
Configuration

11.10.8.2.1 Overview

The log function allows viewing the present alarms/faults and also the last alarms/faults in a more friendly and centralized way than on the equipment HMI, showing the data in tabular form.

All downloaded alarms/faults are saved in a file in order to keep a record for future reference; it is also possible to export the saved data as a csv file.

Below is an overview of the SCA06 log table.

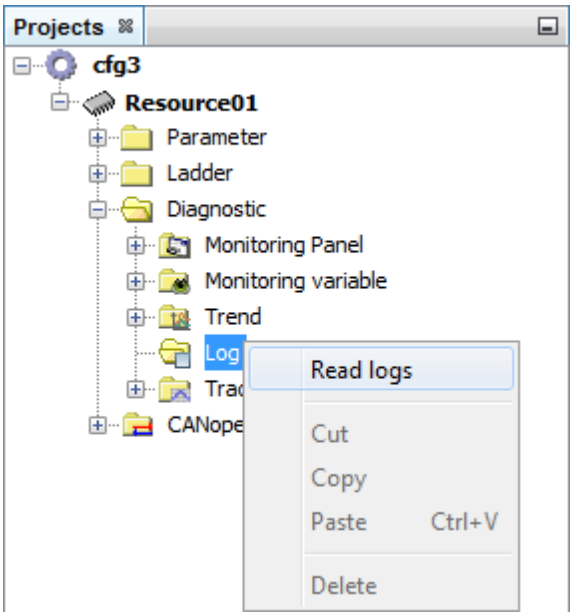


1. **Actions**. Below is the sequential description of each action:
   1.1. **On-Line Reading**: This action constantly searches for new faults/alarms on the equipment; in case any occurs, the table is automatically updated.
   1.2. **Read Logs**: This action reads all faults/alarms only once and updates the table.
   1.3. **Export**: Export of data shown in the table as a CSV file.
2. **Present Fault and Alarm**. This field shows the present fault and alarm of the equipment if present; otherwise, the **OK** message is shown.
3. **Fault and Alarm Table**. This table shows all the faults and alarms, as well as the date when they occurred, code and description.

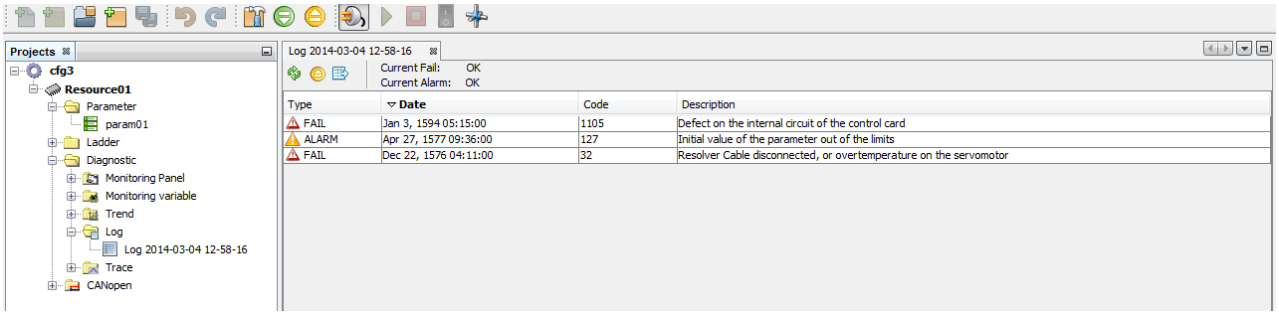
11.10.8.2.2 Configuration

In order to create a new log file, just execute the operation below:

1. Create a new log file by selecting the **Read Logs** option; it is necessary to be connected to the device to execute this operation.



2. After the read logs step is completed, the log file will be automatically created with a name composed of the present data and time and with all the alarms and faults present on the equipment read.



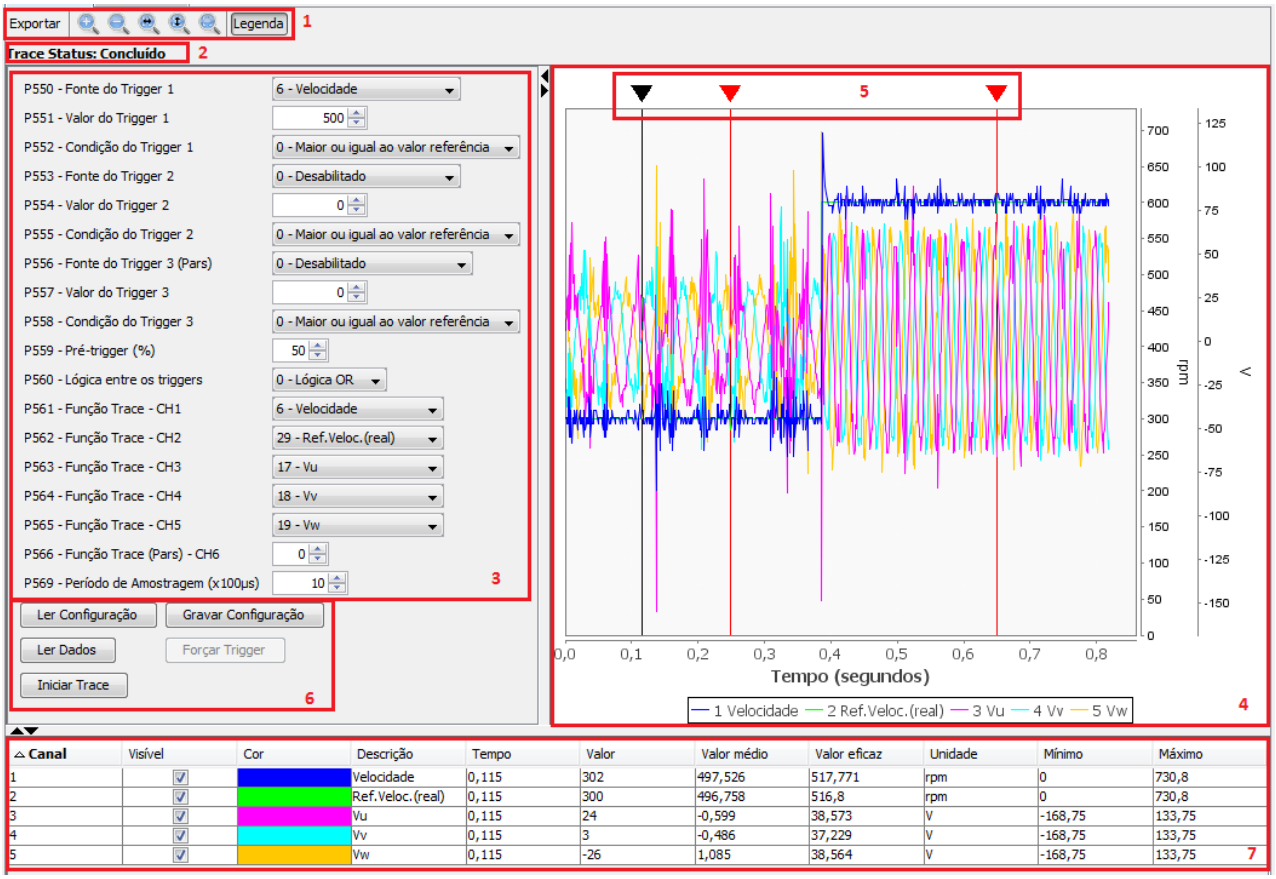
11.10.8.3 Trace

11.10.8.3.1 Overview

The trace function is used to register variables* of interest of the device (such as current, voltage, speed, etc.) when a certain event occurs in the system. Since it triggers the storage of the variables, in the system this event is called trigger, and the user can define up to three trigger conditions and the logic to be used in them (AND or OR logic).
The stored variables can be seen as graphics by using the WPS running on a PC connected via USB or via serial to the device.

NOTE: Up to 6 (six) channels using SCA06; Up to 4 (four) channels using CFW-11.

Below is an overview of the configuration screen of the trace function (example using SCA06).

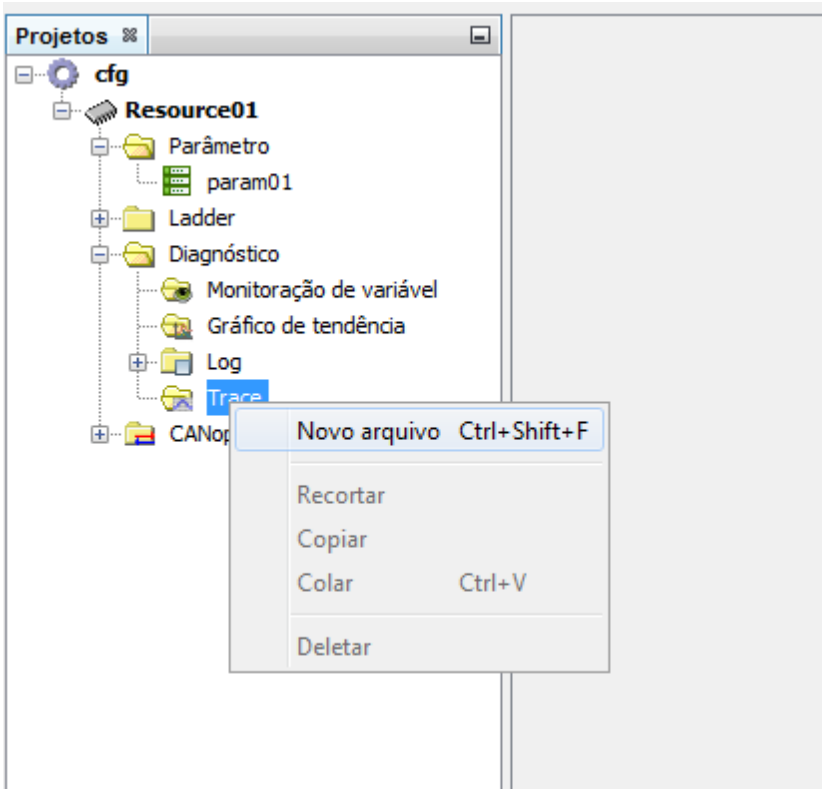


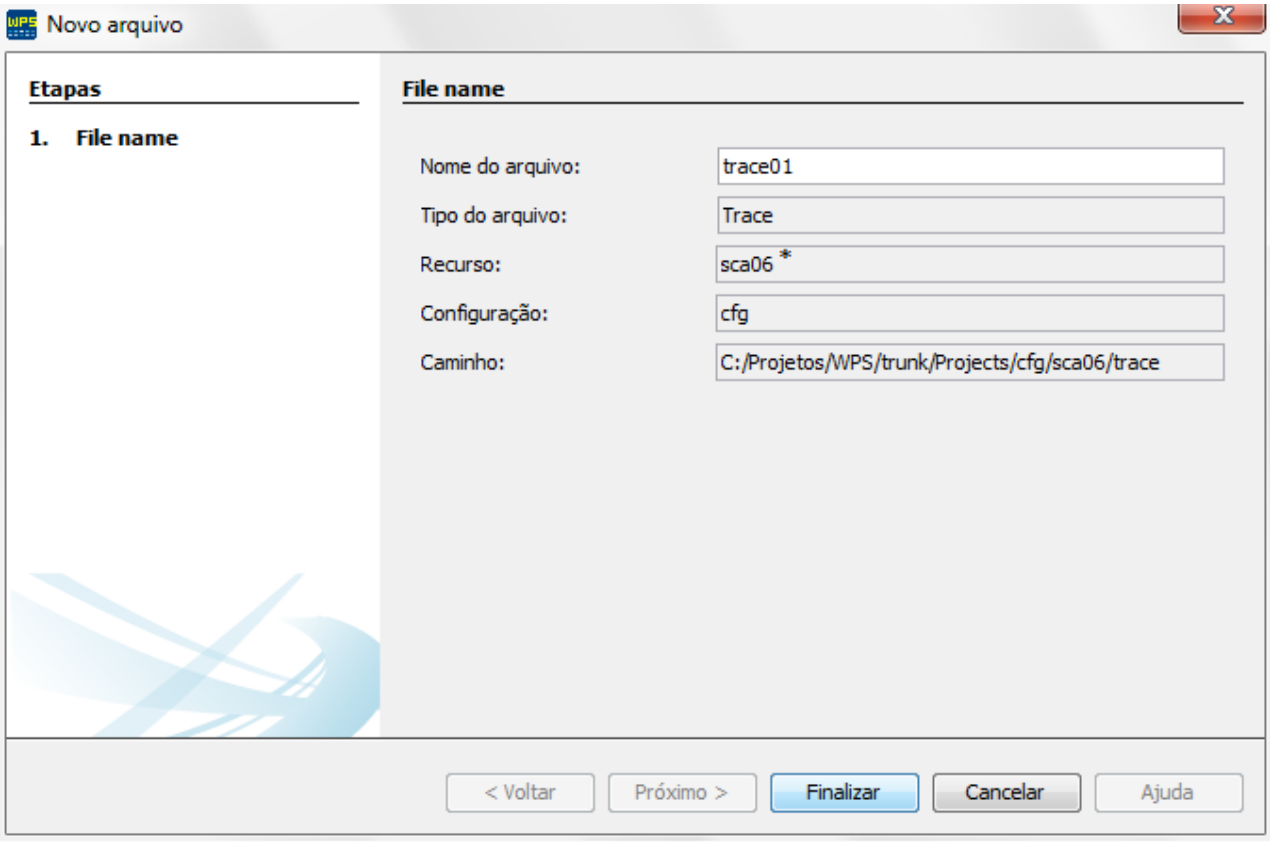
1. **Graphic Zoom**. This bar contains the options to control the graphic, such as export to image file, zoom in, zoom out, set width, set height, se all and show or not show the graphic lettering.
2. **Trace Status**. This item shows the present status of the trace function: not started, trigger occurred and concluded.
3. **Parameters**. In this part are all the parameters that can be configured in the trace routine, such as triggers, conditions, channels to be monitored and sampling period.
4. **Graphic**. In this area is the graphic after the conclusion of trace. In the lower part is the time line and on the right are the values separated by unit of measurement.
5. **Markers**. The markers are within the graphic area. After the graphic is set, just click on the black marker to create red markers (fixed). It is possible to add two fixed markers. Those fixed markers are used to calculate the average and effective values between the two points.
6. **Trace command**. Below is the description of the command functions:
   6.1. **Read configuration**: It reads the trace configuration parameters and updates the parameters on the screen (item 3).
   6.2. **Save configuration**: It sends the trace configuration parameters (item 3) to the equipment.
   6.3. **Read Data**: Command used only when the trace status is concluded, that is, there is already a concluded trace on the equipment, and you just wish to download the data without starting a new trace.
   6.4. **Force Trigger**: Forces the trigger regardless the conditions.
   6.5. **Start Trace**: It starts the trace function.
7. **Channel Table**. This table shows the data of the chosen channels, besides the possibility to hide channels (Visible), change the channel color (Color) and set the graphic limits per unit of measurement (Maximum).

11.10.8.3.2 Configuration

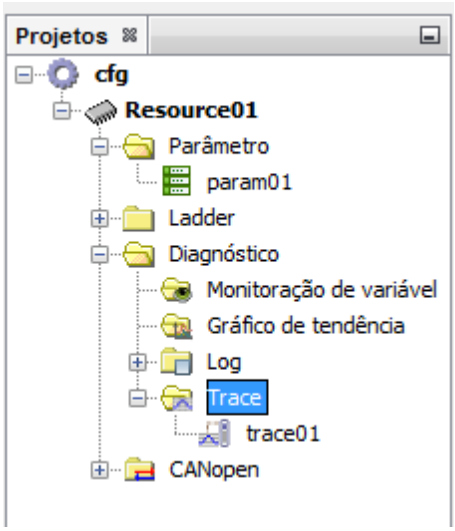
Below is a list of the necessary steps to create a trace configuration:

1. Creation of a new trace file.

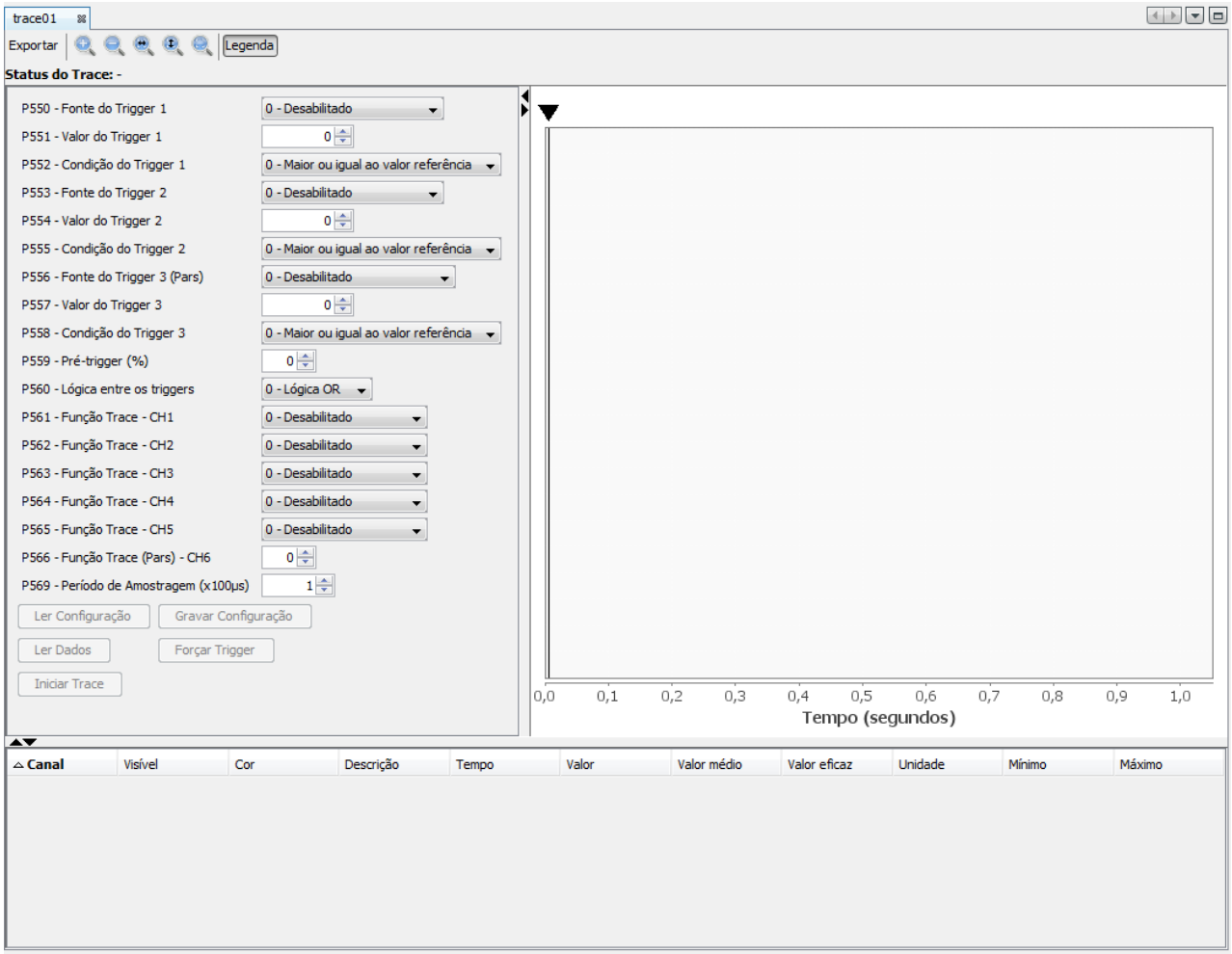




* **Resource:** Resource01, SCA06, CFW300, etc.

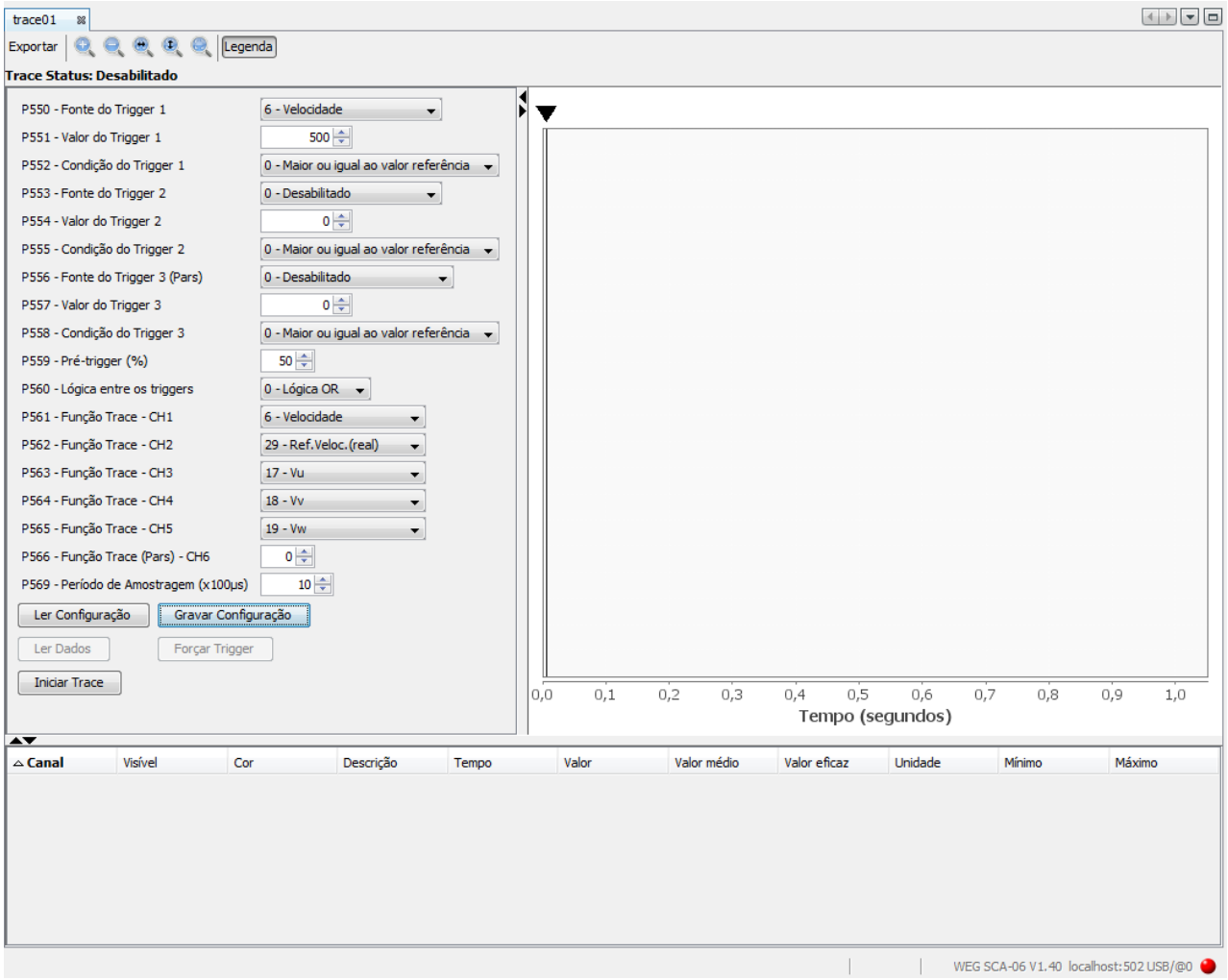


2. After the creation of the trace file, it is necessary to set the desired configurations in the part of parameters.

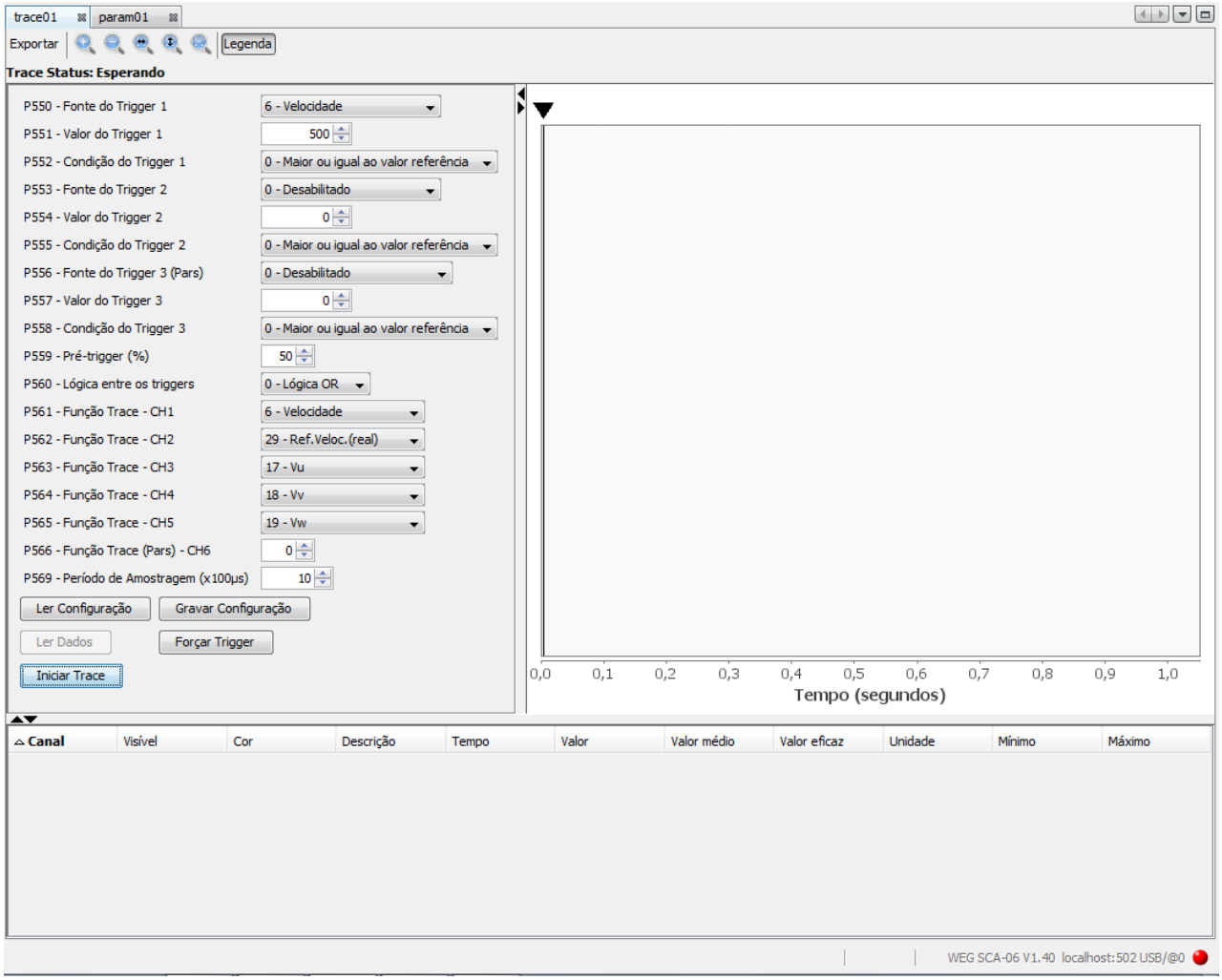


3. After making the desired configurations, just click on save configuration to send them to the equipment.

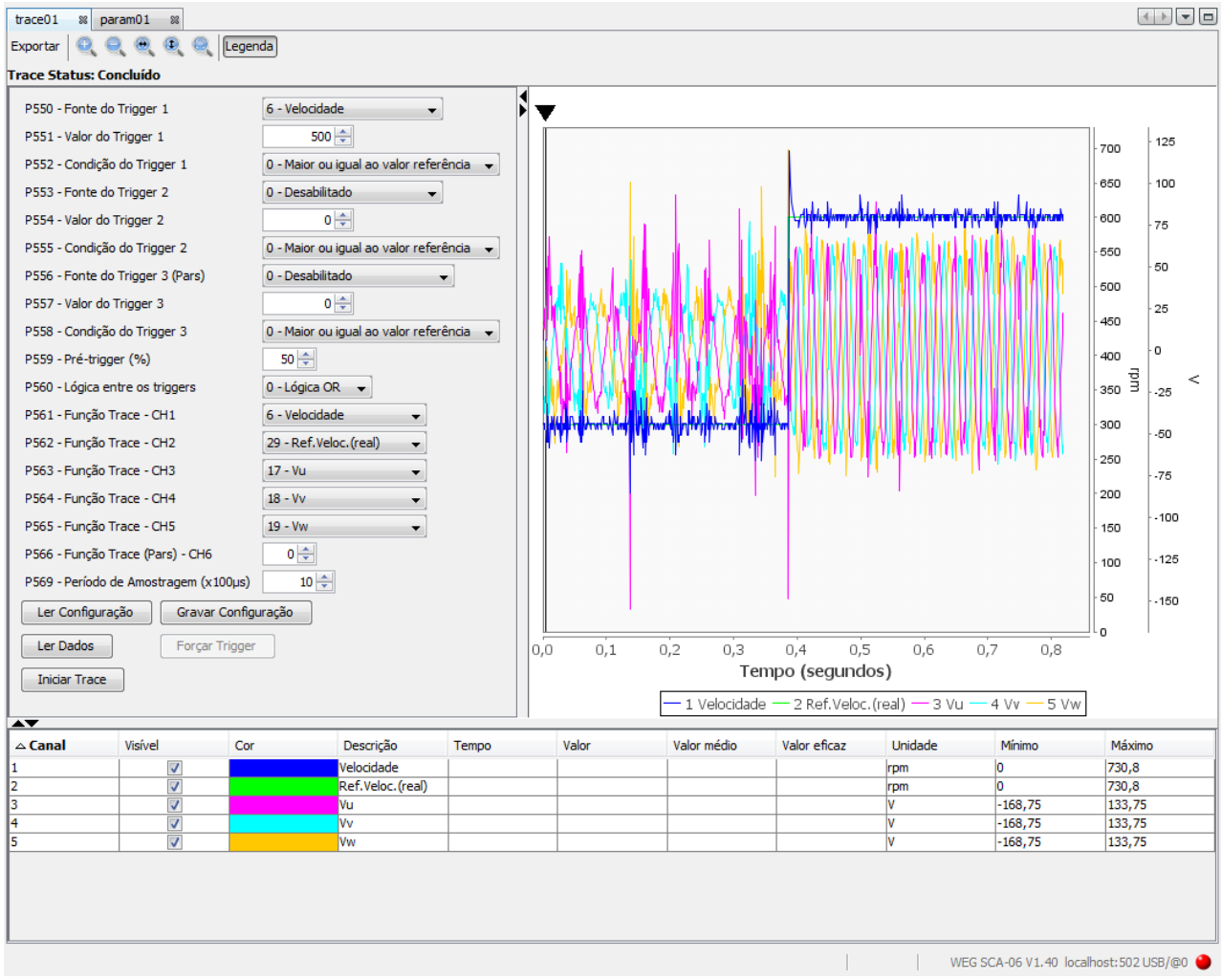
Notice that it is necessary to be connected to the equipment with the option **Connect Device** of the WPS.



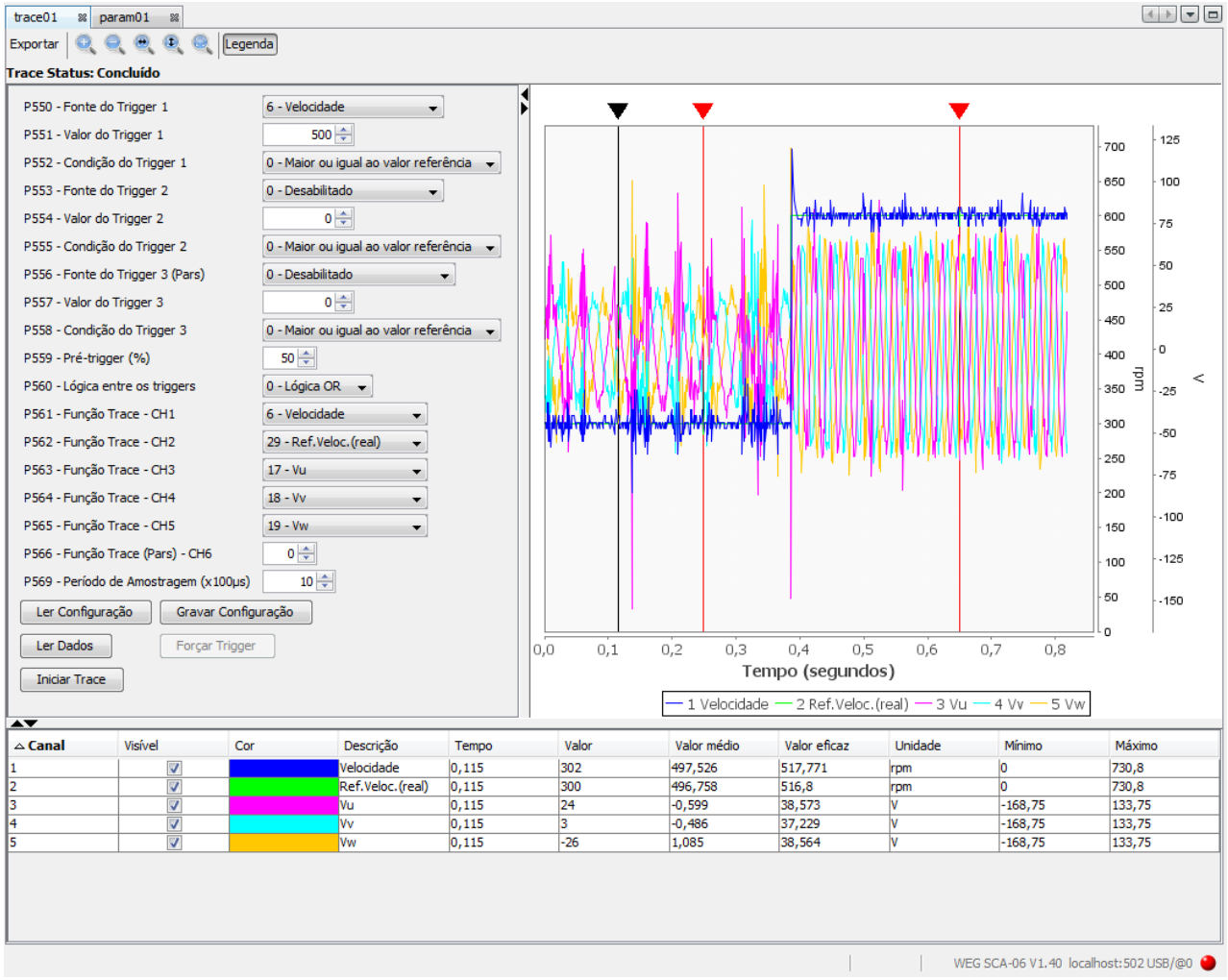
4. After the configurations are saved, just click on **Start Trace**. Notice that the status of the trace function changed to **Waiting**, that is, the tool is now waiting for the trigger execution to set the graphic and show the trace values.



5. After trigger occurs, the graphic and the values will be shown in the table and the trace function status will be **Concluded**.



6. If you wish, you can click on the black cursor of the graphic and add fixed cursors so that the calculation of the average and effective values will be performed for the channels in the defined ranges.



11.11 SSW-06

Enter topic text here.

11.11.1 Description

Soft-Starters are static starters that accelerate, decelerate and protect three-phase induction motors. The control of the voltage applied to the motor by means of adjustments to the firing angle of thyristors allows the soft-starter to start and stop an electric motor smoothly.

The SSW-06, with DSP (Digital Signal Processor) control was designed for high performance on motor starts and stops with an excellent cost-benefit ratio. Easy to set up, it simplifies start-up activities and daily operation.

The SSW-06 is compact optimizing space in electric panels. It already incorporates electric motor protection. It adapts to customer needs through its easy-to-install optional accessories.

The SSW-06 adapts to customer needs through its easy-to-install optional accessories. Thus, a keypad, a communication interface or a motor PTC input can be added to the product.

Refer to the user's manual of the SSW-06 for further details about the product.

NOTE!

This product does not have the Ladder tool available in WPS.
You can use the WLP application if this feature is required.

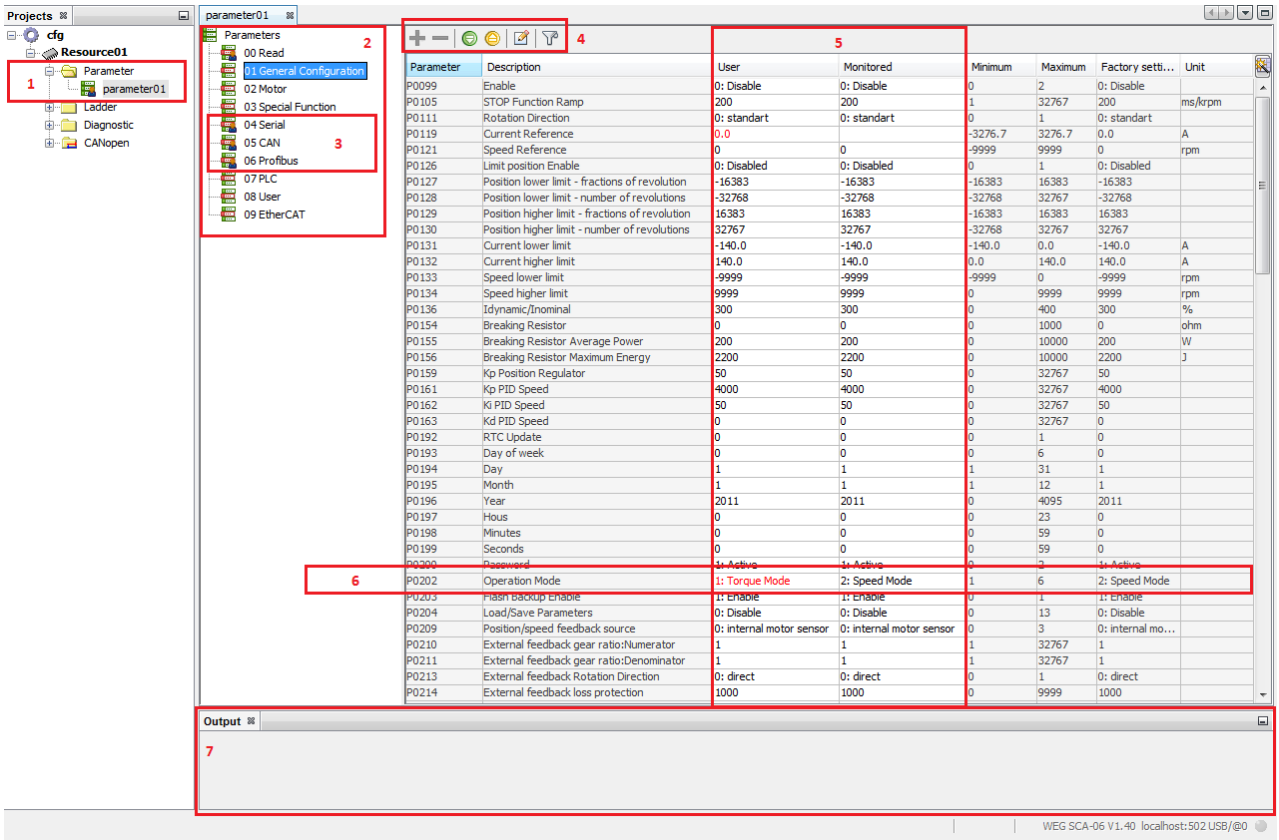
11.11.2Parameters

11.11.2.1 Overview

The parameter configuration screen is used to configure and monitor all the parameters of the equipment, including the user parameters.

NOTE!
The reading and writing of such parameters is done on this screen; only the user parameter configuration must be sent the first time or whenever modified by means of the resource download routine.

Below is an overview of the parameter configuration screen.



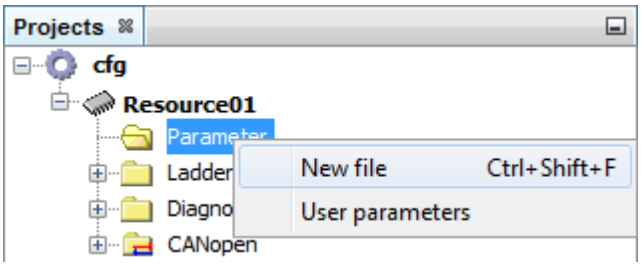
Parameter	Description	User	Monitored	Minimum	Maximum	Factory setti...	Unit
P0099	Enable	0: Disable	0: Disable	0	2	0: Disable	
P0105	STOP Function Ramp	200	200	1	32767	200	ms/krpm
P0111	Rotation Direction	0: standart	0: standart	0	1	0: standart	
P0119	Current Reference	0.0		-3276.7	3276.7	0.0	A
P0121	Speed Reference	0	0	-9999	9999	0	rpm
P0126	Limit position Enable	0: Disabled	0: Disabled	0	1	0: Disabled	
P0127	Position lower limit - fractions of revolution	-16383	-16383	-16383	16383	-16383	
P0128	Position lower limit - number of revolutions	-32768	-32768	-32768	32767	-32768	
P0129	Position higher limit - fractions of revolution	16383	16383	-16383	16383	16383	
P0130	Position higher limit - number of revolutions	32767	32767	-32768	32767	32767	
P0131	Current lower limit	-140.0	-140.0	-140.0	0.0	-140.0	A
P0132	Current higher limit	140.0	140.0	0.0	140.0	140.0	A
P0133	Speed lower limit	-9999	-9999	-9999	0	-9999	rpm
P0134	Speed higher limit	9999	9999	0	9999	9999	rpm
P0136	Idynamic/Inominal	300	300	0	400	300	%
P0154	Breaking Resistor	0	0	0	1000	0	ohm
P0155	Breaking Resistor Average Power	200	200	0	10000	200	W
P0156	Breaking Resistor Maximum Energy	2200	2200	0	10000	2200	J
P0159	Kp Position Regulator	50	50	0	32767	50	
P0161	Kp PID Speed	4000	4000	0	32767	4000	
P0162	Ki PID Speed	50	50	0	32767	50	
P0163	Kd PID Speed	0	0	0	32767	0	
P0192	RTC Update	0	0	0	1	0	
P0193	Day of week	0	0	0	6	0	
P0194	Day	1	1	1	31	1	
P0195	Month	1	1	1	12	1	
P0196	Year	2011	2011	0	4095	2011	
P0197	Hous	0	0	0	23	0	
P0198	Minutes	0	0	0	59	0	
P0199	Seconds	0	0	0	59	0	
P0200	Password	1: Active	1: Active	0	3	1: Active	
P0202	Operation Mode	1: Torque Mode	2: Speed Mode	1	6	2: Speed Mode	
P0203	Flash Backup Enable	1: Enable	1: Enable	0	1	1: Enable	
P0204	Load/Save Parameters	0: Disable	0: Disable	0	13	0: Disable	
P0209	Position/speed feedback source	0: internal motor sensor	0: internal motor sensor	0	3	0: internal mo...	
P0210	External feedback gear ratio:Numerator	1	1	1	32767	1	
P0211	External feedback gear ratio:Denominator	1	1	1	32767	1	
P0213	External feedback Rotation Direction	0: direct	0: direct	0	1	0: direct	
P0214	External feedback loss protection	1000	1000	0	9999	1000	

1. **Parameter files**. In this part are all the parameter configuration files created by the user. Notice that when the file features a person figure on the table, it means this parameter table contains hidden parameters/ group of parameters.
2. **Group of parameters**. This tree shows all the group of parameters. Notice that the same parameter can be in more than one group, and when its value is modified, it will be modified in all the groups to which it belongs.
3. **Modified group of parameters**. Group of parameters which contain the figure of a person on the table means they have hidden parameters.
4. **Commands**. The commands are described below in the order they appear:
   4.1. **Unhide parameter**: In case some parameter has been hidden, this button allows making it visible again.
   4.2. **Hide parameter**: Just select one or more parameters on the table and trigger this command to hide them.
   4.3. **Save table**: It saves the values of the parameters shown on the equipment screen; the sent values are the ones in the User column. The flow is User -> Monitored (equipment)
   4.4. **Read table**: It reads the parameters of the equipment shown in the Monitored column and saves them in the parameter file in the User column. The flow is Monitored (equipment) -> User
   4.5. **User parameters**: It opens a screen to edit the user parameters.
   4.6. **Filter**: It opens a parameter filter option, and it can filter by parameter number or description.
   4.7. **User Parameters and Monitored Parameters**. These two columns show the off-line and on-line parameters, so to speak. The **User** column shows the values contained in the file located on the computer and the **Monitored** column shows the values that are effectively saved on the equipment. Whenever you use the **Save Parameter** option, the sent values will be from the **User** column to the **Monitored** column, that is, File -> Equipment. In case of reading, the flow is the opposite, from the **Monitored** column to the **User** column, that is, Equipment -> File. In case you wish to change the values directly on the equipment without changing it in the file, just click on the monitored column, change the values and the modification will occur on-line.
5. **Modified parameters**: Whenever a parameter value in the **User** column is different from the **Monitored** column, it will be shown in red.
6. **Output**. This screen shows error information in case they occur during the writing or reading of the parameters.

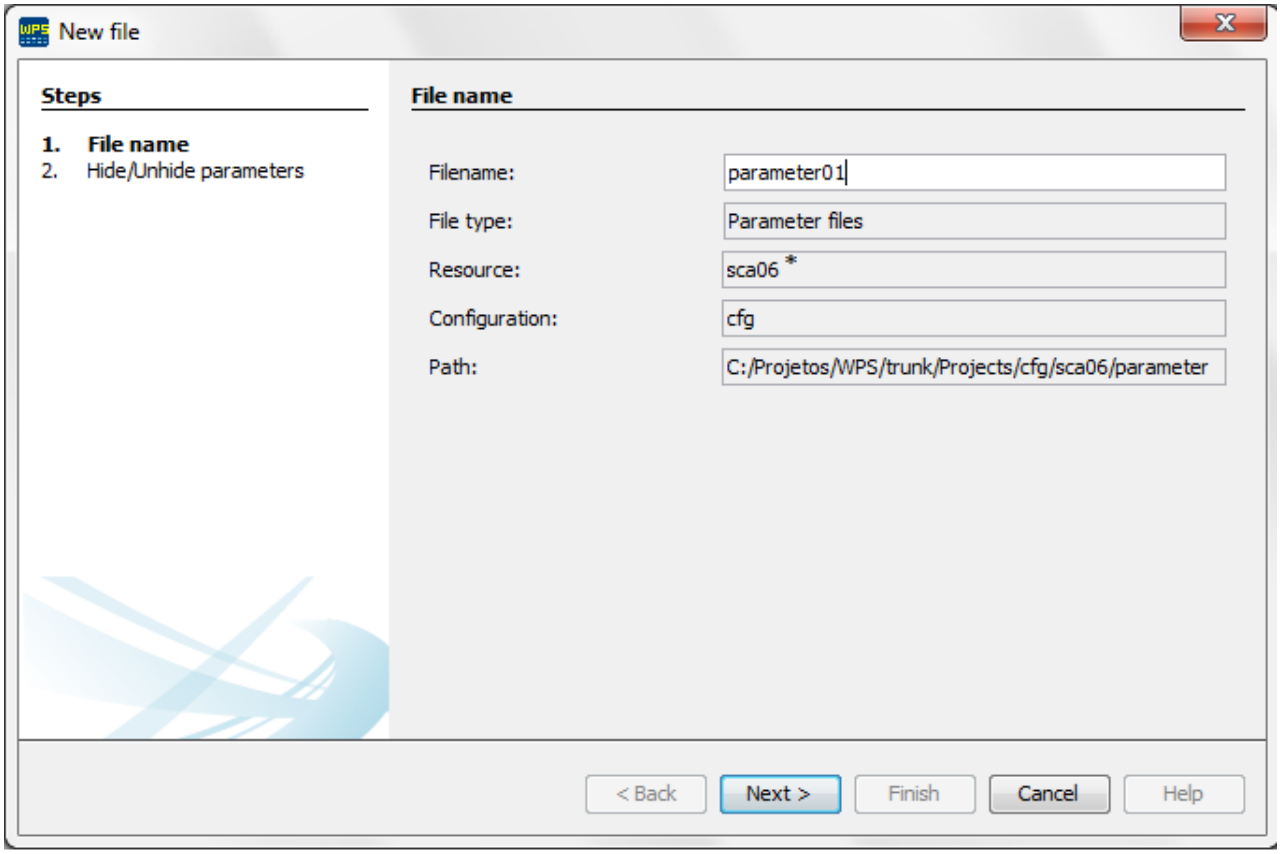
11.11.2.2 Configuration

Below is the list of the required steps to create a parameter file.

1. Create a new parameter file.

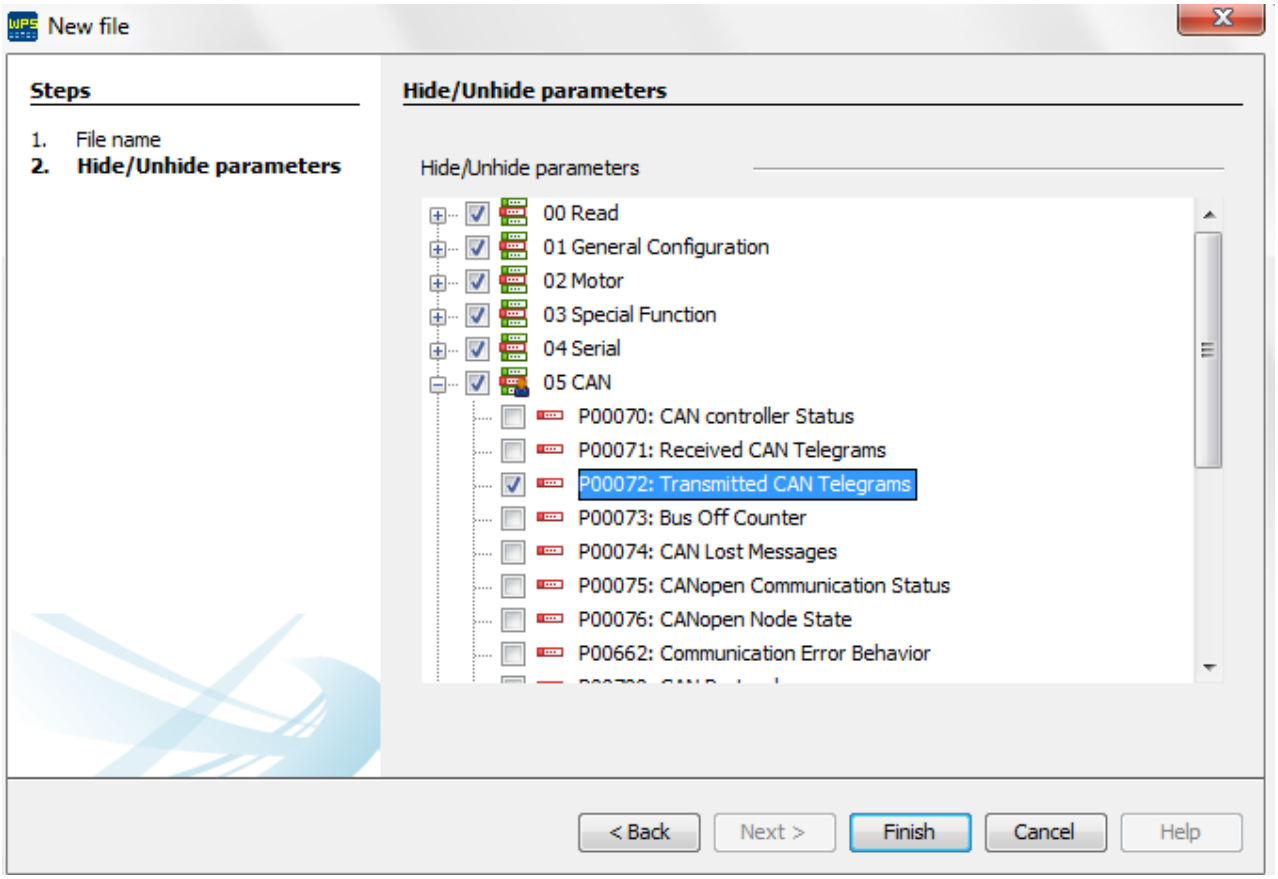


2. Define a name for the parameter file

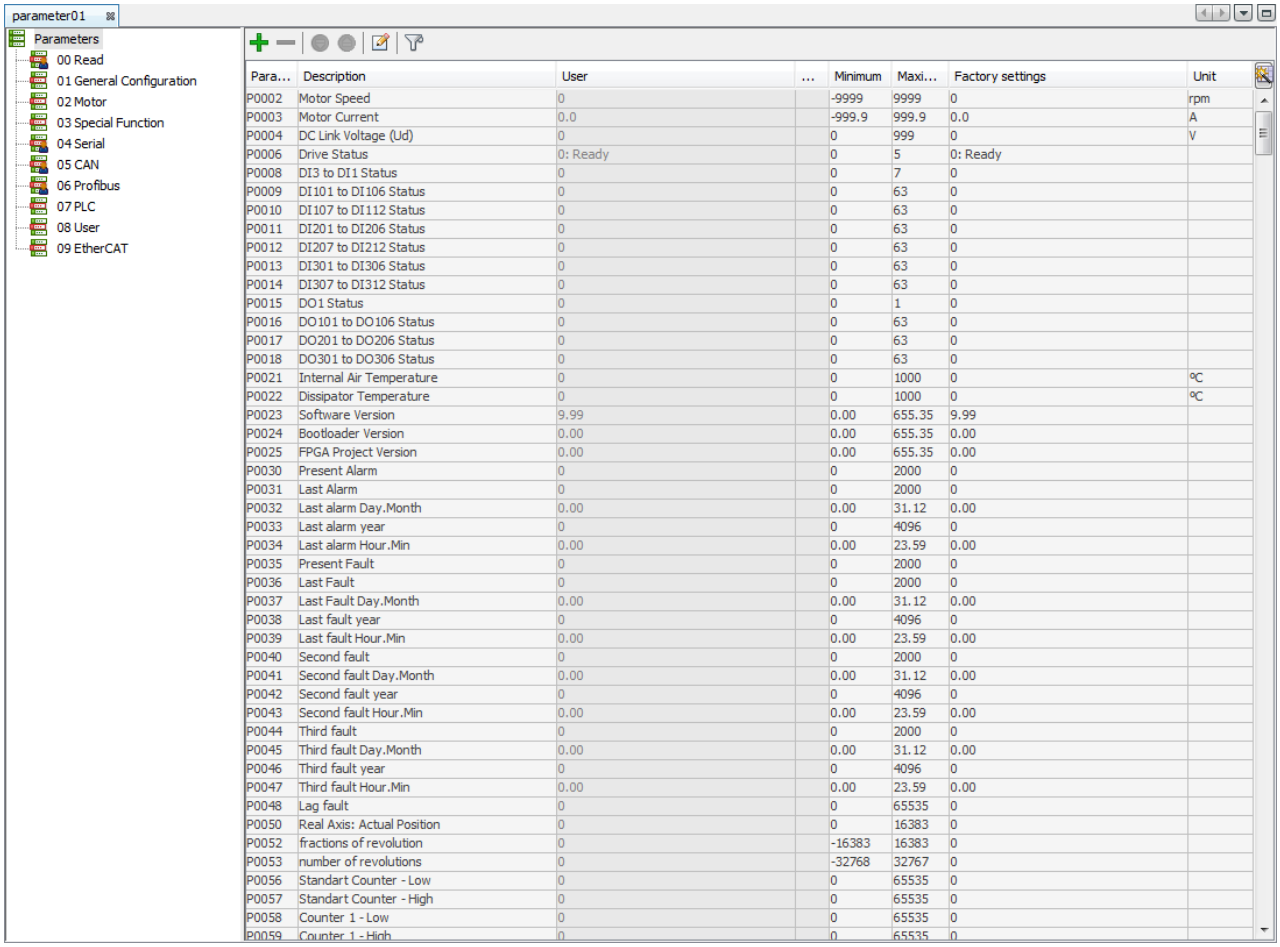


*** Resource: Resource01, SCA06, CFW300, etc.**

3. Configure which parameters you wish to view in your parameter table



4. After performing the steps above, the parameter file will be created and the equipment can be parameterized.

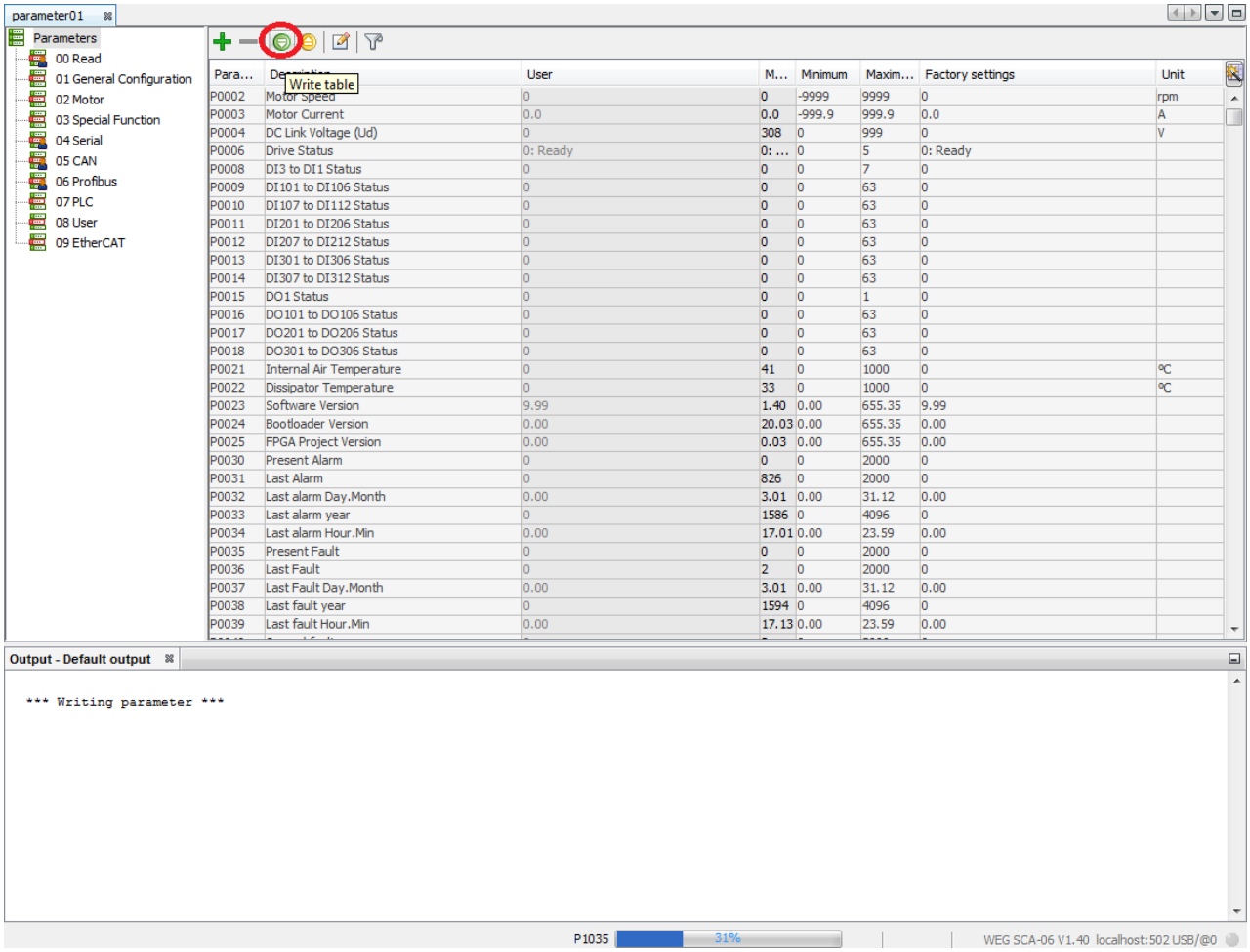


Para...	Description	User	...	Minimum	Maxi...	Factory settings	Unit
P0002	Motor Speed	0		-9999	9999	0	rpm
P0003	Motor Current	0.0		-999.9	999.9	0.0	A
P0004	DC Link Voltage (Ud)	0		0	999	0	V
P0006	Drive Status	0: Ready		0	5	0: Ready	
P0008	DI3 to DI1 Status	0		0	7	0	
P0009	DI101 to DI106 Status	0		0	63	0	
P0010	DI107 to DI112 Status	0		0	63	0	
P0011	DI201 to DI206 Status	0		0	63	0	
P0012	DI207 to DI212 Status	0		0	63	0	
P0013	DI301 to DI306 Status	0		0	63	0	
P0014	DI307 to DI312 Status	0		0	63	0	
P0015	DO1 Status	0		0	1	0	
P0016	DO101 to DO106 Status	0		0	63	0	
P0017	DO201 to DO206 Status	0		0	63	0	
P0018	DO301 to DO306 Status	0		0	63	0	
P0021	Internal Air Temperature	0		0	1000	0	ºC
P0022	Dissipator Temperature	0		0	1000	0	ºC
P0023	Software Version	9.99		0.00	655.35	9.99	
P0024	Bootloader Version	0.00		0.00	655.35	0.00	
P0025	FPGA Project Version	0.00		0.00	655.35	0.00	
P0030	Present Alarm	0		0	2000	0	
P0031	Last Alarm	0		0	2000	0	
P0032	Last alarm Day.Month	0.00		0.00	31.12	0.00	
P0033	Last alarm year	0		0	4096	0	
P0034	Last alarm Hour.Min	0.00		0.00	23.59	0.00	
P0035	Present Fault	0		0	2000	0	
P0036	Last Fault	0		0	2000	0	
P0037	Last Fault Day.Month	0.00		0.00	31.12	0.00	
P0038	Last fault year	0		0	4096	0	
P0039	Last fault Hour.Min	0.00		0.00	23.59	0.00	
P0040	Second fault	0		0	2000	0	
P0041	Second fault Day.Month	0.00		0.00	31.12	0.00	
P0042	Second fault year	0		0	4096	0	
P0043	Second fault Hour.Min	0.00		0.00	23.59	0.00	
P0044	Third fault	0		0	2000	0	
P0045	Third fault Day.Month	0.00		0.00	31.12	0.00	
P0046	Third fault year	0		0	4096	0	
P0047	Third fault Hour.Min	0.00		0.00	23.59	0.00	
P0048	Lag fault	0		0	65535	0	
P0050	Real Axis: Actual Position	0		0	16383	0	
P0052	fractions of revolution	0		-16383	16383	0	
P0053	number of revolutions	0		-32768	32767	0	
P0056	Standart Counter - Low	0		0	65535	0	
P0057	Standart Counter - High	0		0	65535	0	
P0058	Counter 1 - Low	0		0	65535	0	
P0059	Counter 1 - High	0		0	65535	0	

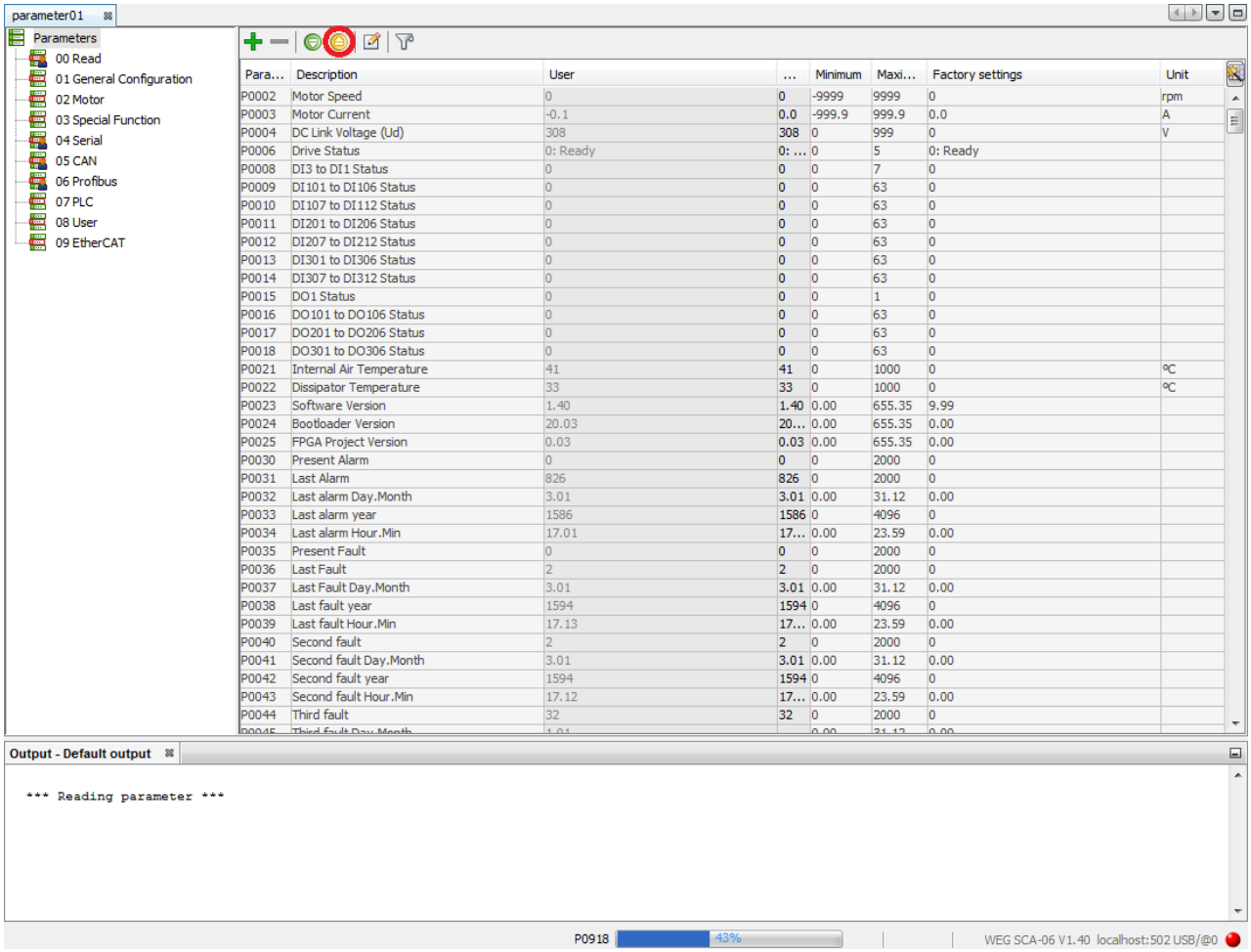
11.11.2.3 Read and Write of Parameters

There are 3 (three) ways to do the reading and writing of the parameters: by means of table, selection and group.

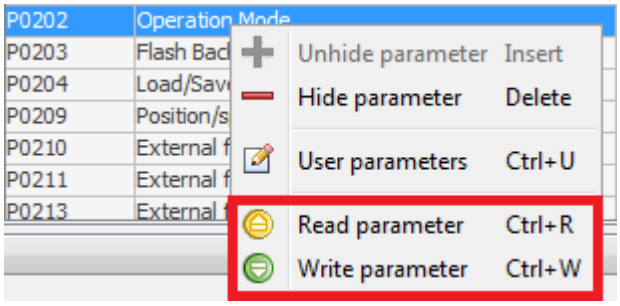
1. **Table writing**. The table writing command will send all visible parameters on the equipment screen. If and error occurs during the sending of some specific parameter, a message will be shown on the output window informing the error. It is important to notice that only visible parameters will be sent; therefore, it is necessary attention to which node of the group of parameters tree you are viewing. Example: If you wish to write all of them without filtering per group, just select the tree root.



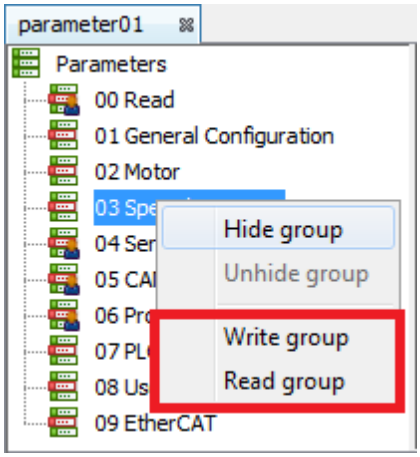
2. **Table reading**. The table reading command will read all the parameters of the equipment. If a error occurs during the reading of some specific parameter, a message will be shown on the output window informing the error. It is important to notice that only visible parameters will be read; therefore, it is necessary attention to which node of the group of parameters tree you are viewing. Example: If you wish to read all of them without filtering per group, just select the tree root.



3. **Reading/writing of specific parameters**. In order to read/write one or more specific parameters, just select them on the table, right click and choose the desired option: read or write parameter.



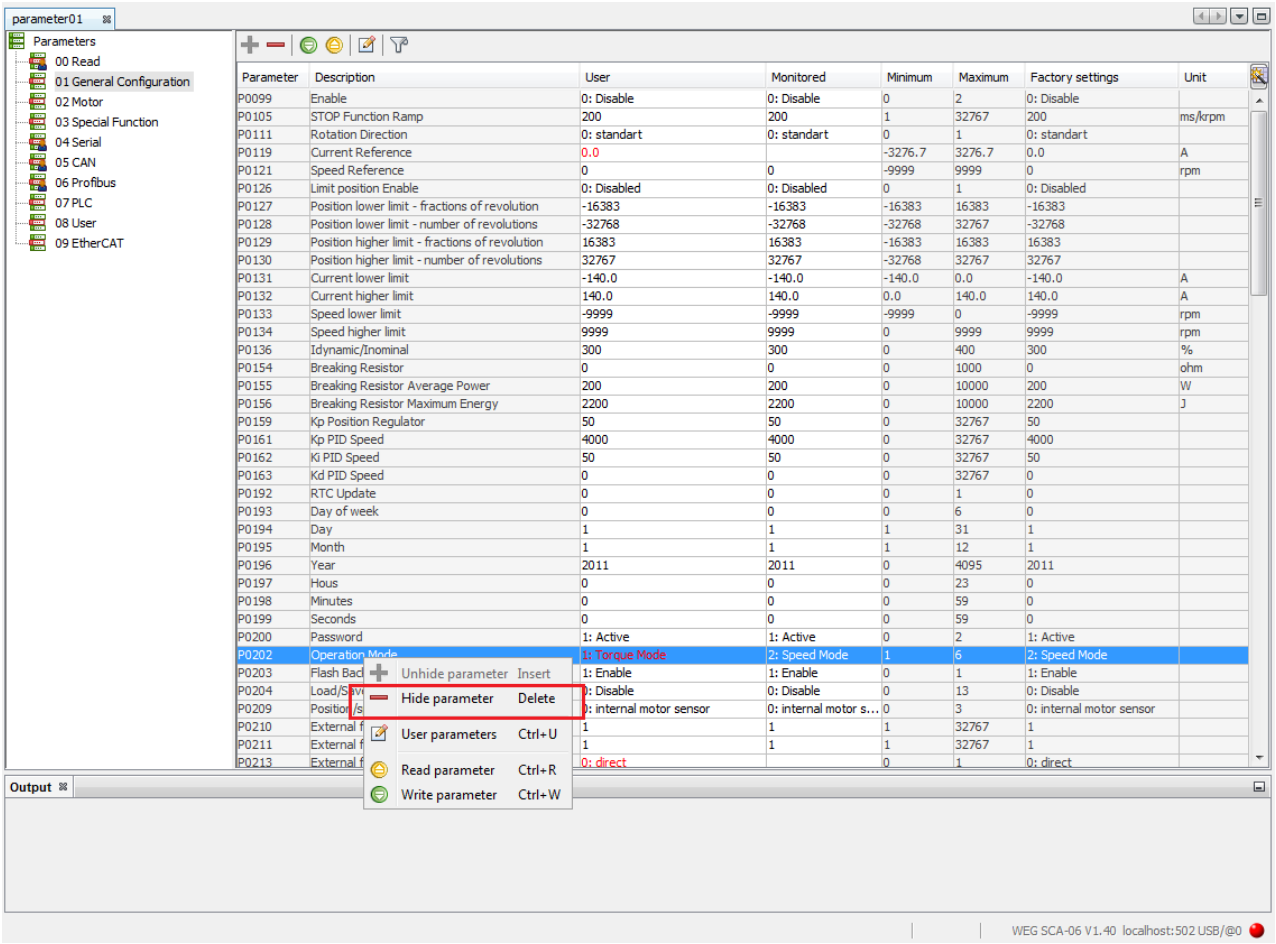
4. **Reading/writing of group of parameters**. In order to read/write only one group of parameters, just select it on the group tree, right click and choose the desired option: read or write group.



11.11.2.4 Hide/Unhide Parameters and Group of Parameters

The parameter can be hidden/unhidden in two ways: individually or in group.

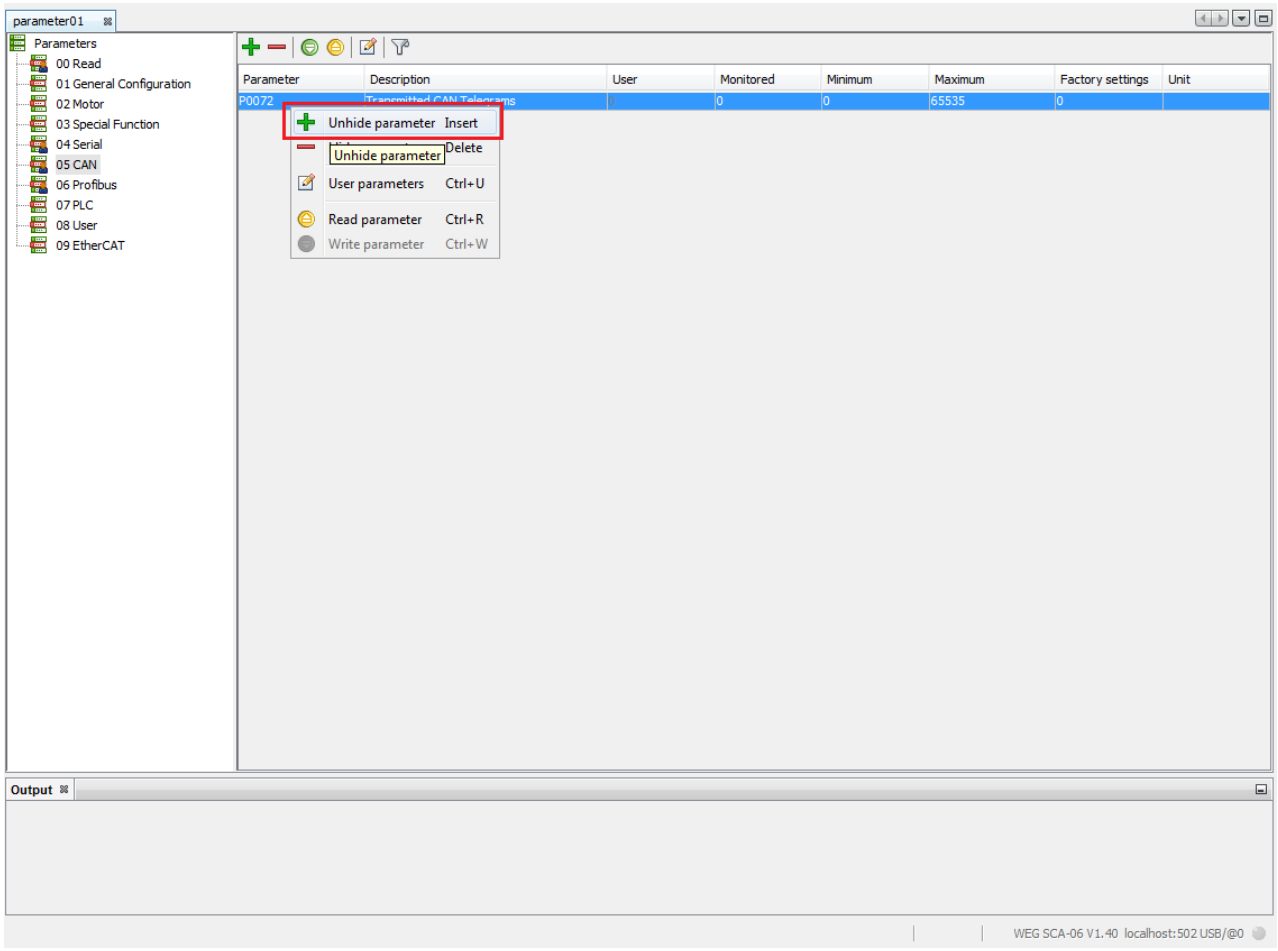
1. **Hide parameters**. In order to hide a parameter individually, just right click on the desired parameters and select the **Hide Parameter** option. You can also press the **Delete** key.

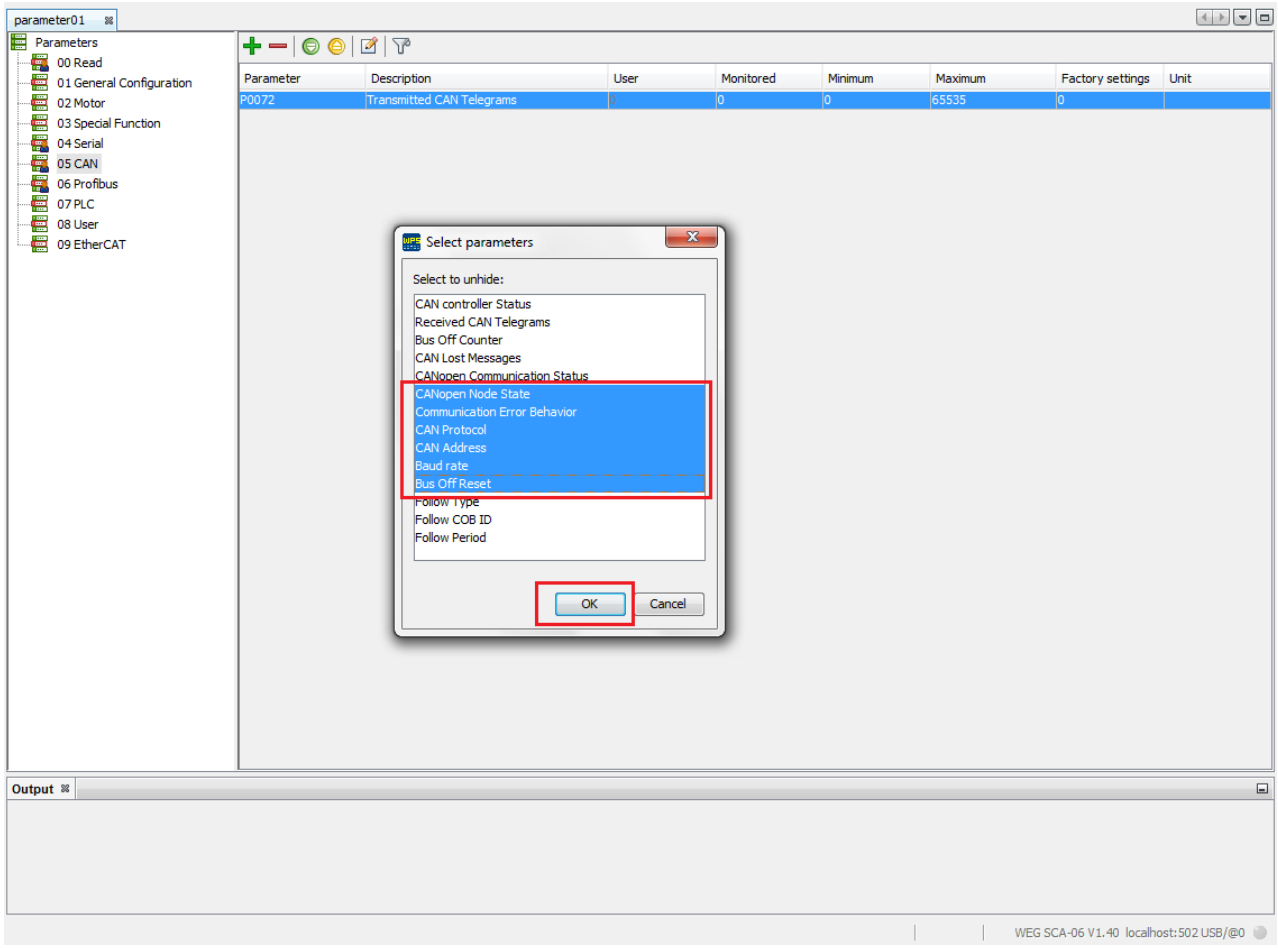


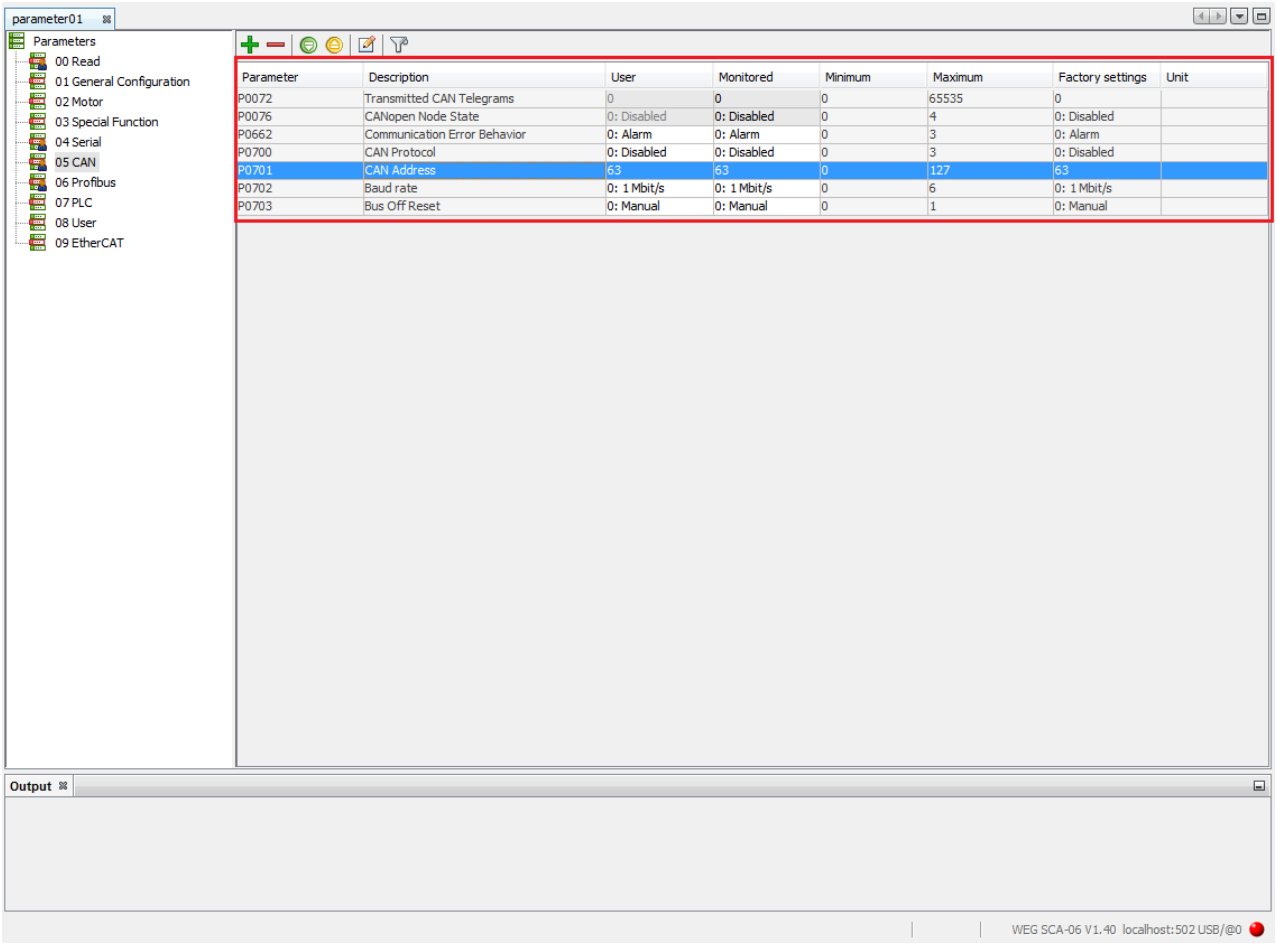
Parameter	Description	User	Monitored	Minimum	Maximum	Factory settings	Unit
P0099	Enable	0: Disable	0: Disable	0	2	0: Disable	
P0105	STOP Function Ramp	200	200	1	32767	200	ms/krpm
P0111	Rotation Direction	0: standart	0: standart	0	1	0: standart	
P0119	Current Reference	0.0		-3276.7	3276.7	0.0	A
P0121	Speed Reference	0	0	-9999	9999	0	rpm
P0126	Limit position Enable	0: Disabled	0: Disabled	0	1	0: Disabled	
P0127	Position lower limit - fractions of revolution	-16383	-16383	-16383	16383	-16383	
P0128	Position lower limit - number of revolutions	-32768	-32768	-32768	32767	-32768	
P0129	Position higher limit - fractions of revolution	16383	16383	-16383	16383	16383	
P0130	Position higher limit - number of revolutions	32767	32767	-32768	32767	32767	
P0131	Current lower limit	-140.0	-140.0	-140.0	0.0	-140.0	A
P0132	Current higher limit	140.0	140.0	0.0	140.0	140.0	A
P0133	Speed lower limit	-9999	-9999	-9999	0	-9999	rpm
P0134	Speed higher limit	9999	9999	0	9999	9999	rpm
P0136	Idynamic/Inominal	300	300	0	400	300	%
P0154	Breaking Resistor	0	0	0	1000	0	ohm
P0155	Breaking Resistor Average Power	200	200	0	10000	200	W
P0156	Breaking Resistor Maximum Energy	2200	2200	0	10000	2200	J
P0159	Kp Position Regulator	50	50	0	32767	50	
P0161	Kp PID Speed	4000	4000	0	32767	4000	
P0162	Ki PID Speed	50	50	0	32767	50	
P0163	Kd PID Speed	0	0	0	32767	0	
P0192	RTC Update	0	0	0	1	0	
P0193	Day of week	0	0	0	6	0	
P0194	Day	1	1	1	31	1	
P0195	Month	1	1	1	12	1	
P0196	Year	2011	2011	0	4095	2011	
P0197	Hous	0	0	0	23	0	
P0198	Minutes	0	0	0	59	0	
P0199	Seconds	0	0	0	59	0	
P0200	Password	1: Active	1: Active	0	2	1: Active	
P0202	Operation Mode	1: Torque Mode	2: Speed Mode	1	6	2: Speed Mode	
P0203	Flash Bac	1: Enable	1: Enable	0	1	1: Enable	
P0204	Load/S	0: Disable	0: Disable	0	13	0: Disable	
P0209	Position/s	0: internal motor sensor	0: internal motor s...	0	3	0: internal motor sensor	
P0210	External f	1	1	1	32767	1	
P0211	External f	1	1	1	32767	1	
P0213	External f	0: direct		0	1	0: direct	

2. **Unhide Parameters**. In order to show hidden parameters, right click and choose the **Unhide Parameters**

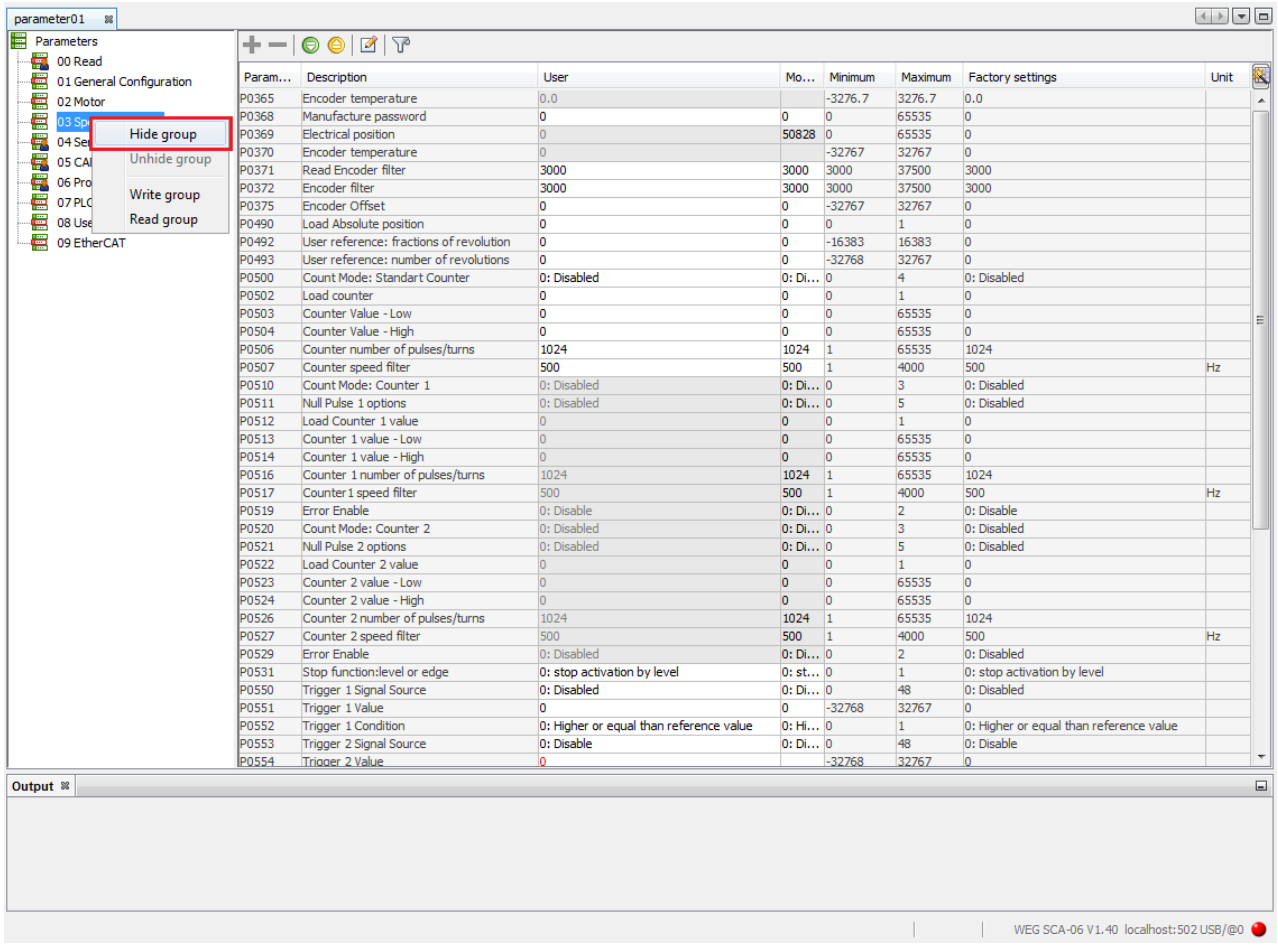
or press the **Insert** key. Then, a window will open and show the hidden parameters. Now, you just have to select the desired parameters and confirm.
Note: The parameters shown on this new window are only those which belong to the current filter according to the selection on the parameter group tree. In the figures below, the CAN group is selected; that means that only the hidden parameters of this group will be shown.

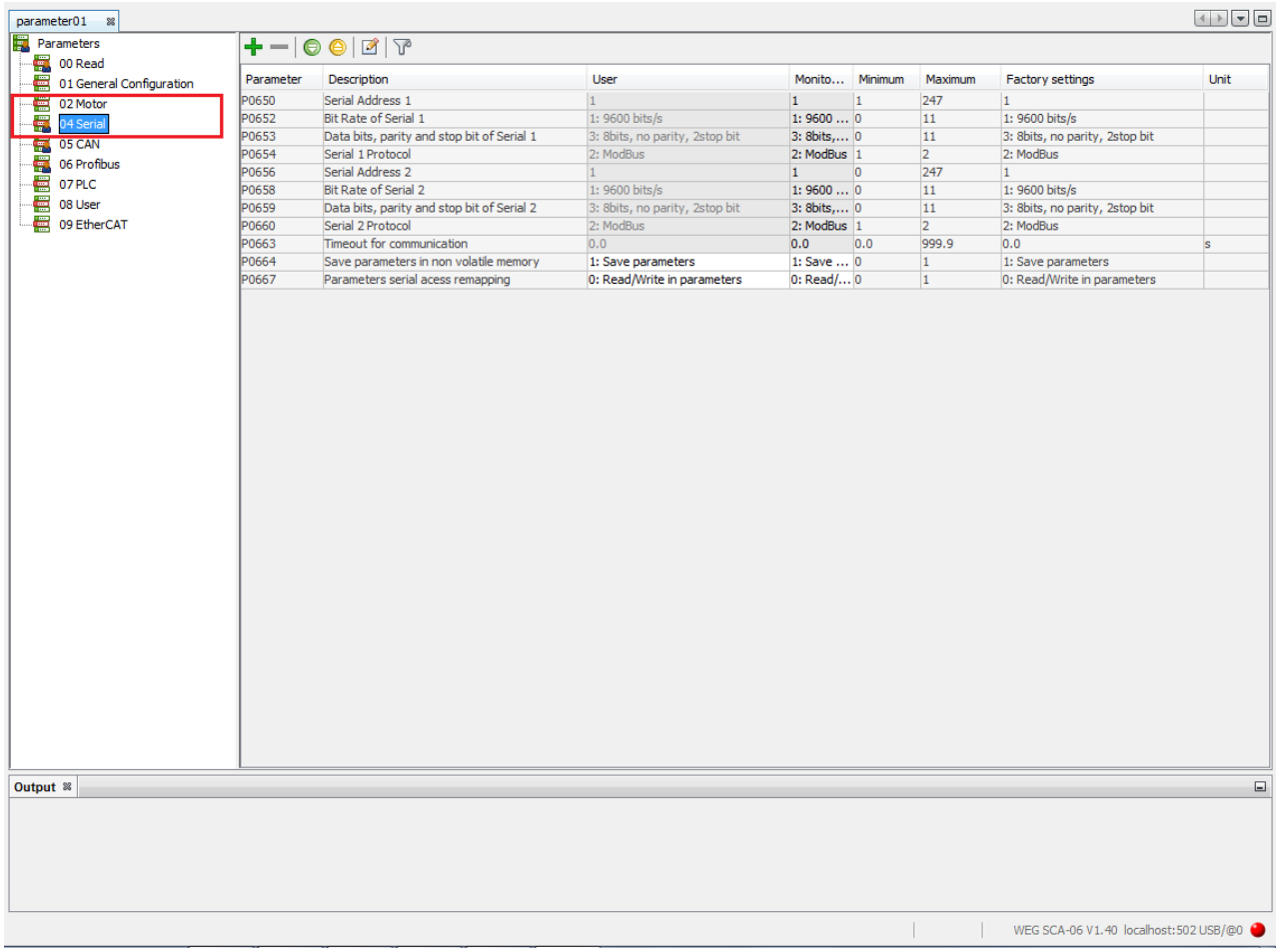



parameter01
Parameters
00 Read
01 General Configuration
02 Motor
03 Special Function
04 Serial
05 CAN
06 Profibus
07 PLC
08 User
09 EtherCAT
Parameter
Description
User
Monitored
Minimum
Maximum
Factory settings
Unit
P0072
Transmitted CAN Telegrams
0
0
65535
0
Select parameters
Select to unhide:
CAN controller Status
Received CAN Telegrams
Bus Off Counter
CAN Lost Messages
CANopen Communication Status
CANopen Node State
Communication Error Behavior
CAN Protocol
CAN Address
Baud rate
Bus Off Reset
Follow Type
Follow COB ID
Follow Period
OK
Cancel
Output
WEG SCA-06 V1.40 localhost:502 USB/@0

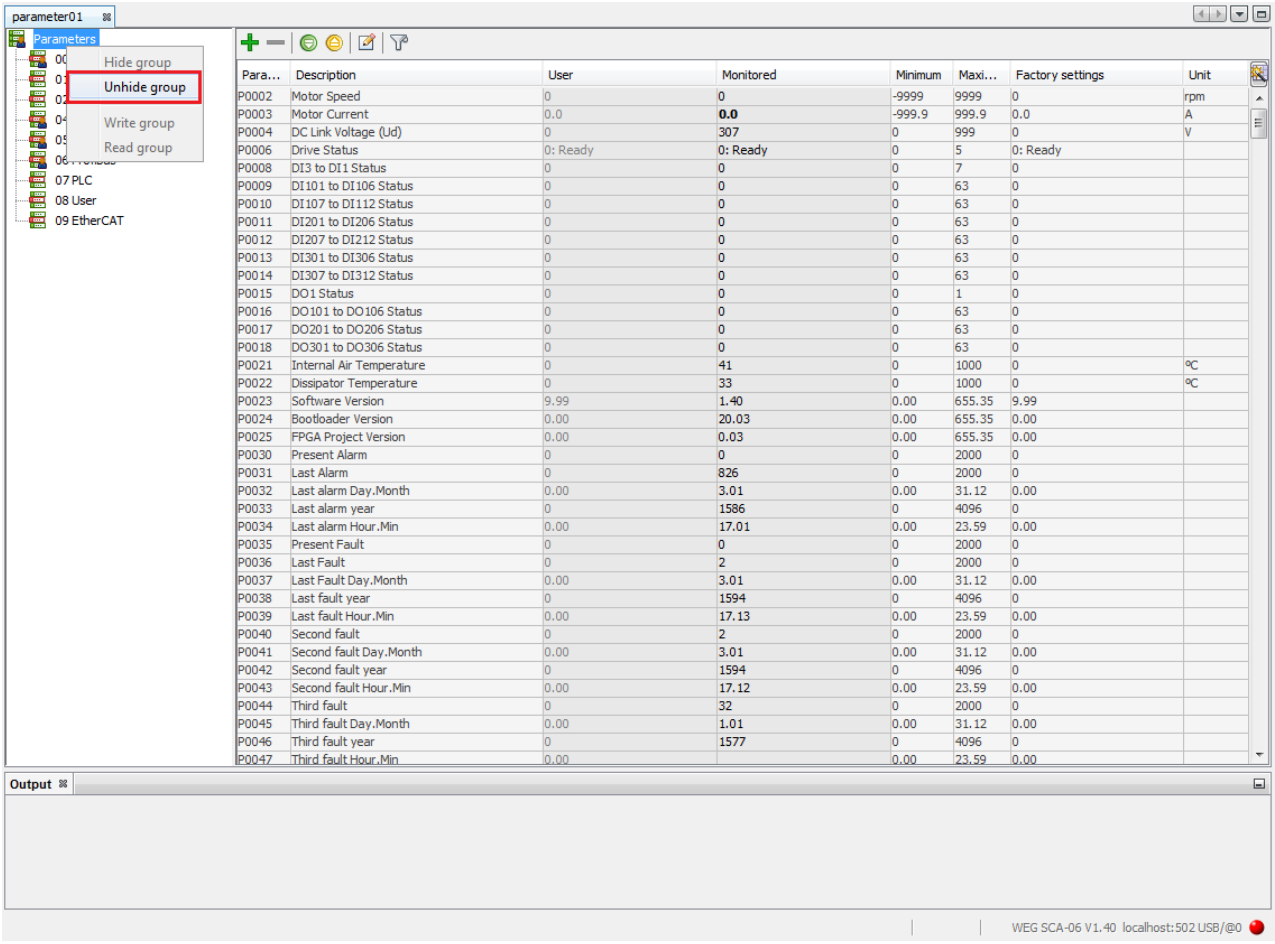


3. **Hide Group of Parameters**. In order to hide a group of parameters, just select the group on the tree and use the **Hide Group** option.

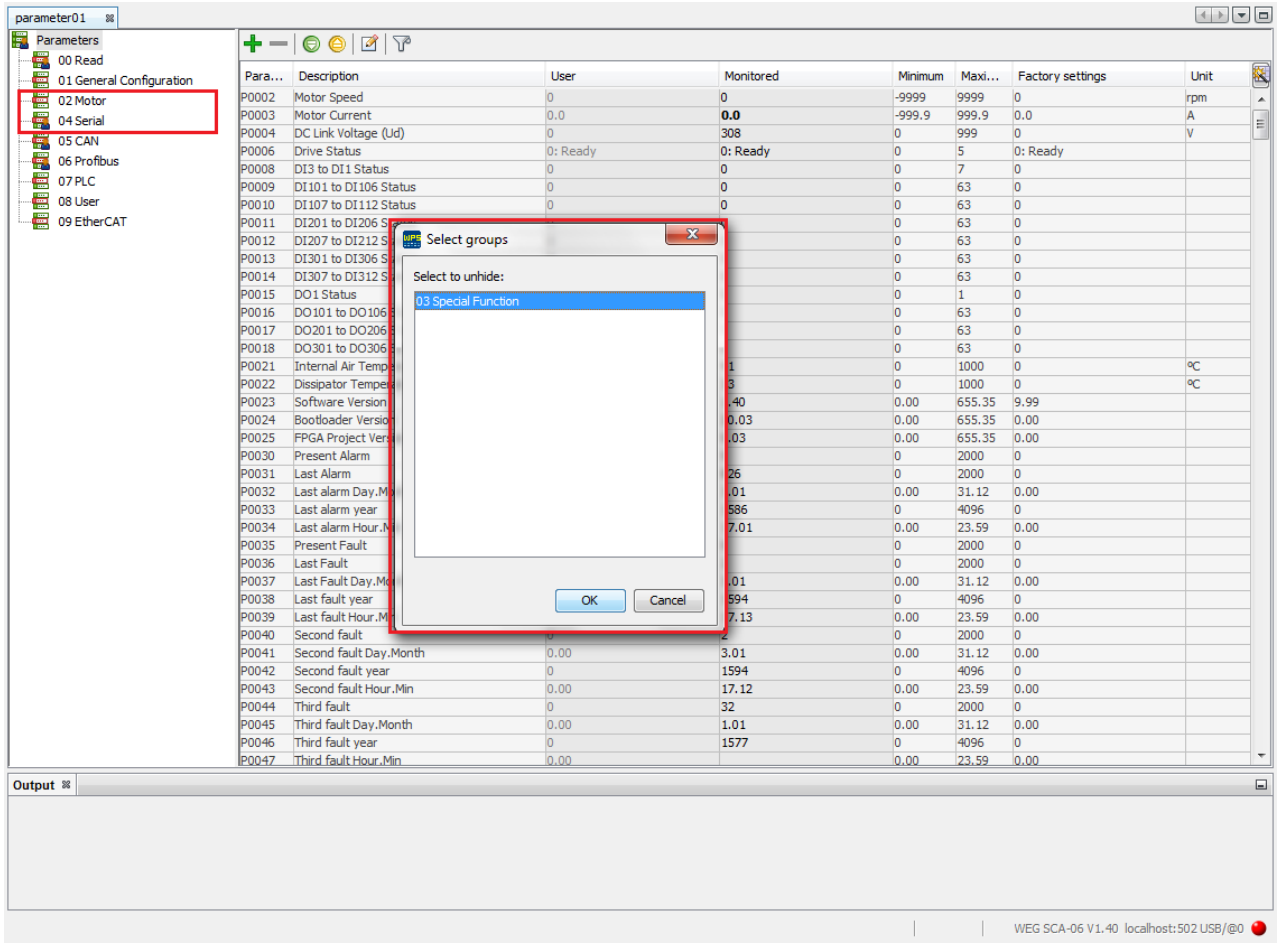

parameter01
Parameters
00 Read
01 General Configuration
02 Motor
03 Sp
04 Se
05 CA
06 Pro
07 PLC
08 Use
09 EtherCAT
Hide group
Unhide group
Write group
Read group
Param...
Description
User
Mo...
Minimum
Maximum
Factory settings
Unit
P0365 Encoder temperature 0.0 -3276.7 3276.7 0.0
P0368 Manufacture password 0 0 0 65535 0
P0369 Electrical position 0 50828 0 65535 0
P0370 Encoder temperature 0 -32767 32767 0
P0371 Read Encoder filter 3000 3000 3000 37500 3000
P0372 Encoder filter 3000 3000 3000 37500 3000
P0375 Encoder Offset 0 0 -32767 32767 0
P0490 Load Absolute position 0 0 0 1 0
P0492 User reference: fractions of revolution 0 0 -16383 16383 0
P0493 User reference: number of revolutions 0 0 -32768 32767 0
P0500 Count Mode: Standart Counter 0: Disabled 0: Di... 0 4 0: Disabled
P0502 Load counter 0 0 0 1 0
P0503 Counter Value - Low 0 0 0 65535 0
P0504 Counter Value - High 0 0 0 65535 0
P0506 Counter number of pulses/turns 1024 1024 1 65535 1024
P0507 Counter speed filter 500 500 1 4000 500 Hz
P0510 Count Mode: Counter 1 0: Disabled 0: Di... 0 3 0: Disabled
P0511 Null Pulse 1 options 0: Disabled 0: Di... 0 5 0: Disabled
P0512 Load Counter 1 value 0 0 0 1 0
P0513 Counter 1 value - Low 0 0 0 65535 0
P0514 Counter 1 value - High 0 0 0 65535 0
P0516 Counter 1 number of pulses/turns 1024 1024 1 65535 1024
P0517 Counter 1 speed filter 500 500 1 4000 500 Hz
P0519 Error Enable 0: Disable 0: Di... 0 2 0: Disable
P0520 Count Mode: Counter 2 0: Disabled 0: Di... 0 3 0: Disabled
P0521 Null Pulse 2 options 0: Disabled 0: Di... 0 5 0: Disabled
P0522 Load Counter 2 value 0 0 0 1 0
P0523 Counter 2 value - Low 0 0 0 65535 0
P0524 Counter 2 value - High 0 0 0 65535 0
P0526 Counter 2 number of pulses/turns 1024 1024 1 65535 1024
P0527 Counter 2 speed filter 500 500 1 4000 500 Hz
P0529 Error Enable 0: Disabled 0: Di... 0 2 0: Disabled
P0531 Stop function:level or edge 0: stop activation by level 0: st... 0 1 0: stop activation by level
P0550 Trigger 1 Signal Source 0: Disabled 0: Di... 0 48 0: Disabled
P0551 Trigger 1 Value 0 0 -32768 32767 0
P0552 Trigger 1 Condition 0: Higher or equal than reference value 0: Hi... 0 1 0: Higher or equal than reference value
P0553 Trigger 2 Signal Source 0: Disable 0: Di... 0 48 0: Disable
P0554 Trigger 2 Value 0 -32768 32767 0
Output
WEG SCA-06 V1.40 localhost:502 USB/@0



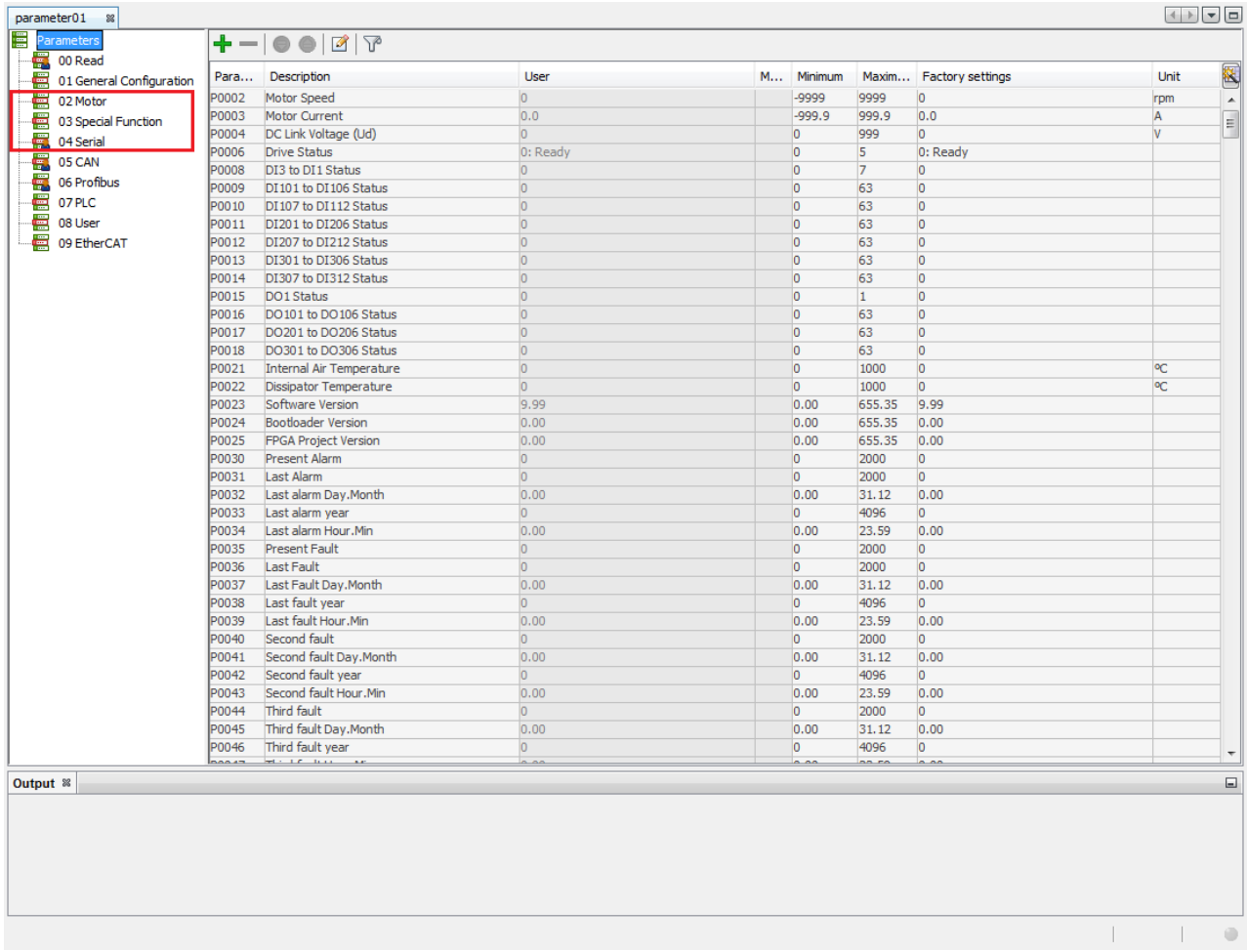
4. **Unhide Group of Parameters**. In order to show a hidden group of parameters, just select the root of the group tree and select the **Unhide Group** option. A window will open showing the groups that are hidden; then just select the group you wish to unhide.



Para...	Description	User	Monitored	Minimum	Maxi...	Factory settings	Unit
P0002	Motor Speed	0	0	-9999	9999	0	rpm
P0003	Motor Current	0.0	**0.0**	-999.9	999.9	0.0	A
P0004	DC Link Voltage (Ud)	0	307	0	999	0	V
P0006	Drive Status	0: Ready	0: Ready	0	5	0: Ready	
P0008	DI3 to DI1 Status	0	0	0	7	0	
P0009	DI101 to DI106 Status	0	0	0	63	0	
P0010	DI107 to DI112 Status	0	0	0	63	0	
P0011	DI201 to DI206 Status	0	0	0	63	0	
P0012	DI207 to DI212 Status	0	0	0	63	0	
P0013	DI301 to DI306 Status	0	0	0	63	0	
P0014	DI307 to DI312 Status	0	0	0	63	0	
P0015	DO1 Status	0	0	0	1	0	
P0016	DO101 to DO106 Status	0	0	0	63	0	
P0017	DO201 to DO206 Status	0	0	0	63	0	
P0018	DO301 to DO306 Status	0	0	0	63	0	
P0021	Internal Air Temperature	0	41	0	1000	0	°C
P0022	Dissipator Temperature	0	33	0	1000	0	°C
P0023	Software Version	9.99	1.40	0.00	655.35	9.99	
P0024	Bootloader Version	0.00	20.03	0.00	655.35	0.00	
P0025	FPGA Project Version	0.00	0.03	0.00	655.35	0.00	
P0030	Present Alarm	0	0	0	2000	0	
P0031	Last Alarm	0	826	0	2000	0	
P0032	Last alarm Day.Month	0.00	3.01	0.00	31.12	0.00	
P0033	Last alarm year	0	1586	0	4096	0	
P0034	Last alarm Hour.Min	0.00	17.01	0.00	23.59	0.00	
P0035	Present Fault	0	0	0	2000	0	
P0036	Last Fault	0	2	0	2000	0	
P0037	Last Fault Day.Month	0.00	3.01	0.00	31.12	0.00	
P0038	Last fault year	0	1594	0	4096	0	
P0039	Last fault Hour.Min	0.00	17.13	0.00	23.59	0.00	
P0040	Second fault	0	2	0	2000	0	
P0041	Second fault Day.Month	0.00	3.01	0.00	31.12	0.00	
P0042	Second fault year	0	1594	0	4096	0	
P0043	Second fault Hour.Min	0.00	17.12	0.00	23.59	0.00	
P0044	Third fault	0	32	0	2000	0	
P0045	Third fault Day.Month	0.00	1.01	0.00	31.12	0.00	
P0046	Third fault year	0	1577	0	4096	0	
P0047	Third fault Hour.Min	0.00		0.00	23.59	0.00	

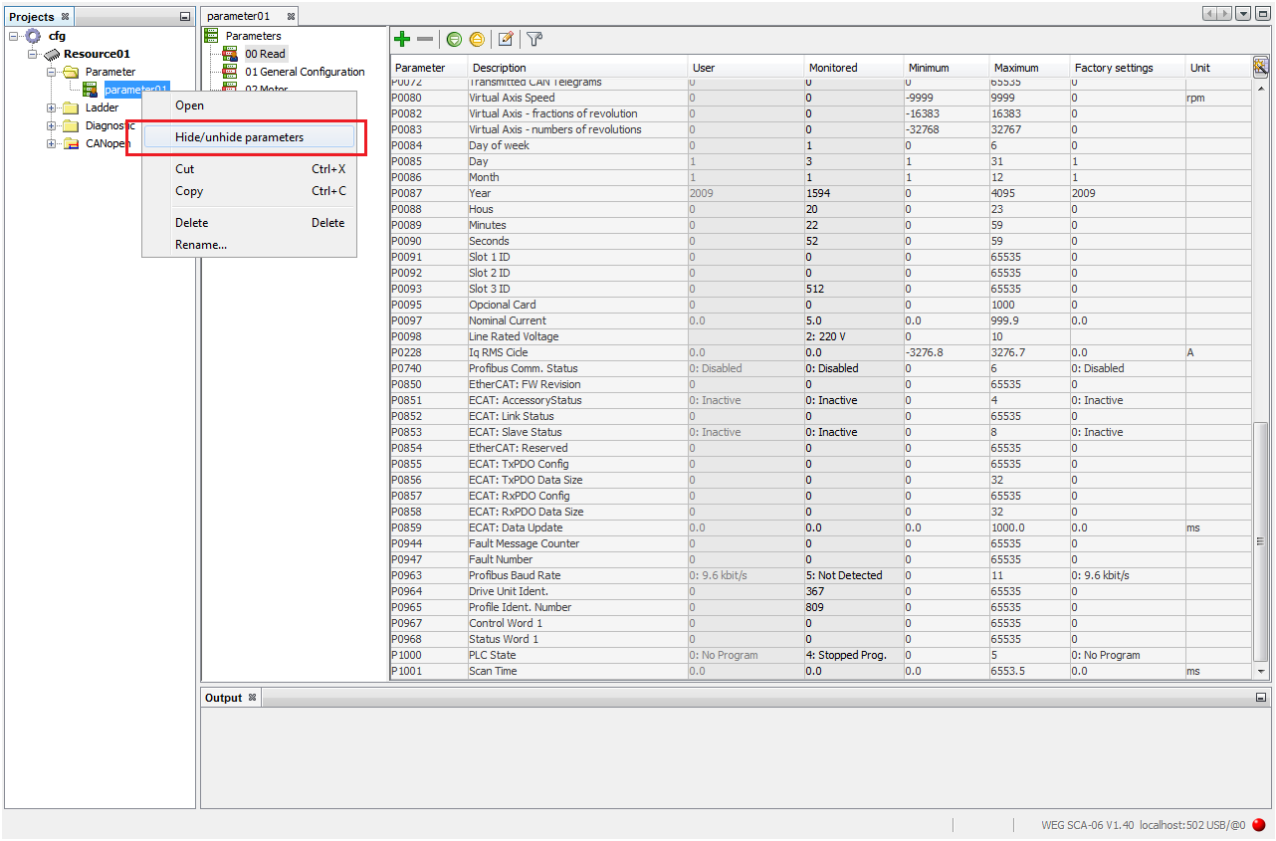


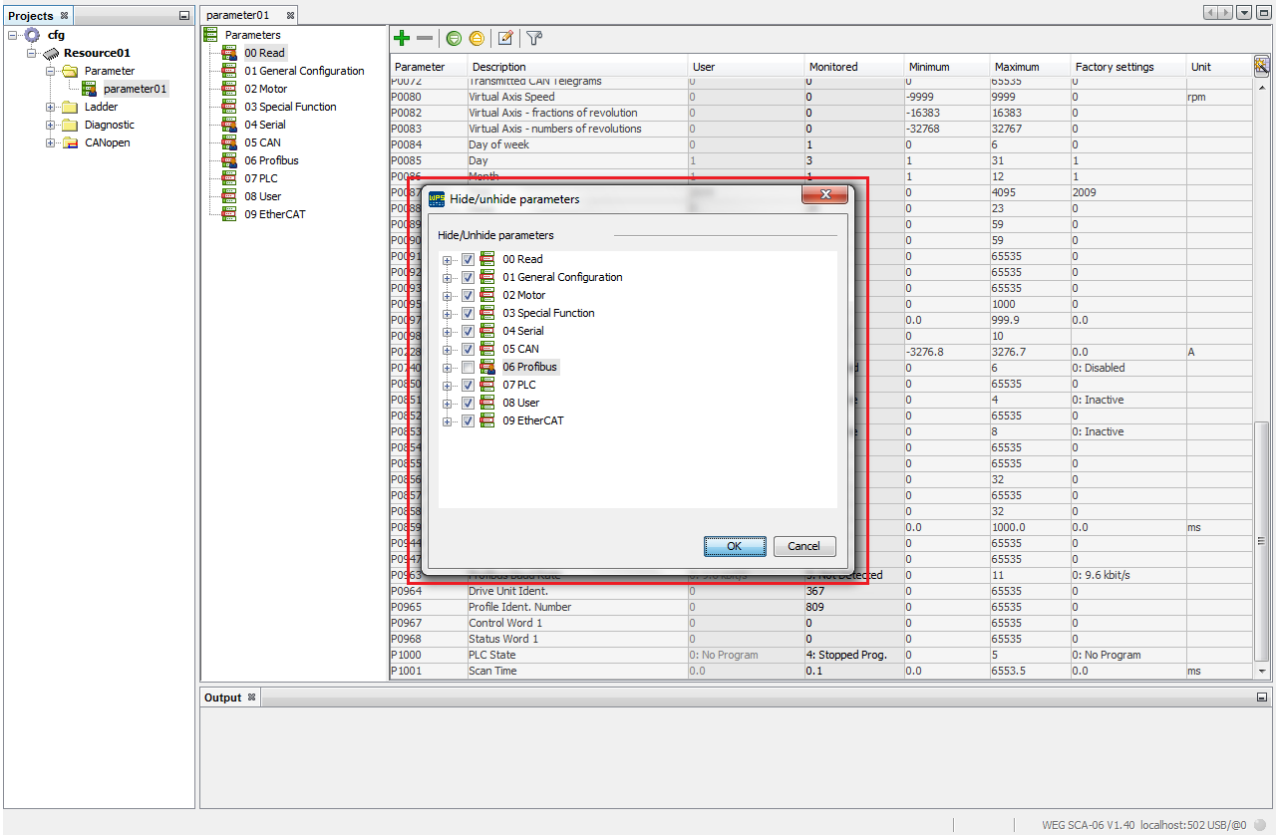
Para...	Description	User	Monitored	Minimum	Maxi...	Factory settings	Unit
P0002	Motor Speed	0	0	-9999	9999	0	rpm
P0003	Motor Current	0.0	**0.0**	-999.9	999.9	0.0	A
P0004	DC Link Voltage (Ud)	0	308	0	999	0	V
P0006	Drive Status	0: Ready	0: Ready	0	5	0: Ready	
P0008	DI3 to DI1 Status	0	0	0	7	0	
P0009	DI101 to DI106 Status	0	0	0	63	0	
P0010	DI107 to DI112 Status	0	0	0	63	0	
P0011	DI201 to DI206 S			0	63	0	
P0012	DI207 to DI212 S			0	63	0	
P0013	DI301 to DI306 S			0	63	0	
P0014	DI307 to DI312 S			0	63	0	
P0015	DO1 Status			0	1	0	
P0016	DO101 to DO106			0	63	0	
P0017	DO201 to DO206			0	63	0	
P0018	DO301 to DO306			0	63	0	
P0021	Internal Air Temp		1	0	1000	0	°C
P0022	Dissipator Temper		3	0	1000	0	°C
P0023	Software Version		.40	0.00	655.35	9.99	
P0024	Bootloader Versio		0.03	0.00	655.35	0.00	
P0025	FPGA Project Ver		.03	0.00	655.35	0.00	
P0030	Present Alarm			0	2000	0	
P0031	Last Alarm		26	0	2000	0	
P0032	Last alarm Day.M		.01	0.00	31.12	0.00	
P0033	Last alarm year		586	0	4096	0	
P0034	Last alarm Hour.M		7.01	0.00	23.59	0.00	
P0035	Present Fault			0	2000	0	
P0036	Last Fault			0	2000	0	
P0037	Last Fault Day.M		.01	0.00	31.12	0.00	
P0038	Last fault year		594	0	4096	0	
P0039	Last fault Hour.M		7.13	0.00	23.59	0.00	
P0040	Second fault		2	0	2000	0	
P0041	Second fault Day.Month	0.00	3.01	0.00	31.12	0.00	
P0042	Second fault year	0	1594	0	4096	0	
P0043	Second fault Hour.Min	0.00	17.12	0.00	23.59	0.00	
P0044	Third fault	0	32	0	2000	0	
P0045	Third fault Day.Month	0.00	1.01	0.00	31.12	0.00	
P0046	Third fault year	0	1577	0	4096	0	
P0047	Third fault Hour.Min	0.00		0.00	23.59	0.00	



Para...	Description	User	M...	Minimum	Maxim...	Factory settings	Unit
P0002	Motor Speed	0		-9999	9999	0	rpm
P0003	Motor Current	0.0		-999.9	999.9	0.0	A
P0004	DC Link Voltage (Ud)	0		0	999	0	V
P0006	Drive Status	0: Ready		0	5	0: Ready	
P0008	DI3 to DI1 Status	0		0	7	0	
P0009	DI101 to DI106 Status	0		0	63	0	
P0010	DI107 to DI112 Status	0		0	63	0	
P0011	DI201 to DI206 Status	0		0	63	0	
P0012	DI207 to DI212 Status	0		0	63	0	
P0013	DI301 to DI306 Status	0		0	63	0	
P0014	DI307 to DI312 Status	0		0	63	0	
P0015	DO1 Status	0		0	1	0	
P0016	DO101 to DO106 Status	0		0	63	0	
P0017	DO201 to DO206 Status	0		0	63	0	
P0018	DO301 to DO306 Status	0		0	63	0	
P0021	Internal Air Temperature	0		0	1000	0	°C
P0022	Dissipator Temperature	0		0	1000	0	°C
P0023	Software Version	9.99		0.00	655.35	9.99	
P0024	Bootloader Version	0.00		0.00	655.35	0.00	
P0025	FPGA Project Version	0.00		0.00	655.35	0.00	
P0030	Present Alarm	0		0	2000	0	
P0031	Last Alarm	0		0	2000	0	
P0032	Last alarm Day.Month	0.00		0.00	31.12	0.00	
P0033	Last alarm year	0		0	4096	0	
P0034	Last alarm Hour.Min	0.00		0.00	23.59	0.00	
P0035	Present Fault	0		0	2000	0	
P0036	Last Fault	0		0	2000	0	
P0037	Last Fault Day.Month	0.00		0.00	31.12	0.00	
P0038	Last fault year	0		0	4096	0	
P0039	Last fault Hour.Min	0.00		0.00	23.59	0.00	
P0040	Second fault	0		0	2000	0	
P0041	Second fault Day.Month	0.00		0.00	31.12	0.00	
P0042	Second fault year	0		0	4096	0	
P0043	Second fault Hour.Min	0.00		0.00	23.59	0.00	
P0044	Third fault	0		0	2000	0	
P0045	Third fault Day.Month	0.00		0.00	31.12	0.00	
P0046	Third fault year	0		0	4096	0	

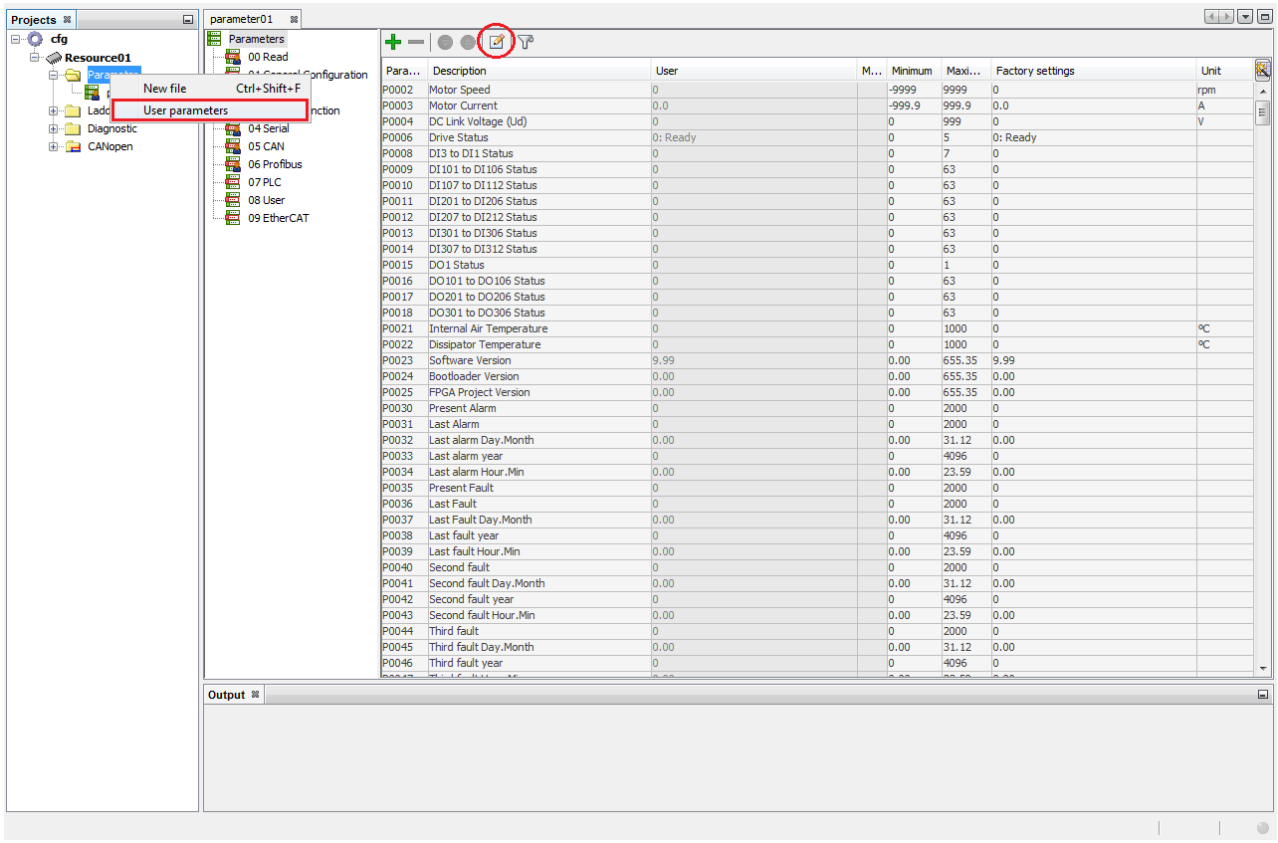
5. **Hide and Show Parameters and Groups of Parameters**. By means of this option, you have full control of the parameters and groups of parameters. It is possible to hide and unhide individual parameters, multiple parameters, individual groups and multiple groups in the same action.


Projects
cfg
Resource01
Parameter
parameter01
Ladder
Diagnostic
CANopen
Open
Hide/unhide parameters
Cut Ctrl+X
Copy Ctrl+C
Delete Delete
Rename...
parameter01
Parameters
00 Read
01 General Configuration
Parameter
Description
User
Monitored
Minimum
Maximum
Factory settings
Unit
P0080 Virtual Axis Speed
P0082 Virtual Axis - fractions of revolution
P0083 Virtual Axis - numbers of revolutions
P0084 Day of week
P0085 Day
P0086 Month
P0087 Year
P0088 Hous
P0089 Minutes
P0090 Seconds
P0091 Slot 1 ID
P0092 Slot 2 ID
P0093 Slot 3 ID
P0095 Opcional Card
P0097 Nominal Current
P0098 Line Rated Voltage
P0228 Iq RMS Cicle
P0740 Profibus Comm. Status
P0850 EtherCAT: FW Revision
P0851 ECAT: AccessoryStatus
P0852 ECAT: Link Status
P0853 ECAT: Slave Status
P0854 EtherCAT: Reserved
P0855 ECAT: TxPDO Config
P0856 ECAT: TxPDO Data Size
P0857 ECAT: RxPDO Config
P0858 ECAT: RxPDO Data Size
P0859 ECAT: Data Update
P0944 Fault Message Counter
P0947 Fault Number
P0963 Profibus Baud Rate
P0964 Drive Unit Ident.
P0965 Profile Ident. Number
P0967 Control Word 1
P0968 Status Word 1
P1000 PLC State
P1001 Scan Time
Output
WEG SCA-06 V1.40 localhost:502 USB/@0


Projects
cfg
Resource01
Parameter
parameter01
Ladder
Diagnostic
CANopen
parameter01
Parameters
00 Read
01 General Configuration
02 Motor
03 Special Function
04 Serial
05 CAN
06 Profibus
07 PLC
08 User
09 EtherCAT
Hide/unhide parameters
Hide/Unhide parameters
OK
Cancel
Output
WEG SCA-06 V1.40 localhost:502 USB/@0

11.11.2.5 User Parameters

In order to open the configuration screen of the user parameters, just click on the **User Parameters** option on the Parameter node of the project tree or click on the icon indicated on the tool bar of the parameter file.



Para...	Description	User	M...	Minimum	Maxi...	Factory settings	Unit
P0002	Motor Speed	0		-9999	9999	0	rpm
P0003	Motor Current	0.0		-999.9	999.9	0.0	A
P0004	DC Link Voltage (Ud)	0		0	999	0	V
P0006	Drive Status	0: Ready		0	5	0: Ready	
P0008	DI3 to DI1 Status	0		0	7	0	
P0009	DI101 to DI106 Status	0		0	63	0	
P0010	DI107 to DI112 Status	0		0	63	0	
P0011	DI201 to DI206 Status	0		0	63	0	
P0012	DI207 to DI212 Status	0		0	63	0	
P0013	DI301 to DI306 Status	0		0	63	0	
P0014	DI307 to DI312 Status	0		0	63	0	
P0015	DO1 Status	0		0	1	0	
P0016	DO101 to DO106 Status	0		0	63	0	
P0017	DO201 to DO206 Status	0		0	63	0	
P0018	DO301 to DO306 Status	0		0	63	0	
P0021	Internal Air Temperature	0		0	1000	0	°C
P0022	Dissipator Temperature	0		0	1000	0	°C
P0023	Software Version	9.99		0.00	655.35	9.99	
P0024	Bootloader Version	0.00		0.00	655.35	0.00	
P0025	FPGA Project Version	0.00		0.00	655.35	0.00	
P0030	Present Alarm	0		0	2000	0	
P0031	Last Alarm	0		0	2000	0	
P0032	Last alarm Day.Month	0.00		0.00	31.12	0.00	
P0033	Last alarm year	0		0	4096	0	
P0034	Last alarm Hour.Min	0.00		0.00	23.59	0.00	
P0035	Present Fault	0		0	2000	0	
P0036	Last Fault	0		0	2000	0	
P0037	Last Fault Day.Month	0.00		0.00	31.12	0.00	
P0038	Last fault year	0		0	4096	0	
P0039	Last fault Hour.Min	0.00		0.00	23.59	0.00	
P0040	Second fault	0		0	2000	0	
P0041	Second fault Day.Month	0.00		0.00	31.12	0.00	
P0042	Second fault year	0		0	4096	0	
P0043	Second fault Hour.Min	0.00		0.00	23.59	0.00	
P0044	Third fault	0		0	2000	0	
P0045	Third fault Day.Month	0.00		0.00	31.12	0.00	
P0046	Third fault year	0		0	4096	0	

Configuration Table.

On the user parameter configuration table, it is possible to define several attributes to the parameters, such as description, minimum and maximum values, unit, digits, data type, etc.

NOTE!
These settings will be automatically displayed in the parameter table. However, to be sent to the device, you need to *download* the resource.

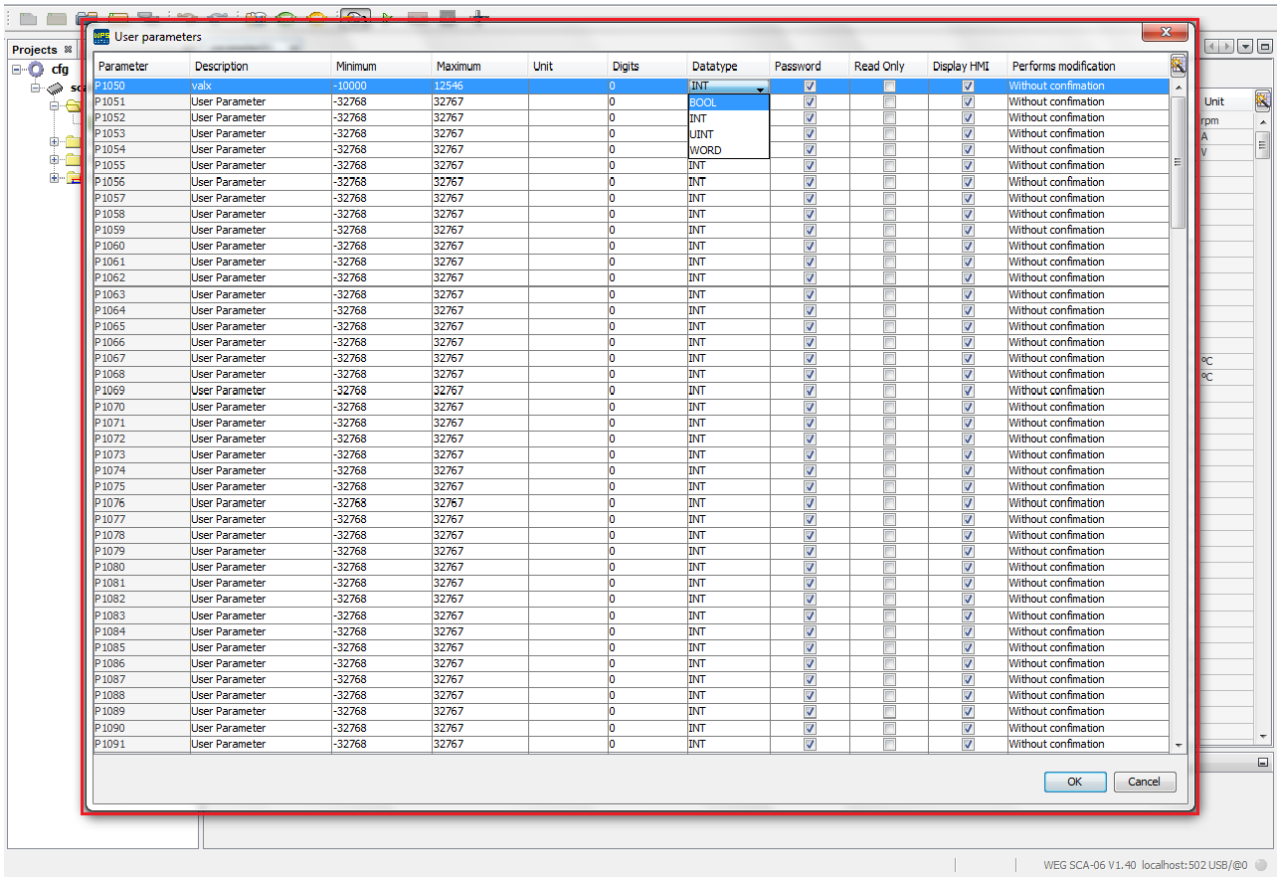


Table fields:

- Parameter: User parameter identification.
- Description: Description of the user parameter in the parameter table. On devices that have text-based HMIs, the description is sent to the machine and displayed on the HMI.
- Minimum: Minimum input value for parameter.
- Maximum: Maximum input value for parameter.
- Unit: Unit displayed on the device's HMI.
- Default: Value loaded when restore factory default is selected.
- Retentive: Retain value after rebooting devices.
- Hexadecimal: Displays the value in hexadecimal.
- Digits: Number of decimal digits for displaying value.
- Datatype: Parameter datatype used by the ladder application.
- Password: Enables password request by changing parameter value.

- Read only: It does not allow the writing of values in the parameter by the communication network or the HMI. Writing is done only by the ladder application.

- Display HMI: Displays the parameter in the HMI.

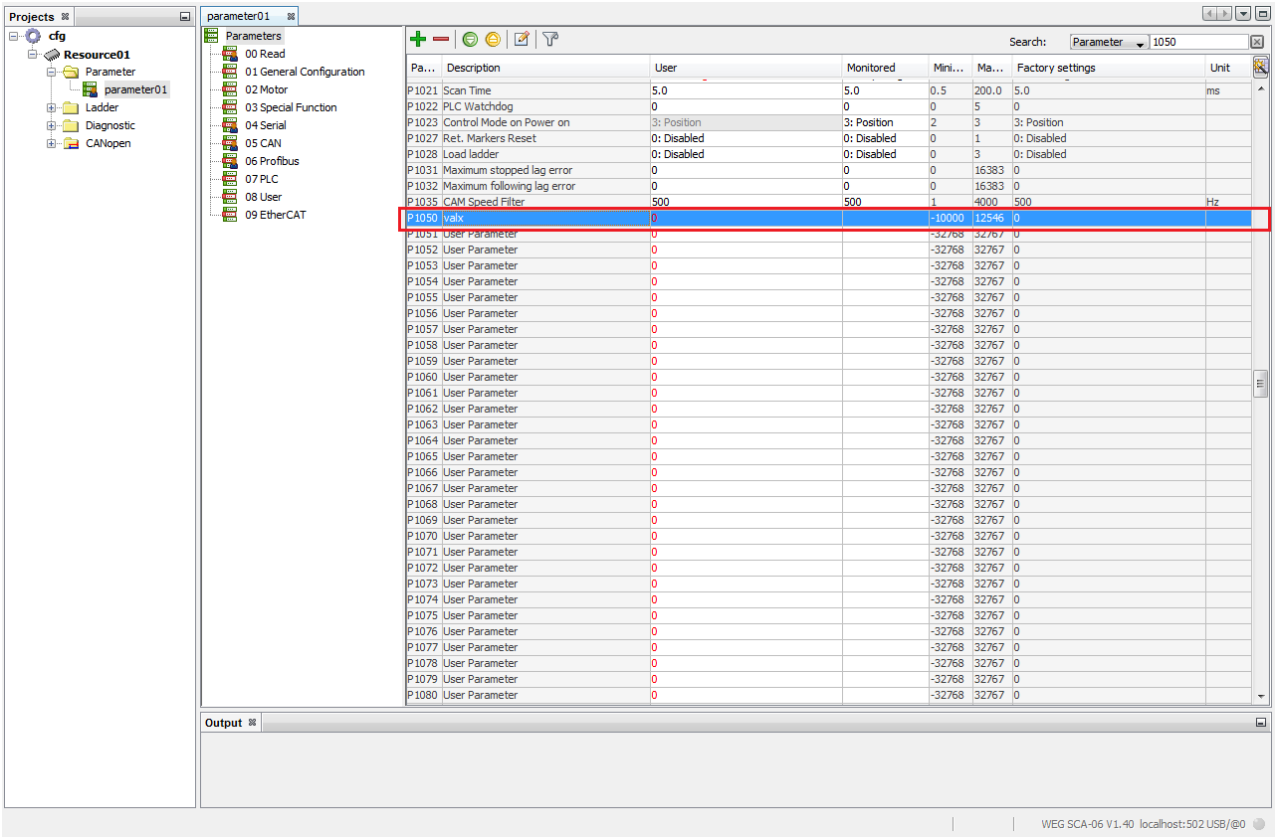
- Performs modification: Confirmation options when changing the parameter:
o No confirmation: Does not prompt for confirmation when changing parameter.
o With confirmation and engine stopped: Request confirmation and allow change only with engine stopped.
o With confirmation: Prompt for confirmation when changing parameter.

- Stopped motor: Perform change only with motor stopped.

- Help: On devices that have text-based HMI, you can edit a help text for the parameter.

View the user parameter

In the parameter table, the user parameters will be shown as they are configured on the configuration screen.



Pa...	Description	User	Monitored	Mini...	Ma...	Factory settings	Unit
P1021	Scan Time	5.0	5.0	0.5	200.0	5.0	ms
P1022	PLC Watchdog	0	0	0	5	0	
P1023	Control Mode on Power on	3: Position	3: Position	2	3	3: Position	
P1027	Ret. Markers Reset	0: Disabled	0: Disabled	0	1	0: Disabled	
P1028	Load ladder	0: Disabled	0: Disabled	0	3	0: Disabled	
P1031	Maximum stopped lag error	0	0	0	16383	0	
P1032	Maximum following lag error	0	0	0	16383	0	
P1035	CAM Speed Filter	500	500	1	4000	500	Hz
P1050	valx	0		-10000	12546	0	
P1051	User Parameter	0		-32768	32767	0	
P1052	User Parameter	0		-32768	32767	0	
P1053	User Parameter	0		-32768	32767	0	
P1054	User Parameter	0		-32768	32767	0	
P1055	User Parameter	0		-32768	32767	0	
P1056	User Parameter	0		-32768	32767	0	
P1057	User Parameter	0		-32768	32767	0	
P1058	User Parameter	0		-32768	32767	0	
P1059	User Parameter	0		-32768	32767	0	
P1060	User Parameter	0		-32768	32767	0	
P1061	User Parameter	0		-32768	32767	0	
P1062	User Parameter	0		-32768	32767	0	
P1063	User Parameter	0		-32768	32767	0	
P1064	User Parameter	0		-32768	32767	0	
P1065	User Parameter	0		-32768	32767	0	
P1066	User Parameter	0		-32768	32767	0	
P1067	User Parameter	0		-32768	32767	0	
P1068	User Parameter	0		-32768	32767	0	
P1069	User Parameter	0		-32768	32767	0	
P1070	User Parameter	0		-32768	32767	0	
P1071	User Parameter	0		-32768	32767	0	
P1072	User Parameter	0		-32768	32767	0	
P1073	User Parameter	0		-32768	32767	0	
P1074	User Parameter	0		-32768	32767	0	
P1075	User Parameter	0		-32768	32767	0	
P1076	User Parameter	0		-32768	32767	0	
P1077	User Parameter	0		-32768	32767	0	
P1078	User Parameter	0		-32768	32767	0	
P1079	User Parameter	0		-32768	32767	0	
P1080	User Parameter	0		-32768	32767	0	

11.12 SSW-07

Enter topic text here.

11.12.1 Description

Soft-Starters are static starters that accelerate, decelerate and protect three-phase induction motors. The control of the voltage applied to the motor by means of adjustments to the firing angle of thyristors allows the soft-starter to start and stop an electric motor smoothly.

The SSW-07, with DSP (Digital Signal Processor) control was designed for high performance on motor starts and stops with an excellent cost-benefit ratio. Easy to set up, it simplifies start-up activities and daily operation.

The SSW-07 is compact optimizing space in electric panels. It already incorporates electric motor protection. It adapts to customer needs through its easy-to-install optional accessories.

The SSW-07 adapts to customer needs through its easy-to-install optional accessories. Thus, a keypad, a communication interface or a motor PTC input can be added to the product.

Refer to the user's manual of the SSW-07 for further details about the product.

NOTE!

This product does not have the Ladder tool available in WPS.
You can use the WLP application if this feature is required.

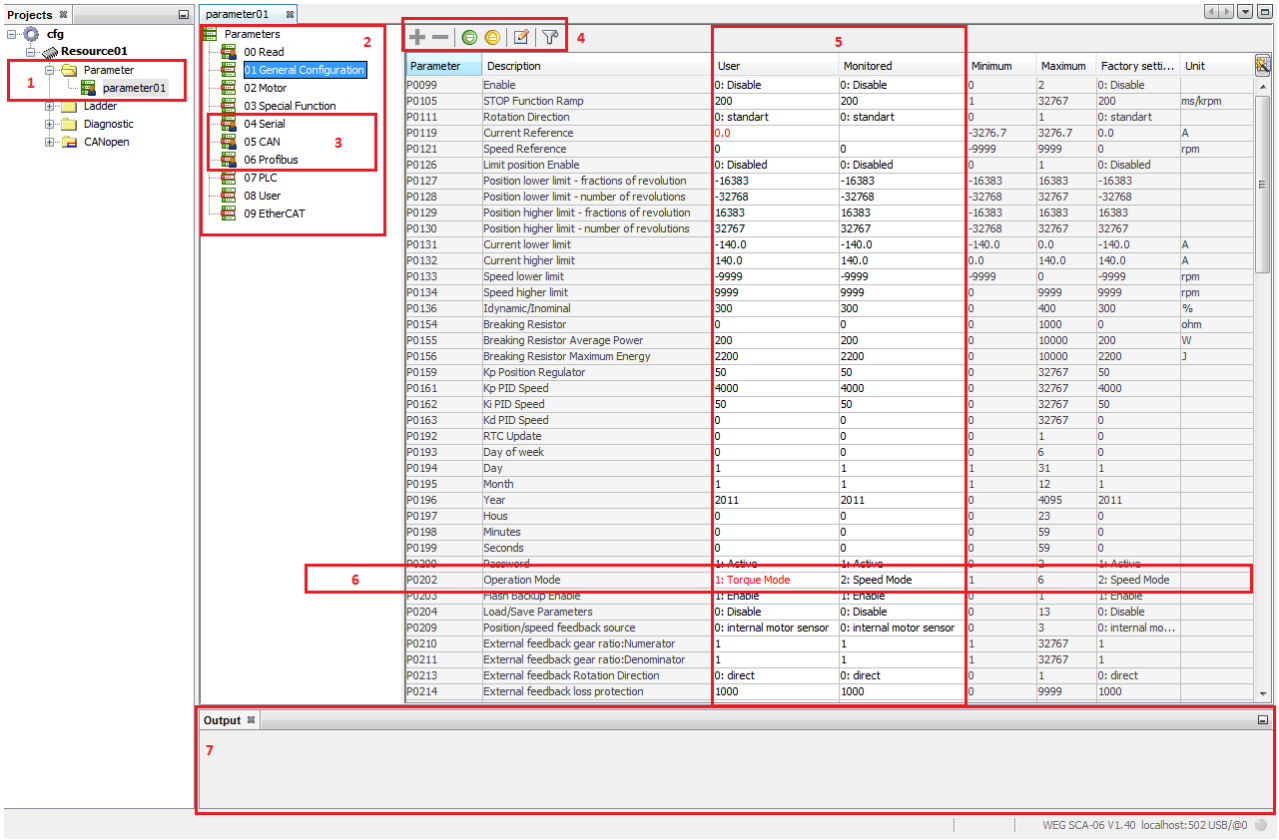
11.12.2Parameters

11.12.2.1 Overview

The parameter configuration screen is used to configure and monitor all the parameters of the equipment, including the user parameters.

NOTE!
The reading and writing of such parameters is done on this screen; only the user parameter configuration must be sent the first time or whenever modified by means of the resource download routine.

Below is an overview of the parameter configuration screen.



1. **Parameter files**. In this part are all the parameter configuration files created by the user. Notice that when the file features a person figure on the table, it means this parameter table contains hidden parameters/ group of parameters.
2. **Group of parameters**. This tree shows all the group of parameters. Notice that the same parameter can be in more than one group, and when its value is modified, it will be modified in all the groups to which it belongs.
3. **Modified group of parameters**. Group of parameters which contain the figure of a person on the table means they have hidden parameters.
4. **Commands**. The commands are described below in the order they appear:
   4.1. **Unhide parameter**: In case some parameter has been hidden, this button allows making it visible again.
   4.2. **Hide parameter**: Just select one or more parameters on the table and trigger this command to hide them.
   4.3. **Save table**: It saves the values of the parameters shown on the equipment screen; the sent values are the ones in the User column. The flow is User -> Monitored (equipment)
   4.4. **Read table**: It reads the parameters of the equipment shown in the Monitored column and saves them in the parameter file in the User column. The flow is Monitored (equipment) -> User
   4.5. **User parameters**: It opens a screen to edit the user parameters.
   4.6. **Filter**: It opens a parameter filter option, and it can filter by parameter number or description.
   4.7. **User Parameters and Monitored Parameters**. These two columns show the off-line and on-line parameters, so to speak. The **User** column shows the values contained in the file located on the computer and the **Monitored** column shows the values that are effectively saved on the equipment. Whenever you use the **Save Parameter** option, the sent values will be from the **User** column to the **Monitored** column, that is, File -> Equipment. In case of reading, the flow is the opposite, from the **Monitored** column to the **User** column, that is, Equipment -> File. In case you wish to change the values directly on the equipment without changing it in the file, just click on the monitored column,

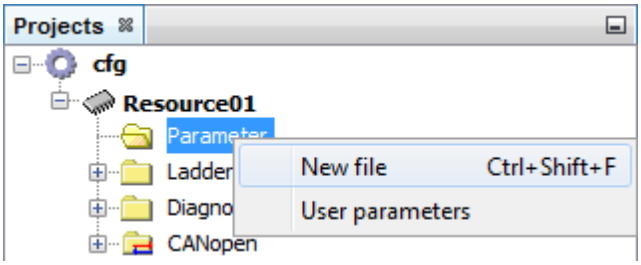
change the values and the modification will occur on-line.

5. **Modified parameters**: Whenever a parameter value in the **User** column is different from the **Monitored** column, it will be shown in red.
6. **Output**. This screen shows error information in case they occur during the writing or reading of the parameters.

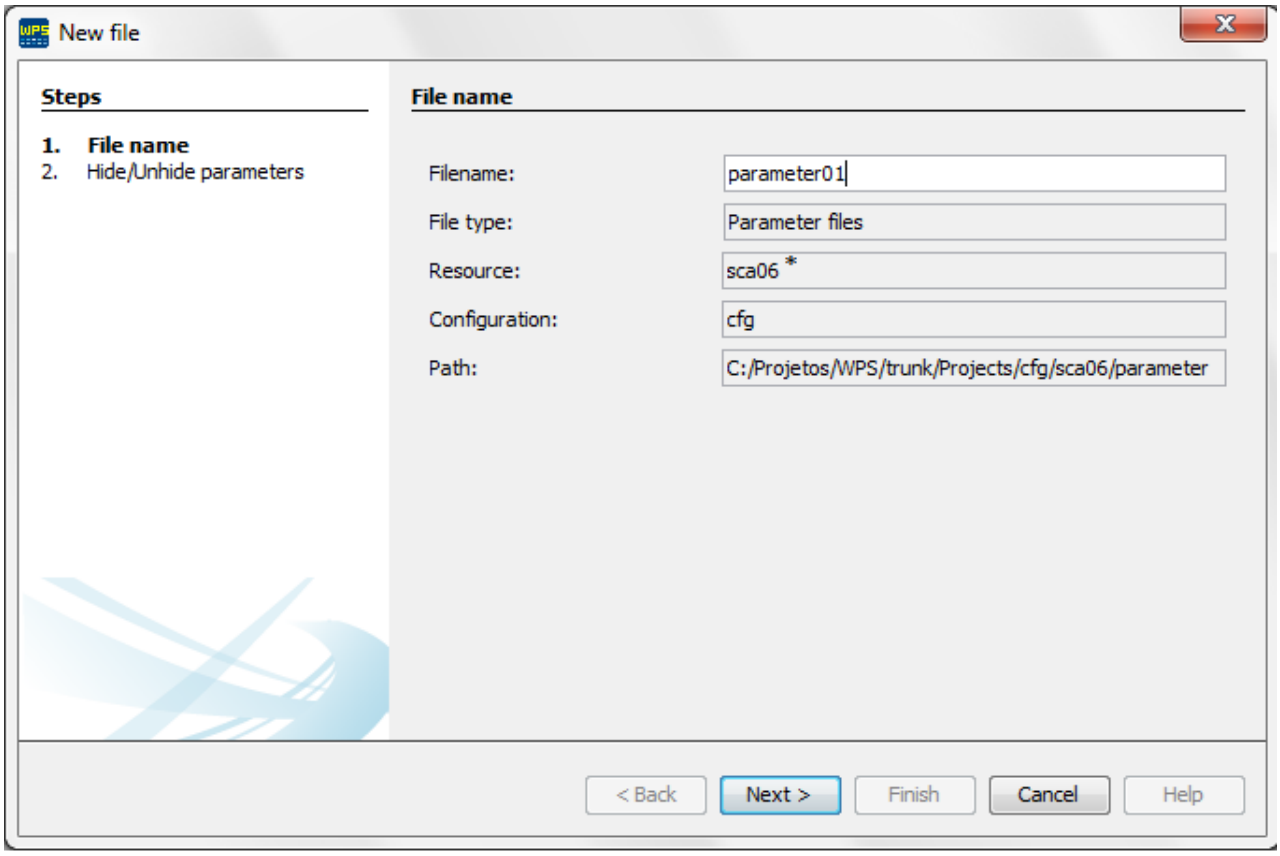
11.12.2.2 Configuration

Below is the list of the required steps to create a parameter file.

1. Create a new parameter file.

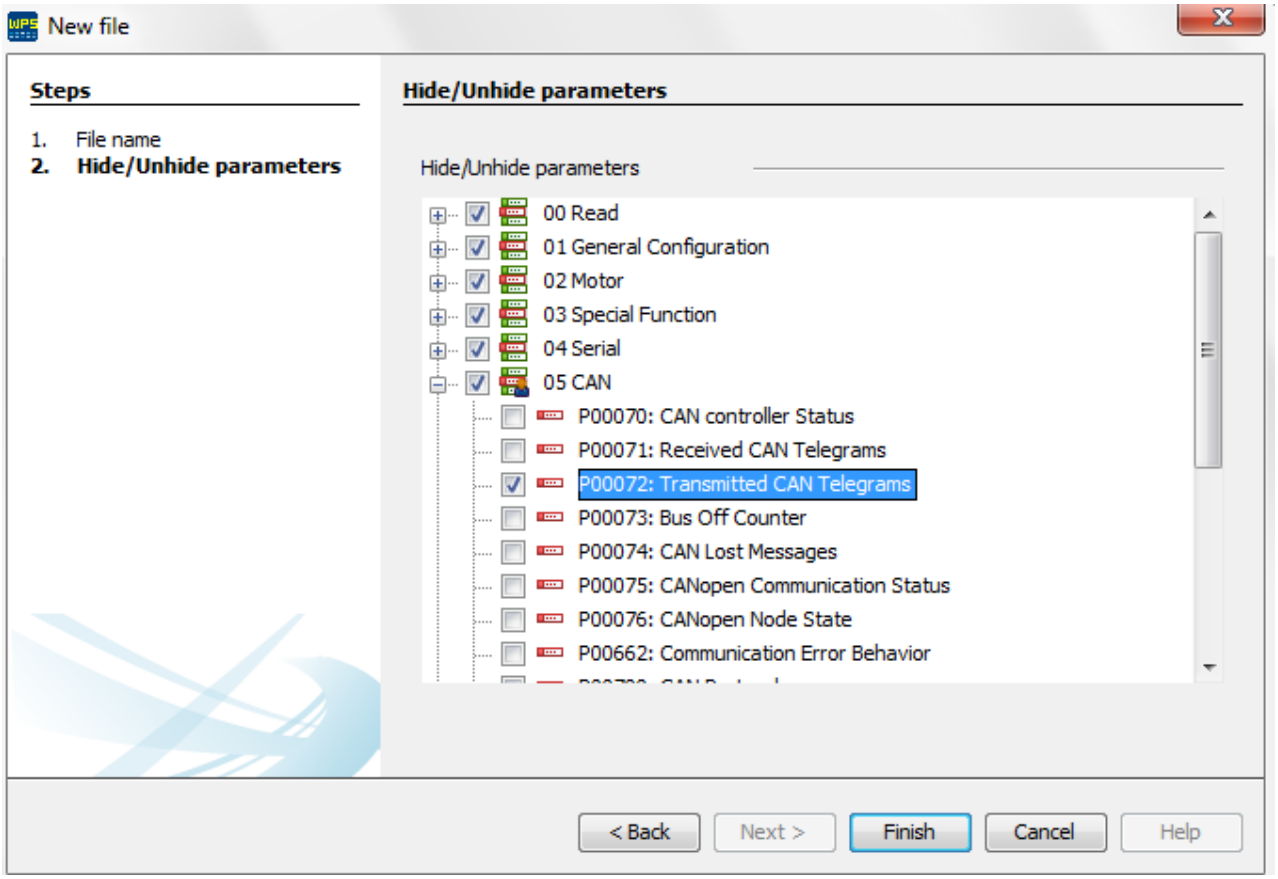


2. Define a name for the parameter file

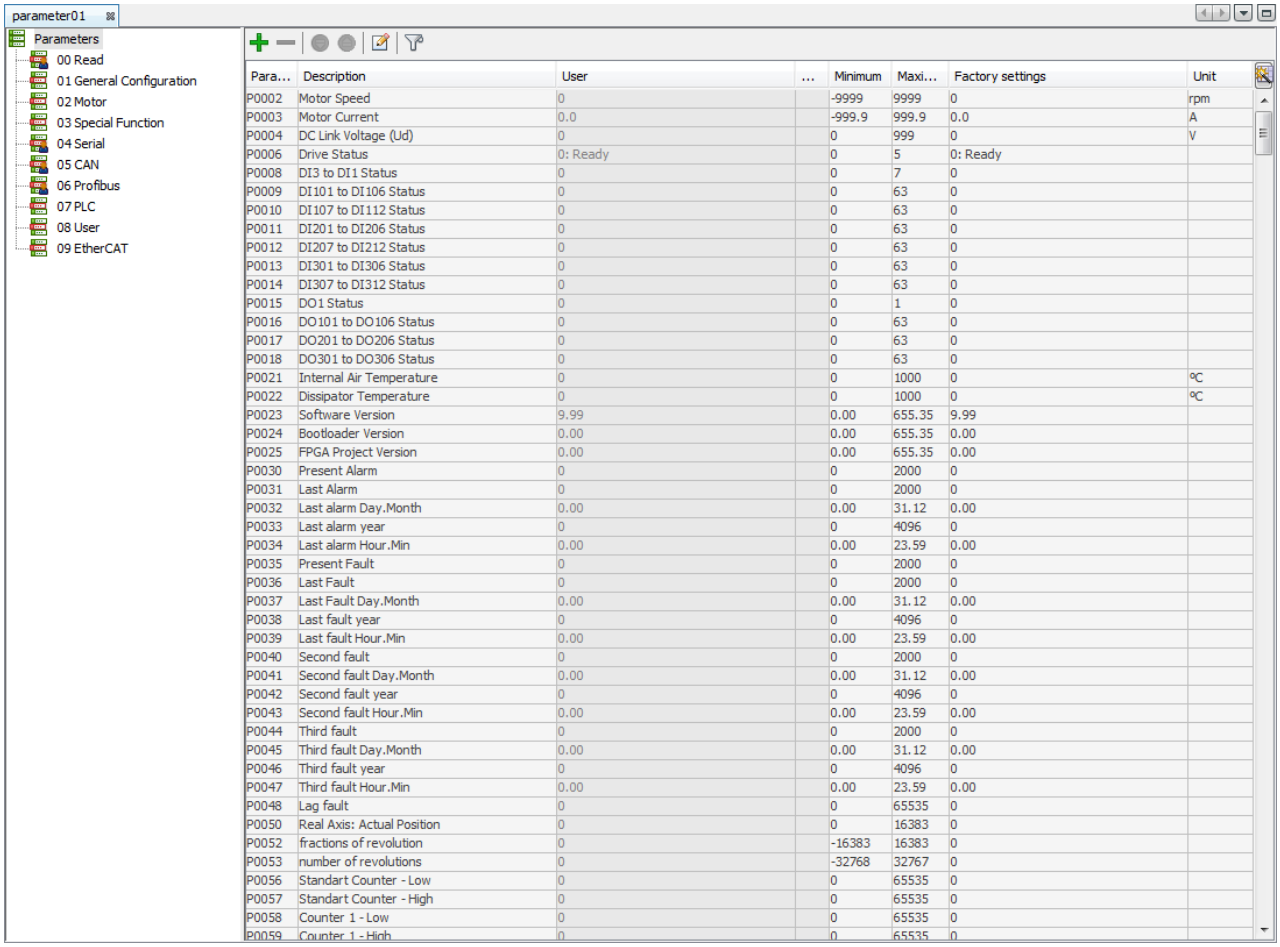


*** Resource: Resource01, SCA06, CFW300, etc.**

3. Configure which parameters you wish to view in your parameter table



4. After performing the steps above, the parameter file will be created and the equipment can be parameterized.

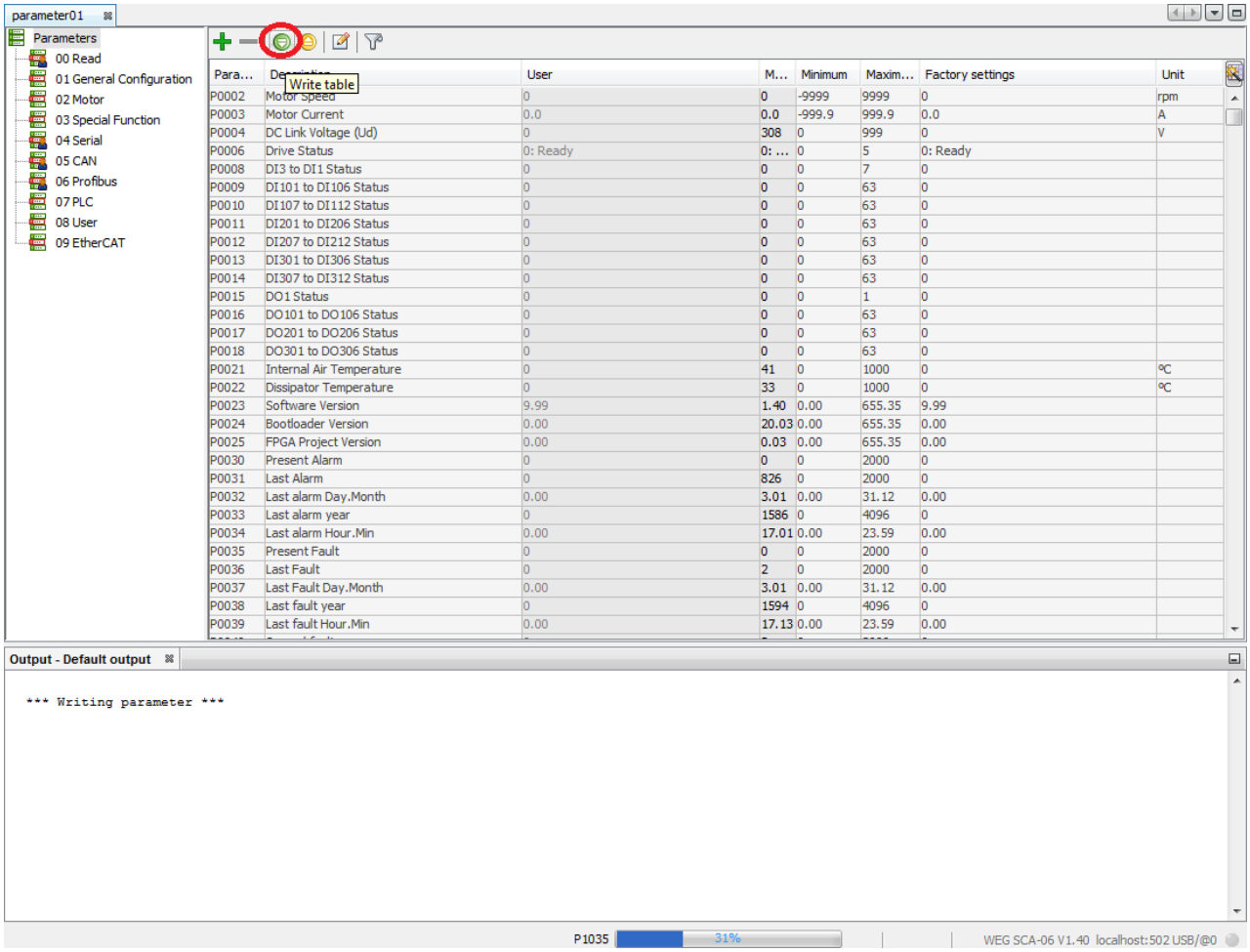


Para...	Description	User	...	Minimum	Maxi...	Factory settings	Unit
P0002	Motor Speed	0		-9999	9999	0	rpm
P0003	Motor Current	0.0		-999.9	999.9	0.0	A
P0004	DC Link Voltage (Ud)	0		0	999	0	V
P0006	Drive Status	0: Ready		0	5	0: Ready	
P0008	DI3 to DI1 Status	0		0	7	0	
P0009	DI101 to DI106 Status	0		0	63	0	
P0010	DI107 to DI112 Status	0		0	63	0	
P0011	DI201 to DI206 Status	0		0	63	0	
P0012	DI207 to DI212 Status	0		0	63	0	
P0013	DI301 to DI306 Status	0		0	63	0	
P0014	DI307 to DI312 Status	0		0	63	0	
P0015	DO1 Status	0		0	1	0	
P0016	DO101 to DO106 Status	0		0	63	0	
P0017	DO201 to DO206 Status	0		0	63	0	
P0018	DO301 to DO306 Status	0		0	63	0	
P0021	Internal Air Temperature	0		0	1000	0	ºC
P0022	Dissipator Temperature	0		0	1000	0	ºC
P0023	Software Version	9.99		0.00	655.35	9.99	
P0024	Bootloader Version	0.00		0.00	655.35	0.00	
P0025	FPGA Project Version	0.00		0.00	655.35	0.00	
P0030	Present Alarm	0		0	2000	0	
P0031	Last Alarm	0		0	2000	0	
P0032	Last alarm Day.Month	0.00		0.00	31.12	0.00	
P0033	Last alarm year	0		0	4096	0	
P0034	Last alarm Hour.Min	0.00		0.00	23.59	0.00	
P0035	Present Fault	0		0	2000	0	
P0036	Last Fault	0		0	2000	0	
P0037	Last Fault Day.Month	0.00		0.00	31.12	0.00	
P0038	Last fault year	0		0	4096	0	
P0039	Last fault Hour.Min	0.00		0.00	23.59	0.00	
P0040	Second fault	0		0	2000	0	
P0041	Second fault Day.Month	0.00		0.00	31.12	0.00	
P0042	Second fault year	0		0	4096	0	
P0043	Second fault Hour.Min	0.00		0.00	23.59	0.00	
P0044	Third fault	0		0	2000	0	
P0045	Third fault Day.Month	0.00		0.00	31.12	0.00	
P0046	Third fault year	0		0	4096	0	
P0047	Third fault Hour.Min	0.00		0.00	23.59	0.00	
P0048	Lag fault	0		0	65535	0	
P0050	Real Axis: Actual Position	0		0	16383	0	
P0052	fractions of revolution	0		-16383	16383	0	
P0053	number of revolutions	0		-32768	32767	0	
P0056	Standart Counter - Low	0		0	65535	0	
P0057	Standart Counter - High	0		0	65535	0	
P0058	Counter 1 - Low	0		0	65535	0	
P0059	Counter 1 - High	0		0	65535	0	

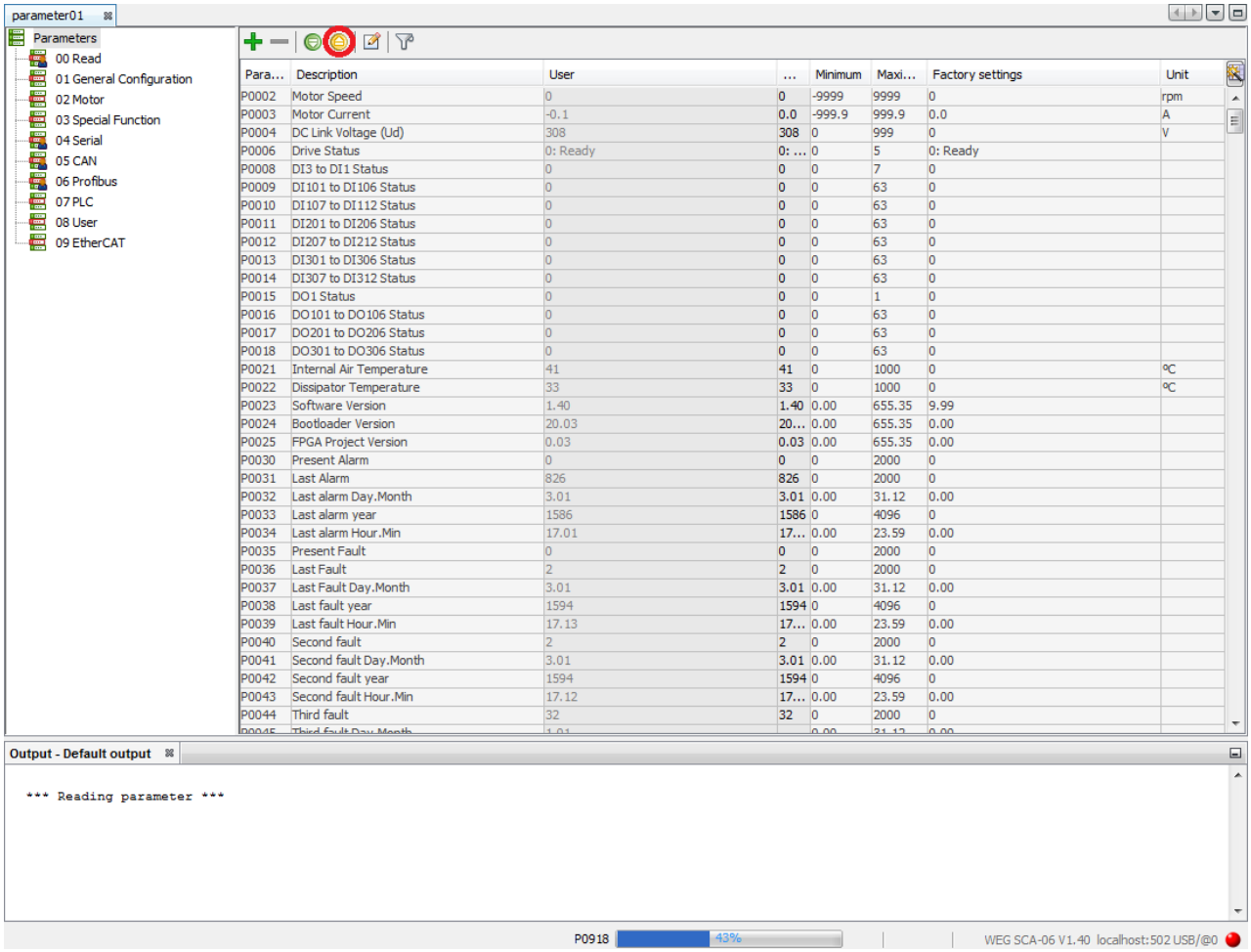
11.12.2.3 Read and Write of Parameters

There are 3 (three) ways to do the reading and writing of the parameters: by means of table, selection and group.

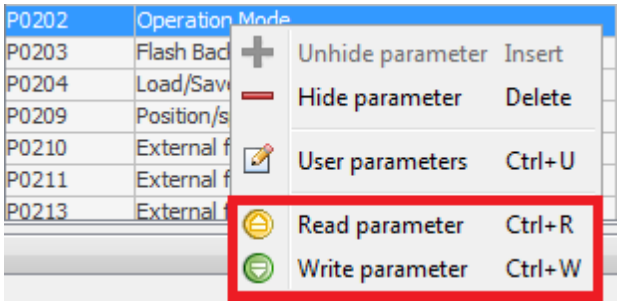
1. **Table writing**. The table writing command will send all visible parameters on the equipment screen. If and error occurs during the sending of some specific parameter, a message will be shown on the output window informing the error. It is important to notice that only visible parameters will be sent; therefore, it is necessary attention to which node of the group of parameters tree you are viewing. Example: If you wish to write all of them without filtering per group, just select the tree root.



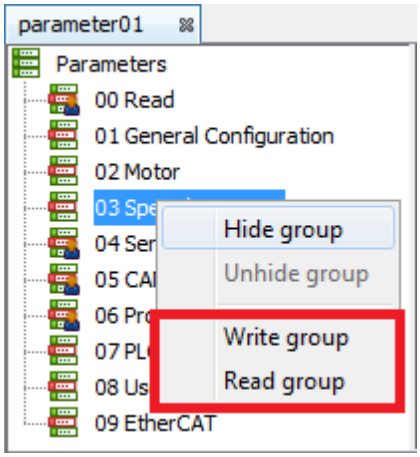
2. **Table reading**. The table reading command will read all the parameters of the equipment. If a error occurs during the reading of some specific parameter, a message will be shown on the output window informing the error. It is important to notice that only visible parameters will be read; therefore, it is necessary attention to which node of the group of parameters tree you are viewing. Example: If you wish to read all of them without filtering per group, just select the tree root.



3. **Reading/writing of specific parameters**. In order to read/write one or more specific parameters, just select them on the table, right click and choose the desired option: read or write parameter.



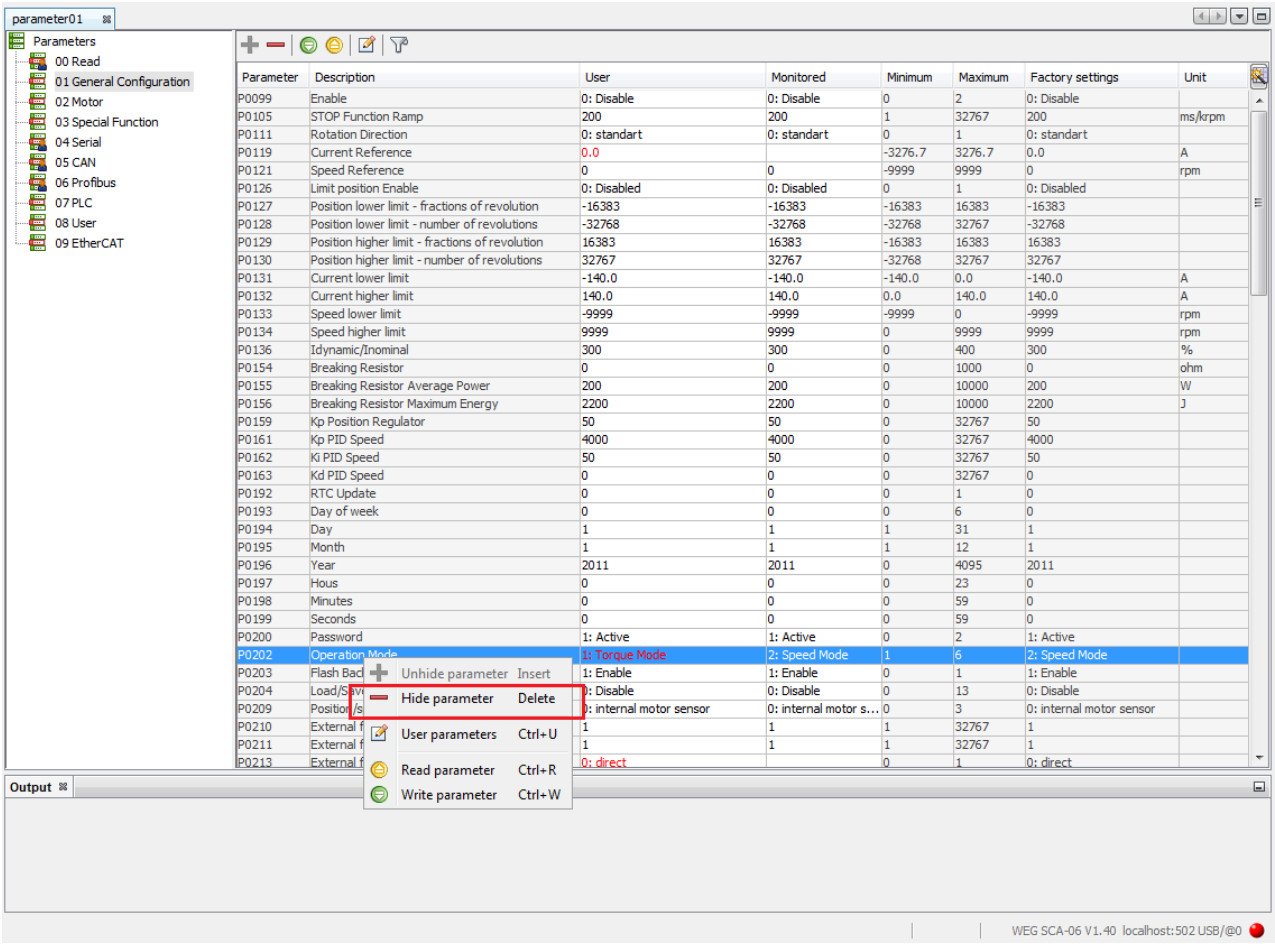
4. **Reading/writing of group of parameters**. In order to read/write only one group of parameters, just select it on the group tree, right click and choose the desired option: read or write group.



11.12.2.4 Hide/Unhide Parameters and Group of Parameters

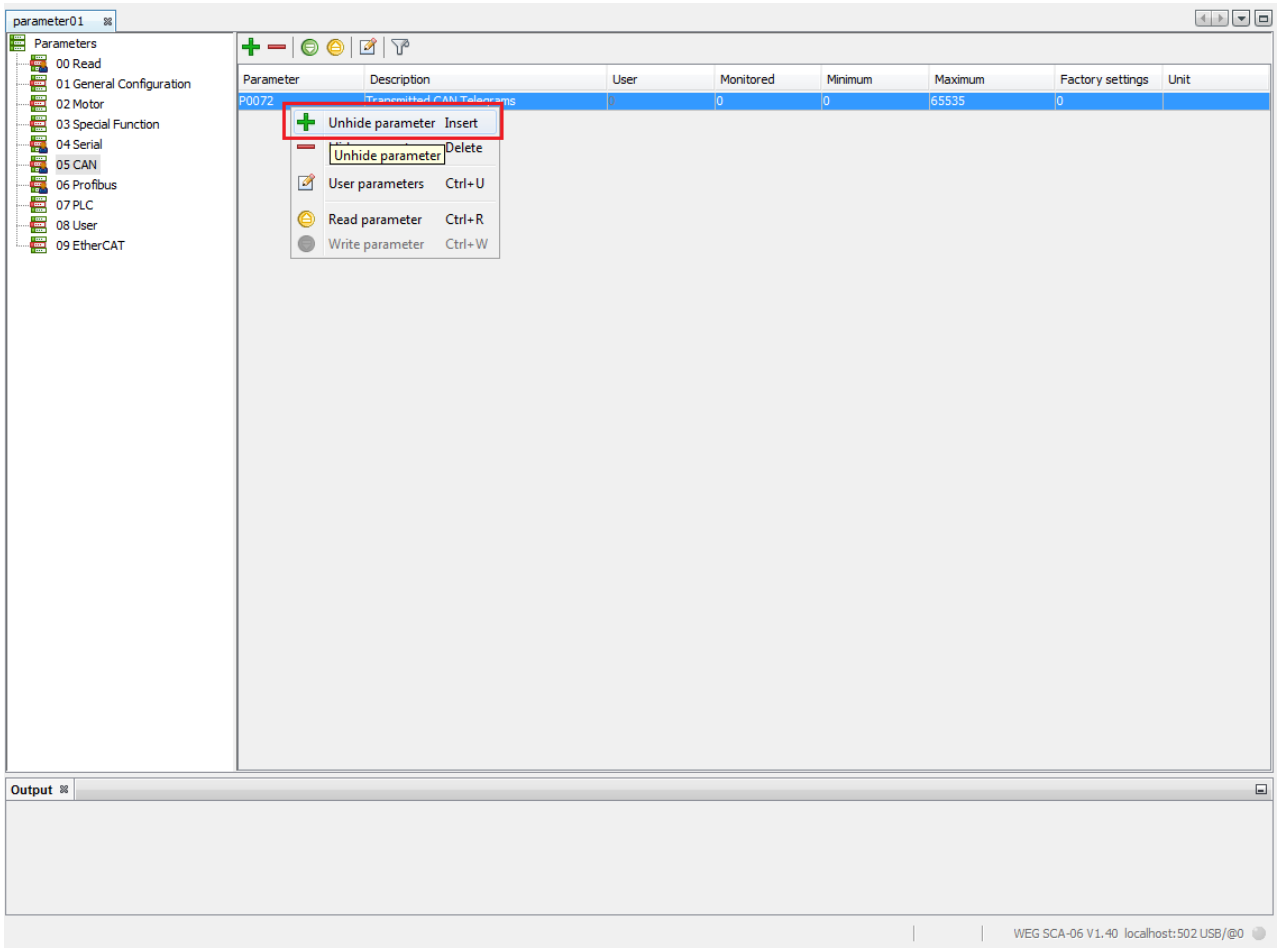
The parameter can be hidden/unhidden in two ways: individually or in group.

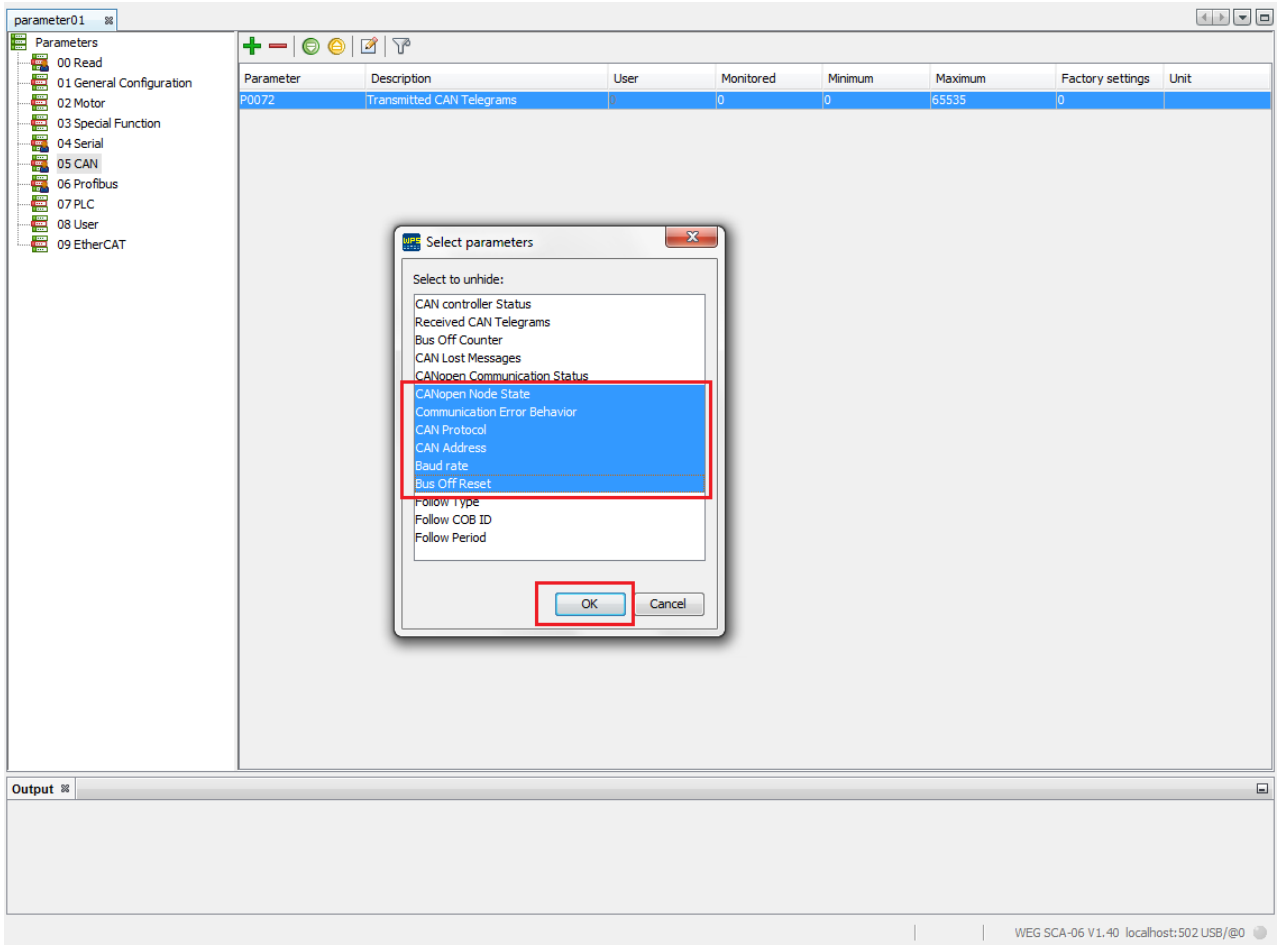
1. **Hide parameters**. In order to hide a parameter individually, just right click on the desired parameters and select the **Hide Parameter** option. You can also press the **Delete** key.

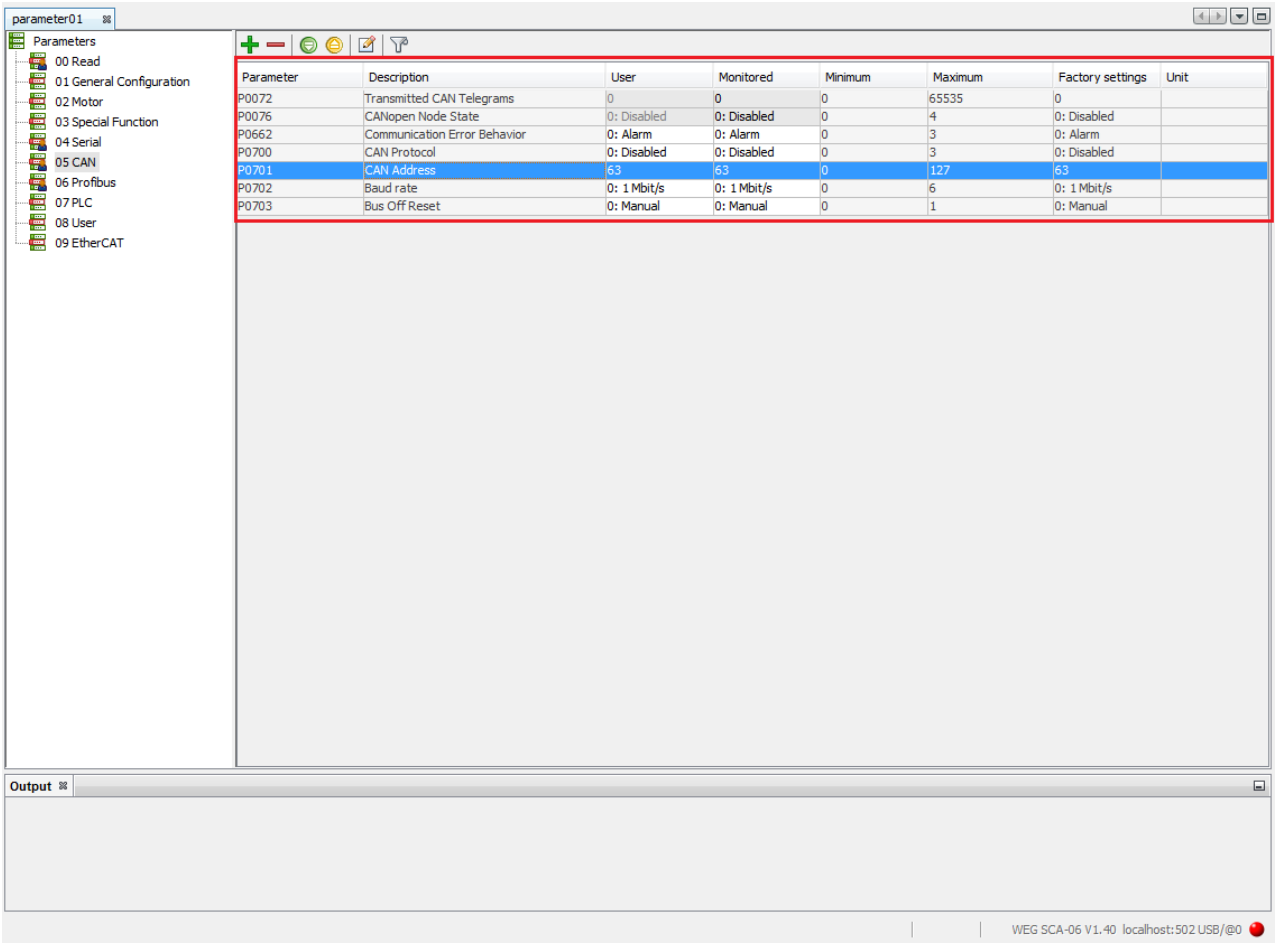


2. **Unhide Parameters**. In order to show hidden parameters, right click and choose the **Unhide Parameters**

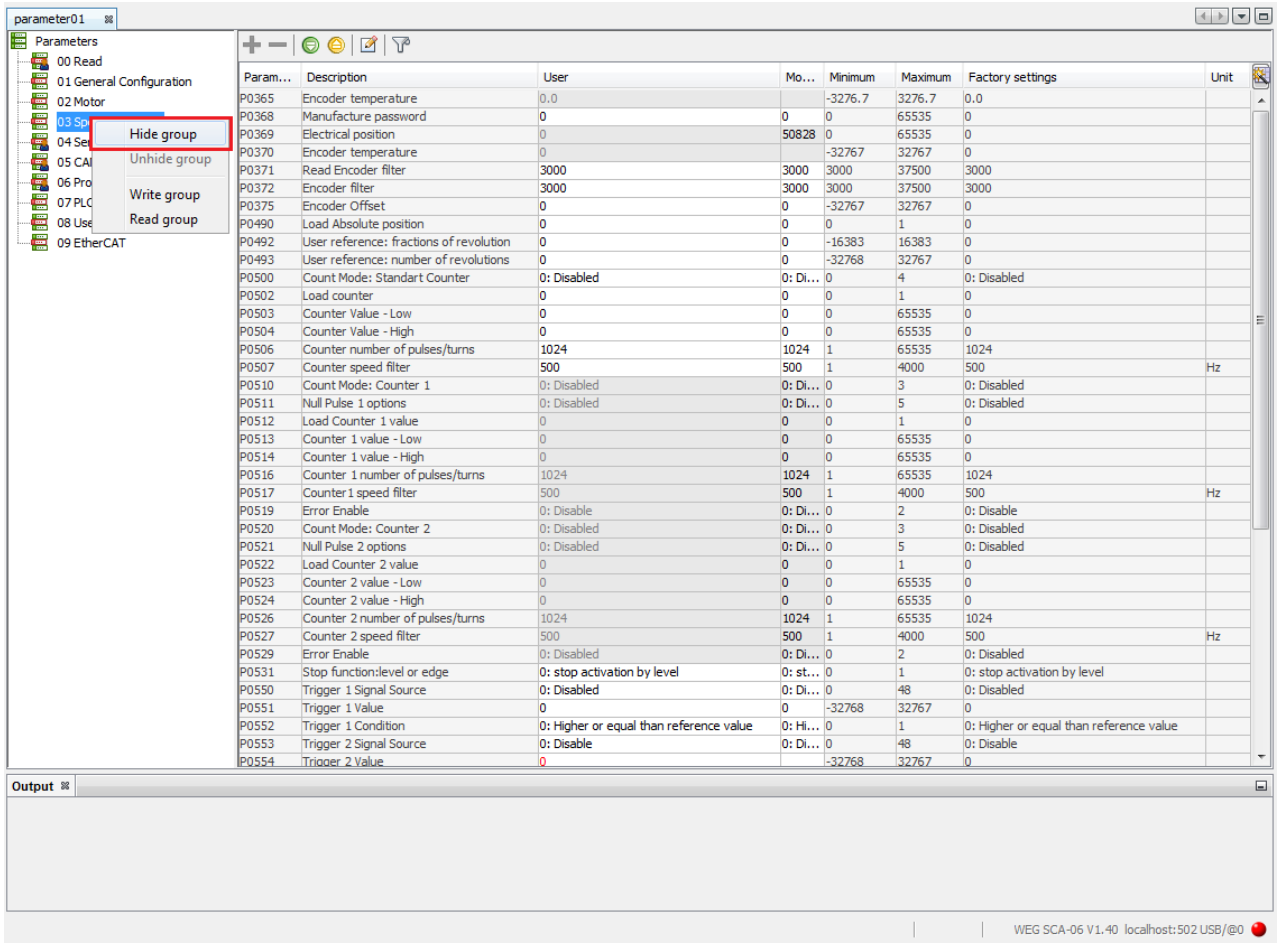
or press the **Insert** key. Then, a window will open and show the hidden parameters. Now, you just have to select the desired parameters and confirm.
Note: The parameters shown on this new window are only those which belong to the current filter according to the selection on the parameter group tree. In the figures below, the CAN group is selected; that means that only the hidden parameters of this group will be shown.

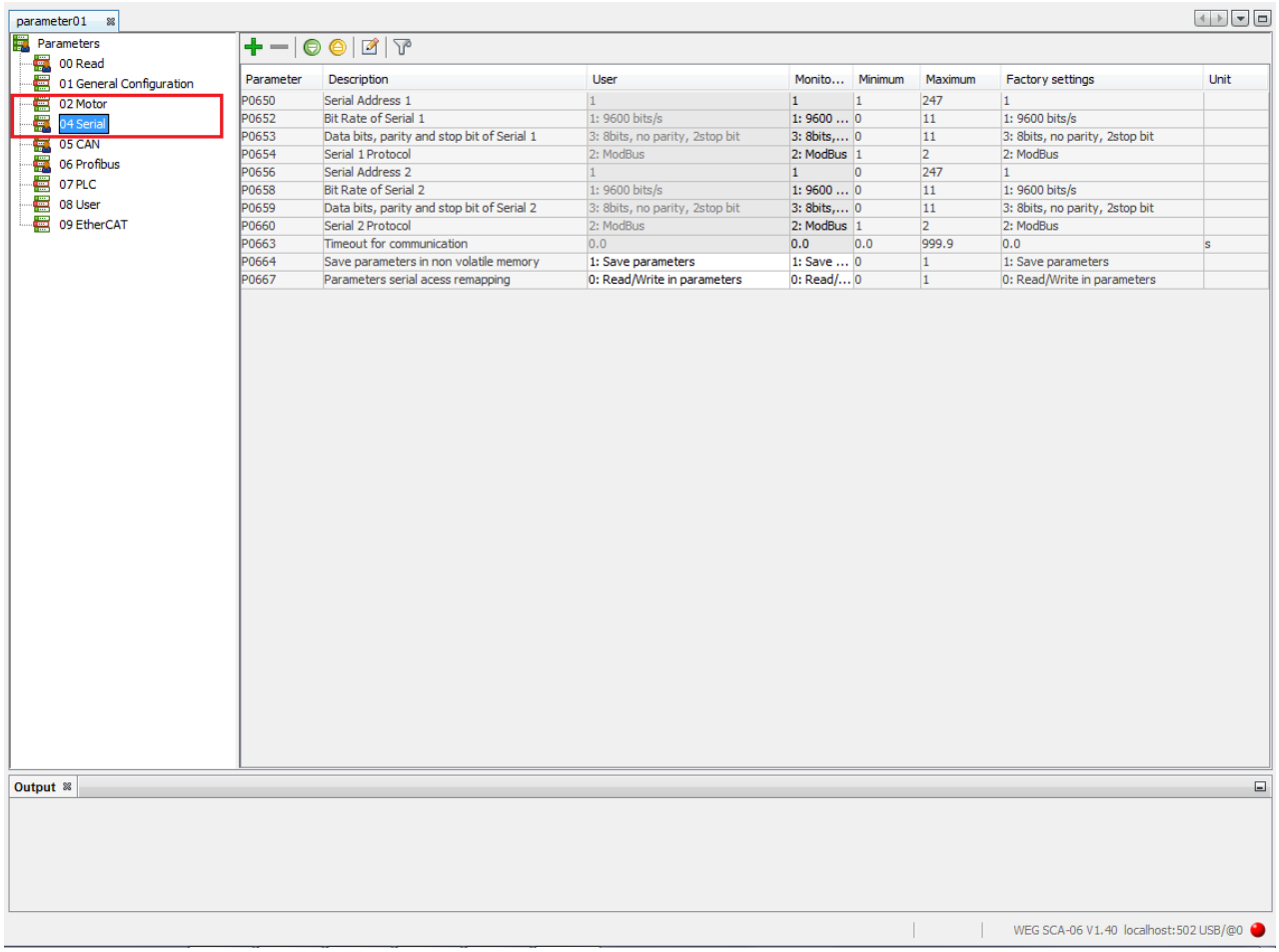



parameter01
Parameters
00 Read
01 General Configuration
02 Motor
03 Special Function
04 Serial
05 CAN
06 Profibus
07 PLC
08 User
09 EtherCAT
Parameter
Description
User
Monitored
Minimum
Maximum
Factory settings
Unit
P0072
Transmitted CAN Telegrams
0
0
65535
0
Select parameters
Select to unhide:
CAN controller Status
Received CAN Telegrams
Bus Off Counter
CAN Lost Messages
CANopen Communication Status
CANopen Node State
Communication Error Behavior
CAN Protocol
CAN Address
Baud rate
Bus Off Reset
Follow Type
Follow COB ID
Follow Period
OK
Cancel
Output
WEG SCA-06 V1.40 localhost:502 USB/@0

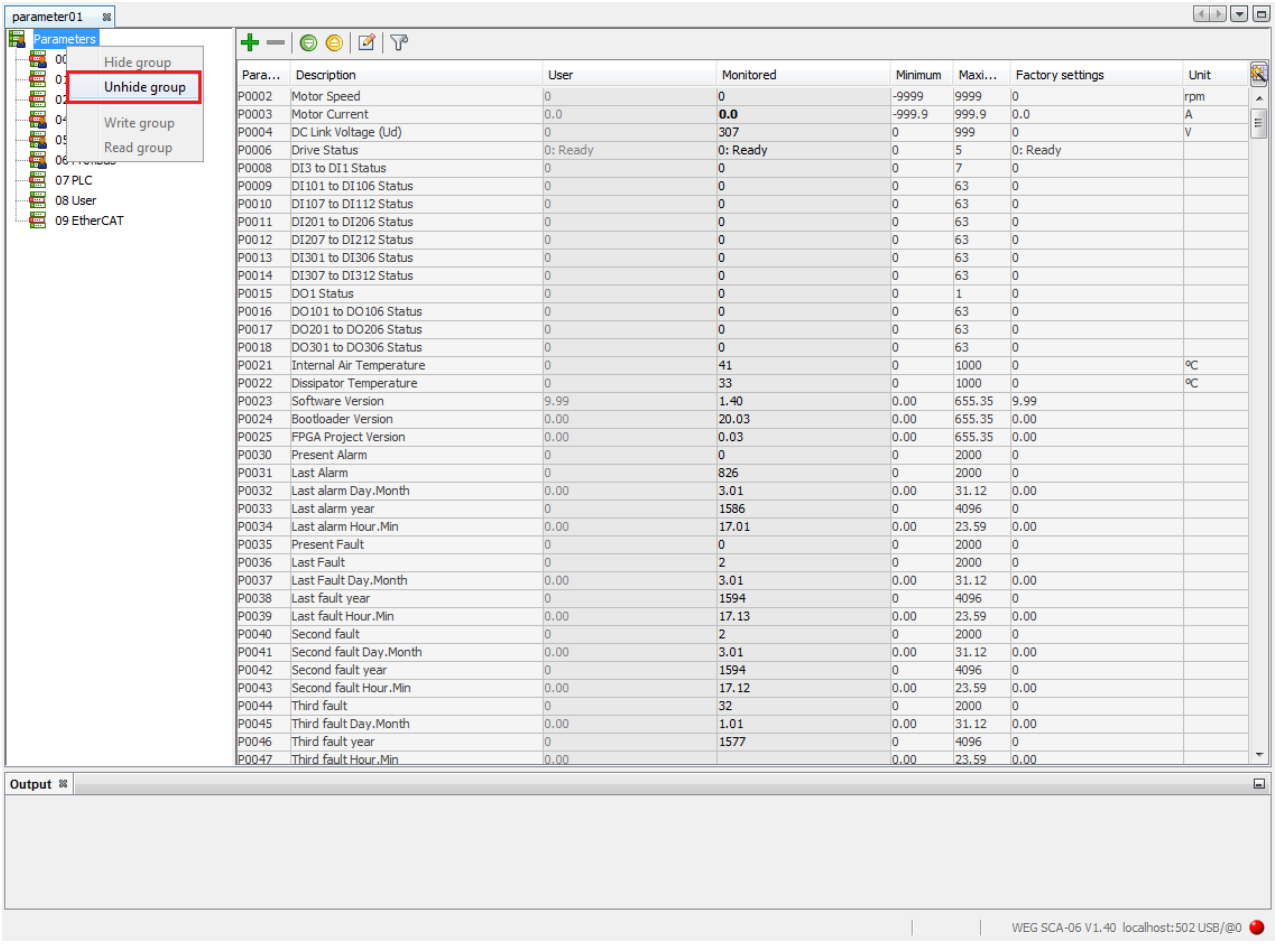


3. **Hide Group of Parameters**. In order to hide a group of parameters, just select the group on the tree and use the **Hide Group** option.

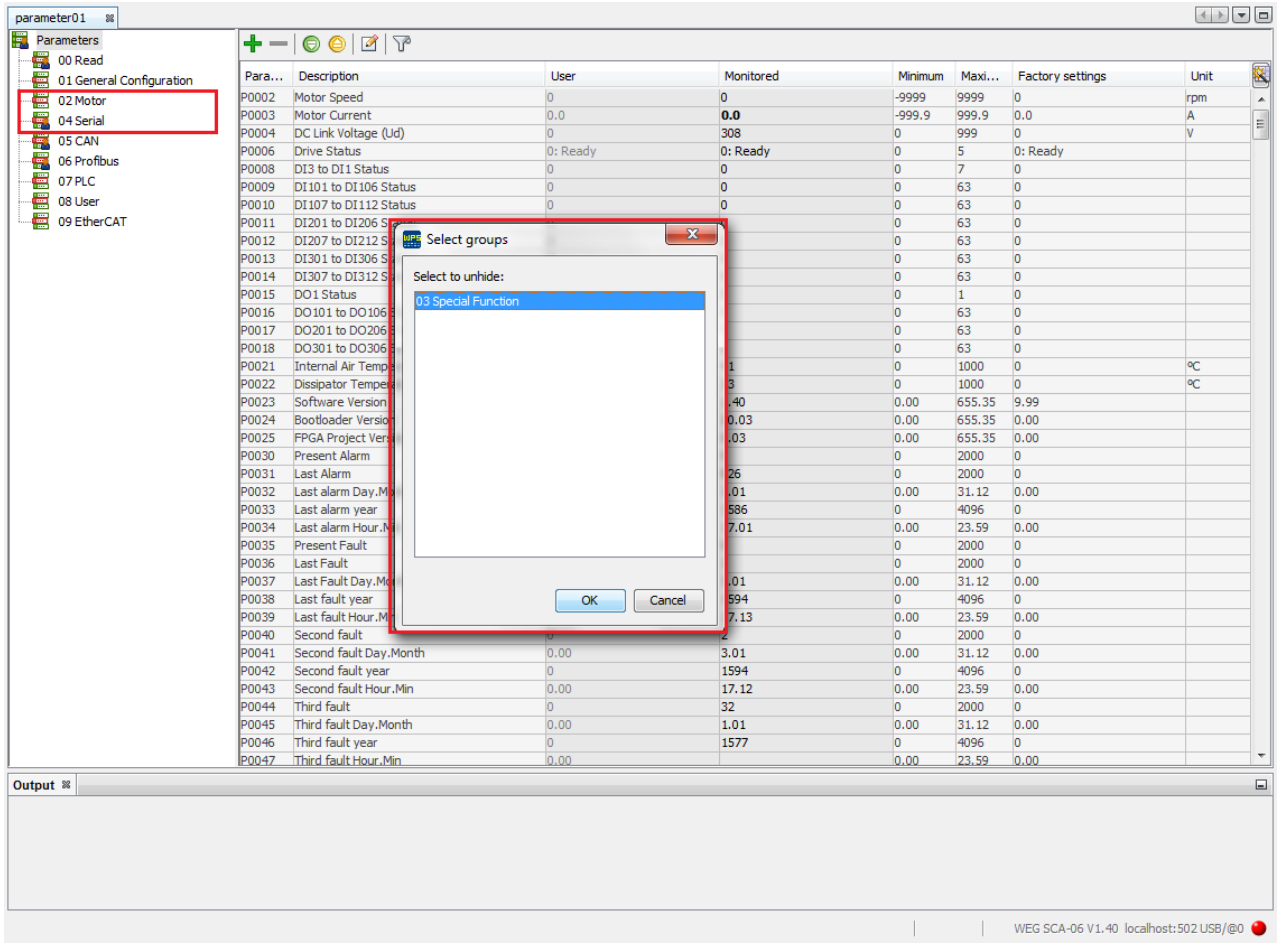

parameter01
Parameters
00 Read
01 General Configuration
02 Motor
03 Sp
04 Se
05 CA
06 Pro
07 PLC
08 Use
09 EtherCAT
Hide group
Unhide group
Write group
Read group
Param...
Description
User
Mo...
Minimum
Maximum
Factory settings
Unit
P0365 Encoder temperature 0.0 -3276.7 3276.7 0.0
P0368 Manufacture password 0 0 0 65535 0
P0369 Electrical position 0 50828 0 65535 0
P0370 Encoder temperature 0 -32767 32767 0
P0371 Read Encoder filter 3000 3000 3000 37500 3000
P0372 Encoder filter 3000 3000 3000 37500 3000
P0375 Encoder Offset 0 0 -32767 32767 0
P0490 Load Absolute position 0 0 0 1 0
P0492 User reference: fractions of revolution 0 0 -16383 16383 0
P0493 User reference: number of revolutions 0 0 -32768 32767 0
P0500 Count Mode: Standart Counter 0: Disabled 0: Di... 0 4 0: Disabled
P0502 Load counter 0 0 0 1 0
P0503 Counter Value - Low 0 0 0 65535 0
P0504 Counter Value - High 0 0 0 65535 0
P0506 Counter number of pulses/turns 1024 1024 1 65535 1024
P0507 Counter speed filter 500 500 1 4000 500 Hz
P0510 Count Mode: Counter 1 0: Disabled 0: Di... 0 3 0: Disabled
P0511 Null Pulse 1 options 0: Disabled 0: Di... 0 5 0: Disabled
P0512 Load Counter 1 value 0 0 0 1 0
P0513 Counter 1 value - Low 0 0 0 65535 0
P0514 Counter 1 value - High 0 0 0 65535 0
P0516 Counter 1 number of pulses/turns 1024 1024 1 65535 1024
P0517 Counter 1 speed filter 500 500 1 4000 500 Hz
P0519 Error Enable 0: Disable 0: Di... 0 2 0: Disable
P0520 Count Mode: Counter 2 0: Disabled 0: Di... 0 3 0: Disabled
P0521 Null Pulse 2 options 0: Disabled 0: Di... 0 5 0: Disabled
P0522 Load Counter 2 value 0 0 0 1 0
P0523 Counter 2 value - Low 0 0 0 65535 0
P0524 Counter 2 value - High 0 0 0 65535 0
P0526 Counter 2 number of pulses/turns 1024 1024 1 65535 1024
P0527 Counter 2 speed filter 500 500 1 4000 500 Hz
P0529 Error Enable 0: Disabled 0: Di... 0 2 0: Disabled
P0531 Stop function:level or edge 0: stop activation by level 0: st... 0 1 0: stop activation by level
P0550 Trigger 1 Signal Source 0: Disabled 0: Di... 0 48 0: Disabled
P0551 Trigger 1 Value 0 0 -32768 32767 0
P0552 Trigger 1 Condition 0: Higher or equal than reference value 0: Hi... 0 1 0: Higher or equal than reference value
P0553 Trigger 2 Signal Source 0: Disable 0: Di... 0 48 0: Disable
P0554 Trigger 2 Value 0 -32768 32767 0
Output
WEG SCA-06 V1.40 localhost:502 USB/@0



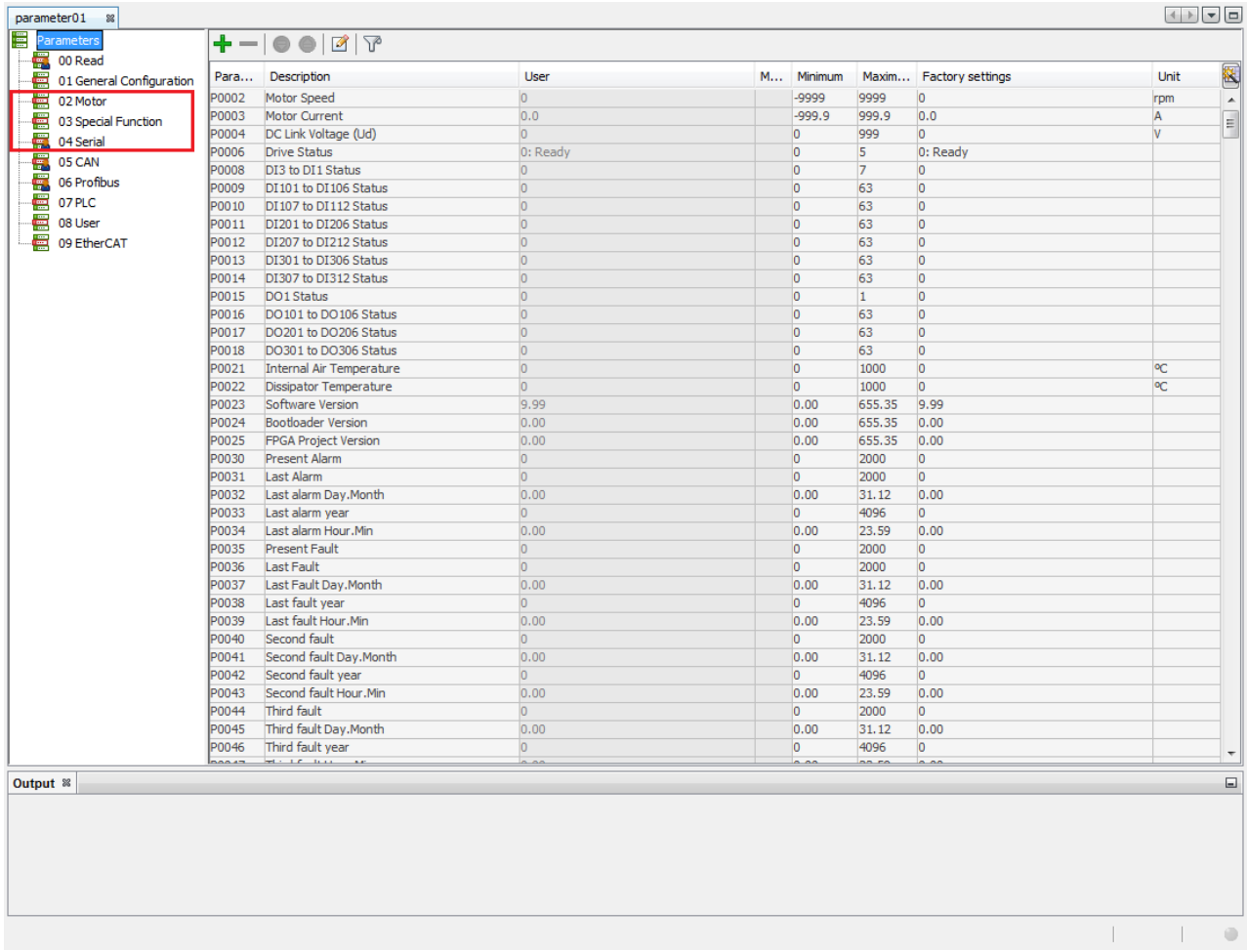
4. **Unhide Group of Parameters**. In order to show a hidden group of parameters, just select the root of the group tree and select the **Unhide Group** option. A window will open showing the groups that are hidden; then just select the group you wish to unhide.



Para...	Description	User	Monitored	Minimum	Maxi...	Factory settings	Unit
P0002	Motor Speed	0	0	-9999	9999	0	rpm
P0003	Motor Current	0.0	**0.0**	-999.9	999.9	0.0	A
P0004	DC Link Voltage (Ud)	0	307	0	999	0	V
P0006	Drive Status	0: Ready	0: Ready	0	5	0: Ready	
P0008	DI3 to DI1 Status	0	0	0	7	0	
P0009	DI101 to DI106 Status	0	0	0	63	0	
P0010	DI107 to DI112 Status	0	0	0	63	0	
P0011	DI201 to DI206 Status	0	0	0	63	0	
P0012	DI207 to DI212 Status	0	0	0	63	0	
P0013	DI301 to DI306 Status	0	0	0	63	0	
P0014	DI307 to DI312 Status	0	0	0	63	0	
P0015	DO1 Status	0	0	0	1	0	
P0016	DO101 to DO106 Status	0	0	0	63	0	
P0017	DO201 to DO206 Status	0	0	0	63	0	
P0018	DO301 to DO306 Status	0	0	0	63	0	
P0021	Internal Air Temperature	0	41	0	1000	0	°C
P0022	Dissipator Temperature	0	33	0	1000	0	°C
P0023	Software Version	9.99	1.40	0.00	655.35	9.99	
P0024	Bootloader Version	0.00	20.03	0.00	655.35	0.00	
P0025	FPGA Project Version	0.00	0.03	0.00	655.35	0.00	
P0030	Present Alarm	0	0	0	2000	0	
P0031	Last Alarm	0	826	0	2000	0	
P0032	Last alarm Day.Month	0.00	3.01	0.00	31.12	0.00	
P0033	Last alarm year	0	1586	0	4096	0	
P0034	Last alarm Hour.Min	0.00	17.01	0.00	23.59	0.00	
P0035	Present Fault	0	0	0	2000	0	
P0036	Last Fault	0	2	0	2000	0	
P0037	Last Fault Day.Month	0.00	3.01	0.00	31.12	0.00	
P0038	Last fault year	0	1594	0	4096	0	
P0039	Last fault Hour.Min	0.00	17.13	0.00	23.59	0.00	
P0040	Second fault	0	2	0	2000	0	
P0041	Second fault Day.Month	0.00	3.01	0.00	31.12	0.00	
P0042	Second fault year	0	1594	0	4096	0	
P0043	Second fault Hour.Min	0.00	17.12	0.00	23.59	0.00	
P0044	Third fault	0	32	0	2000	0	
P0045	Third fault Day.Month	0.00	1.01	0.00	31.12	0.00	
P0046	Third fault year	0	1577	0	4096	0	
P0047	Third fault Hour.Min	0.00		0.00	23.59	0.00	

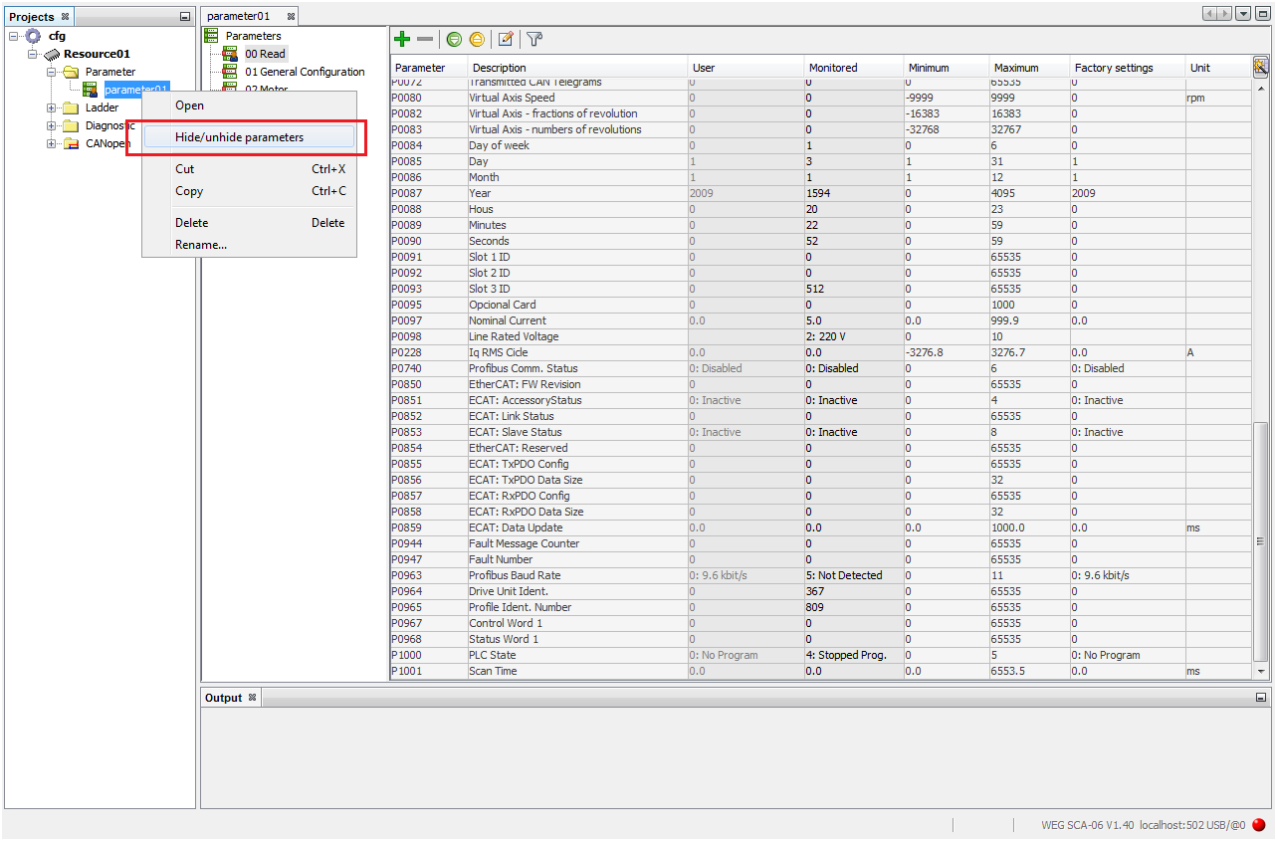


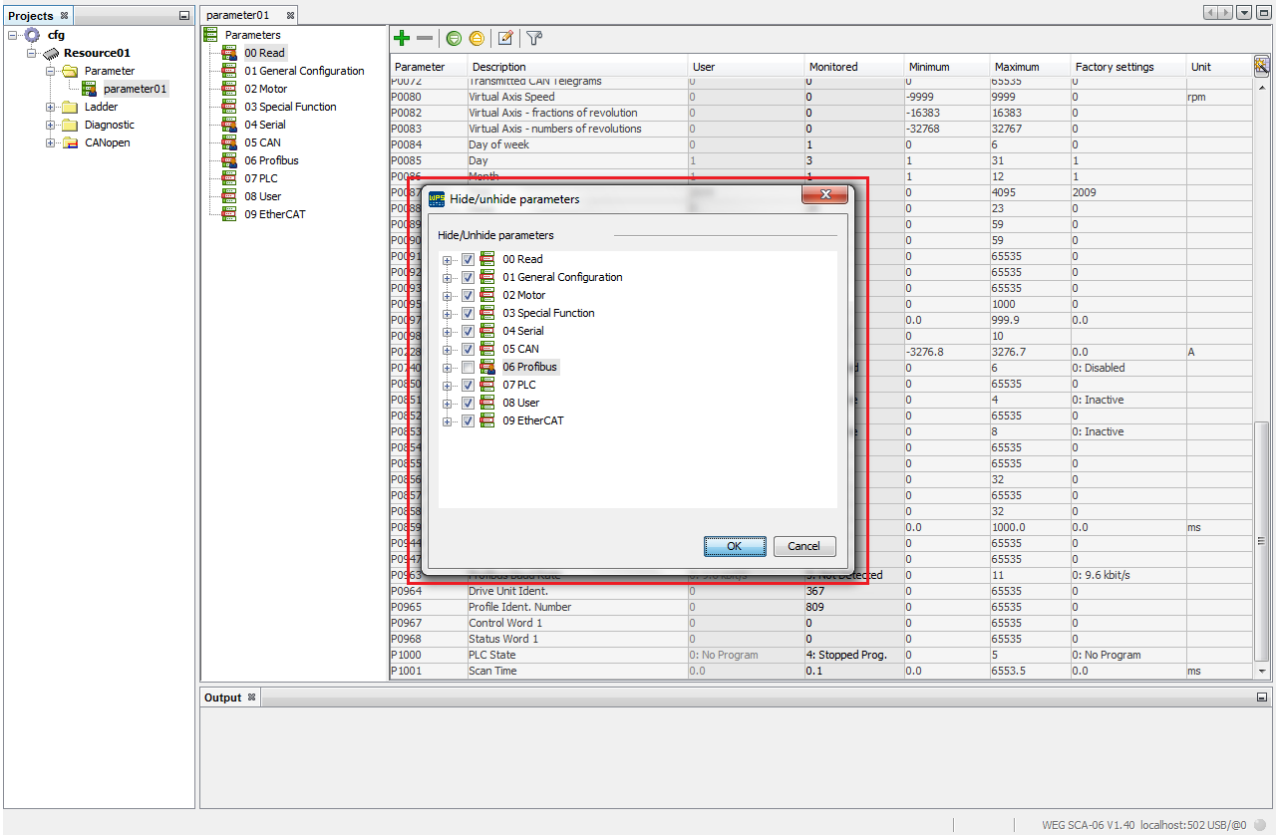
Para...	Description	User	Monitored	Minimum	Maxi...	Factory settings	Unit
P0002	Motor Speed	0	0	-9999	9999	0	rpm
P0003	Motor Current	0.0	**0.0**	-999.9	999.9	0.0	A
P0004	DC Link Voltage (Ud)	0	308	0	999	0	V
P0006	Drive Status	0: Ready	0: Ready	0	5	0: Ready	
P0008	DI3 to DI1 Status	0	0	0	7	0	
P0009	DI101 to DI106 Status	0	0	0	63	0	
P0010	DI107 to DI112 Status	0	0	0	63	0	
P0011	DI201 to DI206 S			0	63	0	
P0012	DI207 to DI212 S			0	63	0	
P0013	DI301 to DI306 S			0	63	0	
P0014	DI307 to DI312 S			0	63	0	
P0015	DO1 Status			0	1	0	
P0016	DO101 to DO106			0	63	0	
P0017	DO201 to DO206			0	63	0	
P0018	DO301 to DO306			0	63	0	
P0021	Internal Air Temp		1	0	1000	0	°C
P0022	Dissipator Temper		3	0	1000	0	°C
P0023	Software Version		.40	0.00	655.35	9.99	
P0024	Bootloader Versio		0.03	0.00	655.35	0.00	
P0025	FPGA Project Ver		.03	0.00	655.35	0.00	
P0030	Present Alarm			0	2000	0	
P0031	Last Alarm		26	0	2000	0	
P0032	Last alarm Day.M		.01	0.00	31.12	0.00	
P0033	Last alarm year		586	0	4096	0	
P0034	Last alarm Hour.M		7.01	0.00	23.59	0.00	
P0035	Present Fault			0	2000	0	
P0036	Last Fault			0	2000	0	
P0037	Last Fault Day.Mo		.01	0.00	31.12	0.00	
P0038	Last fault year		594	0	4096	0	
P0039	Last fault Hour.M		7.13	0.00	23.59	0.00	
P0040	Second fault		2	0	2000	0	
P0041	Second fault Day.Month	0.00	3.01	0.00	31.12	0.00	
P0042	Second fault year	0	1594	0	4096	0	
P0043	Second fault Hour.Min	0.00	17.12	0.00	23.59	0.00	
P0044	Third fault	0	32	0	2000	0	
P0045	Third fault Day.Month	0.00	1.01	0.00	31.12	0.00	
P0046	Third fault year	0	1577	0	4096	0	
P0047	Third fault Hour.Min	0.00		0.00	23.59	0.00	



Para...	Description	User	M...	Minimum	Maxim...	Factory settings	Unit
P0002	Motor Speed	0		-9999	9999	0	rpm
P0003	Motor Current	0.0		-999.9	999.9	0.0	A
P0004	DC Link Voltage (Ud)	0		0	999	0	V
P0006	Drive Status	0: Ready		0	5	0: Ready	
P0008	DI3 to DI1 Status	0		0	7	0	
P0009	DI101 to DI106 Status	0		0	63	0	
P0010	DI107 to DI112 Status	0		0	63	0	
P0011	DI201 to DI206 Status	0		0	63	0	
P0012	DI207 to DI212 Status	0		0	63	0	
P0013	DI301 to DI306 Status	0		0	63	0	
P0014	DI307 to DI312 Status	0		0	63	0	
P0015	DO1 Status	0		0	1	0	
P0016	DO101 to DO106 Status	0		0	63	0	
P0017	DO201 to DO206 Status	0		0	63	0	
P0018	DO301 to DO306 Status	0		0	63	0	
P0021	Internal Air Temperature	0		0	1000	0	°C
P0022	Dissipator Temperature	0		0	1000	0	°C
P0023	Software Version	9.99		0.00	655.35	9.99	
P0024	Bootloader Version	0.00		0.00	655.35	0.00	
P0025	FPGA Project Version	0.00		0.00	655.35	0.00	
P0030	Present Alarm	0		0	2000	0	
P0031	Last Alarm	0		0	2000	0	
P0032	Last alarm Day.Month	0.00		0.00	31.12	0.00	
P0033	Last alarm year	0		0	4096	0	
P0034	Last alarm Hour.Min	0.00		0.00	23.59	0.00	
P0035	Present Fault	0		0	2000	0	
P0036	Last Fault	0		0	2000	0	
P0037	Last Fault Day.Month	0.00		0.00	31.12	0.00	
P0038	Last fault year	0		0	4096	0	
P0039	Last fault Hour.Min	0.00		0.00	23.59	0.00	
P0040	Second fault	0		0	2000	0	
P0041	Second fault Day.Month	0.00		0.00	31.12	0.00	
P0042	Second fault year	0		0	4096	0	
P0043	Second fault Hour.Min	0.00		0.00	23.59	0.00	
P0044	Third fault	0		0	2000	0	
P0045	Third fault Day.Month	0.00		0.00	31.12	0.00	
P0046	Third fault year	0		0	4096	0	

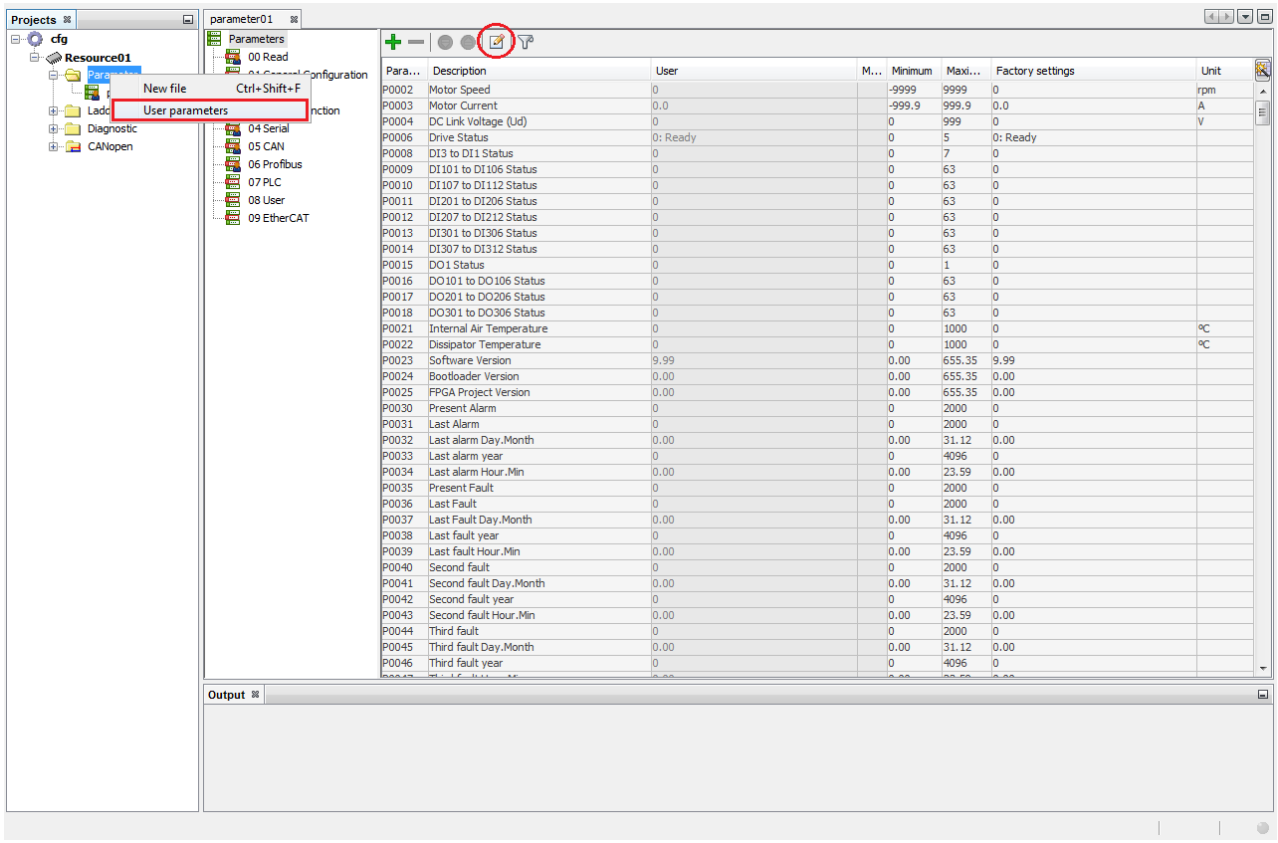
5. **Hide and Show Parameters and Groups of Parameters**. By means of this option, you have full control of the parameters and groups of parameters. It is possible to hide and unhide individual parameters, multiple parameters, individual groups and multiple groups in the same action.


Projects
cfg
Resource01
Parameter
parameter01
Ladder
Diagnostic
CANopen
parameter01
Parameters
00 Read
01 General Configuration
Open
Hide/unhide parameters
Cut Ctrl+X
Copy Ctrl+C
Delete Delete
Rename...
Parameter
Description
User
Monitored
Minimum
Maximum
Factory settings
Unit
P0080 Virtual Axis Speed 0 0 -9999 9999 0 rpm
P0082 Virtual Axis - fractions of revolution 0 0 -16383 16383 0
P0083 Virtual Axis - numbers of revolutions 0 0 -32768 32767 0
P0084 Day of week 0 1 0 6 0
P0085 Day 1 3 1 31 1
P0086 Month 1 1 1 12 1
P0087 Year 2009 1594 0 4095 2009
P0088 Hous 0 20 0 23 0
P0089 Minutes 0 22 0 59 0
P0090 Seconds 0 52 0 59 0
P0091 Slot 1 ID 0 0 0 65535 0
P0092 Slot 2 ID 0 0 0 65535 0
P0093 Slot 3 ID 0 512 0 65535 0
P0095 Opcional Card 0 0 0 1000 0
P0097 Nominal Current 0.0 5.0 0.0 999.9 0.0
P0098 Line Rated Voltage 2: 220 V 0 10
P0228 Iq RMS Cicle 0.0 0.0 -3276.8 3276.7 0.0 A
P0740 Profibus Comm. Status 0: Disabled 0: Disabled 0 6 0: Disabled
P0850 EtherCAT: FW Revision 0 0 0 65535 0
P0851 ECAT: AccessoryStatus 0: Inactive 0: Inactive 0 4 0: Inactive
P0852 ECAT: Link Status 0 0 0 65535 0
P0853 ECAT: Slave Status 0: Inactive 0: Inactive 0 8 0: Inactive
P0854 EtherCAT: Reserved 0 0 0 65535 0
P0855 ECAT: TxPDO Config 0 0 0 65535 0
P0856 ECAT: TxPDO Data Size 0 0 0 32 0
P0857 ECAT: RxPDO Config 0 0 0 65535 0
P0858 ECAT: RxPDO Data Size 0 0 0 32 0
P0859 ECAT: Data Update 0.0 0.0 0.0 1000.0 0.0 ms
P0944 Fault Message Counter 0 0 0 65535 0
P0947 Fault Number 0 0 0 65535 0
P0963 Profibus Baud Rate 0: 9.6 kbit/s 5: Not Detected 0 11 0: 9.6 kbit/s
P0964 Drive Unit Ident. 0 367 0 65535 0
P0965 Profile Ident. Number 0 809 0 65535 0
P0967 Control Word 1 0 0 0 65535 0
P0968 Status Word 1 0 0 0 65535 0
P1000 PLC State 0: No Program 4: Stopped Prog. 0 5 0: No Program
P1001 Scan Time 0.0 0.0 0.0 6553.5 0.0 ms
Output
WEG SCA-06 V1.40 localhost:502 USB/@0


Projects
cfg
Resource01
Parameter
parameter01
Ladder
Diagnostic
CANopen
parameter01
Parameters
00 Read
01 General Configuration
02 Motor
03 Special Function
04 Serial
05 CAN
06 Profibus
07 PLC
08 User
09 EtherCAT
Hide/unhide parameters
Hide/Unhide parameters
00 Read
01 General Configuration
02 Motor
03 Special Function
04 Serial
05 CAN
06 Profibus
07 PLC
08 User
09 EtherCAT
OK
Cancel
Parameter
Description
User
Monitored
Minimum
Maximum
Factory settings
Unit
P0080 Virtual Axis Speed 0 0 -9999 9999 0 rpm
P0082 Virtual Axis - fractions of revolution 0 0 -16383 16383 0
P0083 Virtual Axis - numbers of revolutions 0 0 -32768 32767 0
P0084 Day of week 0 1 0 6 0
P0085 Day 1 3 1 31 1
P0086 Month 1 1 1 12 1
P0087 Year 2009 1594 0 4095 2009
P0088 Hous 0 20 0 23 0
P0089 Minutes 0 22 0 59 0
P0090 Seconds 0 52 0 59 0
P0091 Slot 1 ID 0 0 0 65535 0
P0092 Slot 2 ID 0 0 0 65535 0
P0093 Slot 3 ID 0 512 0 65535 0
P0095 Opcional Card 0 0 0 1000 0
P0097 Nominal Current 0.0 5.0 0.0 999.9 0.0
P0098 Line Rated Voltage 2: 220 V 0 10
P0228 Iq RMS Cicle 0.0 0.0 -3276.8 3276.7 0.0 A
P0740 Profibus Comm. Status 0: Disabled 0: Disabled 0 6 0: Disabled
P0850 EtherCAT: FW Revision 0 0 0 65535 0
P0851 ECAT: AccessoryStatus 0: Inactive 0: Inactive 0 4 0: Inactive
P0852 ECAT: Link Status 0 0 0 65535 0
P0853 ECAT: Slave Status 0: Inactive 0: Inactive 0 8 0: Inactive
P0854 EtherCAT: Reserved 0 0 0 65535 0
P0855 ECAT: TxPDO Config 0 0 0 65535 0
P0856 ECAT: TxPDO Data Size 0 0 0 32 0
P0857 ECAT: RxPDO Config 0 0 0 65535 0
P0858 ECAT: RxPDO Data Size 0 0 0 32 0
P0859 ECAT: Data Update 0.0 0.0 0.0 1000.0 0.0 ms
P0944 Fault Message Counter 0 0 0 65535 0
P0947 Fault Number 0 0 0 65535 0
P0963 Profibus Baud Rate 0: 9.6 kbit/s 5: Not Detected 0 11 0: 9.6 kbit/s
P0964 Drive Unit Ident. 0 367 0 65535 0
P0965 Profile Ident. Number 0 809 0 65535 0
P0967 Control Word 1 0 0 0 65535 0
P0968 Status Word 1 0 0 0 65535 0
P1000 PLC State 0: No Program 4: Stopped Prog. 0 5 0: No Program
P1001 Scan Time 0.0 0.1 0.0 6553.5 0.0 ms
Output
WEG SCA-06 V1.40 localhost:502 USB/@0

11.12.2.5 User Parameters

In order to open the configuration screen of the user parameters, just click on the **User Parameters** option on the Parameter node of the project tree or click on the icon indicated on the tool bar of the parameter file.



Para...	Description	User	M...	Minimum	Maxi...	Factory settings	Unit
P0002	Motor Speed	0		-9999	9999	0	rpm
P0003	Motor Current	0.0		-999.9	999.9	0.0	A
P0004	DC Link Voltage (Ud)	0		0	999	0	V
P0006	Drive Status	0: Ready		0	5	0: Ready	
P0008	DI3 to DI1 Status	0		0	7	0	
P0009	DI101 to DI106 Status	0		0	63	0	
P0010	DI107 to DI112 Status	0		0	63	0	
P0011	DI201 to DI206 Status	0		0	63	0	
P0012	DI207 to DI212 Status	0		0	63	0	
P0013	DI301 to DI306 Status	0		0	63	0	
P0014	DI307 to DI312 Status	0		0	63	0	
P0015	DO1 Status	0		0	1	0	
P0016	DO101 to DO106 Status	0		0	63	0	
P0017	DO201 to DO206 Status	0		0	63	0	
P0018	DO301 to DO306 Status	0		0	63	0	
P0021	Internal Air Temperature	0		0	1000	0	°C
P0022	Dissipator Temperature	0		0	1000	0	°C
P0023	Software Version	9.99		0.00	655.35	9.99	
P0024	Bootloader Version	0.00		0.00	655.35	0.00	
P0025	FPGA Project Version	0.00		0.00	655.35	0.00	
P0030	Present Alarm	0		0	2000	0	
P0031	Last Alarm	0		0	2000	0	
P0032	Last alarm Day.Month	0.00		0.00	31.12	0.00	
P0033	Last alarm year	0		0	4096	0	
P0034	Last alarm Hour.Min	0.00		0.00	23.59	0.00	
P0035	Present Fault	0		0	2000	0	
P0036	Last Fault	0		0	2000	0	
P0037	Last Fault Day.Month	0.00		0.00	31.12	0.00	
P0038	Last fault year	0		0	4096	0	
P0039	Last fault Hour.Min	0.00		0.00	23.59	0.00	
P0040	Second fault	0		0	2000	0	
P0041	Second fault Day.Month	0.00		0.00	31.12	0.00	
P0042	Second fault year	0		0	4096	0	
P0043	Second fault Hour.Min	0.00		0.00	23.59	0.00	
P0044	Third fault	0		0	2000	0	
P0045	Third fault Day.Month	0.00		0.00	31.12	0.00	
P0046	Third fault year	0		0	4096	0	

Configuration Table.

On the user parameter configuration table, it is possible to define several attributes to the parameters, such as description, minimum and maximum values, unit, digits, data type, etc.

NOTE!
These settings will be automatically displayed in the parameter table. However, to be sent to the device, you need to *download* the resource.

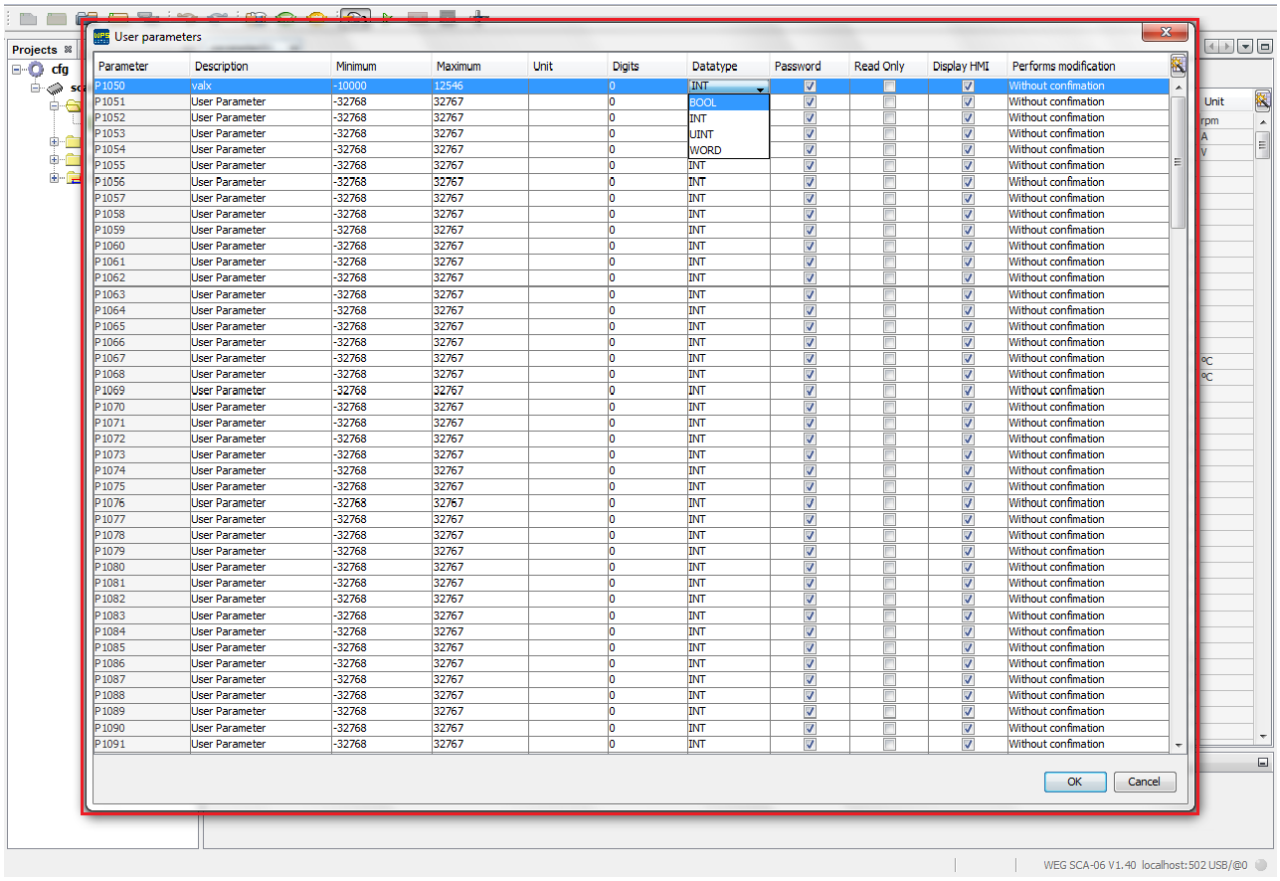


Table fields:

- Parameter: User parameter identification.
- Description: Description of the user parameter in the parameter table. On devices that have text-based HMIs, the description is sent to the machine and displayed on the HMI.
- Minimum: Minimum input value for parameter.
- Maximum: Maximum input value for parameter.
- Unit: Unit displayed on the device's HMI.
- Default: Value loaded when restore factory default is selected.
- Retentive: Retain value after rebooting devices.
- Hexadecimal: Displays the value in hexadecimal.
- Digits: Number of decimal digits for displaying value.
- Datatype: Parameter datatype used by the ladder application.
- Password: Enables password request by changing parameter value.

- Read only: It does not allow the writing of values in the parameter by the communication network or the HMI. Writing is done only by the ladder application.

- Display HMI: Displays the parameter in the HMI.

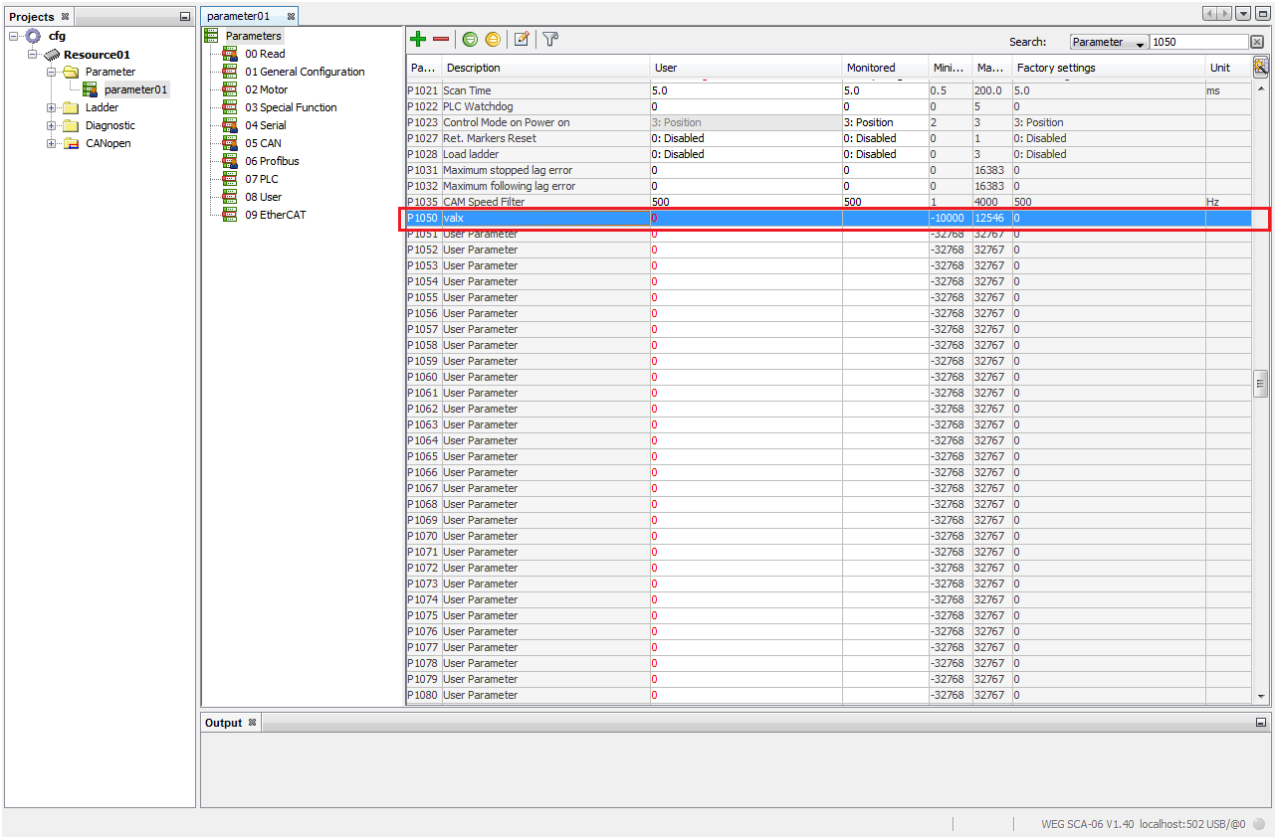
- Performs modification: Confirmation options when changing the parameter:
o No confirmation: Does not prompt for confirmation when changing parameter.
o With confirmation and engine stopped: Request confirmation and allow change only with engine stopped.
o With confirmation: Prompt for confirmation when changing parameter.

- Stopped motor: Perform change only with motor stopped.

- Help: On devices that have text-based HMI, you can edit a help text for the parameter.

View the user parameter

In the parameter table, the user parameters will be shown as they are configured on the configuration screen.



11.13 SSW-08

Enter topic text here.

11.13.1 Description

WEG soft starters are fitted with micro processors. They are high-tech and were designed to ensure the highest performance during starts and stops of induction motors. WEG Soft-Starters are static starting switches, intended for the acceleration, deceleration and protection of three-phase, induction motors. The voltage control applied to the motor, by setting the thyristor firing angle, allows smooth starts and stops.

With proper variable settings, the torque produced is set to the requirement of the load, ensuring that the demanded current of the start is the lowest actually required.

Refer to the user's manual of the SSW-08 for further details about the product.

NOTE!

This product does not have the Ladder tool available in WPS.
You can use the WLP application if this feature is required.

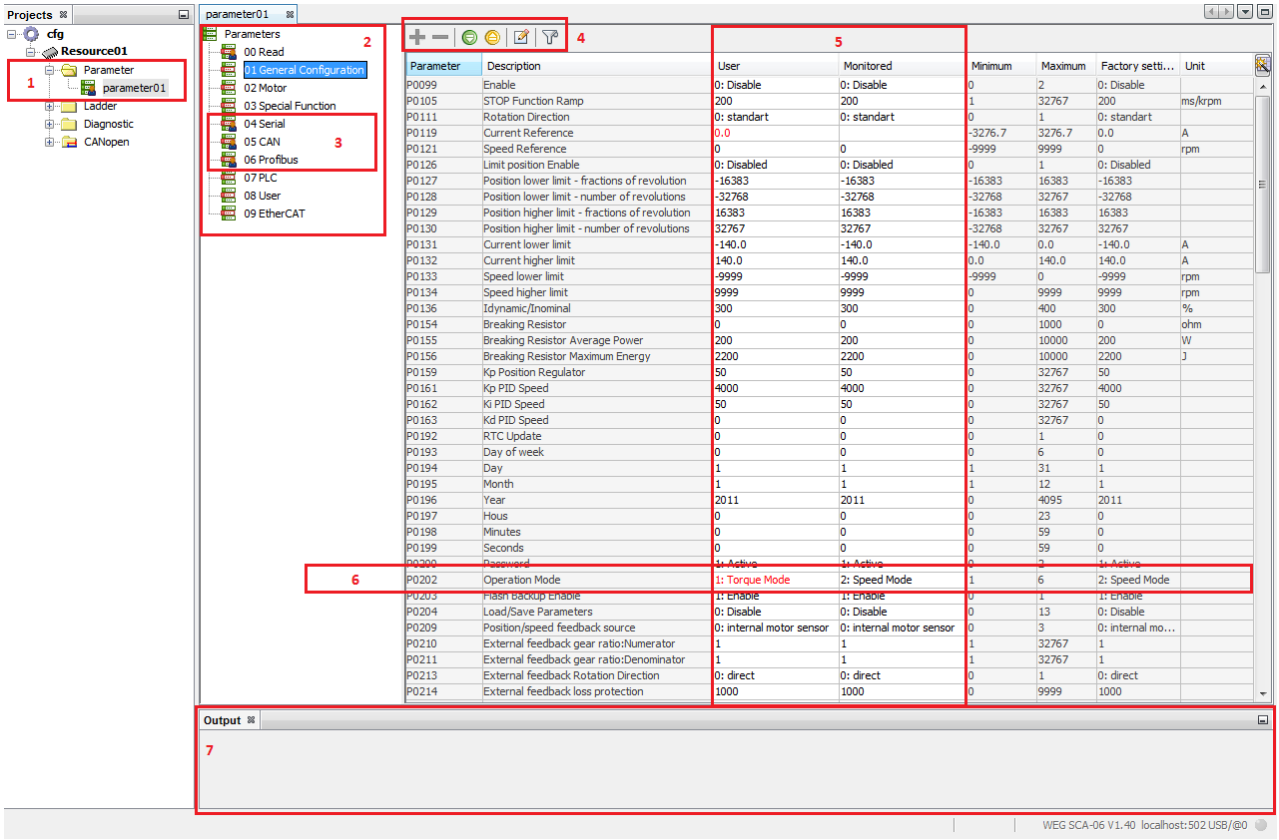
11.13.2Parameters

11.13.2.1 Overview

The parameter configuration screen is used to configure and monitor all the parameters of the equipment, including the user parameters.

NOTE!
The reading and writing of such parameters is done on this screen; only the user parameter configuration must be sent the first time or whenever modified by means of the resource download routine.

Below is an overview of the parameter configuration screen.



Parameter	Description	User	Monitored	Minimum	Maximum	Factory setti...	Unit
P0099	Enable	0: Disable	0: Disable	0	2	0: Disable	
P0105	STOP Function Ramp	200	200	1	32767	200	ms/krpm
P0111	Rotation Direction	0: standart	0: standart	0	1	0: standart	
P0119	Current Reference	0.0		-3276.7	3276.7	0.0	A
P0121	Speed Reference	0	0	-9999	9999	0	rpm
P0126	Limit position Enable	0: Disabled	0: Disabled	0	1	0: Disabled	
P0127	Position lower limit - fractions of revolution	-16383	-16383	-16383	16383	-16383	
P0128	Position lower limit - number of revolutions	-32768	-32768	-32768	32767	-32768	
P0129	Position higher limit - fractions of revolution	16383	16383	-16383	16383	16383	
P0130	Position higher limit - number of revolutions	32767	32767	-32768	32767	32767	
P0131	Current lower limit	-140.0	-140.0	-140.0	0.0	-140.0	A
P0132	Current higher limit	140.0	140.0	0.0	140.0	140.0	A
P0133	Speed lower limit	-9999	-9999	-9999	0	-9999	rpm
P0134	Speed higher limit	9999	9999	0	9999	9999	rpm
P0136	Idynamic/Inominal	300	300	0	400	300	%
P0154	Breaking Resistor	0	0	0	1000	0	ohm
P0155	Breaking Resistor Average Power	200	200	0	10000	200	W
P0156	Breaking Resistor Maximum Energy	2200	2200	0	10000	2200	J
P0159	Kp Position Regulator	50	50	0	32767	50	
P0161	Kp PID Speed	4000	4000	0	32767	4000	
P0162	Ki PID Speed	50	50	0	32767	50	
P0163	Kd PID Speed	0	0	0	32767	0	
P0192	RTC Update	0	0	0	1	0	
P0193	Day of week	0	0	0	6	0	
P0194	Day	1	1	1	31	1	
P0195	Month	1	1	1	12	1	
P0196	Year	2011	2011	0	4095	2011	
P0197	Hous	0	0	0	23	0	
P0198	Minutes	0	0	0	59	0	
P0199	Seconds	0	0	0	59	0	
P0200	Password	1: Active	1: Active	0	2	1: Active	
P0202	Operation Mode	1: Torque Mode	2: Speed Mode	1	6	2: Speed Mode	
P0203	Flash Backup Enable	1: Enable	1: Enable	0	1	1: Enable	
P0204	Load/Save Parameters	0: Disable	0: Disable	0	13	0: Disable	
P0209	Position/speed feedback source	0: internal motor sensor	0: internal motor sensor	0	3	0: internal mo...	
P0210	External feedback gear ratio:Numerator	1	1	1	32767	1	
P0211	External feedback gear ratio:Denominator	1	1	1	32767	1	
P0213	External feedback Rotation Direction	0: direct	0: direct	0	1	0: direct	
P0214	External feedback loss protection	1000	1000	0	9999	1000	

1. **Parameter files**. In this part are all the parameter configuration files created by the user. Notice that when the file features a person figure on the table, it means this parameter table contains hidden parameters/ group of parameters.
2. **Group of parameters**. This tree shows all the group of parameters. Notice that the same parameter can be in more than one group, and when its value is modified, it will be modified in all the groups to which it belongs.
3. **Modified group of parameters**. Group of parameters which contain the figure of a person on the table means they have hidden parameters.
4. **Commands**. The commands are described below in the order they appear:
   - 4.1. **Unhide parameter**: In case some parameter has been hidden, this button allows making it visible again.
   - 4.2. **Hide parameter**: Just select one or more parameters on the table and trigger this command to hide them.
   - 4.3. **Save table**: It saves the values of the parameters shown on the equipment screen; the sent values are the ones in the User column. The flow is User -> Monitored (equipment)
   - 4.4. **Read table**: It reads the parameters of the equipment shown in the Monitored column and saves them in the parameter file in the User column. The flow is Monitored (equipment) -> User
   - 4.5. **User parameters**: It opens a screen to edit the user parameters.
   - 4.6. **Filter**: It opens a parameter filter option, and it can filter by parameter number or description.
   - 4.7. **User Parameters and Monitored Parameters**. These two columns show the off-line and on-line parameters, so to speak. The **User** column shows the values contained in the file located on the computer and the **Monitored** column shows the values that are effectively saved on the equipment. Whenever you use the **Save Parameter** option, the sent values will be from the **User** column to the **Monitored** column, that is, File -> Equipment. In case of reading, the flow is the opposite, from the **Monitored** column to the **User** column, that is, Equipment -> File. In case you wish to change the values directly on the equipment without changing it in the file, just click on the monitored column,

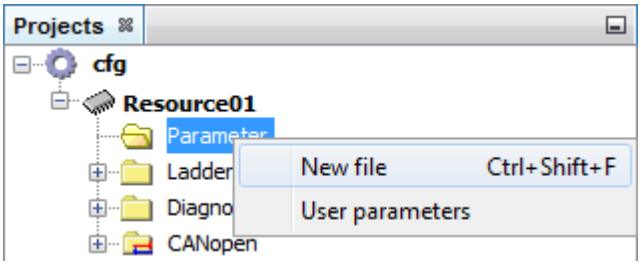
change the values and the modification will occur on-line.

5. **Modified parameters**: Whenever a parameter value in the **User** column is different from the **Monitored** column, it will be shown in red.
6. **Output**. This screen shows error information in case they occur during the writing or reading of the parameters.

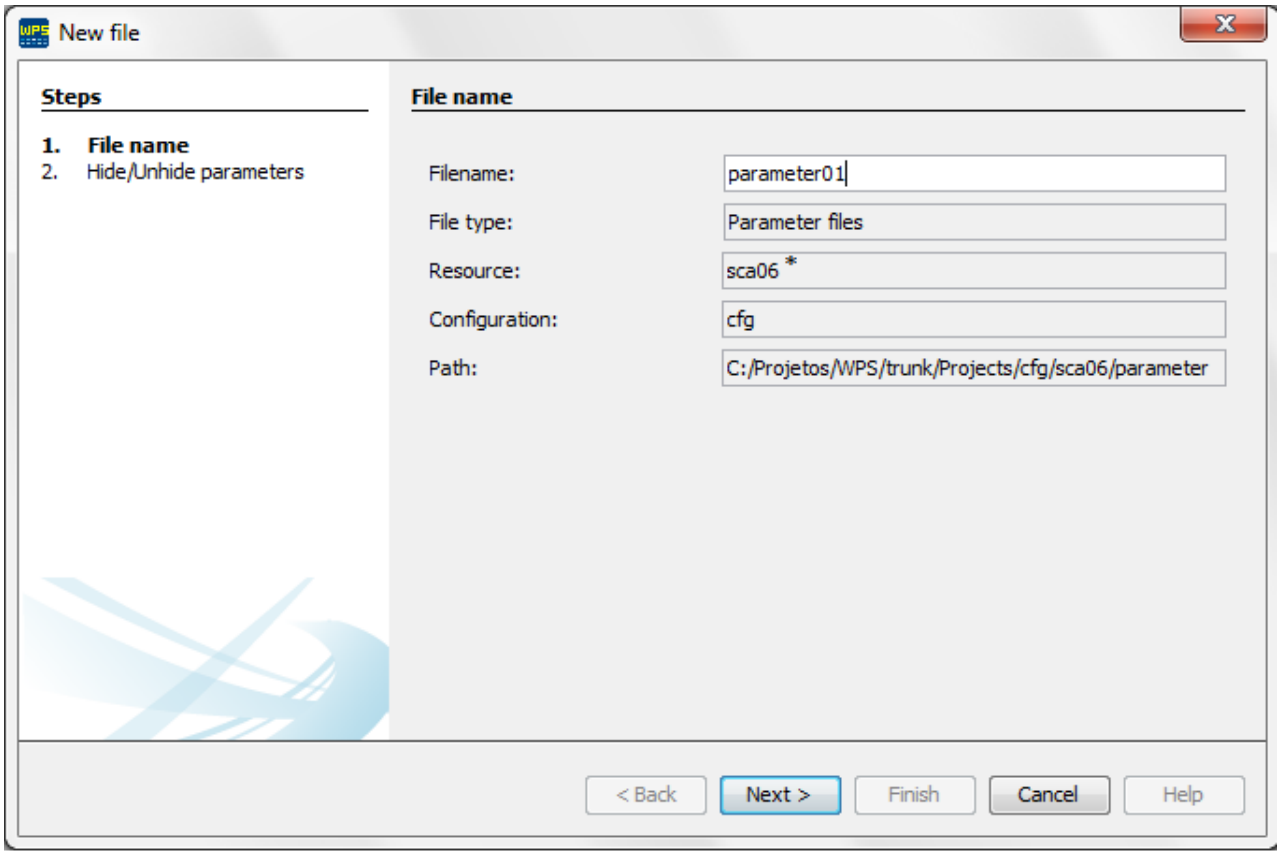
11.13.2.2 Configuration

Below is the list of the required steps to create a parameter file.

1. Create a new parameter file.

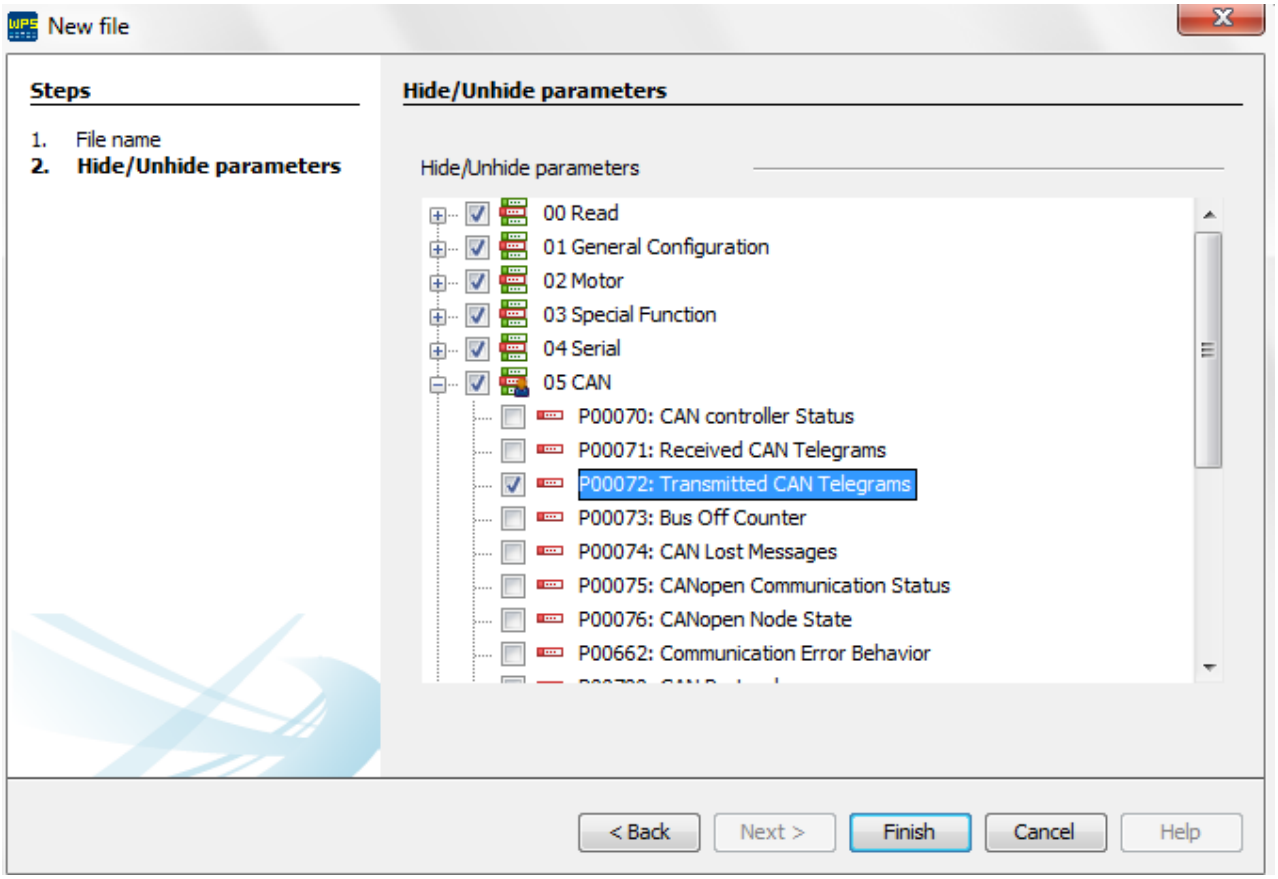


2. Define a name for the parameter file

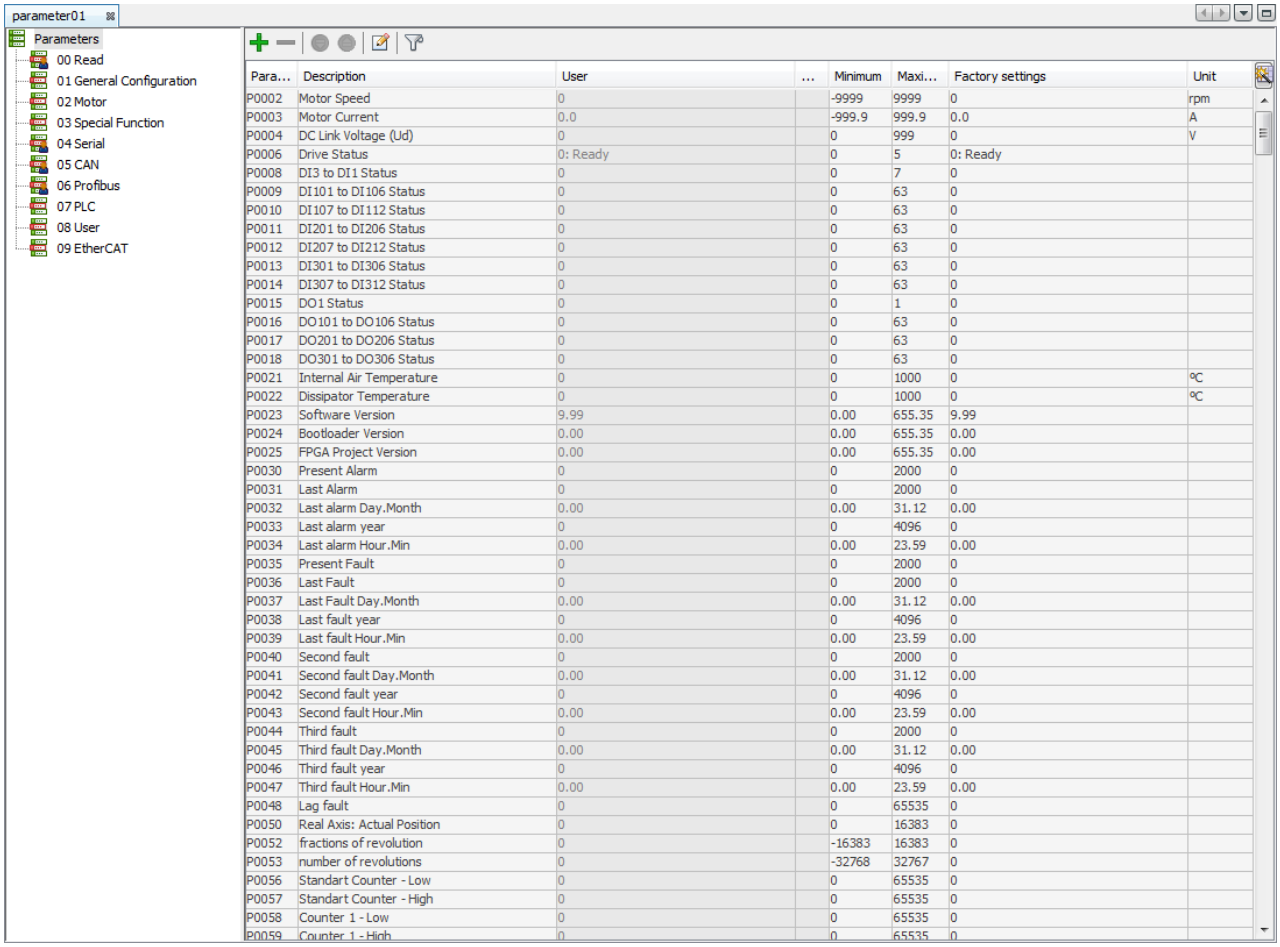


*** Resource: Resource01, SCA06, CFW300, etc.**

3. Configure which parameters you wish to view in your parameter table



4. After performing the steps above, the parameter file will be created and the equipment can be parameterized.

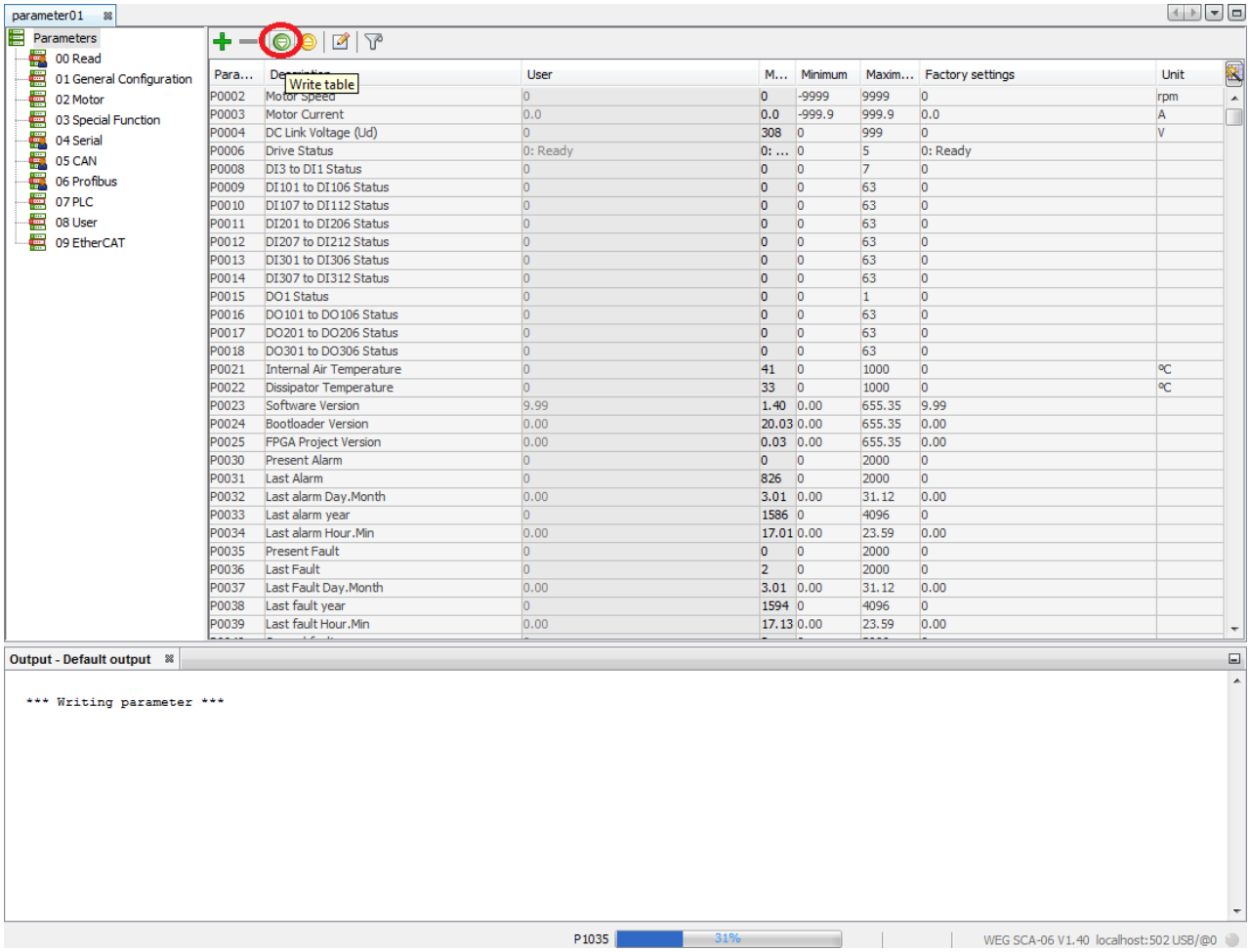


Para...	Description	User	...	Minimum	Maxi...	Factory settings	Unit
P0002	Motor Speed	0		-9999	9999	0	rpm
P0003	Motor Current	0.0		-999.9	999.9	0.0	A
P0004	DC Link Voltage (Ud)	0		0	999	0	V
P0006	Drive Status	0: Ready		0	5	0: Ready	
P0008	DI3 to DI1 Status	0		0	7	0	
P0009	DI101 to DI106 Status	0		0	63	0	
P0010	DI107 to DI112 Status	0		0	63	0	
P0011	DI201 to DI206 Status	0		0	63	0	
P0012	DI207 to DI212 Status	0		0	63	0	
P0013	DI301 to DI306 Status	0		0	63	0	
P0014	DI307 to DI312 Status	0		0	63	0	
P0015	DO1 Status	0		0	1	0	
P0016	DO101 to DO106 Status	0		0	63	0	
P0017	DO201 to DO206 Status	0		0	63	0	
P0018	DO301 to DO306 Status	0		0	63	0	
P0021	Internal Air Temperature	0		0	1000	0	ºC
P0022	Dissipator Temperature	0		0	1000	0	ºC
P0023	Software Version	9.99		0.00	655.35	9.99	
P0024	Bootloader Version	0.00		0.00	655.35	0.00	
P0025	FPGA Project Version	0.00		0.00	655.35	0.00	
P0030	Present Alarm	0		0	2000	0	
P0031	Last Alarm	0		0	2000	0	
P0032	Last alarm Day.Month	0.00		0.00	31.12	0.00	
P0033	Last alarm year	0		0	4096	0	
P0034	Last alarm Hour.Min	0.00		0.00	23.59	0.00	
P0035	Present Fault	0		0	2000	0	
P0036	Last Fault	0		0	2000	0	
P0037	Last Fault Day.Month	0.00		0.00	31.12	0.00	
P0038	Last fault year	0		0	4096	0	
P0039	Last fault Hour.Min	0.00		0.00	23.59	0.00	
P0040	Second fault	0		0	2000	0	
P0041	Second fault Day.Month	0.00		0.00	31.12	0.00	
P0042	Second fault year	0		0	4096	0	
P0043	Second fault Hour.Min	0.00		0.00	23.59	0.00	
P0044	Third fault	0		0	2000	0	
P0045	Third fault Day.Month	0.00		0.00	31.12	0.00	
P0046	Third fault year	0		0	4096	0	
P0047	Third fault Hour.Min	0.00		0.00	23.59	0.00	
P0048	Lag fault	0		0	65535	0	
P0050	Real Axis: Actual Position	0		0	16383	0	
P0052	fractions of revolution	0		-16383	16383	0	
P0053	number of revolutions	0		-32768	32767	0	
P0056	Standart Counter - Low	0		0	65535	0	
P0057	Standart Counter - High	0		0	65535	0	
P0058	Counter 1 - Low	0		0	65535	0	
P0059	Counter 1 - High	0		0	65535	0	

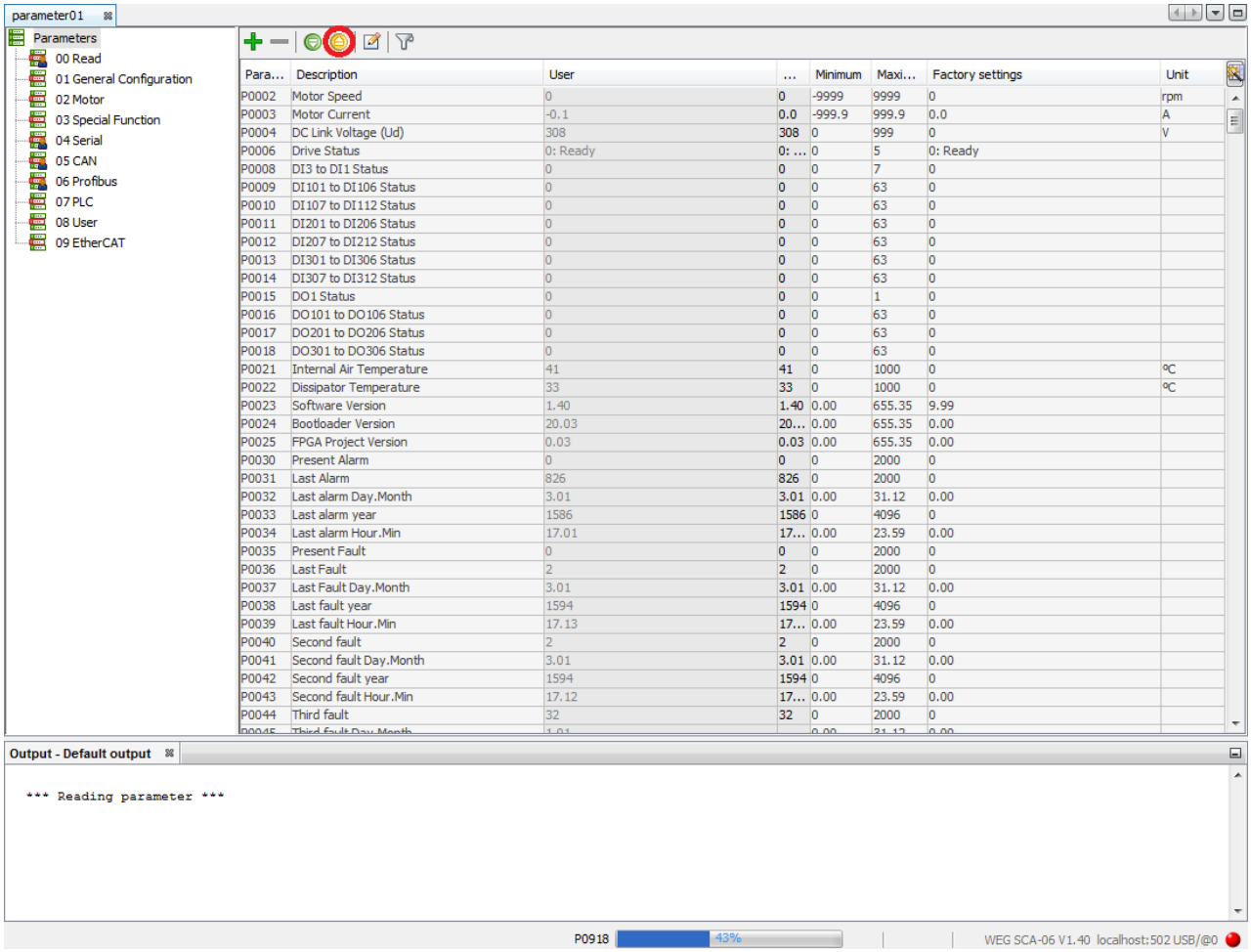
11.13.2.3 Read and Write of Parameters

There are 3 (three) ways to do the reading and writing of the parameters: by means of table, selection and group.

1. **Table writing**. The table writing command will send all visible parameters on the equipment screen. If and error occurs during the sending of some specific parameter, a message will be shown on the output window informing the error. It is important to notice that only visible parameters will be sent; therefore, it is necessary attention to which node of the group of parameters tree you are viewing. Example: If you wish to write all of them without filtering per group, just select the tree root.

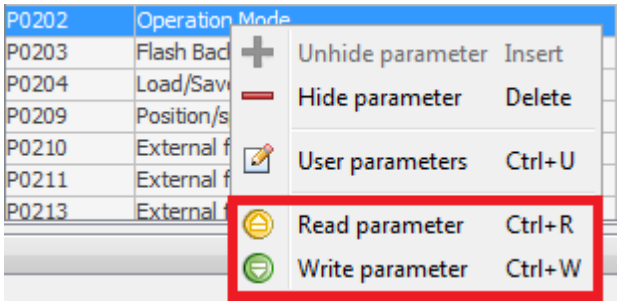


2. **Table reading**. The table reading command will read all the parameters of the equipment. If a error occurs during the reading of some specific parameter, a message will be shown on the output window informing the error. It is important to notice that only visible parameters will be read; therefore, it is necessary attention to which node of the group of parameters tree you are viewing. Example: If you wish to read all of them without filtering per group, just select the tree root.

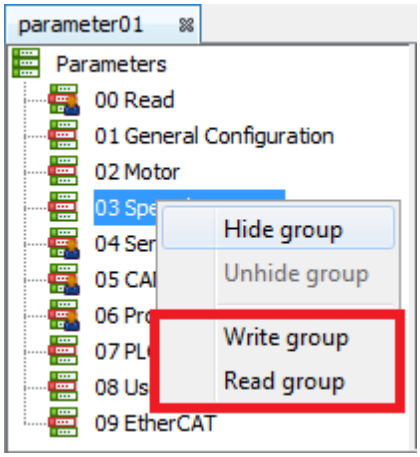


Para...	Description	User	...	Minimum	Maxi...	Factory settings	Unit
P0002	Motor Speed	0	0	-9999	9999	0	rpm
P0003	Motor Current	-0.1	0.0	-999.9	999.9	0.0	A
P0004	DC Link Voltage (Ud)	308	308	0	999	0	V
P0006	Drive Status	0: Ready	0: ...	0	5	0: Ready	
P0008	DI3 to DI1 Status	0	0	0	7	0	
P0009	DI101 to DI106 Status	0	0	0	63	0	
P0010	DI107 to DI112 Status	0	0	0	63	0	
P0011	DI201 to DI206 Status	0	0	0	63	0	
P0012	DI207 to DI212 Status	0	0	0	63	0	
P0013	DI301 to DI306 Status	0	0	0	63	0	
P0014	DI307 to DI312 Status	0	0	0	63	0	
P0015	DO1 Status	0	0	0	1	0	
P0016	DO101 to DO106 Status	0	0	0	63	0	
P0017	DO201 to DO206 Status	0	0	0	63	0	
P0018	DO301 to DO306 Status	0	0	0	63	0	
P0021	Internal Air Temperature	41	41	0	1000	0	°C
P0022	Dissipator Temperature	33	33	0	1000	0	°C
P0023	Software Version	1.40	1.40	0.00	655.35	9.99	
P0024	Bootloader Version	20.03	20...	0.00	655.35	0.00	
P0025	FPGA Project Version	0.03	0.03	0.00	655.35	0.00	
P0030	Present Alarm	0	0	0	2000	0	
P0031	Last Alarm	826	826	0	2000	0	
P0032	Last alarm Day.Month	3.01	3.01	0.00	31.12	0.00	
P0033	Last alarm year	1586	1586	0	4096	0	
P0034	Last alarm Hour.Min	17.01	17...	0.00	23.59	0.00	
P0035	Present Fault	0	0	0	2000	0	
P0036	Last Fault	2	2	0	2000	0	
P0037	Last Fault Day.Month	3.01	3.01	0.00	31.12	0.00	
P0038	Last fault year	1594	1594	0	4096	0	
P0039	Last fault Hour.Min	17.13	17...	0.00	23.59	0.00	
P0040	Second fault	2	2	0	2000	0	
P0041	Second fault Day.Month	3.01	3.01	0.00	31.12	0.00	
P0042	Second fault year	1594	1594	0	4096	0	
P0043	Second fault Hour.Min	17.12	17...	0.00	23.59	0.00	
P0044	Third fault	32	32	0	2000	0	

3. **Reading/writing of specific parameters**. In order to read/write one or more specific parameters, just select them on the table, right click and choose the desired option: read or write parameter.



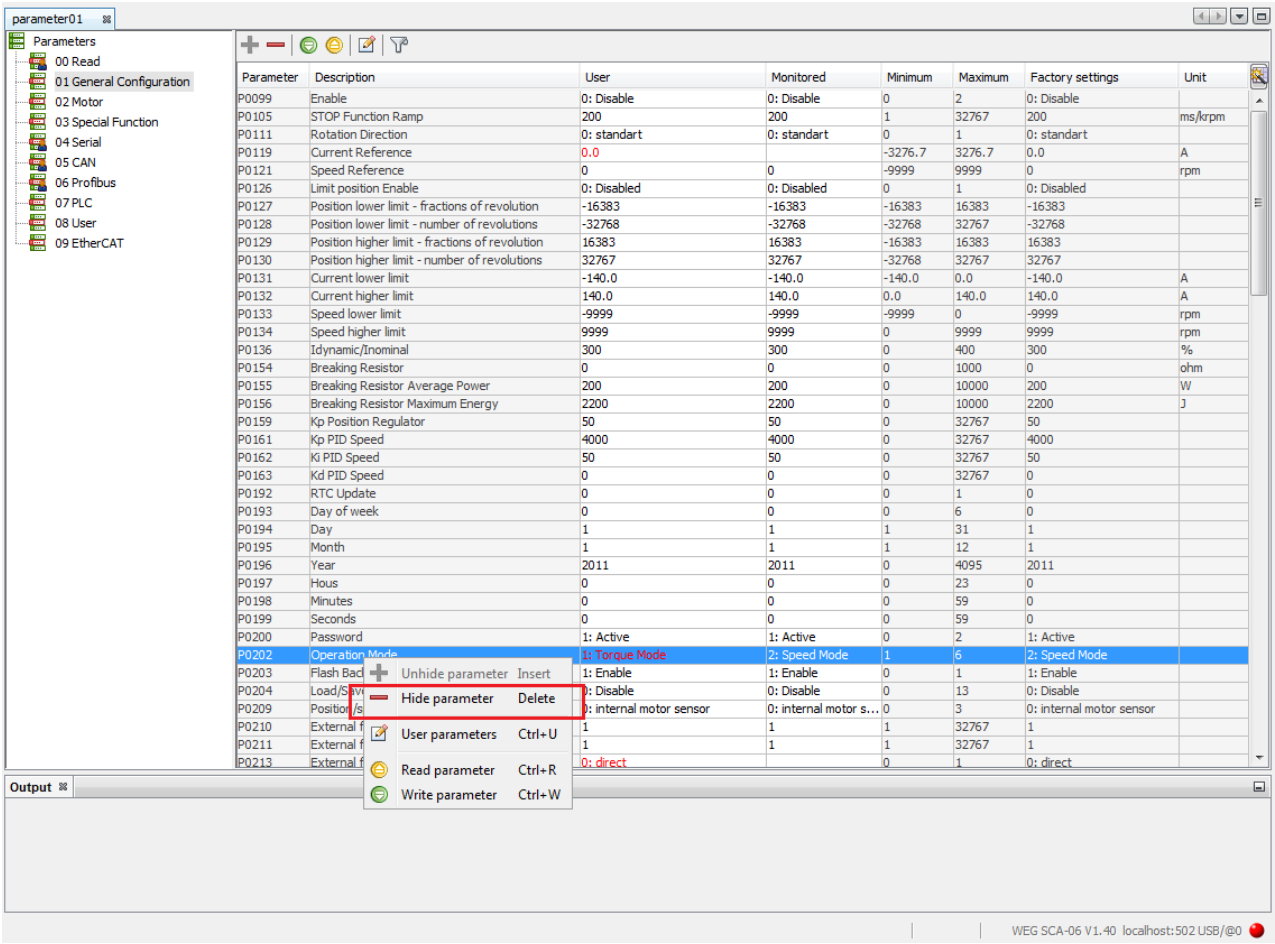
4. **Reading/writing of group of parameters**. In order to read/write only one group of parameters, just select it on the group tree, right click and choose the desired option: read or write group.



11.13.2.4 Hide/Unhide Parameters and Group of Parameters

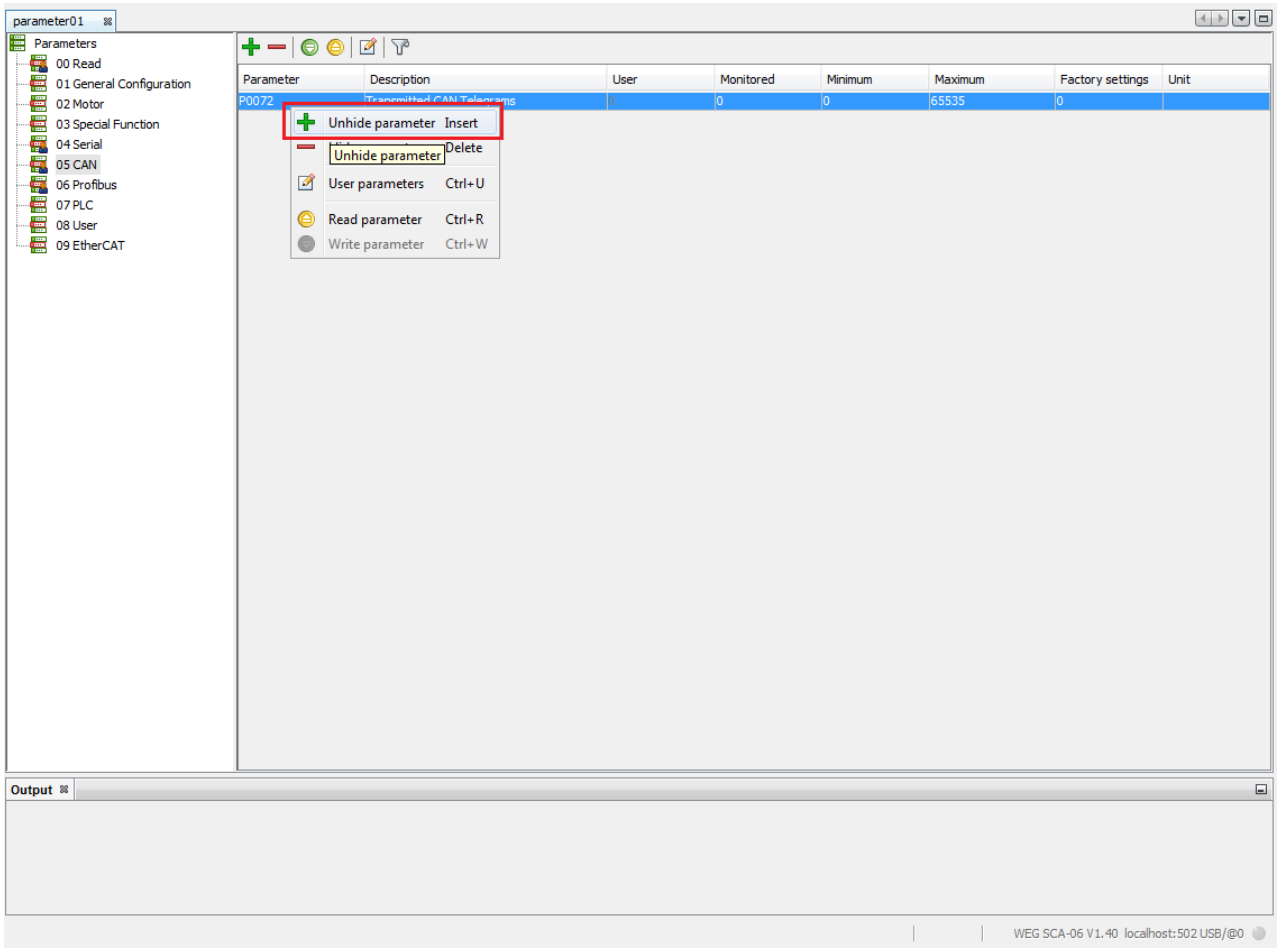
The parameter can be hidden/unhidden in two ways: individually or in group.

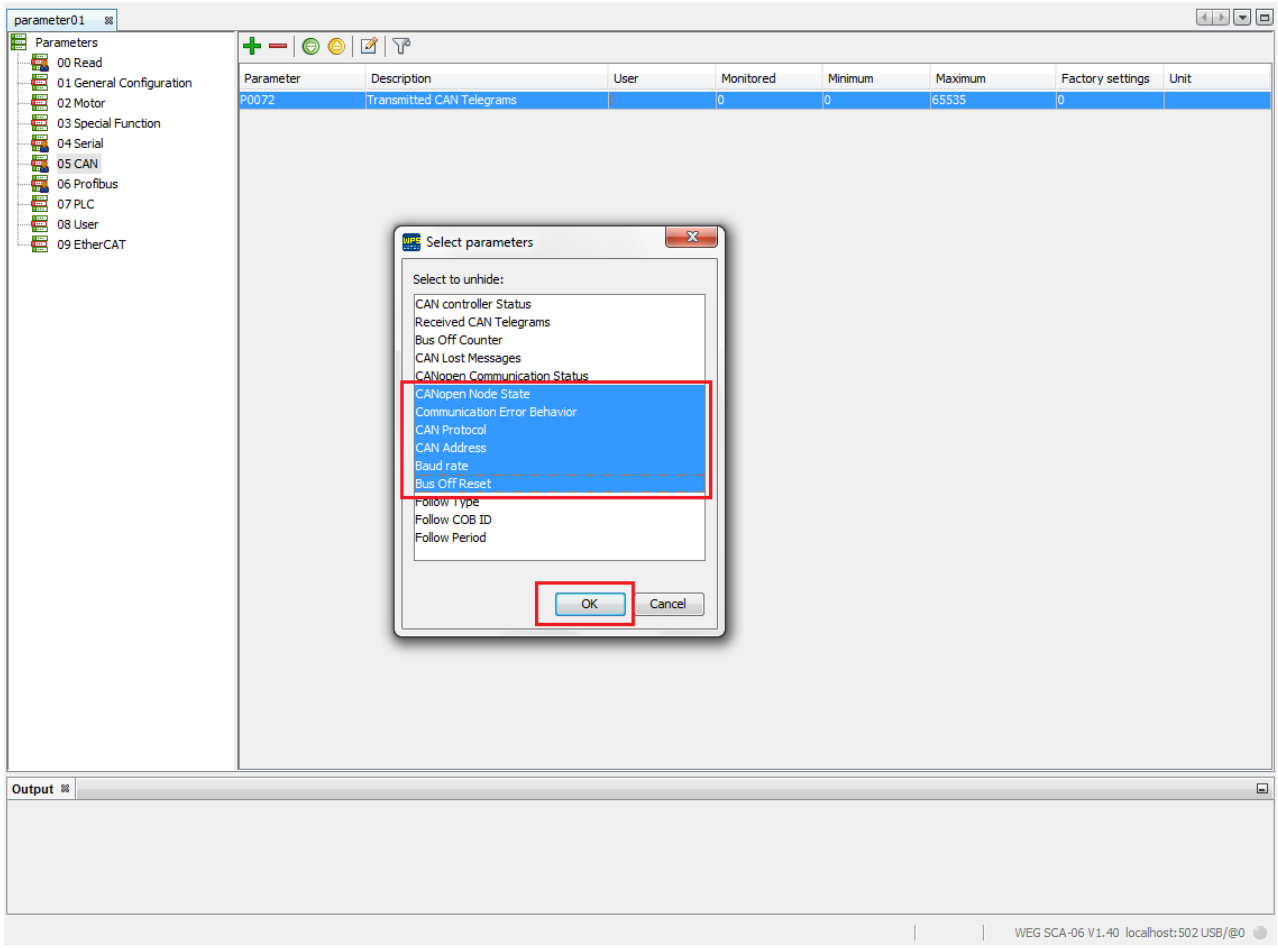
1. **Hide parameters**. In order to hide a parameter individually, just right click on the desired parameters and select the **Hide Parameter** option. You can also press the **Delete** key.

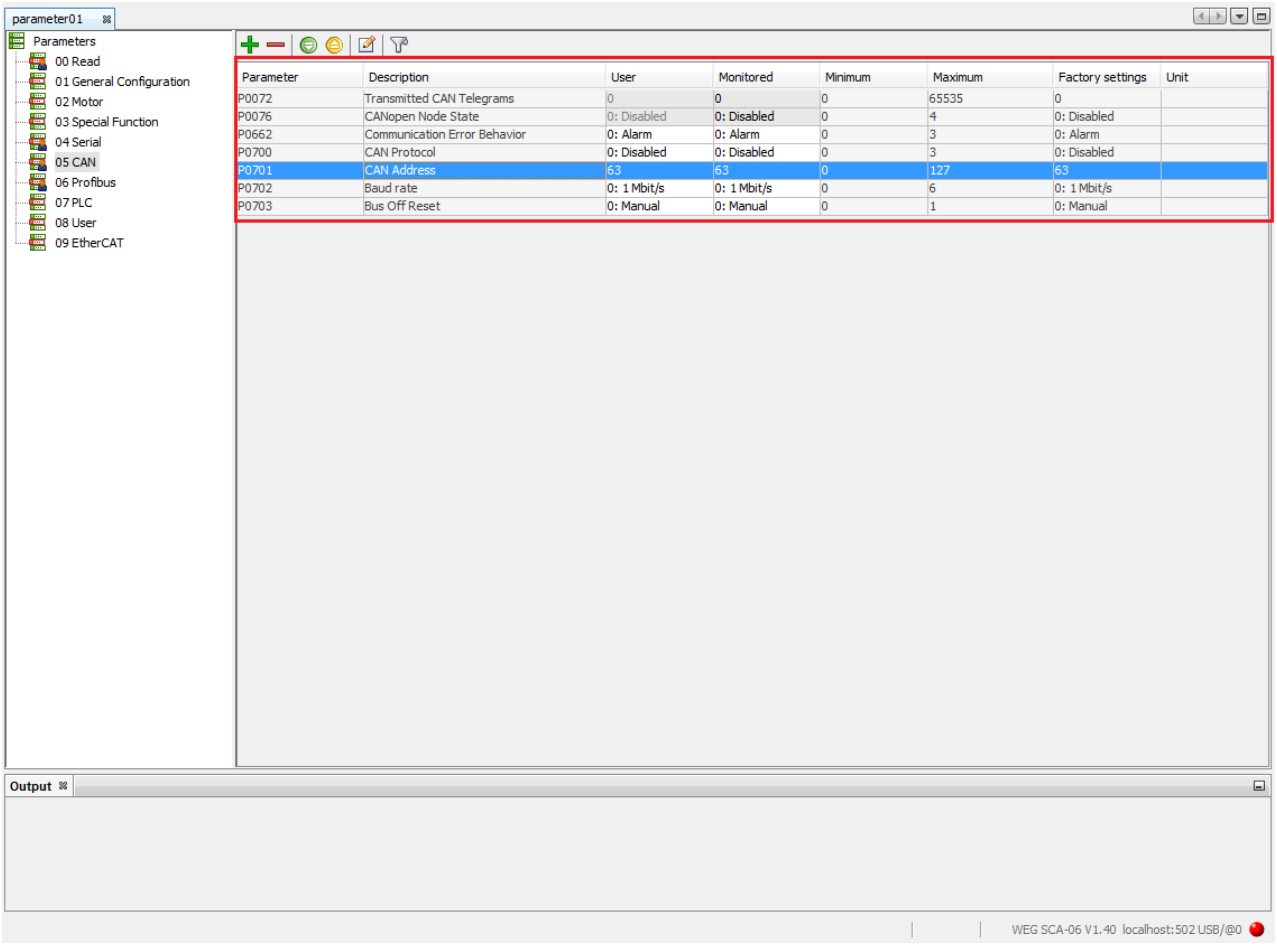


2. **Unhide Parameters**. In order to show hidden parameters, right click and choose the **Unhide Parameters**

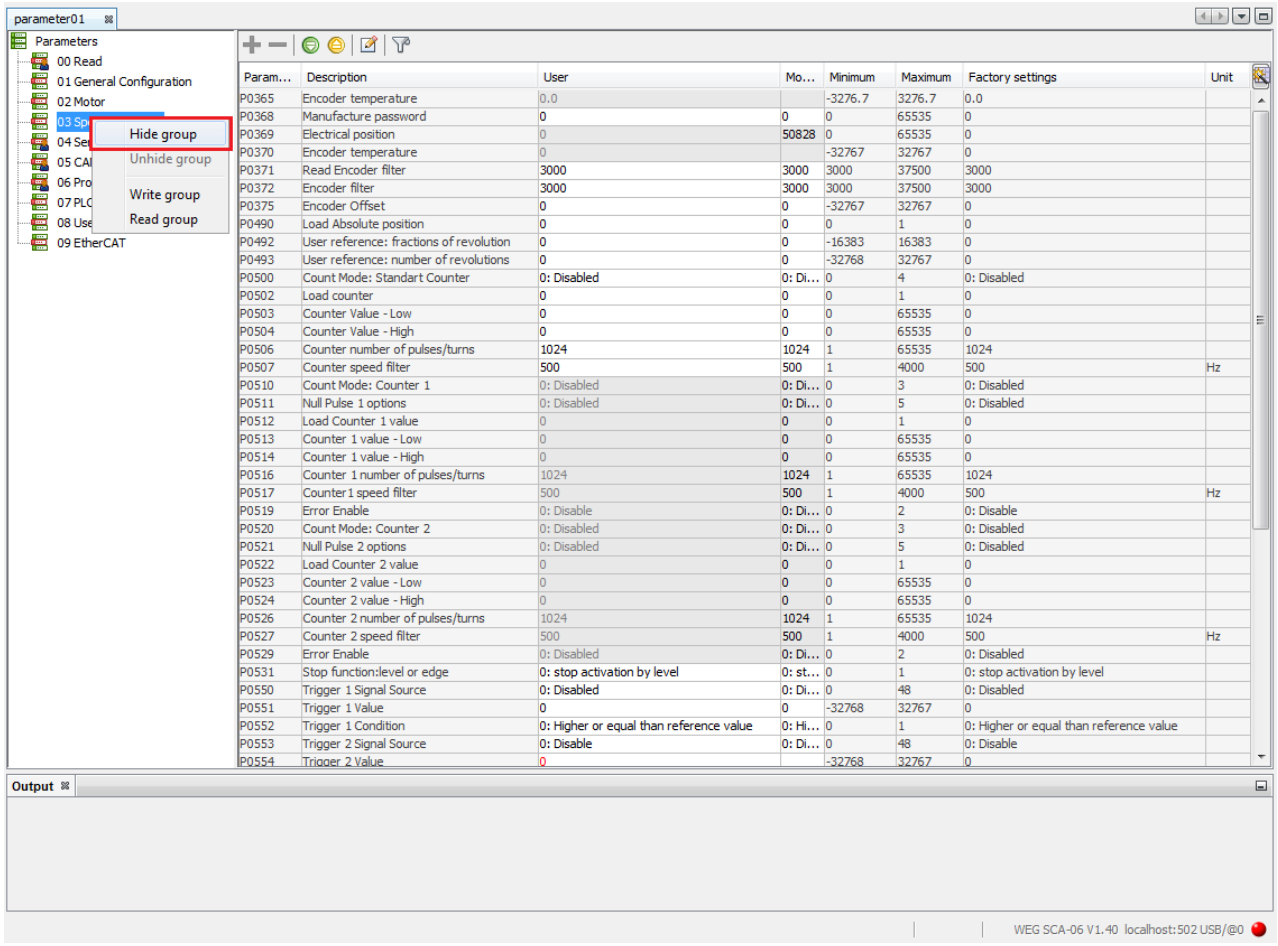
or press the **Insert** key. Then, a window will open and show the hidden parameters. Now, you just have to select the desired parameters and confirm.
Note: The parameters shown on this new window are only those which belong to the current filter according to the selection on the parameter group tree. In the figures below, the CAN group is selected; that means that only the hidden parameters of this group will be shown.

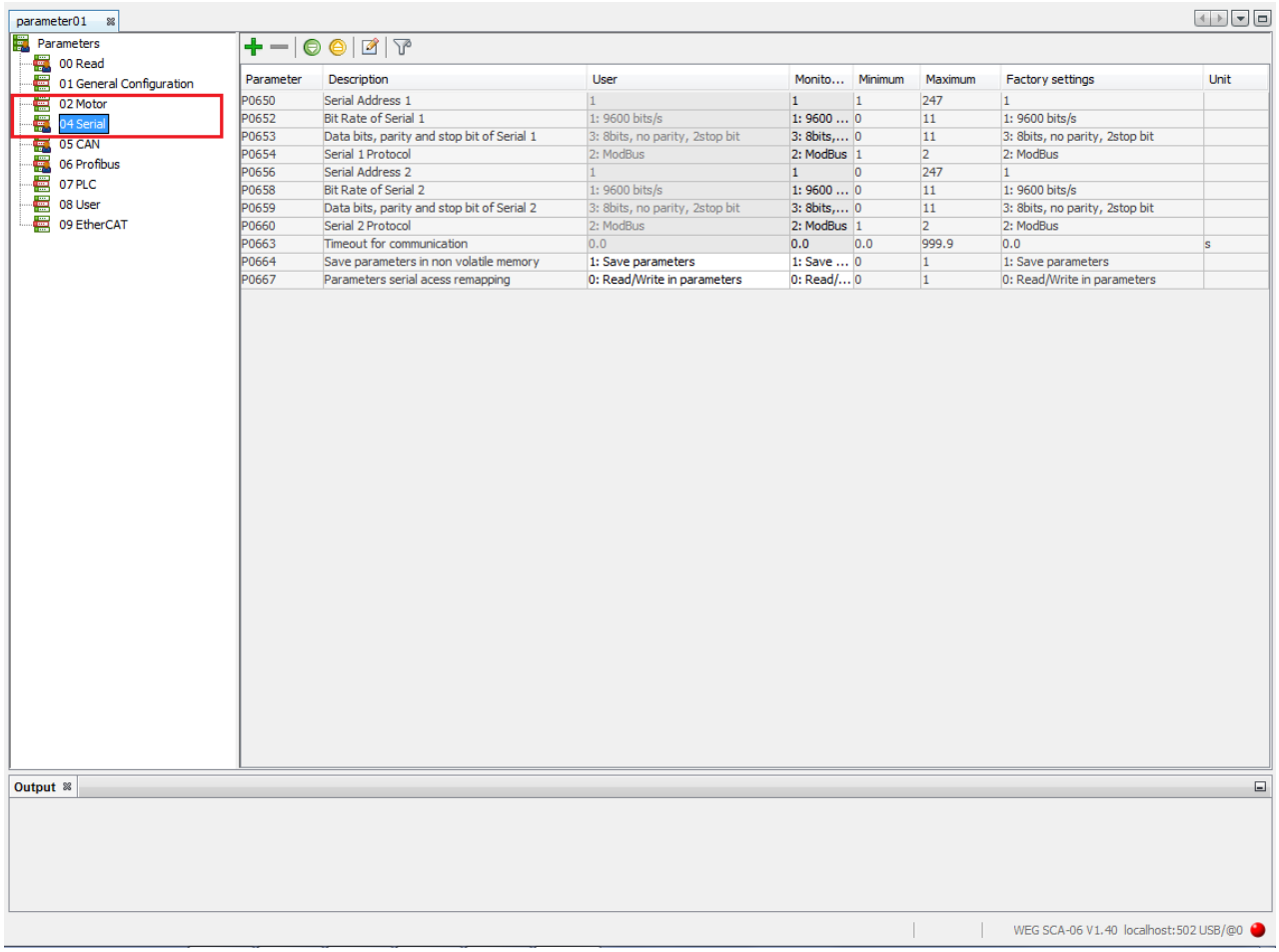



parameter01
Parameters
00 Read
01 General Configuration
02 Motor
03 Special Function
04 Serial
05 CAN
06 Profibus
07 PLC
08 User
09 EtherCAT
Parameter
Description
User
Monitored
Minimum
Maximum
Factory settings
Unit
P0072
Transmitted CAN Telegrams
0
0
65535
Select parameters
Select to unhide:
CAN controller Status
Received CAN Telegrams
Bus Off Counter
CAN Lost Messages
CANopen Communication Status
CANopen Node State
Communication Error Behavior
CAN Protocol
CAN Address
Baud rate
Bus Off Reset
Follow Type
Follow COB ID
Follow Period
OK
Cancel
Output
WEG SCA-06 V1.40 localhost:502 USB/@0

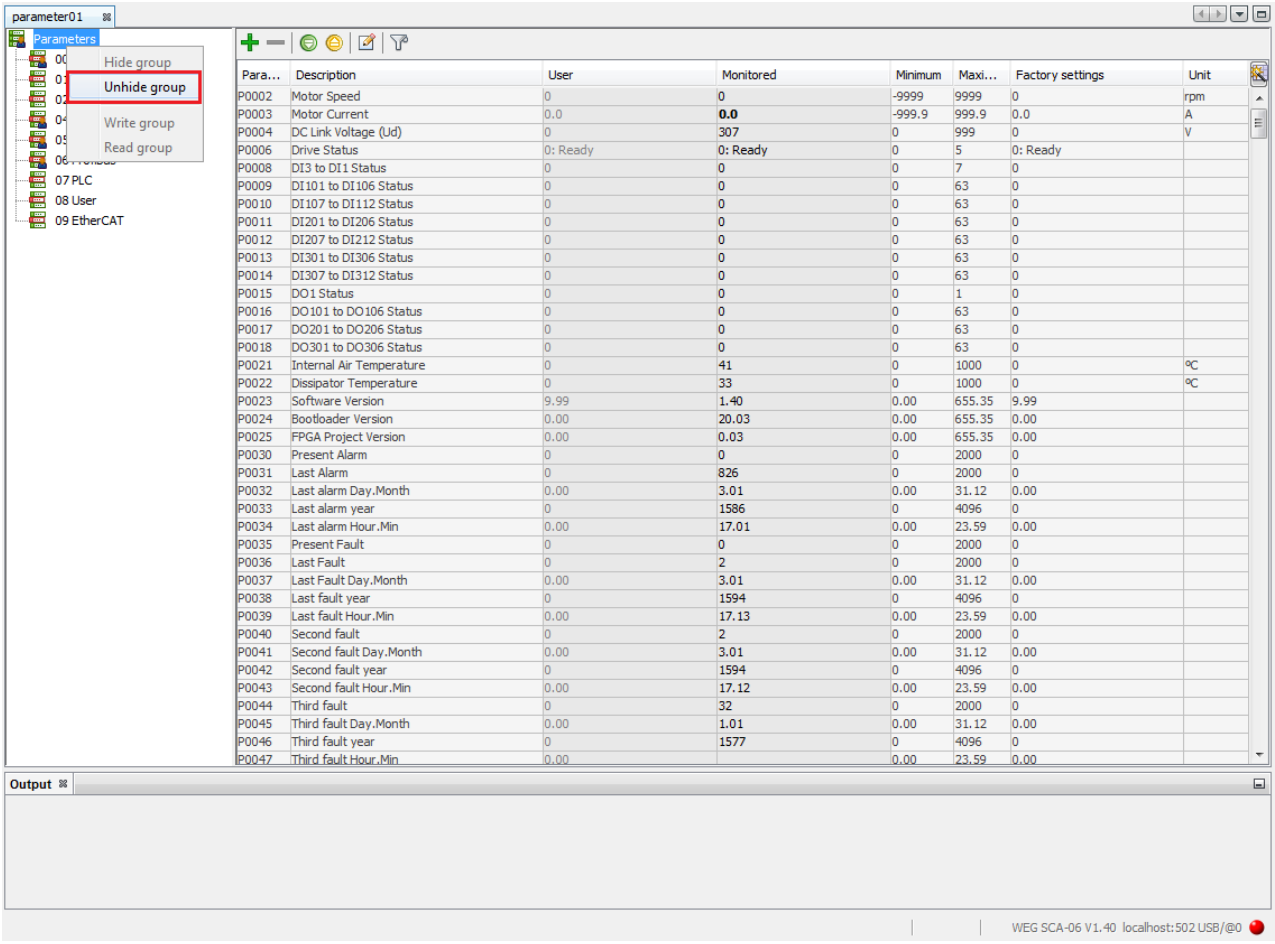


3. **Hide Group of Parameters**. In order to hide a group of parameters, just select the group on the tree and use the **Hide Group** option.

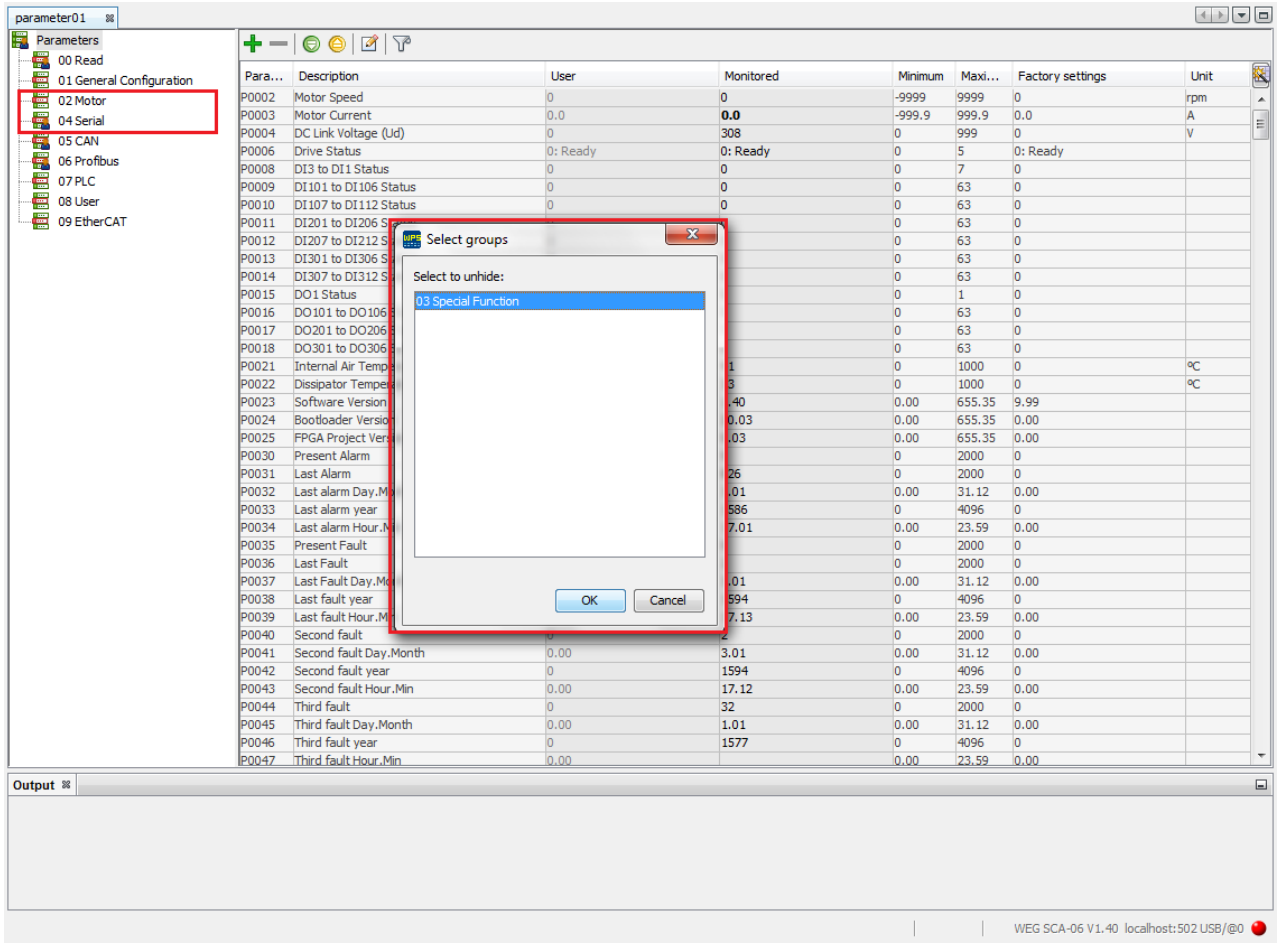

parameter01
Parameters
00 Read
01 General Configuration
02 Motor
03 Sp
04 Se
05 CA
06 Pro
07 PLC
08 Use
09 EtherCAT
Hide group
Unhide group
Write group
Read group
Param...
Description
User
Mo...
Minimum
Maximum
Factory settings
Unit
P0365 Encoder temperature 0.0 -3276.7 3276.7 0.0
P0368 Manufacture password 0 0 0 65535 0
P0369 Electrical position 0 50828 0 65535 0
P0370 Encoder temperature 0 -32767 32767 0
P0371 Read Encoder filter 3000 3000 3000 37500 3000
P0372 Encoder filter 3000 3000 3000 37500 3000
P0375 Encoder Offset 0 0 -32767 32767 0
P0490 Load Absolute position 0 0 0 1 0
P0492 User reference: fractions of revolution 0 0 -16383 16383 0
P0493 User reference: number of revolutions 0 0 -32768 32767 0
P0500 Count Mode: Standart Counter 0: Disabled 0: Di... 0 4 0: Disabled
P0502 Load counter 0 0 0 1 0
P0503 Counter Value - Low 0 0 0 65535 0
P0504 Counter Value - High 0 0 0 65535 0
P0506 Counter number of pulses/turns 1024 1024 1 65535 1024
P0507 Counter speed filter 500 500 1 4000 500 Hz
P0510 Count Mode: Counter 1 0: Disabled 0: Di... 0 3 0: Disabled
P0511 Null Pulse 1 options 0: Disabled 0: Di... 0 5 0: Disabled
P0512 Load Counter 1 value 0 0 0 1 0
P0513 Counter 1 value - Low 0 0 0 65535 0
P0514 Counter 1 value - High 0 0 0 65535 0
P0516 Counter 1 number of pulses/turns 1024 1024 1 65535 1024
P0517 Counter1 speed filter 500 500 1 4000 500 Hz
P0519 Error Enable 0: Disable 0: Di... 0 2 0: Disable
P0520 Count Mode: Counter 2 0: Disabled 0: Di... 0 3 0: Disabled
P0521 Null Pulse 2 options 0: Disabled 0: Di... 0 5 0: Disabled
P0522 Load Counter 2 value 0 0 0 1 0
P0523 Counter 2 value - Low 0 0 0 65535 0
P0524 Counter 2 value - High 0 0 0 65535 0
P0526 Counter 2 number of pulses/turns 1024 1024 1 65535 1024
P0527 Counter 2 speed filter 500 500 1 4000 500 Hz
P0529 Error Enable 0: Disabled 0: Di... 0 2 0: Disabled
P0531 Stop function:level or edge 0: stop activation by level 0: st... 0 1 0: stop activation by level
P0550 Trigger 1 Signal Source 0: Disabled 0: Di... 0 48 0: Disabled
P0551 Trigger 1 Value 0 0 -32768 32767 0
P0552 Trigger 1 Condition 0: Higher or equal than reference value 0: Hi... 0 1 0: Higher or equal than reference value
P0553 Trigger 2 Signal Source 0: Disable 0: Di... 0 48 0: Disable
P0554 Trigger 2 Value 0 -32768 32767 0
Output
WEG SCA-06 V1.40 localhost:502 USB/@0



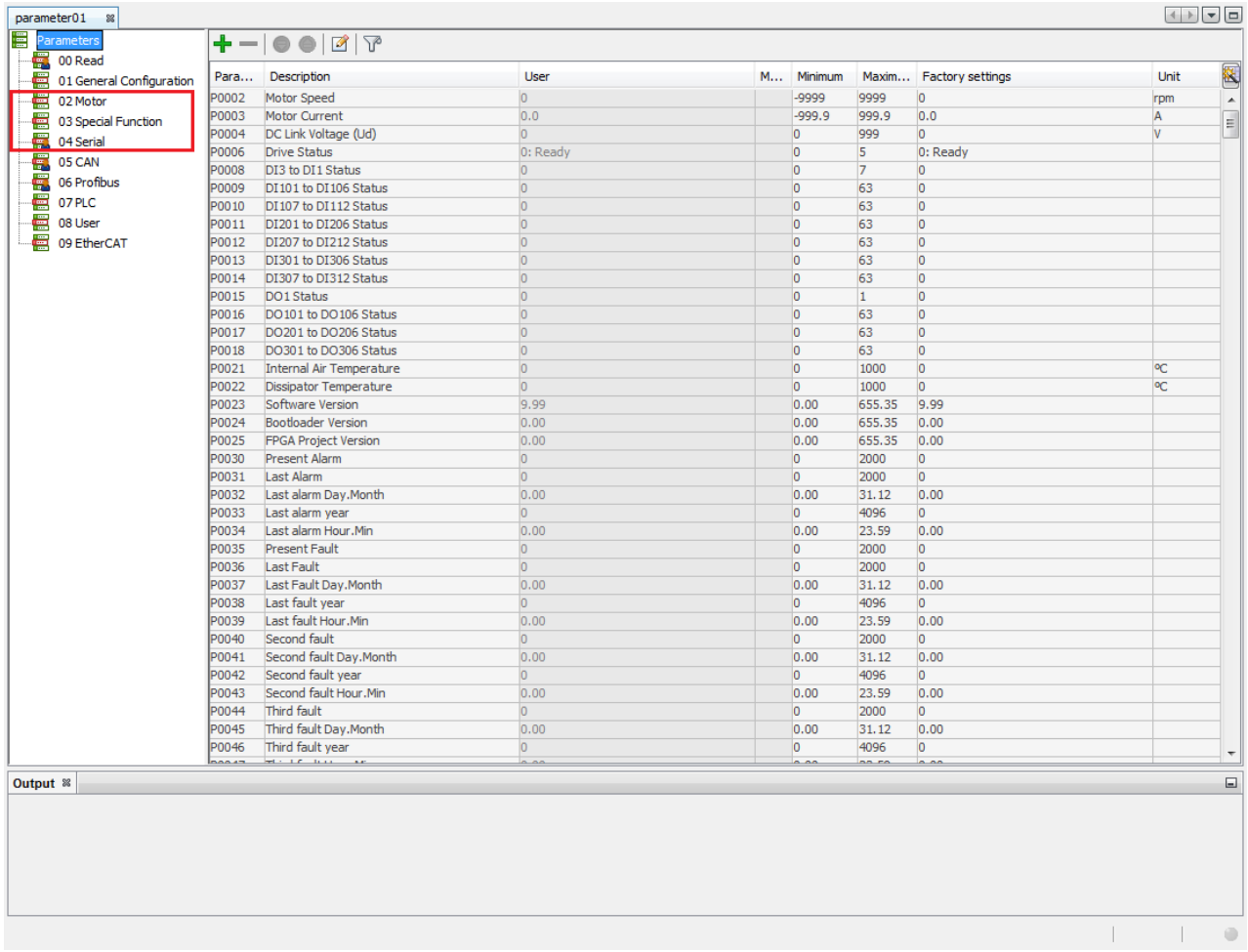
4. **Unhide Group of Parameters**. In order to show a hidden group of parameters, just select the root of the group tree and select the **Unhide Group** option. A window will open showing the groups that are hidden; then just select the group you wish to unhide.



Para...	Description	User	Monitored	Minimum	Maxi...	Factory settings	Unit
P0002	Motor Speed	0	0	-9999	9999	0	rpm
P0003	Motor Current	0.0	**0.0**	-999.9	999.9	0.0	A
P0004	DC Link Voltage (Ud)	0	307	0	999	0	V
P0006	Drive Status	0: Ready	0: Ready	0	5	0: Ready	
P0008	DI3 to DI1 Status	0	0	0	7	0	
P0009	DI101 to DI106 Status	0	0	0	63	0	
P0010	DI107 to DI112 Status	0	0	0	63	0	
P0011	DI201 to DI206 Status	0	0	0	63	0	
P0012	DI207 to DI212 Status	0	0	0	63	0	
P0013	DI301 to DI306 Status	0	0	0	63	0	
P0014	DI307 to DI312 Status	0	0	0	63	0	
P0015	DO1 Status	0	0	0	1	0	
P0016	DO101 to DO106 Status	0	0	0	63	0	
P0017	DO201 to DO206 Status	0	0	0	63	0	
P0018	DO301 to DO306 Status	0	0	0	63	0	
P0021	Internal Air Temperature	0	41	0	1000	0	°C
P0022	Dissipator Temperature	0	33	0	1000	0	°C
P0023	Software Version	9.99	1.40	0.00	655.35	9.99	
P0024	Bootloader Version	0.00	20.03	0.00	655.35	0.00	
P0025	FPGA Project Version	0.00	0.03	0.00	655.35	0.00	
P0030	Present Alarm	0	0	0	2000	0	
P0031	Last Alarm	0	826	0	2000	0	
P0032	Last alarm Day.Month	0.00	3.01	0.00	31.12	0.00	
P0033	Last alarm year	0	1586	0	4096	0	
P0034	Last alarm Hour.Min	0.00	17.01	0.00	23.59	0.00	
P0035	Present Fault	0	0	0	2000	0	
P0036	Last Fault	0	2	0	2000	0	
P0037	Last Fault Day.Month	0.00	3.01	0.00	31.12	0.00	
P0038	Last fault year	0	1594	0	4096	0	
P0039	Last fault Hour.Min	0.00	17.13	0.00	23.59	0.00	
P0040	Second fault	0	2	0	2000	0	
P0041	Second fault Day.Month	0.00	3.01	0.00	31.12	0.00	
P0042	Second fault year	0	1594	0	4096	0	
P0043	Second fault Hour.Min	0.00	17.12	0.00	23.59	0.00	
P0044	Third fault	0	32	0	2000	0	
P0045	Third fault Day.Month	0.00	1.01	0.00	31.12	0.00	
P0046	Third fault year	0	1577	0	4096	0	
P0047	Third fault Hour.Min	0.00		0.00	23.59	0.00	

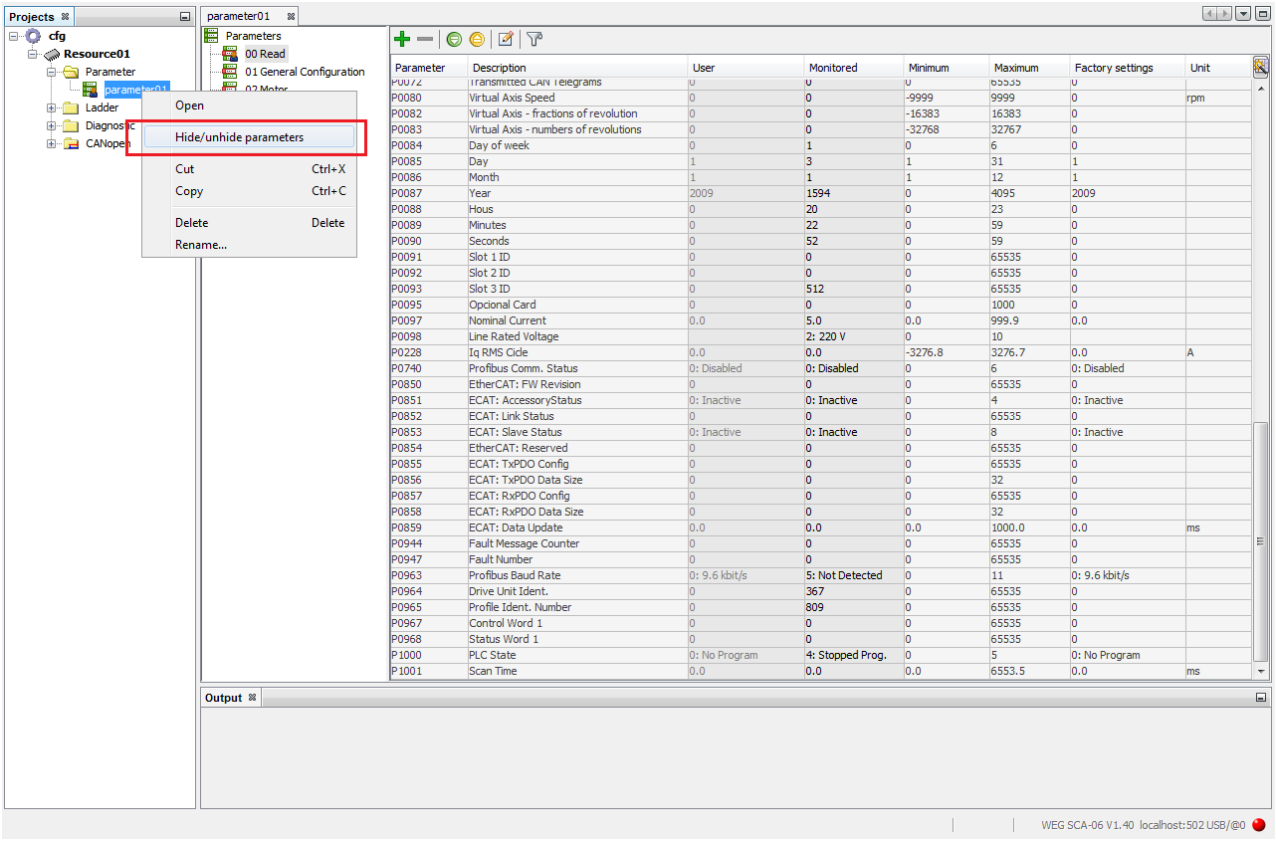


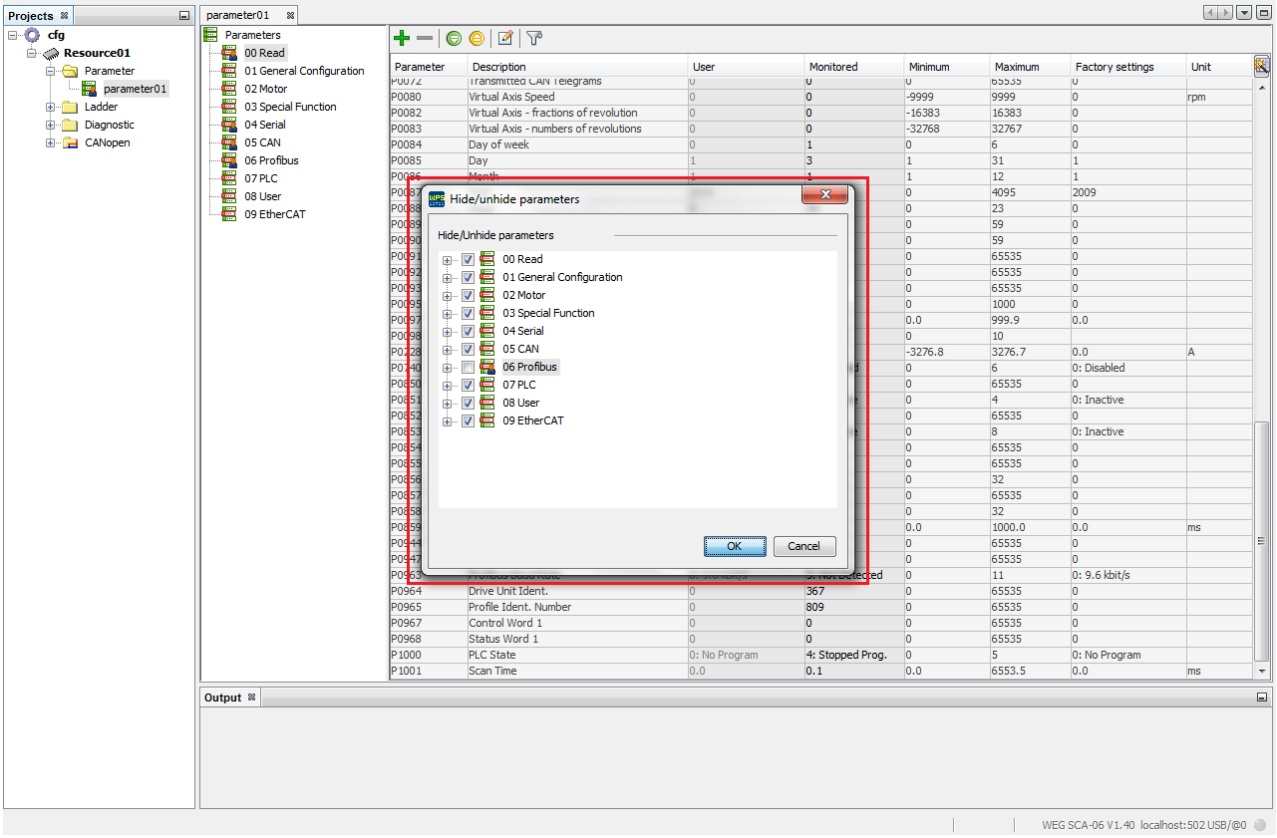
Para...	Description	User	Monitored	Minimum	Maxi...	Factory settings	Unit
P0002	Motor Speed	0	0	-9999	9999	0	rpm
P0003	Motor Current	0.0	**0.0**	-999.9	999.9	0.0	A
P0004	DC Link Voltage (Ud)	0	308	0	999	0	V
P0006	Drive Status	0: Ready	0: Ready	0	5	0: Ready	
P0008	DI3 to DI1 Status	0	0	0	7	0	
P0009	DI101 to DI106 Status	0	0	0	63	0	
P0010	DI107 to DI112 Status	0	0	0	63	0	
P0011	DI201 to DI206 S			0	63	0	
P0012	DI207 to DI212 S			0	63	0	
P0013	DI301 to DI306 S			0	63	0	
P0014	DI307 to DI312 S			0	63	0	
P0015	DO1 Status			0	1	0	
P0016	DO101 to DO106			0	63	0	
P0017	DO201 to DO206			0	63	0	
P0018	DO301 to DO306			0	63	0	
P0021	Internal Air Temp		1	0	1000	0	°C
P0022	Dissipator Temper		3	0	1000	0	°C
P0023	Software Version		.40	0.00	655.35	9.99	
P0024	Bootloader Versio		0.03	0.00	655.35	0.00	
P0025	FPGA Project Ver		.03	0.00	655.35	0.00	
P0030	Present Alarm			0	2000	0	
P0031	Last Alarm		26	0	2000	0	
P0032	Last alarm Day.M		.01	0.00	31.12	0.00	
P0033	Last alarm year		586	0	4096	0	
P0034	Last alarm Hour.M		7.01	0.00	23.59	0.00	
P0035	Present Fault			0	2000	0	
P0036	Last Fault			0	2000	0	
P0037	Last Fault Day.M		.01	0.00	31.12	0.00	
P0038	Last fault year		594	0	4096	0	
P0039	Last fault Hour.M		7.13	0.00	23.59	0.00	
P0040	Second fault		2	0	2000	0	
P0041	Second fault Day.Month	0.00	3.01	0.00	31.12	0.00	
P0042	Second fault year	0	1594	0	4096	0	
P0043	Second fault Hour.Min	0.00	17.12	0.00	23.59	0.00	
P0044	Third fault	0	32	0	2000	0	
P0045	Third fault Day.Month	0.00	1.01	0.00	31.12	0.00	
P0046	Third fault year	0	1577	0	4096	0	
P0047	Third fault Hour.Min	0.00		0.00	23.59	0.00	



Para...	Description	User	M...	Minimum	Maxim...	Factory settings	Unit
P0002	Motor Speed	0		-9999	9999	0	rpm
P0003	Motor Current	0.0		-999.9	999.9	0.0	A
P0004	DC Link Voltage (Ud)	0		0	999	0	V
P0006	Drive Status	0: Ready		0	5	0: Ready	
P0008	DI3 to DI1 Status	0		0	7	0	
P0009	DI101 to DI106 Status	0		0	63	0	
P0010	DI107 to DI112 Status	0		0	63	0	
P0011	DI201 to DI206 Status	0		0	63	0	
P0012	DI207 to DI212 Status	0		0	63	0	
P0013	DI301 to DI306 Status	0		0	63	0	
P0014	DI307 to DI312 Status	0		0	63	0	
P0015	DO1 Status	0		0	1	0	
P0016	DO101 to DO106 Status	0		0	63	0	
P0017	DO201 to DO206 Status	0		0	63	0	
P0018	DO301 to DO306 Status	0		0	63	0	
P0021	Internal Air Temperature	0		0	1000	0	°C
P0022	Dissipator Temperature	0		0	1000	0	°C
P0023	Software Version	9.99		0.00	655.35	9.99	
P0024	Bootloader Version	0.00		0.00	655.35	0.00	
P0025	FPGA Project Version	0.00		0.00	655.35	0.00	
P0030	Present Alarm	0		0	2000	0	
P0031	Last Alarm	0		0	2000	0	
P0032	Last alarm Day.Month	0.00		0.00	31.12	0.00	
P0033	Last alarm year	0		0	4096	0	
P0034	Last alarm Hour.Min	0.00		0.00	23.59	0.00	
P0035	Present Fault	0		0	2000	0	
P0036	Last Fault	0		0	2000	0	
P0037	Last Fault Day.Month	0.00		0.00	31.12	0.00	
P0038	Last fault year	0		0	4096	0	
P0039	Last fault Hour.Min	0.00		0.00	23.59	0.00	
P0040	Second fault	0		0	2000	0	
P0041	Second fault Day.Month	0.00		0.00	31.12	0.00	
P0042	Second fault year	0		0	4096	0	
P0043	Second fault Hour.Min	0.00		0.00	23.59	0.00	
P0044	Third fault	0		0	2000	0	
P0045	Third fault Day.Month	0.00		0.00	31.12	0.00	
P0046	Third fault year	0		0	4096	0	

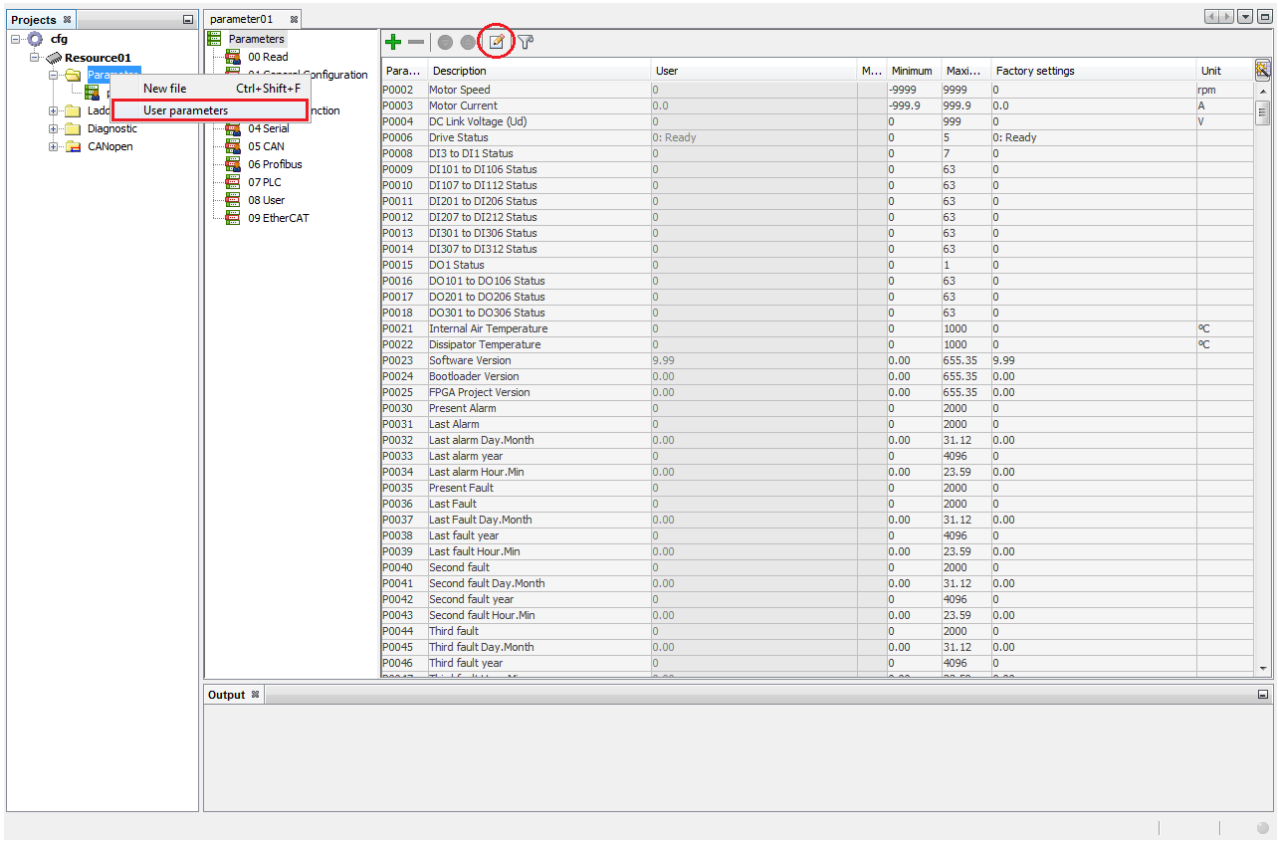
5. **Hide and Show Parameters and Groups of Parameters**. By means of this option, you have full control of the parameters and groups of parameters. It is possible to hide and unhide individual parameters, multiple parameters, individual groups and multiple groups in the same action.


Projects
cfg
Resource01
Parameter
parameter01
Ladder
Diagnostic
CANopen
parameter01
Parameters
00 Read
01 General Configuration
Open
Hide/unhide parameters
Cut Ctrl+X
Copy Ctrl+C
Delete Delete
Rename...
Parameter
Description
User
Monitored
Minimum
Maximum
Factory settings
Unit
P0080 Virtual Axis Speed 0 0 -9999 9999 0 rpm
P0082 Virtual Axis - fractions of revolution 0 0 -16383 16383 0
P0083 Virtual Axis - numbers of revolutions 0 0 -32768 32767 0
P0084 Day of week 0 1 0 6 0
P0085 Day 1 3 1 31 1
P0086 Month 1 1 1 12 1
P0087 Year 2009 1594 0 4095 2009
P0088 Hous 0 20 0 23 0
P0089 Minutes 0 22 0 59 0
P0090 Seconds 0 52 0 59 0
P0091 Slot 1 ID 0 0 0 65535 0
P0092 Slot 2 ID 0 0 0 65535 0
P0093 Slot 3 ID 0 512 0 65535 0
P0095 Opcional Card 0 0 0 1000 0
P0097 Nominal Current 0.0 5.0 0.0 999.9 0.0
P0098 Line Rated Voltage 2: 220 V 0 10
P0228 Iq RMS Cicle 0.0 0.0 -3276.8 3276.7 0.0 A
P0740 Profibus Comm. Status 0: Disabled 0: Disabled 0 6 0: Disabled
P0850 EtherCAT: FW Revision 0 0 0 65535 0
P0851 ECAT: AccessoryStatus 0: Inactive 0: Inactive 0 4 0: Inactive
P0852 ECAT: Link Status 0 0 0 65535 0
P0853 ECAT: Slave Status 0: Inactive 0: Inactive 0 8 0: Inactive
P0854 EtherCAT: Reserved 0 0 0 65535 0
P0855 ECAT: TxPDO Config 0 0 0 65535 0
P0856 ECAT: TxPDO Data Size 0 0 0 32 0
P0857 ECAT: RxPDO Config 0 0 0 65535 0
P0858 ECAT: RxPDO Data Size 0 0 0 32 0
P0859 ECAT: Data Update 0.0 0.0 0.0 1000.0 0.0 ms
P0944 Fault Message Counter 0 0 0 65535 0
P0947 Fault Number 0 0 0 65535 0
P0963 Profibus Baud Rate 0: 9.6 kbit/s 5: Not Detected 0 11 0: 9.6 kbit/s
P0964 Drive Unit Ident. 0 367 0 65535 0
P0965 Profile Ident. Number 0 809 0 65535 0
P0967 Control Word 1 0 0 0 65535 0
P0968 Status Word 1 0 0 0 65535 0
P1000 PLC State 0: No Program 4: Stopped Prog. 0 5 0: No Program
P1001 Scan Time 0.0 0.0 0.0 6553.5 0.0 ms
Output
WEG SCA-06 V1.40 localhost:502 USB/@0


Projects
cfg
Resource01
Parameter
parameter01
Ladder
Diagnostic
CANopen
parameter01
Parameters
00 Read
01 General Configuration
02 Motor
03 Special Function
04 Serial
05 CAN
06 Profibus
07 PLC
08 User
09 EtherCAT
Hide/unhide parameters
Hide/Unhide parameters
00 Read
01 General Configuration
02 Motor
03 Special Function
04 Serial
05 CAN
06 Profibus
07 PLC
08 User
09 EtherCAT
OK
Cancel
Parameter
Description
User
Monitored
Minimum
Maximum
Factory settings
Unit
P0080 Virtual Axis Speed 0 0 -9999 9999 0 rpm
P0082 Virtual Axis - fractions of revolution 0 0 -16383 16383 0
P0083 Virtual Axis - numbers of revolutions 0 0 -32768 32767 0
P0084 Day of week 0 1 0 6 0
P0085 Day 1 3 1 31 1
P0964 Drive Unit Ident. 0 367 0 65535 0
P0965 Profile Ident. Number 0 809 0 65535 0
P0967 Control Word 1 0 0 0 65535 0
P0968 Status Word 1 0 0 0 65535 0
P1000 PLC State 0: No Program 4: Stopped Prog. 0 5 0: No Program
P1001 Scan Time 0.0 0.1 0.0 6553.5 0.0 ms
Output
WEG SCA-06 V1.40 localhost:502 USB/@0

11.13.2.5 User Parameters

In order to open the configuration screen of the user parameters, just click on the **User Parameters** option on the Parameter node of the project tree or click on the icon indicated on the tool bar of the parameter file.



Para...	Description	User	M...	Minimum	Maxi...	Factory settings	Unit
P0002	Motor Speed	0		-9999	9999	0	rpm
P0003	Motor Current	0.0		-999.9	999.9	0.0	A
P0004	DC Link Voltage (Ud)	0		0	999	0	V
P0006	Drive Status	0: Ready		0	5	0: Ready	
P0008	DI3 to DI1 Status	0		0	7	0	
P0009	DI101 to DI106 Status	0		0	63	0	
P0010	DI107 to DI112 Status	0		0	63	0	
P0011	DI201 to DI206 Status	0		0	63	0	
P0012	DI207 to DI212 Status	0		0	63	0	
P0013	DI301 to DI306 Status	0		0	63	0	
P0014	DI307 to DI312 Status	0		0	63	0	
P0015	DO1 Status	0		0	1	0	
P0016	DO101 to DO106 Status	0		0	63	0	
P0017	DO201 to DO206 Status	0		0	63	0	
P0018	DO301 to DO306 Status	0		0	63	0	
P0021	Internal Air Temperature	0		0	1000	0	°C
P0022	Dissipator Temperature	0		0	1000	0	°C
P0023	Software Version	9.99		0.00	655.35	9.99	
P0024	Bootloader Version	0.00		0.00	655.35	0.00	
P0025	FPGA Project Version	0.00		0.00	655.35	0.00	
P0030	Present Alarm	0		0	2000	0	
P0031	Last Alarm	0		0	2000	0	
P0032	Last alarm Day.Month	0.00		0.00	31.12	0.00	
P0033	Last alarm year	0		0	4096	0	
P0034	Last alarm Hour.Min	0.00		0.00	23.59	0.00	
P0035	Present Fault	0		0	2000	0	
P0036	Last Fault	0		0	2000	0	
P0037	Last Fault Day.Month	0.00		0.00	31.12	0.00	
P0038	Last fault year	0		0	4096	0	
P0039	Last fault Hour.Min	0.00		0.00	23.59	0.00	
P0040	Second fault	0		0	2000	0	
P0041	Second fault Day.Month	0.00		0.00	31.12	0.00	
P0042	Second fault year	0		0	4096	0	
P0043	Second fault Hour.Min	0.00		0.00	23.59	0.00	
P0044	Third fault	0		0	2000	0	
P0045	Third fault Day.Month	0.00		0.00	31.12	0.00	
P0046	Third fault year	0		0	4096	0	

Configuration Table.

On the user parameter configuration table, it is possible to define several attributes to the parameters, such as description, minimum and maximum values, unit, digits, data type, etc.

NOTE!
These settings will be automatically displayed in the parameter table. However, to be sent to the device, you need to *download* the resource.

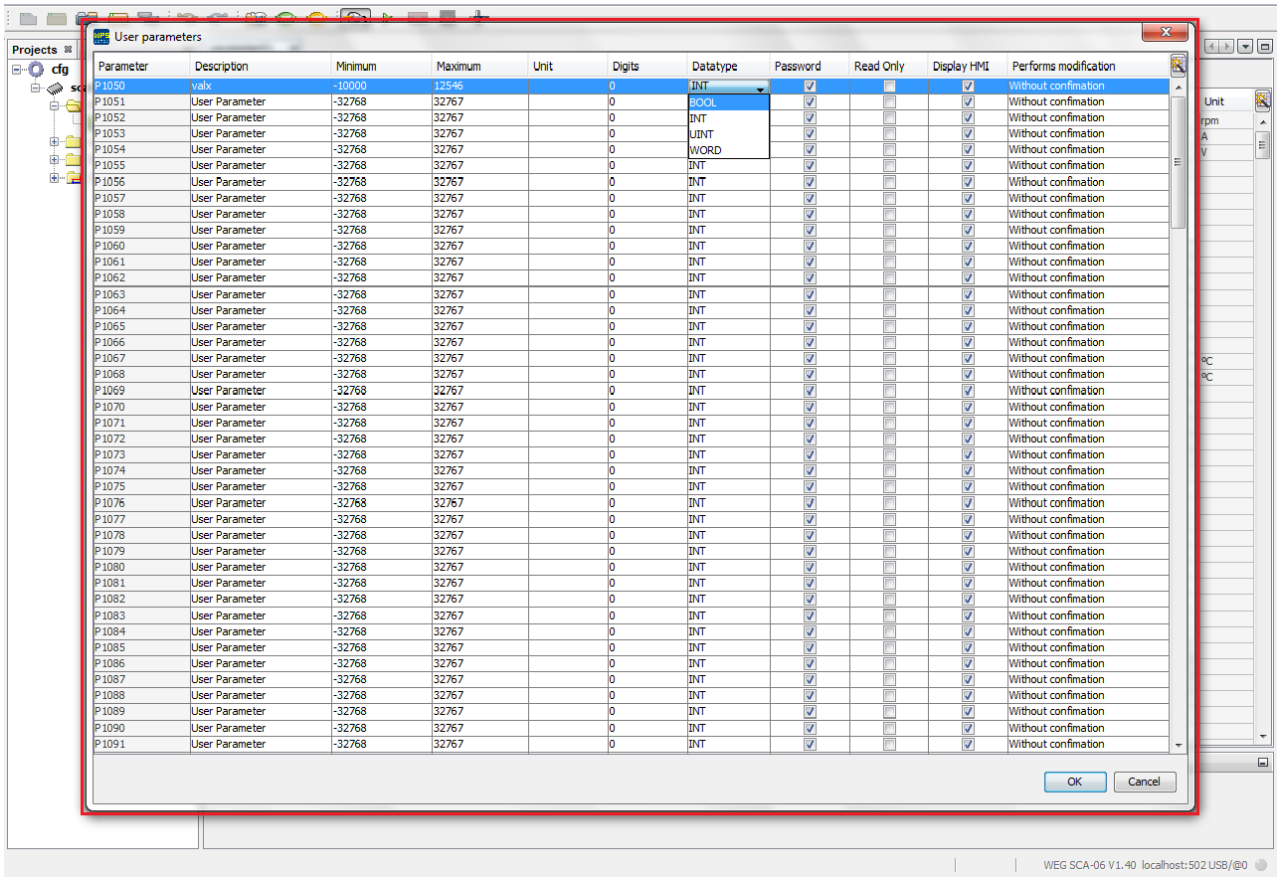


Table fields:

- Parameter: User parameter identification.
- Description: Description of the user parameter in the parameter table. On devices that have text-based HMIs, the description is sent to the machine and displayed on the HMI.
- Minimum: Minimum input value for parameter.
- Maximum: Maximum input value for parameter.
- Unit: Unit displayed on the device's HMI.
- Default: Value loaded when restore factory default is selected.
- Retentive: Retain value after rebooting devices.
- Hexadecimal: Displays the value in hexadecimal.
- Digits: Number of decimal digits for displaying value.
- Datatype: Parameter datatype used by the ladder application.
- Password: Enables password request by changing parameter value.

- Read only: It does not allow the writing of values in the parameter by the communication network or the HMI. Writing is done only by the ladder application.

- Display HMI: Displays the parameter in the HMI.

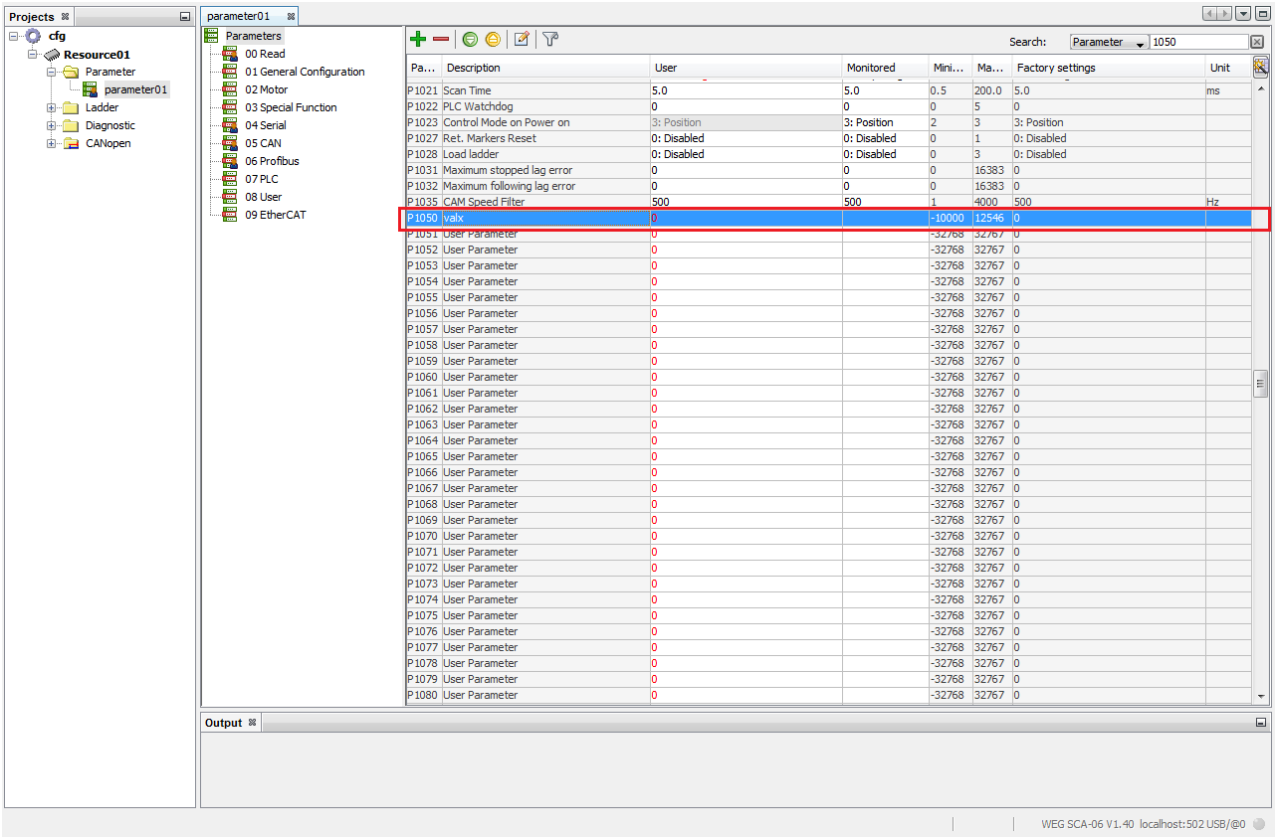
- Performs modification: Confirmation options when changing the parameter:
o No confirmation: Does not prompt for confirmation when changing parameter.
o With confirmation and engine stopped: Request confirmation and allow change only with engine stopped.
o With confirmation: Prompt for confirmation when changing parameter.

- Stopped motor: Perform change only with motor stopped.

- Help: On devices that have text-based HMI, you can edit a help text for the parameter.

View the user parameter

In the parameter table, the user parameters will be shown as they are configured on the configuration screen.



Pa...	Description	User	Monitored	Mini...	Ma...	Factory settings	Unit
P1021	Scan Time	5.0	5.0	0.5	200.0	5.0	ms
P1022	PLC Watchdog	0	0	0	5	0	
P1023	Control Mode on Power on	3: Position	3: Position	2	3	3: Position	
P1027	Ret. Markers Reset	0: Disabled	0: Disabled	0	1	0: Disabled	
P1028	Load ladder	0: Disabled	0: Disabled	0	3	0: Disabled	
P1031	Maximum stopped lag error	0	0	0	16383	0	
P1032	Maximum following lag error	0	0	0	16383	0	
P1035	CAM Speed Filter	500	500	1	4000	500	Hz
P1050	valx	0		-10000	12546	0	
P1051	User Parameter	0		-32768	32767	0	
P1052	User Parameter	0		-32768	32767	0	
P1053	User Parameter	0		-32768	32767	0	
P1054	User Parameter	0		-32768	32767	0	
P1055	User Parameter	0		-32768	32767	0	
P1056	User Parameter	0		-32768	32767	0	
P1057	User Parameter	0		-32768	32767	0	
P1058	User Parameter	0		-32768	32767	0	
P1059	User Parameter	0		-32768	32767	0	
P1060	User Parameter	0		-32768	32767	0	
P1061	User Parameter	0		-32768	32767	0	
P1062	User Parameter	0		-32768	32767	0	
P1063	User Parameter	0		-32768	32767	0	
P1064	User Parameter	0		-32768	32767	0	
P1065	User Parameter	0		-32768	32767	0	
P1066	User Parameter	0		-32768	32767	0	
P1067	User Parameter	0		-32768	32767	0	
P1068	User Parameter	0		-32768	32767	0	
P1069	User Parameter	0		-32768	32767	0	
P1070	User Parameter	0		-32768	32767	0	
P1071	User Parameter	0		-32768	32767	0	
P1072	User Parameter	0		-32768	32767	0	
P1073	User Parameter	0		-32768	32767	0	
P1074	User Parameter	0		-32768	32767	0	
P1075	User Parameter	0		-32768	32767	0	
P1076	User Parameter	0		-32768	32767	0	
P1077	User Parameter	0		-32768	32767	0	
P1078	User Parameter	0		-32768	32767	0	
P1079	User Parameter	0		-32768	32767	0	
P1080	User Parameter	0		-32768	32767	0	

11.14 SSW900

11.14.1 Description

The "Soft-Starter WEG 900" is a high-performance product that allows the starting/stopping control and protection of three-phase induction motors. Thus it prevents mechanical shocks on the load, current peaks in the supply line and damage to the motor.

Refer to the user's manual of the SSW900 for further details about the product.

11.14.2I/O's

Hardware information can be found in the Manual of the SSW900 at the website www.weg.net.

Digital Inputs

Address	Bit	Modbus	Tag	Description
%IB0	0	16000	DI1	Digital Input 1
%IB0	1	16001	DI2	Digital Input 2
%IB0	2	16002	DI3	Digital Input 3
%IB0	3	16003	DI4	Digital Input 4
%IB0	4	16004	DI5	Digital Input 5
%IB0	5	16005	DI6	Digital Input 6

Digital Outputs

Address	Bit	Modbus	Tag	Description
%QB0	0	16000	DO1	Digital Output 1
%QB0	1	16001	DO2	Digital Output 2
%QB0	2	16002	DO3	Digital Output 3

Analog Outputs

Address	Bit	Modbus	Tag	Description
%QW1	--	5001	AO1	Analog Output 1

11.14.3System Markers

The following variables contained in teh **GLOBAL_SYSTEM** group of the variables table, have the fixed tag. The tag of system markers were divided into groups and subgroups, wher:do:

Grupos:
- SSW: reading and writing variables of the SSW900 soft-starter.

Subgroups:
- STS: reading variable (status);
- CMD: writing variable (command).

Reading System Markers (Status)

Rading - Modbus Function 02 "Read Discrete Inputs"

Address	Bit	Modbus	Tag	Description
Ladder				
%SB6000	0	0	SYS_FREQ_2HZ	Oscilator with frequency of 2 Hz
%SB6000	1	1	SYS_PULSE_1SCAN	Pulse during the first scan cycle
%SB6000	2	2	SYS_FALSE	Always in 0
%SB6000	3	3	SYS_TRUE	Always in 1

Address	Bit	Modbus	Tag	Description
Logical Status				
%SB6002	0	16	SSW_STS_MOTOR_RUNNING	The soft-starter is turning the motor
%SB6002	1	17	SSW_STS_GENERAL_ENABLED	The soft-starter is general enabled and ready to run motor
%SB6002	2	18	SSW_STS_JOG_ACTIVE	The JOG function is active
%SB6002	3	19	SSW_STS_INITIAL_TEST	Executing initial tests before starting the engine
%SB6002	4	20	SSW_STS_ACCEL_RAMP	Motor in acceleration ramp
%SB6002	5	21	SSW_STS_FULL_VOLTAGE	Full voltage is being applied to the motor
%SB6002	6	22	SSW_STS_BYPASS	The bypass contactor is closed
%SB6002	7	23	SSW_STS_DECEL_RAMP	Motor in deceleration ramp
%SB6003	0	24	SSW_STS_LOC_REM_MODE	Soft-starter is in local or remote mode (0-Local, 1-Remote)
%SB6003	1	25	SSW_STS_BRAKING	Braking is active
%SB6003	2	26	SSW_STS_REVERSING_DIRECTION	The motor is reversing the direction of rotation
%SB6003	3	27	SSW_STS_FWD_REV_DIRECTION	Motor running in reverse or direct direction (0-Reverse, 1-Direct)
%SB6003	4	28	SSW_STS_START_DELAY	Time before startin the motor
%SB6003	5	29	SSW_STS_RESTART_DELAY	Time after motor stop
%SB6003	6	30	SSW_STS_ALARM_ACTIVE	Soft-starter has an active alarm
%SB6003	7	31	SSW_STS_FAULT_ACTIVE	Soft-starter is in fault state

Address	Bit	Modbus	Tag	Description
HMI keys				
%SB6004	0	32	SSW_STS_KEY_BACK	BACK key pressed
%SB6004	1	33	SSW_STS_KEY_UP	UP key pressed
%SB6004	2	34	SSW_STS_KEY_HELP	HELP key pressed
%SB6004	3	35	SSW_STS_KEY_DIRECTION	DIRECTION key pressed
%SB6004	4	36	SSW_STS_KEY_DOWN	DOWN key pressed
%SB6004	5	37	SSW_STS_KEY_LOC_REM	LOC/REM key pressed
%SB6004	6	38	SSW_STS_KEY_START	START (I) key pressed
%SB6004	7	39	SSW_STS_KEY_JOG	JOG key pressed
%SB6005	0	40	SSW_STS_KEY_STOP	STOP (0) key pressed
%SB6005	1	41	SSW_STS_KEY_RIGHT	RIGHT key pressed
%SB6005	2	42	SSW_STS_KEY_ENTER	ENTER key pressed
%SB6005	3	43	SSW_STS_KEY_LEFT	LEFT key pressed

Writing / Reading System Markers (Command)

Reading - Modbus Function 01 "Read Coils"
Writing - Modbus Function 05 "Write Single Coil" and 15 "Write Multiple Coils"

Address	Bit	Modbus	Tag	Description
Logical Command				
%CB6006	0	0	SSW_CMD_RUN_STOP	Run the motor (0-Stop, 1-Run)
%CB6006	1	1	SSW_CMD_GENERAL_ENABLE	Enables the soft-starter allowing the motor operation (0-Disable, 1-Enable)
%CB6006	2	2	SSW_CMD_JOG	Enables the JOG function (0-Disable, 1-Enable)
%CB6006	3	3	SSW_CMD_DIRECTION	Reverses the direction of rotation of motor
%CB6006	4	4	SSW_CMD_LOC_REM	Selects the soft-starter operation mode (0-Local, 1-Remote)
%CB6006	7	7	SSW_CMD_FAULT_RESET	Executes the fault reset command

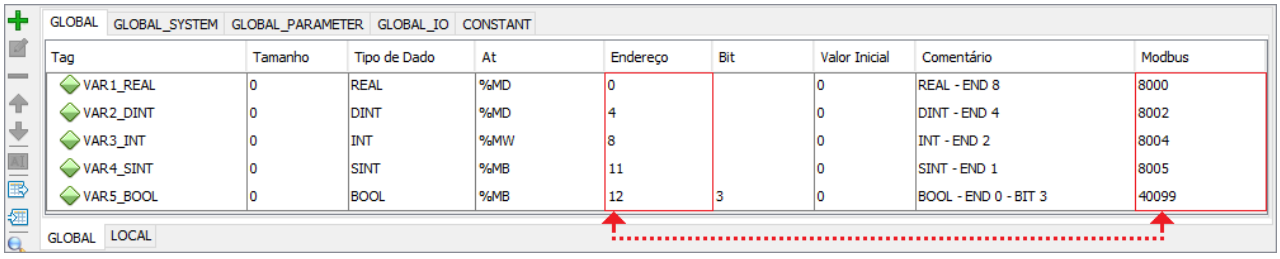
11.14.4Volatile Markers

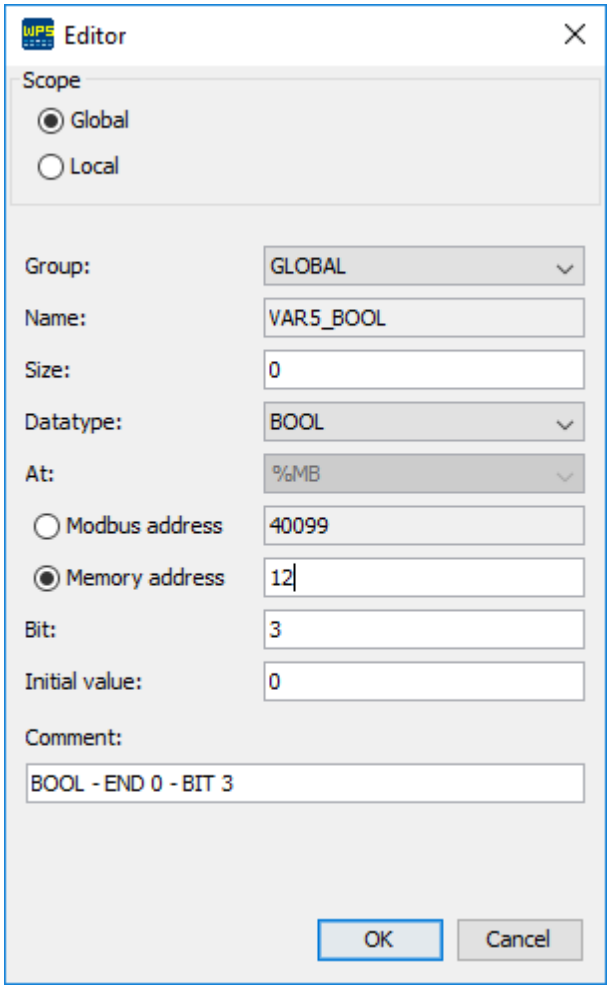
Only the variables created in the **GLOBAL** group of the variables table, called Global Volatile Markers, and that have a user-specified address, will be accessible by modbus protocol, when their are within the following range:

TYPE	Address Range	Modbus Map	Description
%MB: BOOL	0 - 7343	40000 - 49999 *(Coil)*	Each modbus address represents 1 bit of the address content. After address 1250, the bit is not accessible via modbus.
%MB: BYTE, SINT, USINT **%MW:** WORD, INT, DINT **%MD:** DWORD, DINT, UDINT, REAL		8000 - 11671 *(Register)*	Each modbus address represents 2 bytes of the address content. After address 7344, the data is not accessible via modbus.

Modbus functions with access to volatile global markers with address defined:
1. Read - Modbus Function 01 "Read Coils"
2. Read - Modbus Function 03 "Read Holding Registers"
3. Write - Modbus Function 05 "Write Single Coil"
4. Write - Modbus Function 06 "Write Single Register"
5. Write - Modbus Function 15 "Write Multiple Coils"
6. Write - Modbus Function 16 "Write Multiple Registers"

Example of volatile global markers declaration with address defined in variables table of the ladder editor:

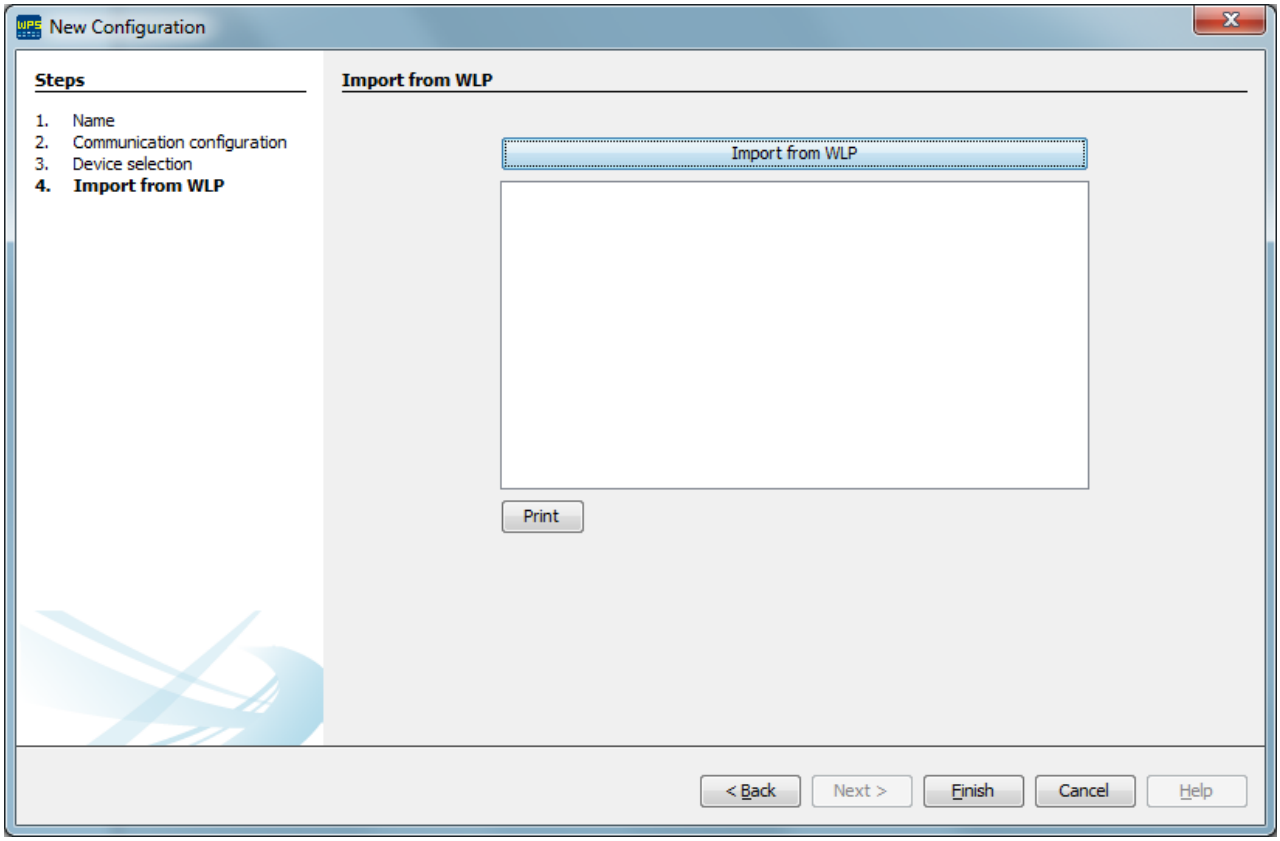




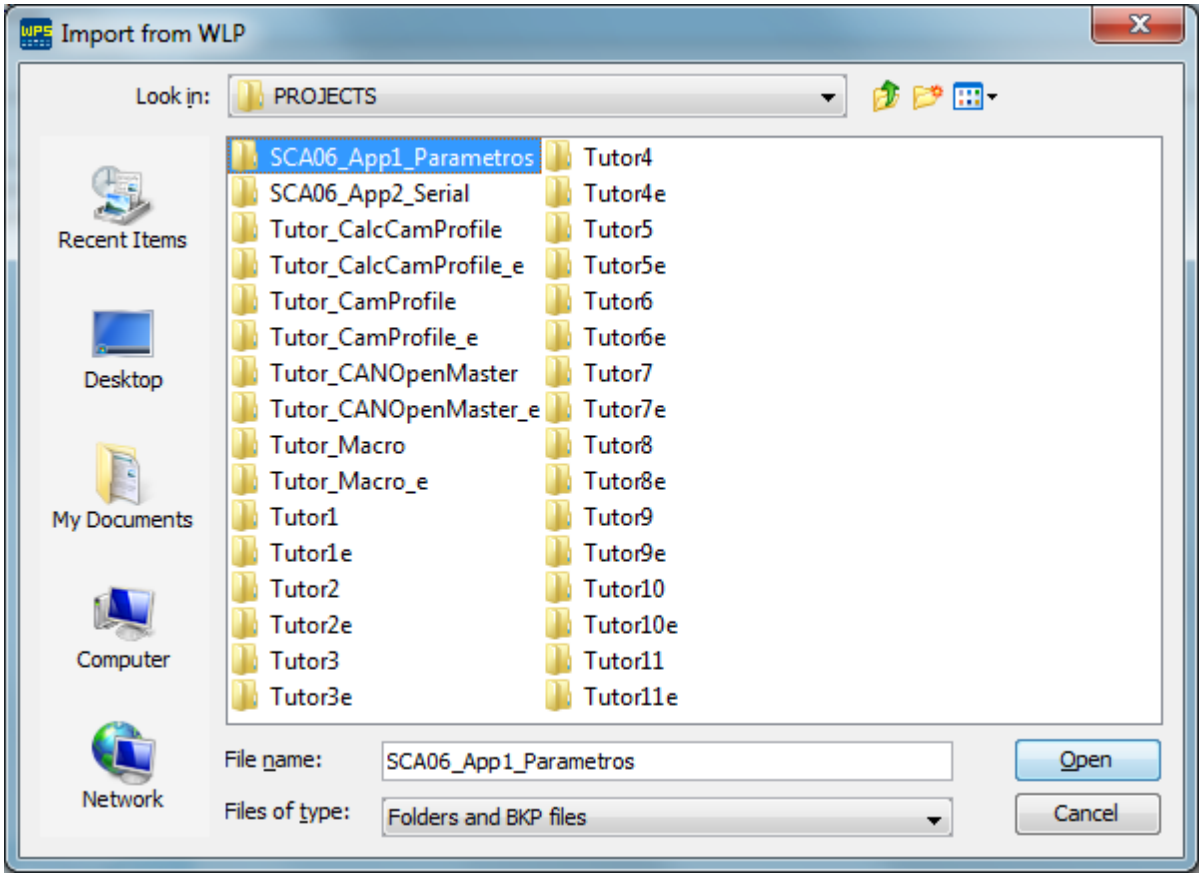
11.14.5Import from WLP

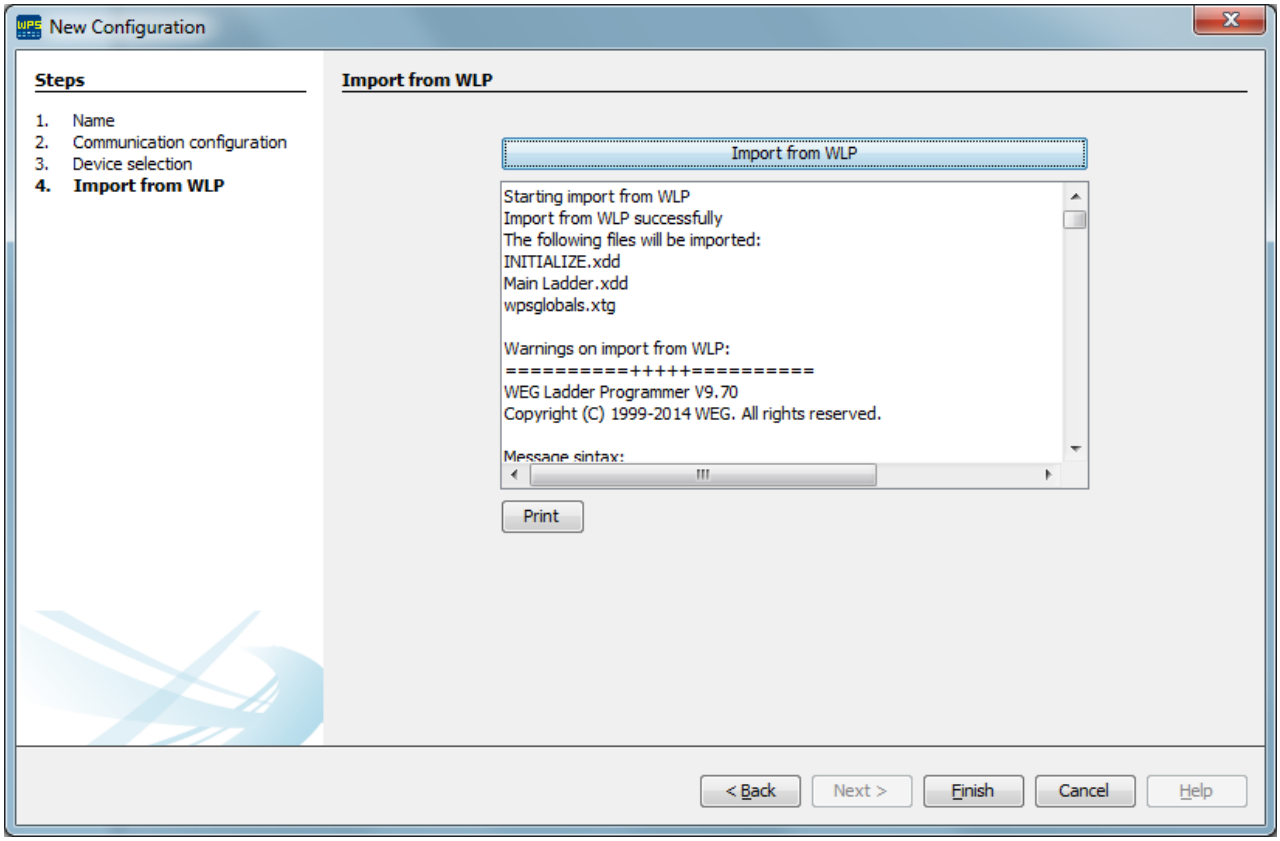
The function import from WLP is utilized to import Ladder developed on WLP software to equipment (device).

The import from WLP can be executed during the resource creation.



1. To execute the import WLP function click the Import from WLP button and select the WLP project folder or the WLP BKP file.


Import from WLP
Look in:
PROJECTS
SCA06_App1_Parametros
SCA06_App2_Serial
Tutor_CalcCamProfile
Tutor_CalcCamProfile_e
Tutor_CamProfile
Tutor_CamProfile_e
Tutor_CANOpenMaster
Tutor_CANOpenMaster_e
Tutor_Macro
Tutor_Macro_e
Tutor1
Tutor1e
Tutor2
Tutor2e
Tutor3
Tutor3e
Tutor4
Tutor4e
Tutor5
Tutor5e
Tutor6
Tutor6e
Tutor7
Tutor7e
Tutor8
Tutor8e
Tutor9
Tutor9e
Tutor10
Tutor10e
Tutor11
Tutor11e
Recent Items
Desktop
My Documents
Computer
Network
File name:
SCA06_App1_Parametros
Files of type:
Folders and BKP files
Open
Cancel



2. After import from WLP completed successfully click the Finish button to copy the imported files to new resource.

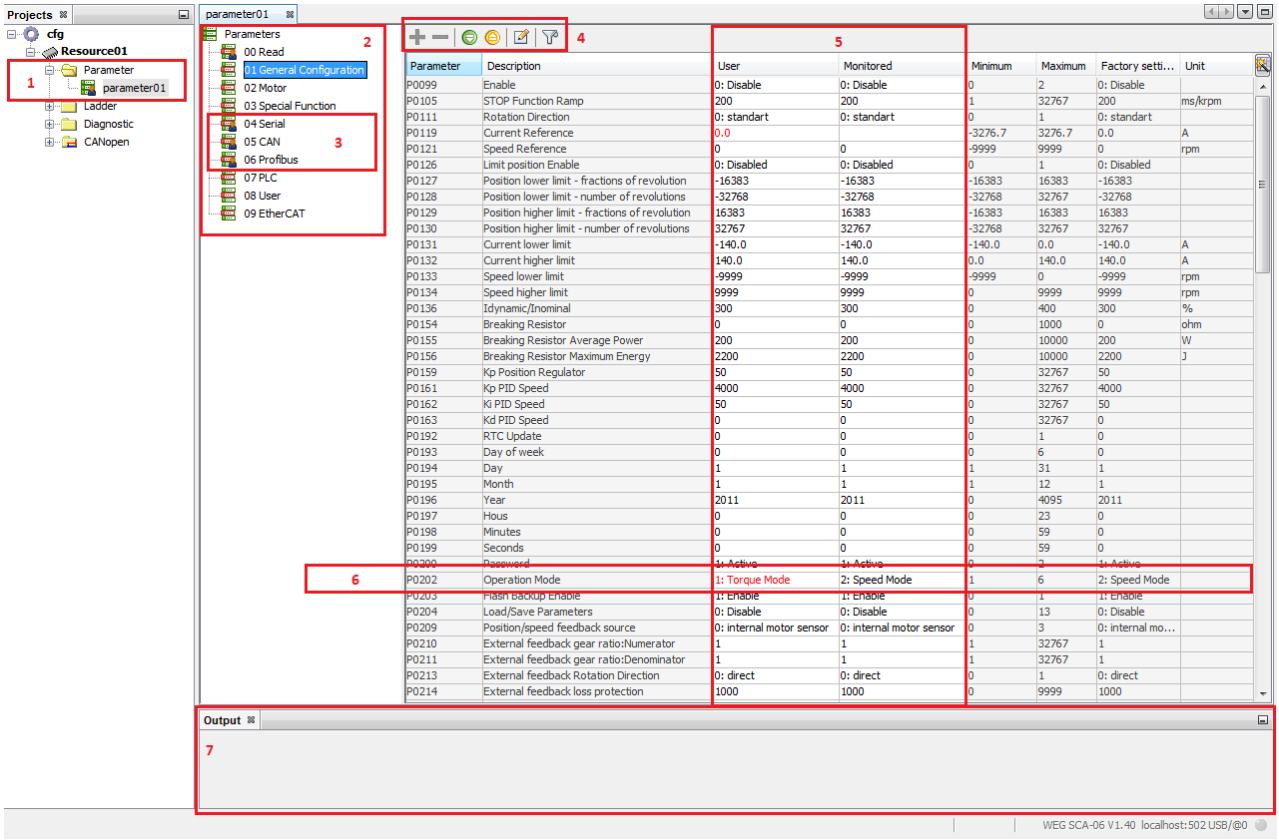
11.14.6 Parameters

11.14.6.1 Overview

The parameter configuration screen is used to configure and monitor all the parameters of the equipment, including the user parameters.

NOTE!
The reading and writing of such parameters is done on this screen; only the user parameter configuration must be sent the first time or whenever modified by means of the resource download routine.

Below is an overview of the parameter configuration screen.



1. **Parameter files**. In this part are all the parameter configuration files created by the user. Notice that when the file features a person figure on the table, it means this parameter table contains hidden parameters/ group of parameters.
2. **Group of parameters**. This tree shows all the group of parameters. Notice that the same parameter can be in more than one group, and when its value is modified, it will be modified in all the groups to which it belongs.
3. **Modified group of parameters**. Group of parameters which contain the figure of a person on the table means they have hidden parameters.
4. **Commands**. The commands are described below in the order they appear:
   4.1. **Unhide parameter**: In case some parameter has been hidden, this button allows making it visible again.
   4.2. **Hide parameter**: Just select one or more parameters on the table and trigger this command to hide them.
   4.3. **Save table**: It saves the values of the parameters shown on the equipment screen; the sent values are the ones in the User column. The flow is User -> Monitored (equipment)
   4.4. **Read table**: It reads the parameters of the equipment shown in the Monitored column and saves them in the parameter file in the User column. The flow is Monitored (equipment) -> User
   4.5. **User parameters**: It opens a screen to edit the user parameters.
   4.6. **Filter**: It opens a parameter filter option, and it can filter by parameter number or description.
   4.7. **User Parameters and Monitored Parameters**. These two columns show the off-line and on-line parameters, so to speak. The **User** column shows the values contained in the file located on the computer and the **Monitored** column shows the values that are effectively saved on the equipment. Whenever you use the **Save Parameter** option, the sent values will be from the **User** column to the **Monitored** column, that is, File -> Equipment. In case of reading, the flow is the opposite, from the **Monitored** column to the **User** column, that is, Equipment -> File. In case you wish to change the values directly on the equipment without changing it in the file, just click on the monitored column,

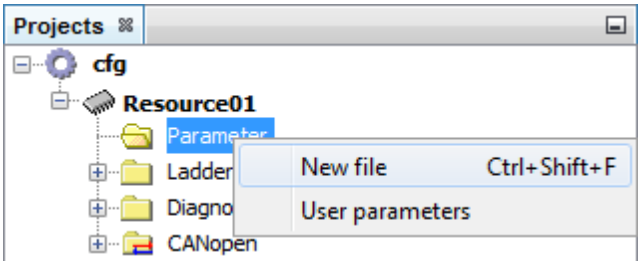
change the values and the modification will occur on-line.

5. **Modified parameters**: Whenever a parameter value in the **User** column is different from the **Monitored** column, it will be shown in red.
6. **Output**. This screen shows error information in case they occur during the writing or reading of the parameters.

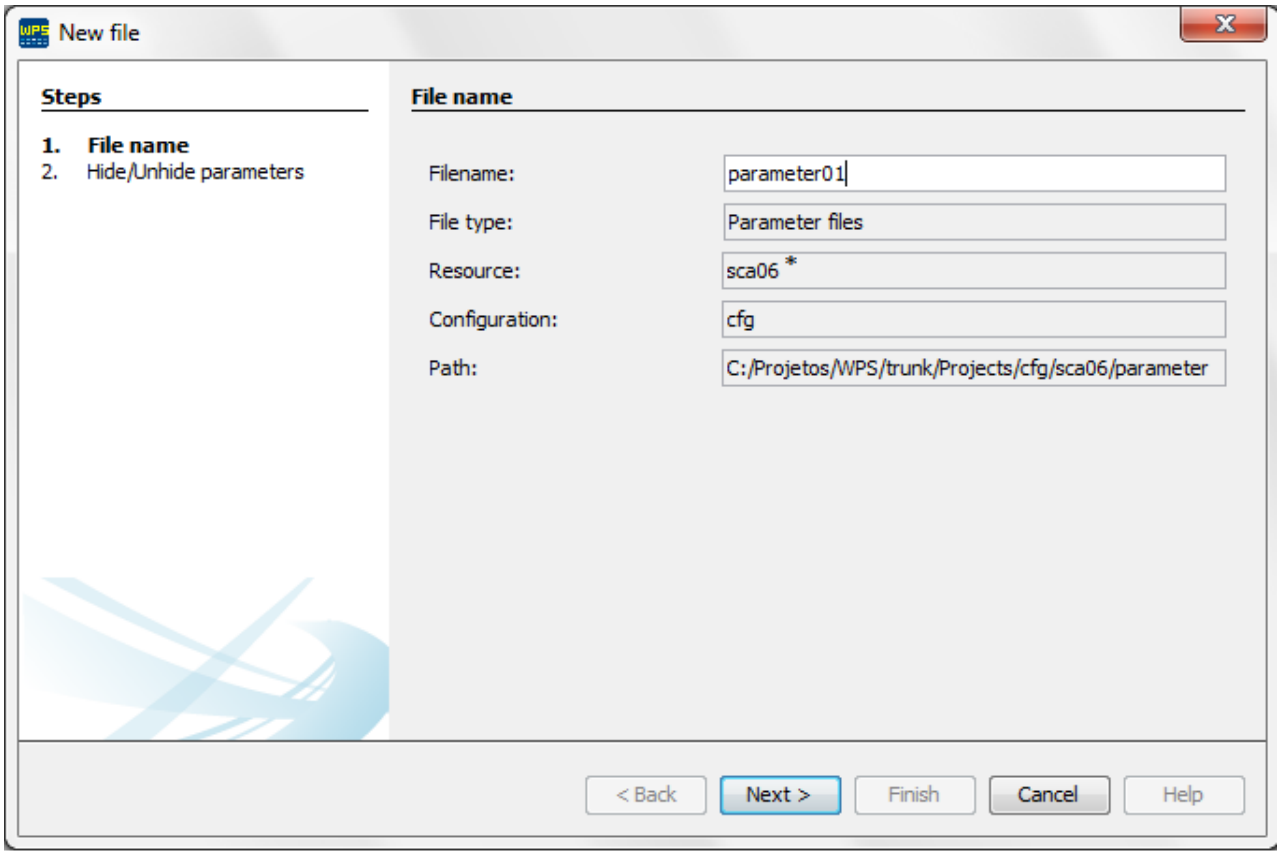
11.14.6.2 Configuration

Below is the list of the required steps to create a parameter file.

1. Create a new parameter file.

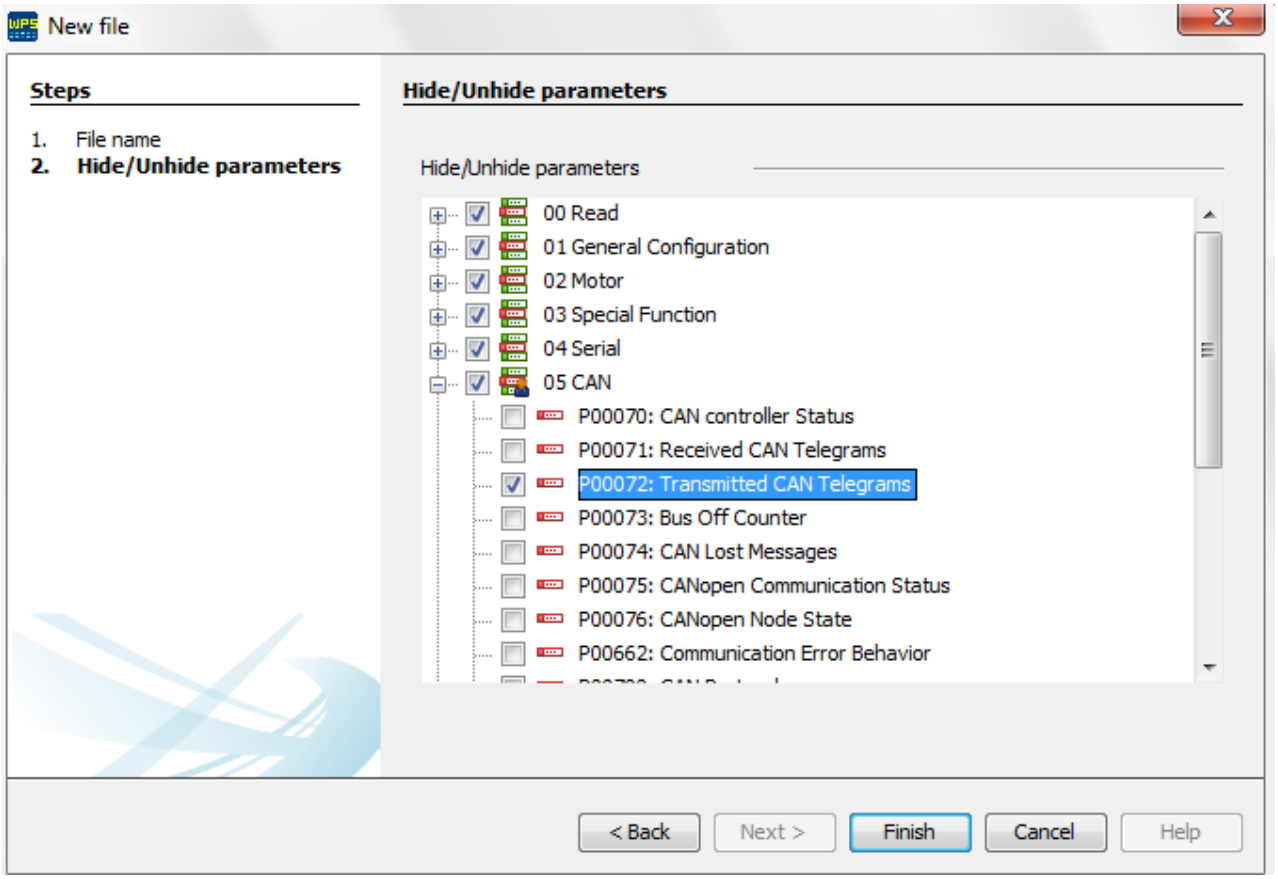


2. Define a name for the parameter file

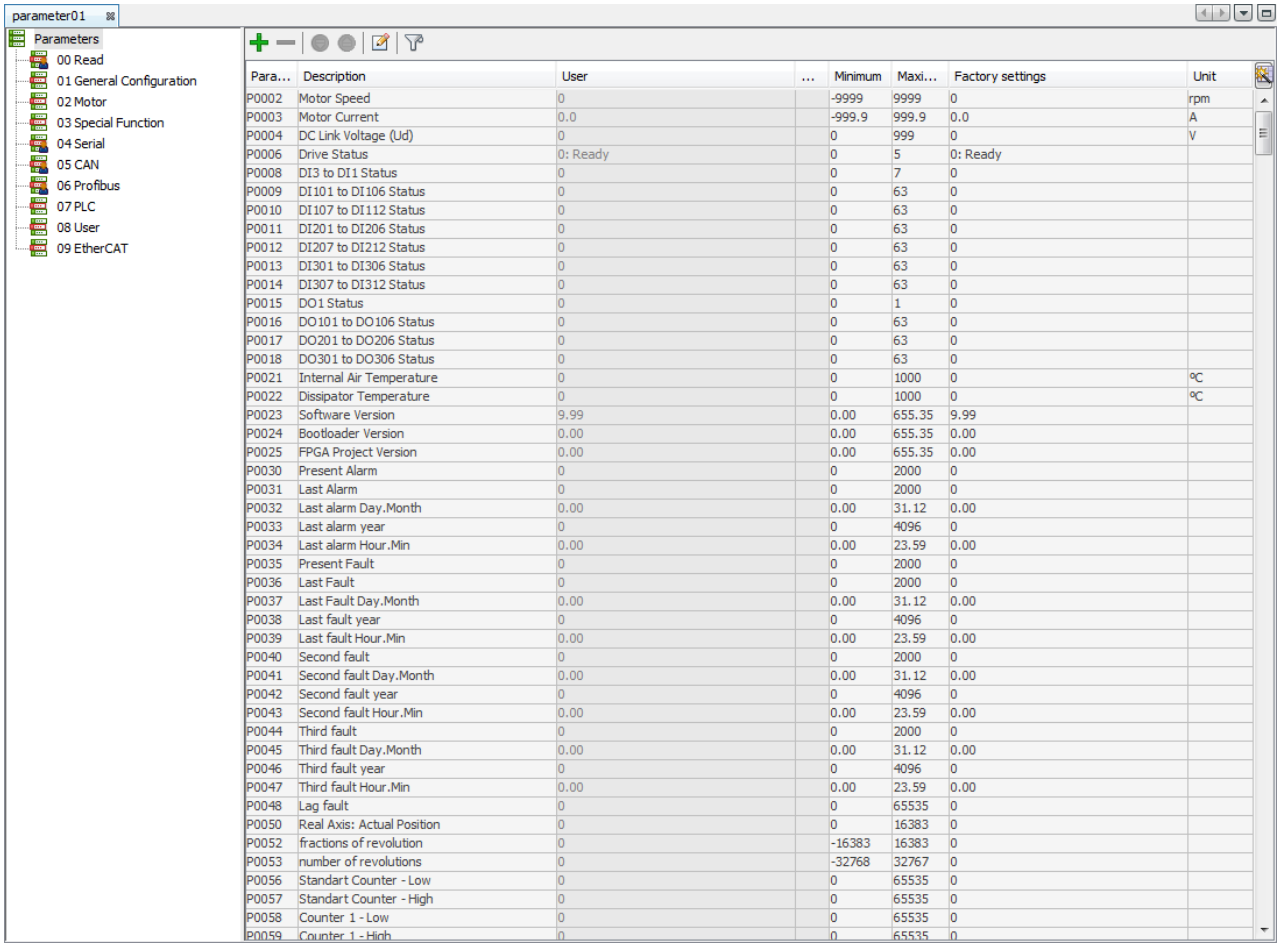


*** Resource: Resource01, SCA06, CFW300, etc.**

3. Configure which parameters you wish to view in your parameter table



4. After performing the steps above, the parameter file will be created and the equipment can be parameterized.

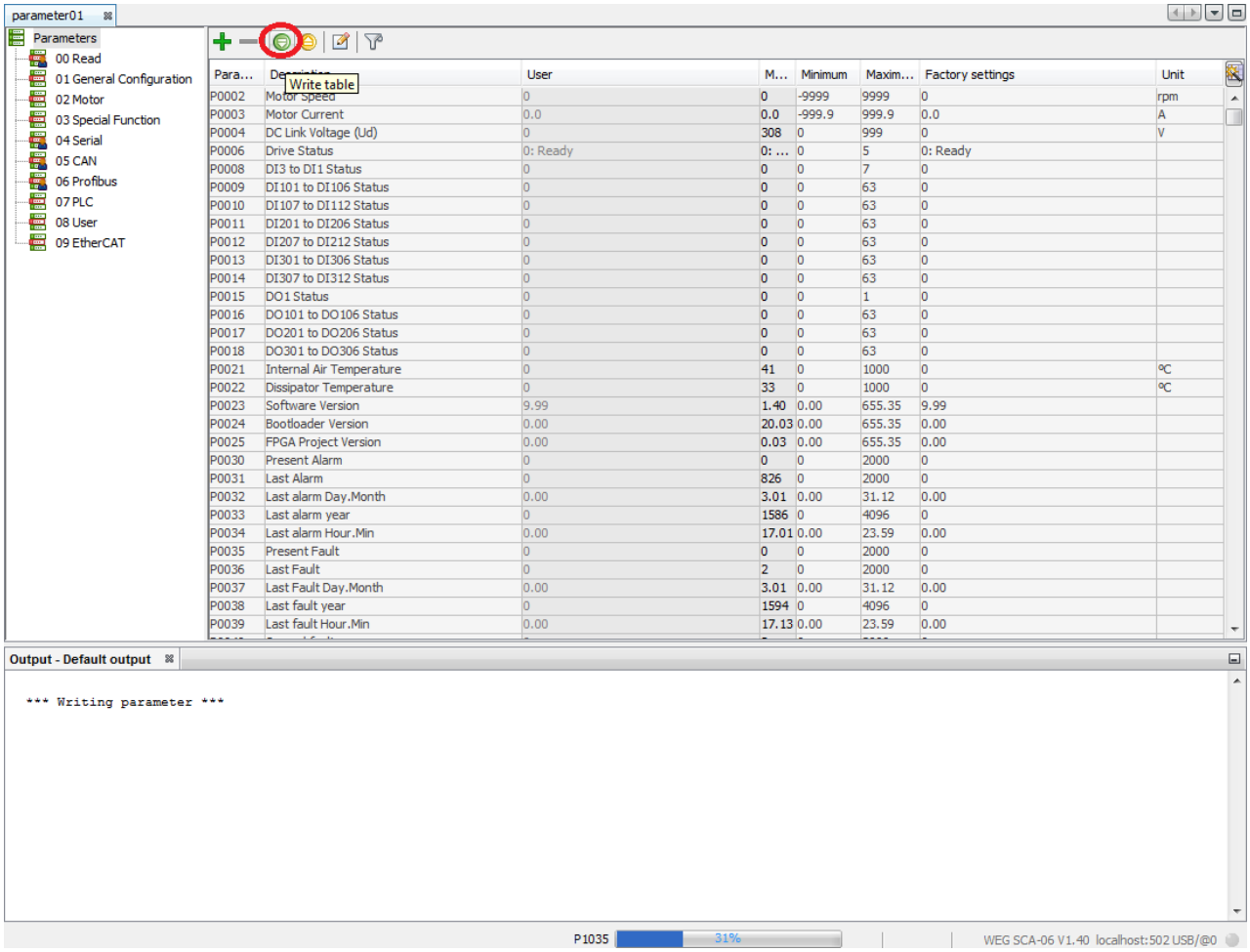


Para...	Description	User	...	Minimum	Maxi...	Factory settings	Unit
P0002	Motor Speed	0		-9999	9999	0	rpm
P0003	Motor Current	0.0		-999.9	999.9	0.0	A
P0004	DC Link Voltage (Ud)	0		0	999	0	V
P0006	Drive Status	0: Ready		0	5	0: Ready	
P0008	DI3 to DI1 Status	0		0	7	0	
P0009	DI101 to DI106 Status	0		0	63	0	
P0010	DI107 to DI112 Status	0		0	63	0	
P0011	DI201 to DI206 Status	0		0	63	0	
P0012	DI207 to DI212 Status	0		0	63	0	
P0013	DI301 to DI306 Status	0		0	63	0	
P0014	DI307 to DI312 Status	0		0	63	0	
P0015	DO1 Status	0		0	1	0	
P0016	DO101 to DO106 Status	0		0	63	0	
P0017	DO201 to DO206 Status	0		0	63	0	
P0018	DO301 to DO306 Status	0		0	63	0	
P0021	Internal Air Temperature	0		0	1000	0	ºC
P0022	Dissipator Temperature	0		0	1000	0	ºC
P0023	Software Version	9.99		0.00	655.35	9.99	
P0024	Bootloader Version	0.00		0.00	655.35	0.00	
P0025	FPGA Project Version	0.00		0.00	655.35	0.00	
P0030	Present Alarm	0		0	2000	0	
P0031	Last Alarm	0		0	2000	0	
P0032	Last alarm Day.Month	0.00		0.00	31.12	0.00	
P0033	Last alarm year	0		0	4096	0	
P0034	Last alarm Hour.Min	0.00		0.00	23.59	0.00	
P0035	Present Fault	0		0	2000	0	
P0036	Last Fault	0		0	2000	0	
P0037	Last Fault Day.Month	0.00		0.00	31.12	0.00	
P0038	Last fault year	0		0	4096	0	
P0039	Last fault Hour.Min	0.00		0.00	23.59	0.00	
P0040	Second fault	0		0	2000	0	
P0041	Second fault Day.Month	0.00		0.00	31.12	0.00	
P0042	Second fault year	0		0	4096	0	
P0043	Second fault Hour.Min	0.00		0.00	23.59	0.00	
P0044	Third fault	0		0	2000	0	
P0045	Third fault Day.Month	0.00		0.00	31.12	0.00	
P0046	Third fault year	0		0	4096	0	
P0047	Third fault Hour.Min	0.00		0.00	23.59	0.00	
P0048	Lag fault	0		0	65535	0	
P0050	Real Axis: Actual Position	0		0	16383	0	
P0052	fractions of revolution	0		-16383	16383	0	
P0053	number of revolutions	0		-32768	32767	0	
P0056	Standart Counter - Low	0		0	65535	0	
P0057	Standart Counter - High	0		0	65535	0	
P0058	Counter 1 - Low	0		0	65535	0	
P0059	Counter 1 - High	0		0	65535	0	

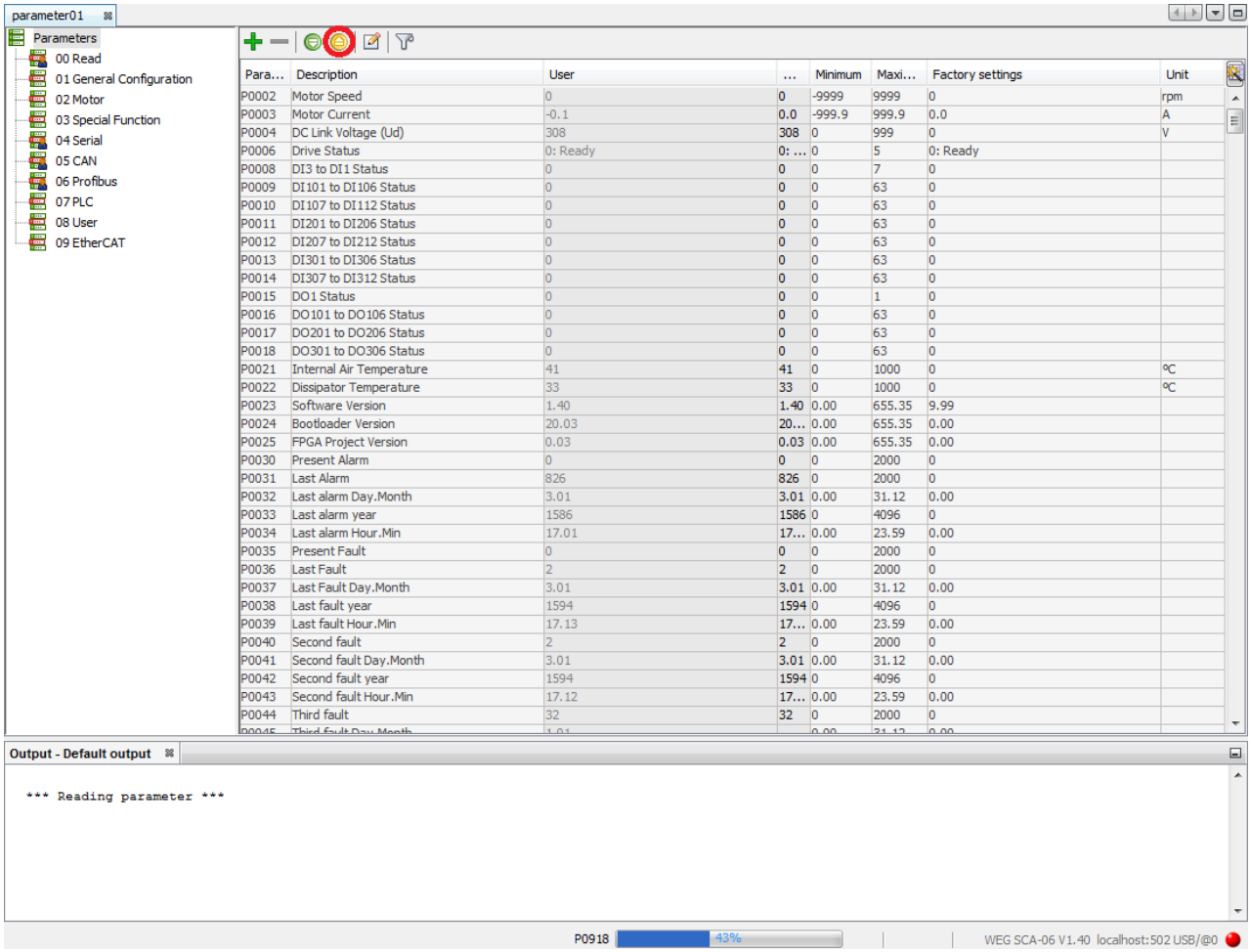
11.14.6.3 Read and Write of Parameters

There are 3 (three) ways to do the reading and writing of the parameters: by means of table, selection and group.

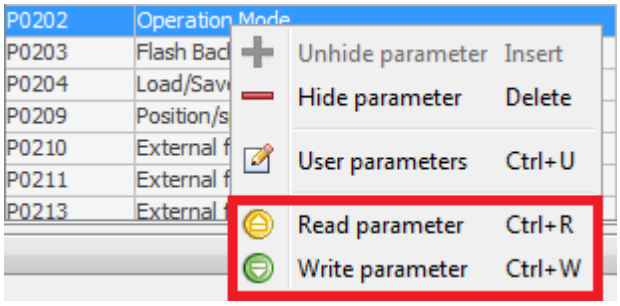
1. **Table writing**. The table writing command will send all visible parameters on the equipment screen. If and error occurs during the sending of some specific parameter, a message will be shown on the output window informing the error. It is important to notice that only visible parameters will be sent; therefore, it is necessary attention to which node of the group of parameters tree you are viewing. Example: If you wish to write all of them without filtering per group, just select the tree root.



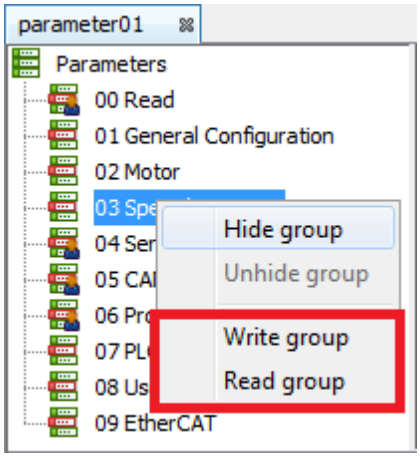
2. **Table reading**. The table reading command will read all the parameters of the equipment. If a error occurs during the reading of some specific parameter, a message will be shown on the output window informing the error. It is important to notice that only visible parameters will be read; therefore, it is necessary attention to which node of the group of parameters tree you are viewing. Example: If you wish to read all of them without filtering per group, just select the tree root.



3. **Reading/writing of specific parameters**. In order to read/write one or more specific parameters, just select them on the table, right click and choose the desired option: read or write parameter.



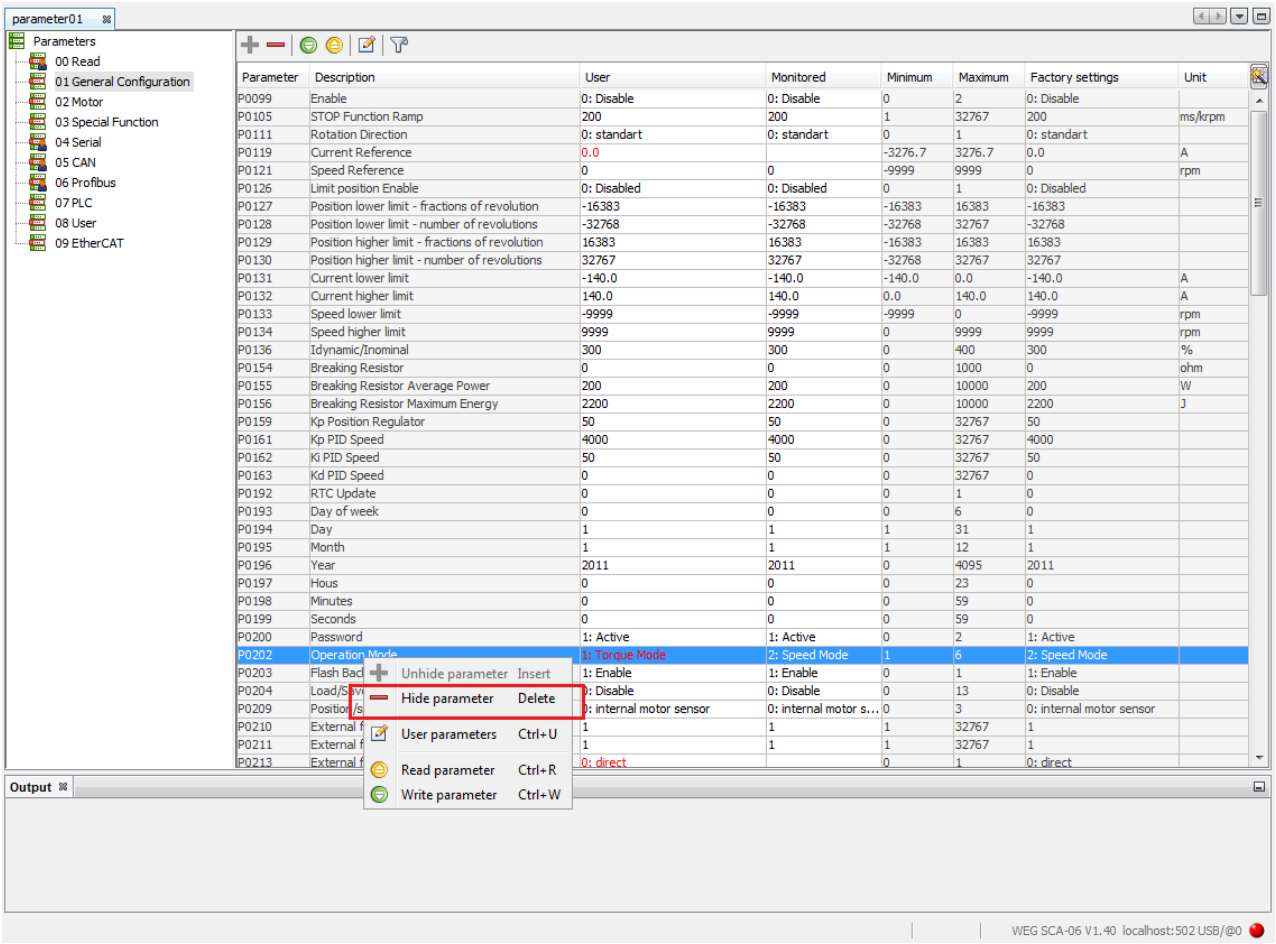
4. **Reading/writing of group of parameters**. In order to read/write only one group of parameters, just select it on the group tree, right click and choose the desired option: read or write group.



11.14.6.4 Hide/Unhide Parameters and Group of Parameters_2

The parameter can be hidden/unhidden in two ways: individually or in group.

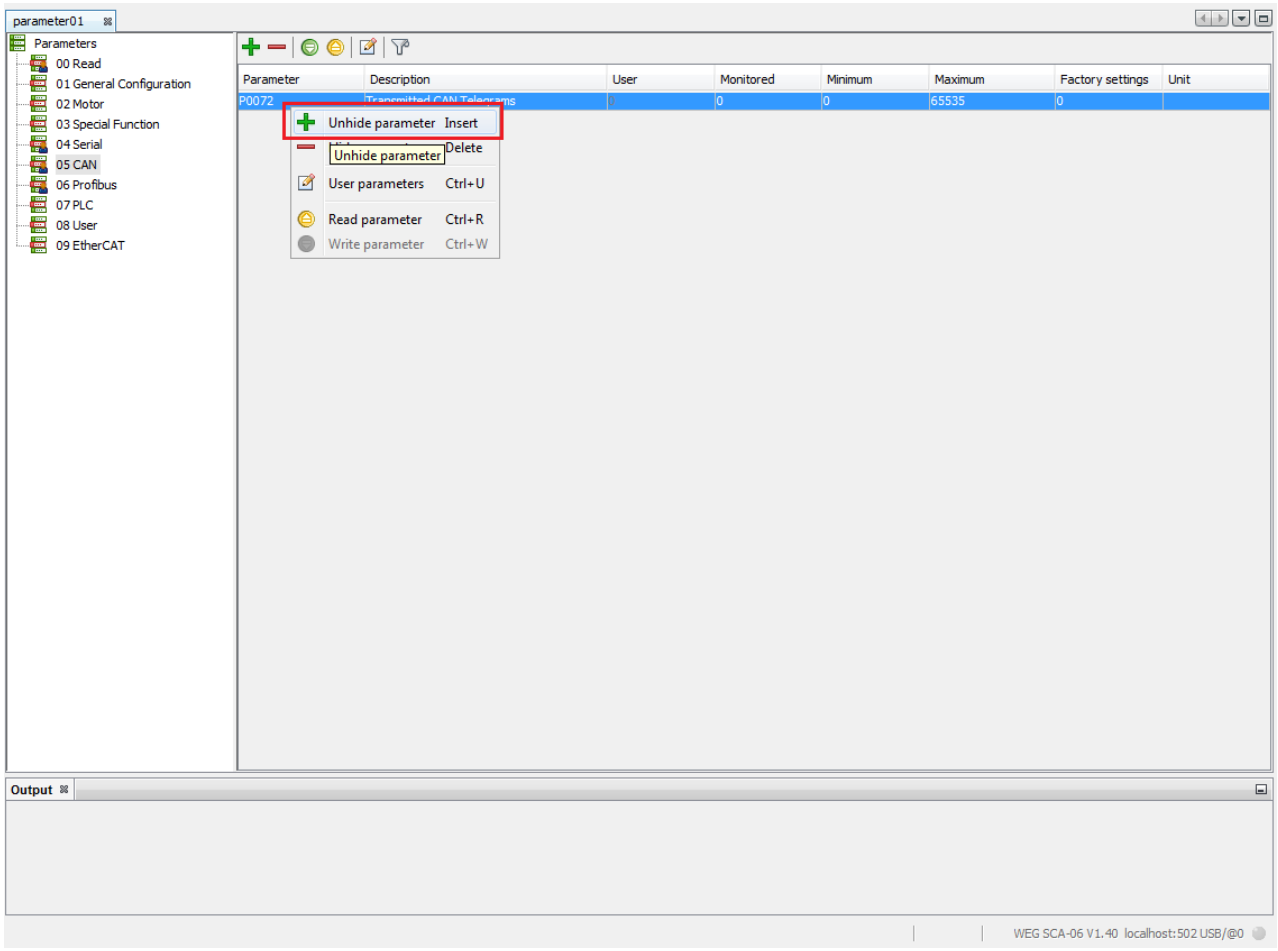
1. **Hide parameters**. In order to hide a parameter individually, just right click on the desired parameters and select the **Hide Parameter** option. You can also press the **Delete** key.

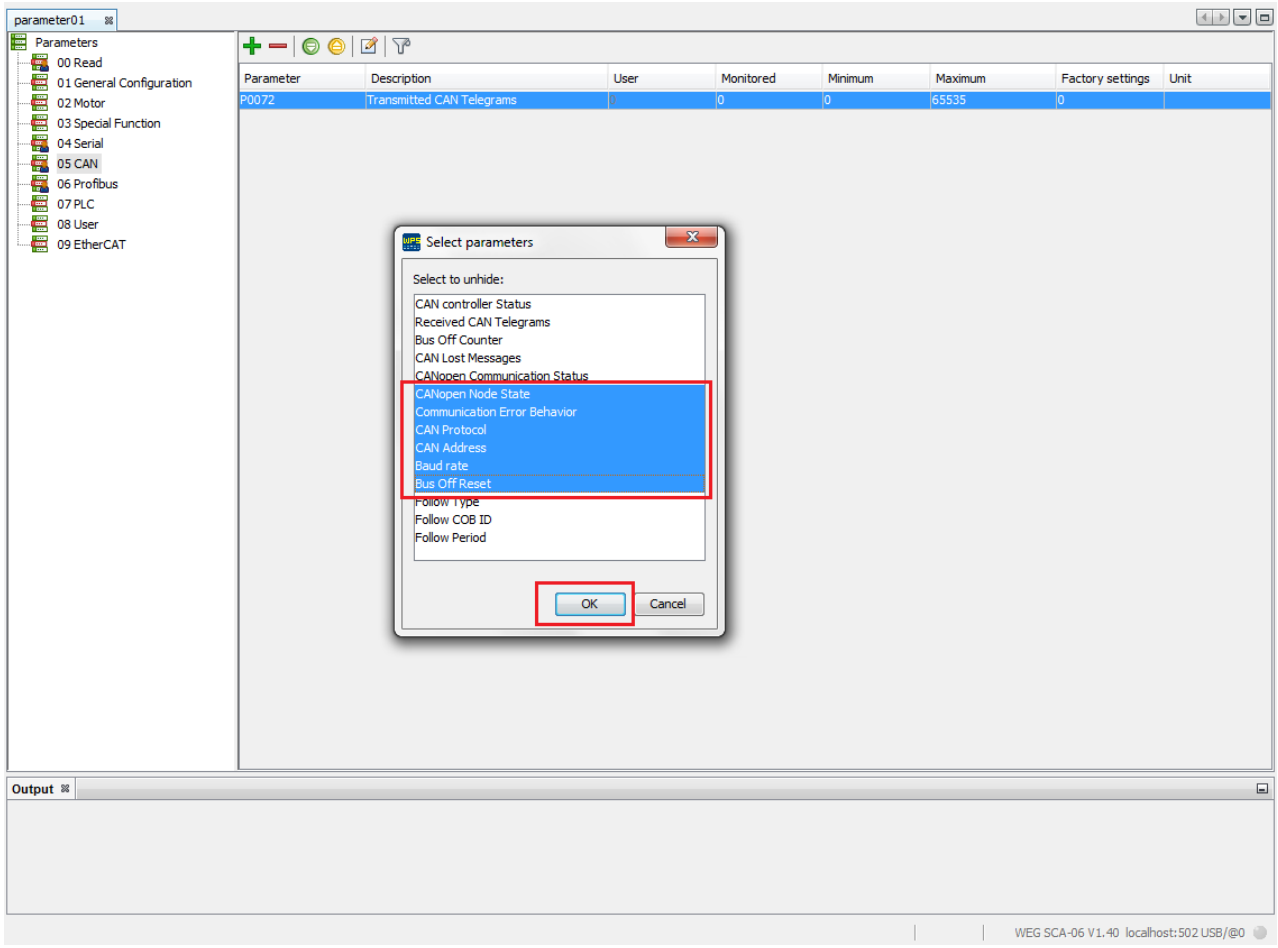


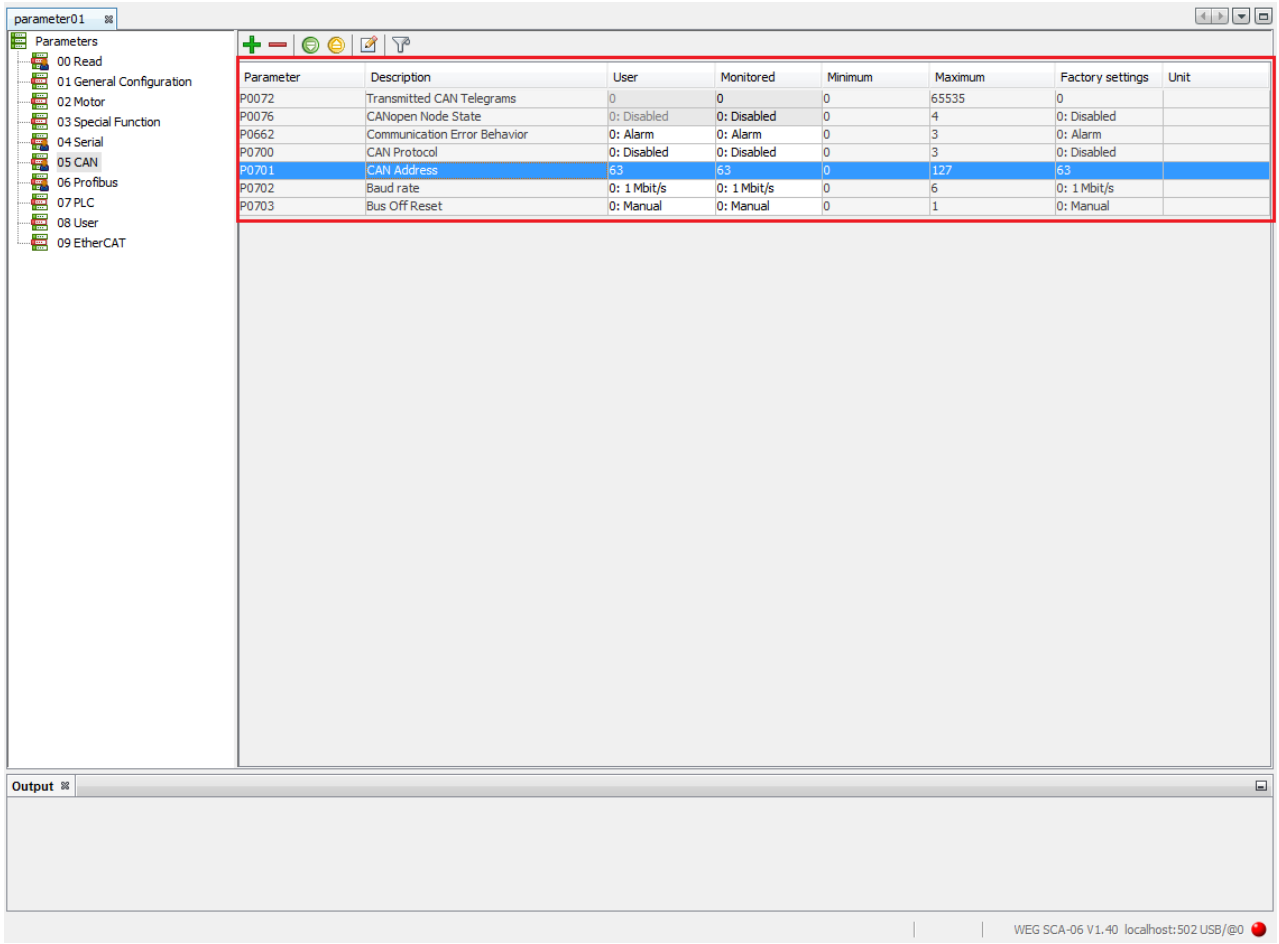
Parameter	Description	User	Monitored	Minimum	Maximum	Factory settings	Unit
P0099	Enable	0: Disable	0: Disable	0	2	0: Disable	
P0105	STOP Function Ramp	200	200	1	32767	200	ms/krpm
P0111	Rotation Direction	0: standart	0: standart	0	1	0: standart	
P0119	Current Reference	0.0		-3276.7	3276.7	0.0	A
P0121	Speed Reference	0	0	-9999	9999	0	rpm
P0126	Limit position Enable	0: Disabled	0: Disabled	0	1	0: Disabled	
P0127	Position lower limit - fractions of revolution	-16383	-16383	-16383	16383	-16383	
P0128	Position lower limit - number of revolutions	-32768	-32768	-32768	32767	-32768	
P0129	Position higher limit - fractions of revolution	16383	16383	-16383	16383	16383	
P0130	Position higher limit - number of revolutions	32767	32767	-32768	32767	32767	
P0131	Current lower limit	-140.0	-140.0	-140.0	0.0	-140.0	A
P0132	Current higher limit	140.0	140.0	0.0	140.0	140.0	A
P0133	Speed lower limit	-9999	-9999	-9999	0	-9999	rpm
P0134	Speed higher limit	9999	9999	0	9999	9999	rpm
P0136	Idynamic/Inominal	300	300	0	400	300	%
P0154	Breaking Resistor	0	0	0	1000	0	ohm
P0155	Breaking Resistor Average Power	200	200	0	10000	200	W
P0156	Breaking Resistor Maximum Energy	2200	2200	0	10000	2200	J
P0159	Kp Position Regulator	50	50	0	32767	50	
P0161	Kp PID Speed	4000	4000	0	32767	4000	
P0162	Ki PID Speed	50	50	0	32767	50	
P0163	Kd PID Speed	0	0	0	32767	0	
P0192	RTC Update	0	0	0	1	0	
P0193	Day of week	0	0	0	6	0	
P0194	Day	1	1	1	31	1	
P0195	Month	1	1	1	12	1	
P0196	Year	2011	2011	0	4095	2011	
P0197	Hous	0	0	0	23	0	
P0198	Minutes	0	0	0	59	0	
P0199	Seconds	0	0	0	59	0	
P0200	Password	1: Active	1: Active	0	2	1: Active	
P0202	Operation Mode	1: Torque Mode	2: Speed Mode	1	6	2: Speed Mode	
P0203	Flash Bac	1: Enable	1: Enable	0	1	1: Enable	
P0204	Load/S	0: Disable	0: Disable	0	13	0: Disable	
P0209	Position/s	0: internal motor sensor	0: internal motor s...	0	3	0: internal motor sensor	
P0210	External f	1	1	1	32767	1	
P0211	External f	1	1	1	32767	1	
P0213	External f	0: direct		0	1	0: direct	

2. **Unhide Parameters**. In order to show hidden parameters, right click and choose the **Unhide Parameters**

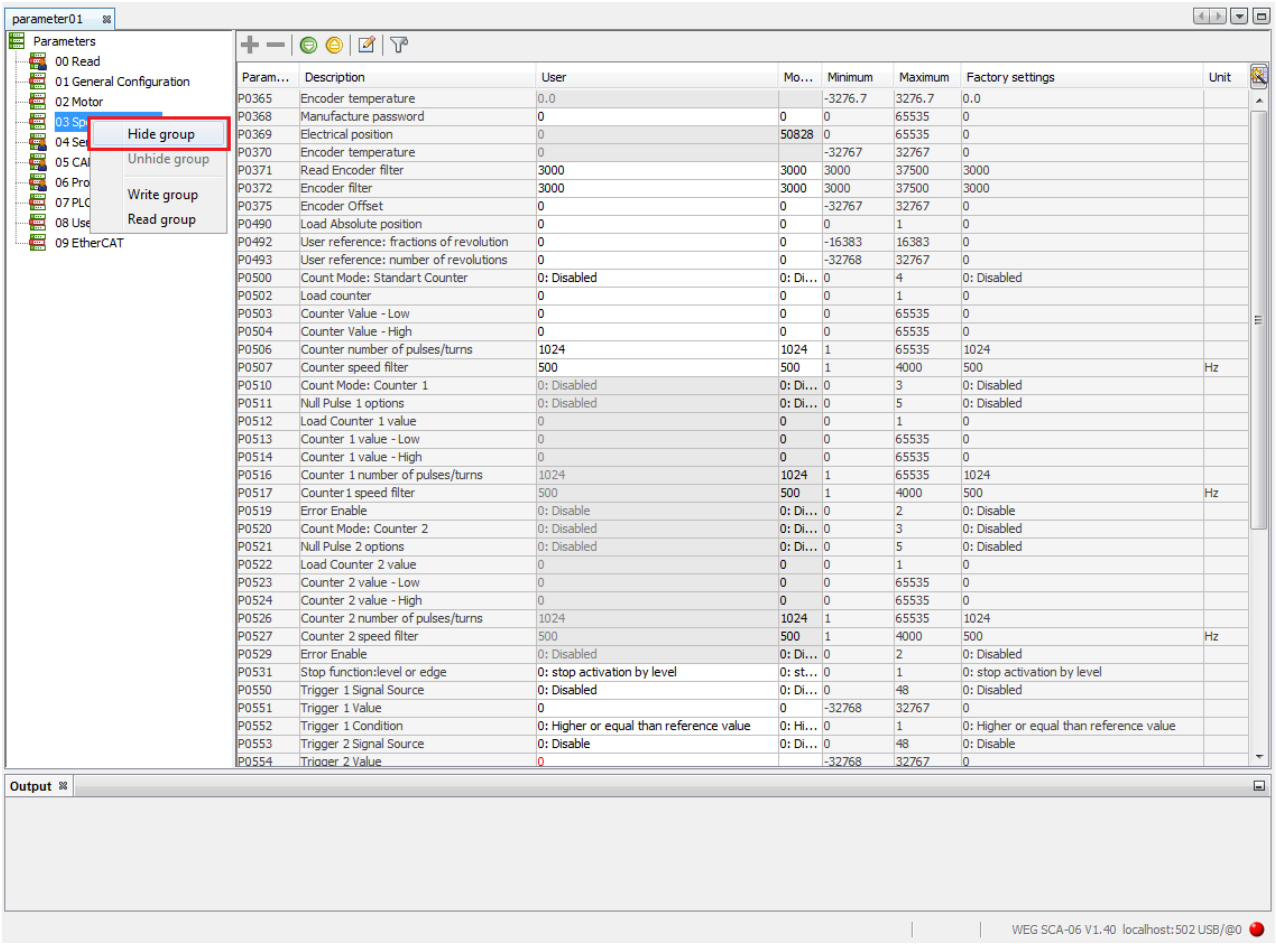
or press the **Insert** key. Then, a window will open and show the hidden parameters. Now, you just have to select the desired parameters and confirm.
Note: The parameters shown on this new window are only those which belong to the current filter according to the selection on the parameter group tree. In the figures below, the CAN group is selected; that means that only the hidden parameters of this group will be shown.

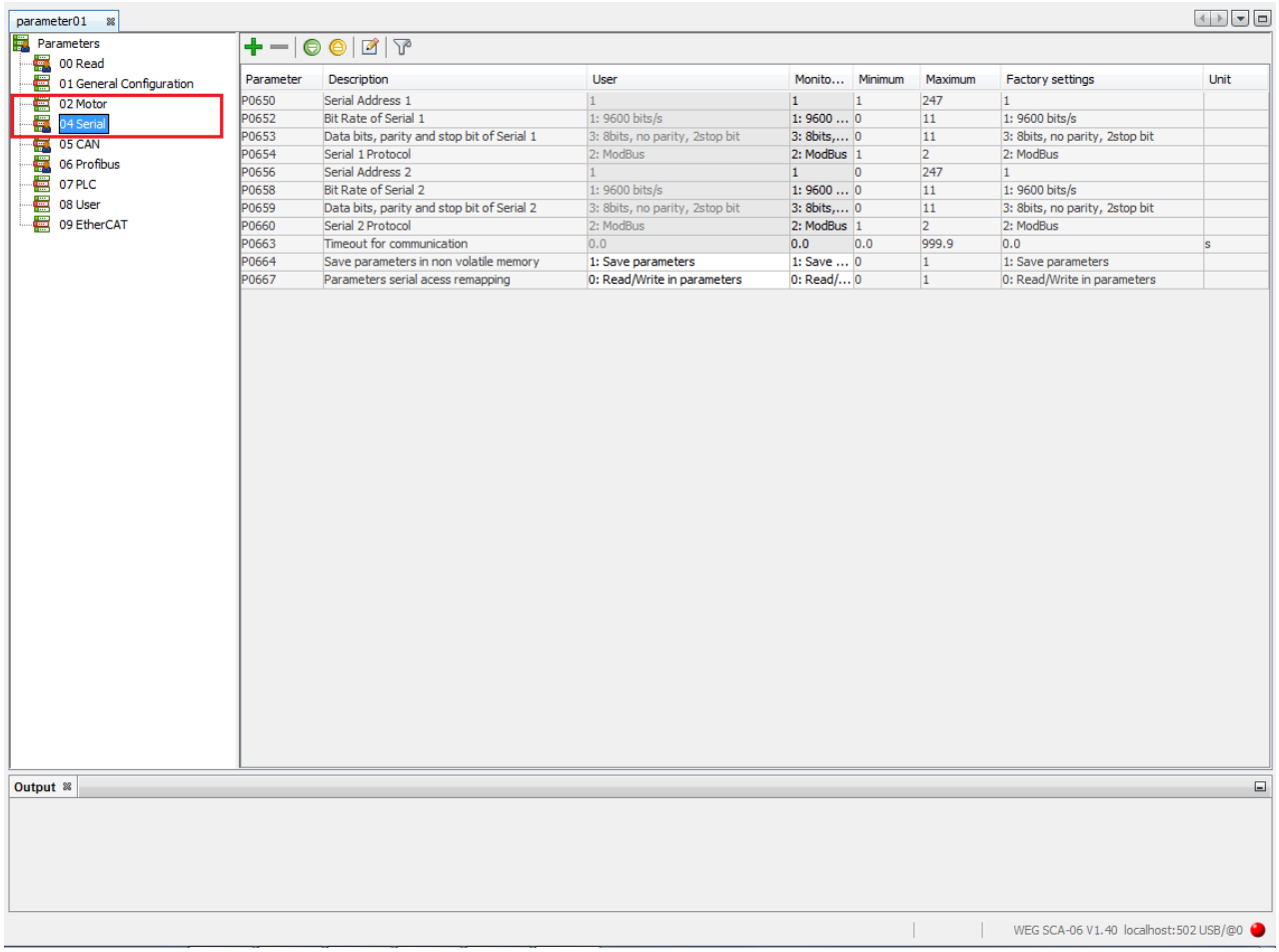



parameter01
Parameters
00 Read
01 General Configuration
02 Motor
03 Special Function
04 Serial
05 CAN
06 Profibus
07 PLC
08 User
09 EtherCAT
Parameter
Description
User
Monitored
Minimum
Maximum
Factory settings
Unit
P0072
Transmitted CAN Telegrams
0
0
65535
0
Select parameters
Select to unhide:
CAN controller Status
Received CAN Telegrams
Bus Off Counter
CAN Lost Messages
CANopen Communication Status
CANopen Node State
Communication Error Behavior
CAN Protocol
CAN Address
Baud rate
Bus Off Reset
Follow Type
Follow COB ID
Follow Period
OK
Cancel
Output
WEG SCA-06 V1.40 localhost:502 USB/@0

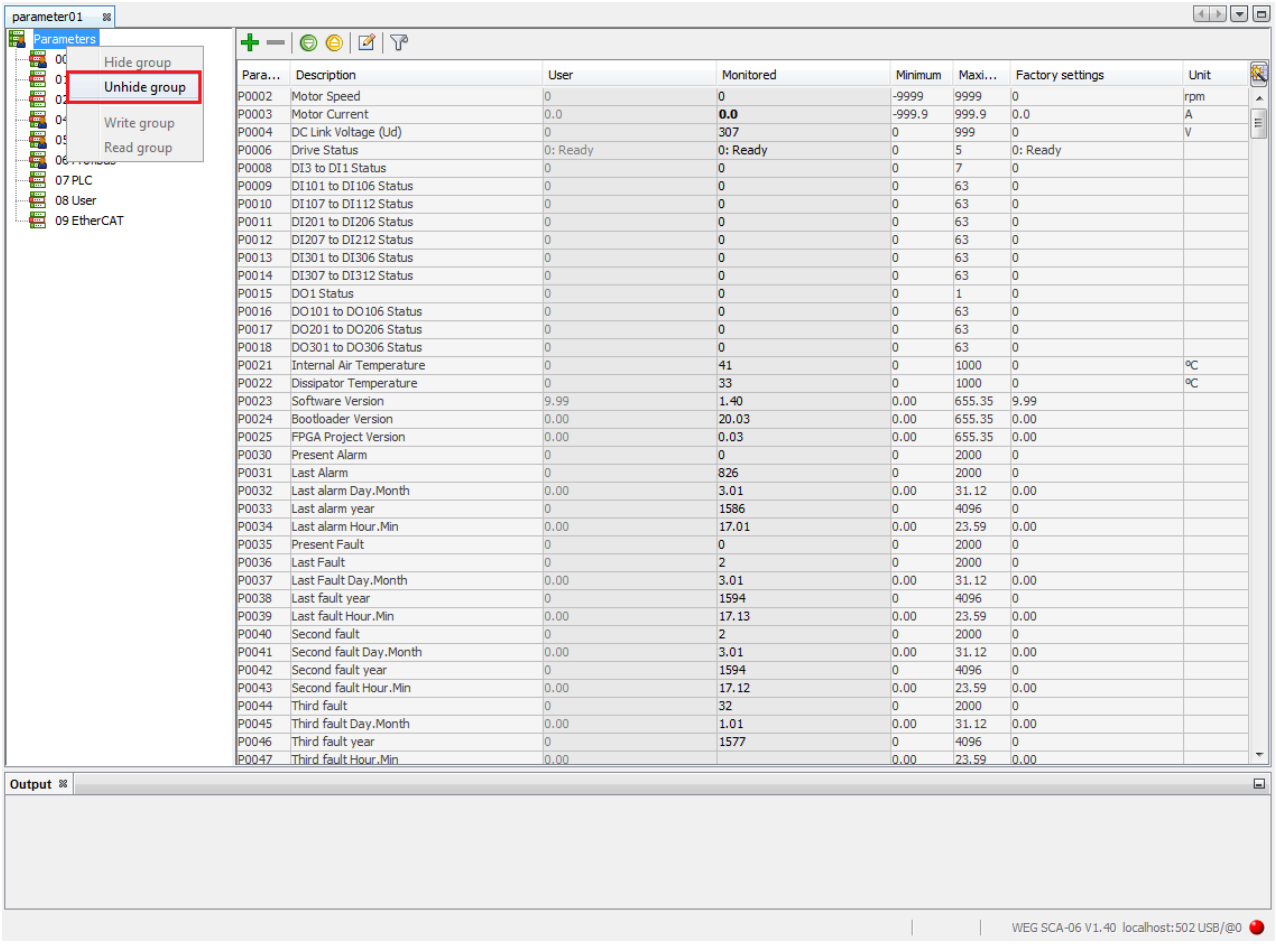


3. **Hide Group of Parameters**. In order to hide a group of parameters, just select the group on the tree and use the **Hide Group** option.

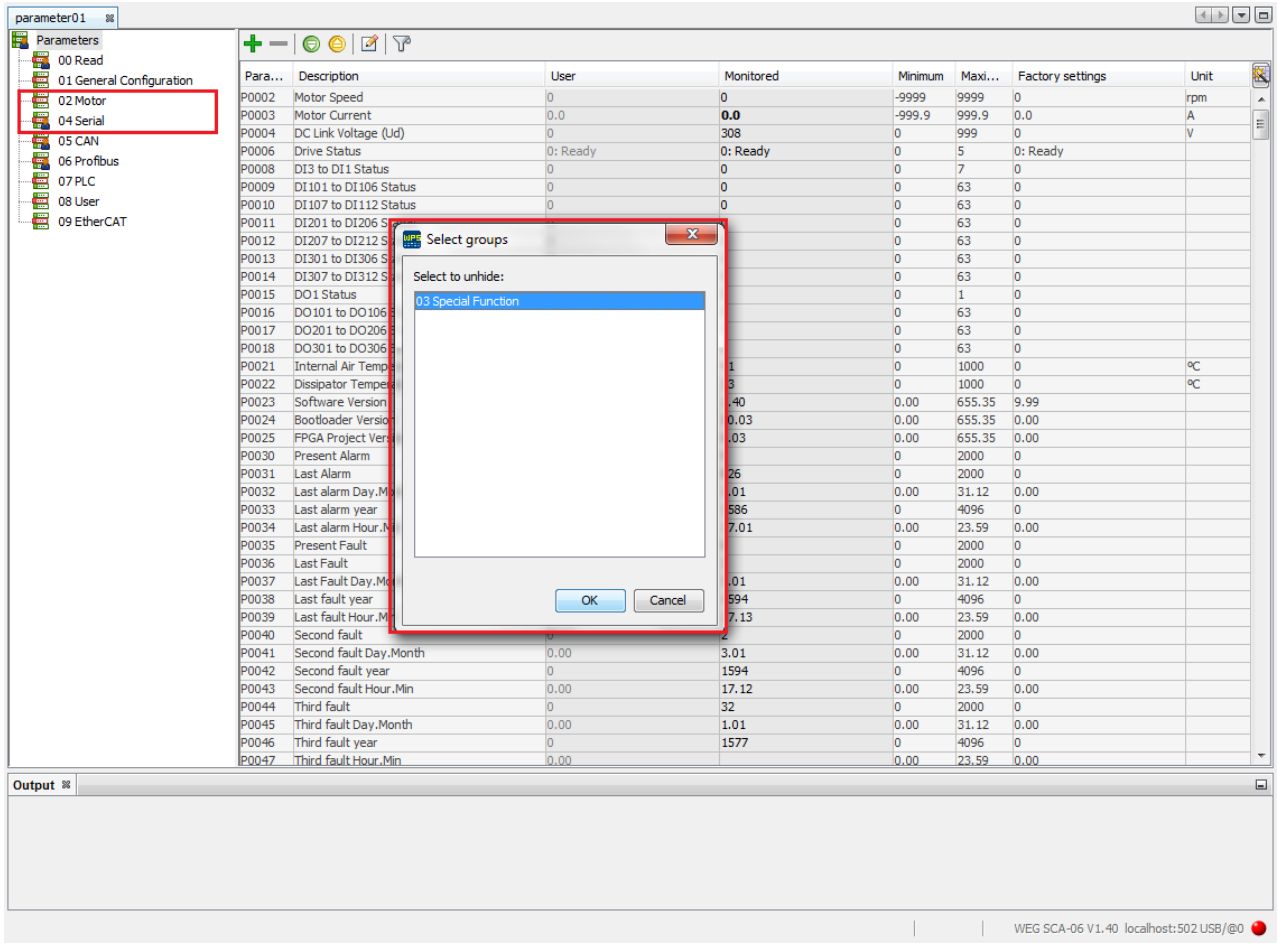

parameter01
Parameters
00 Read
01 General Configuration
02 Motor
03 Sp
04 Se
05 CA
06 Pro
07 PLC
08 Use
09 EtherCAT
Hide group
Unhide group
Write group
Read group
Param...
Description
User
Mo...
Minimum
Maximum
Factory settings
Unit
P0365 Encoder temperature 0.0 -3276.7 3276.7 0.0
P0368 Manufacture password 0 0 0 65535 0
P0369 Electrical position 0 50828 0 65535 0
P0370 Encoder temperature 0 -32767 32767 0
P0371 Read Encoder filter 3000 3000 3000 37500 3000
P0372 Encoder filter 3000 3000 3000 37500 3000
P0375 Encoder Offset 0 0 -32767 32767 0
P0490 Load Absolute position 0 0 0 1 0
P0492 User reference: fractions of revolution 0 0 -16383 16383 0
P0493 User reference: number of revolutions 0 0 -32768 32767 0
P0500 Count Mode: Standart Counter 0: Disabled 0: Di... 0 4 0: Disabled
P0502 Load counter 0 0 0 1 0
P0503 Counter Value - Low 0 0 0 65535 0
P0504 Counter Value - High 0 0 0 65535 0
P0506 Counter number of pulses/turns 1024 1024 1 65535 1024
P0507 Counter speed filter 500 500 1 4000 500 Hz
P0510 Count Mode: Counter 1 0: Disabled 0: Di... 0 3 0: Disabled
P0511 Null Pulse 1 options 0: Disabled 0: Di... 0 5 0: Disabled
P0512 Load Counter 1 value 0 0 0 1 0
P0513 Counter 1 value - Low 0 0 0 65535 0
P0514 Counter 1 value - High 0 0 0 65535 0
P0516 Counter 1 number of pulses/turns 1024 1024 1 65535 1024
P0517 Counter1 speed filter 500 500 1 4000 500 Hz
P0519 Error Enable 0: Disable 0: Di... 0 2 0: Disable
P0520 Count Mode: Counter 2 0: Disabled 0: Di... 0 3 0: Disabled
P0521 Null Pulse 2 options 0: Disabled 0: Di... 0 5 0: Disabled
P0522 Load Counter 2 value 0 0 0 1 0
P0523 Counter 2 value - Low 0 0 0 65535 0
P0524 Counter 2 value - High 0 0 0 65535 0
P0526 Counter 2 number of pulses/turns 1024 1024 1 65535 1024
P0527 Counter 2 speed filter 500 500 1 4000 500 Hz
P0529 Error Enable 0: Disabled 0: Di... 0 2 0: Disabled
P0531 Stop function:level or edge 0: stop activation by level 0: st... 0 1 0: stop activation by level
P0550 Trigger 1 Signal Source 0: Disabled 0: Di... 0 48 0: Disabled
P0551 Trigger 1 Value 0 0 -32768 32767 0
P0552 Trigger 1 Condition 0: Higher or equal than reference value 0: Hi... 0 1 0: Higher or equal than reference value
P0553 Trigger 2 Signal Source 0: Disable 0: Di... 0 48 0: Disable
P0554 Trigger 2 Value 0 -32768 32767 0
Output
WEG SCA-06 V1.40 localhost:502 USB/@0



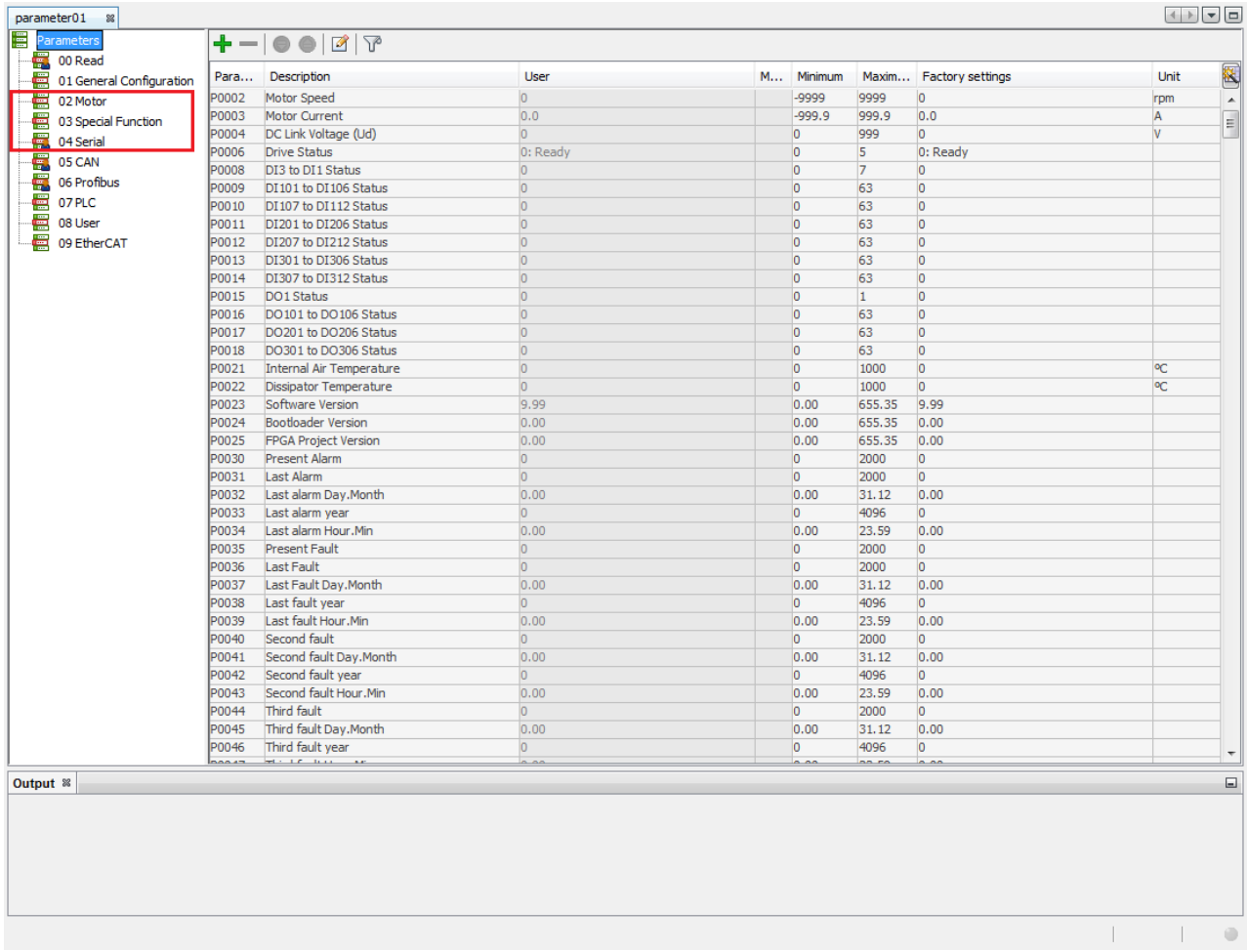
4. **Unhide Group of Parameters**. In order to show a hidden group of parameters, just select the root of the group tree and select the **Unhide Group** option. A window will open showing the groups that are hidden; then just select the group you wish to unhide.



Para...	Description	User	Monitored	Minimum	Maxi...	Factory settings	Unit
P0002	Motor Speed	0	0	-9999	9999	0	rpm
P0003	Motor Current	0.0	**0.0**	-999.9	999.9	0.0	A
P0004	DC Link Voltage (Ud)	0	307	0	999	0	V
P0006	Drive Status	0: Ready	0: Ready	0	5	0: Ready	
P0008	DI3 to DI1 Status	0	0	0	7	0	
P0009	DI101 to DI106 Status	0	0	0	63	0	
P0010	DI107 to DI112 Status	0	0	0	63	0	
P0011	DI201 to DI206 Status	0	0	0	63	0	
P0012	DI207 to DI212 Status	0	0	0	63	0	
P0013	DI301 to DI306 Status	0	0	0	63	0	
P0014	DI307 to DI312 Status	0	0	0	63	0	
P0015	DO1 Status	0	0	0	1	0	
P0016	DO101 to DO106 Status	0	0	0	63	0	
P0017	DO201 to DO206 Status	0	0	0	63	0	
P0018	DO301 to DO306 Status	0	0	0	63	0	
P0021	Internal Air Temperature	0	41	0	1000	0	°C
P0022	Dissipator Temperature	0	33	0	1000	0	°C
P0023	Software Version	9.99	1.40	0.00	655.35	9.99	
P0024	Bootloader Version	0.00	20.03	0.00	655.35	0.00	
P0025	FPGA Project Version	0.00	0.03	0.00	655.35	0.00	
P0030	Present Alarm	0	0	0	2000	0	
P0031	Last Alarm	0	826	0	2000	0	
P0032	Last alarm Day.Month	0.00	3.01	0.00	31.12	0.00	
P0033	Last alarm year	0	1586	0	4096	0	
P0034	Last alarm Hour.Min	0.00	17.01	0.00	23.59	0.00	
P0035	Present Fault	0	0	0	2000	0	
P0036	Last Fault	0	2	0	2000	0	
P0037	Last Fault Day.Month	0.00	3.01	0.00	31.12	0.00	
P0038	Last fault year	0	1594	0	4096	0	
P0039	Last fault Hour.Min	0.00	17.13	0.00	23.59	0.00	
P0040	Second fault	0	2	0	2000	0	
P0041	Second fault Day.Month	0.00	3.01	0.00	31.12	0.00	
P0042	Second fault year	0	1594	0	4096	0	
P0043	Second fault Hour.Min	0.00	17.12	0.00	23.59	0.00	
P0044	Third fault	0	32	0	2000	0	
P0045	Third fault Day.Month	0.00	1.01	0.00	31.12	0.00	
P0046	Third fault year	0	1577	0	4096	0	
P0047	Third fault Hour.Min	0.00		0.00	23.59	0.00	

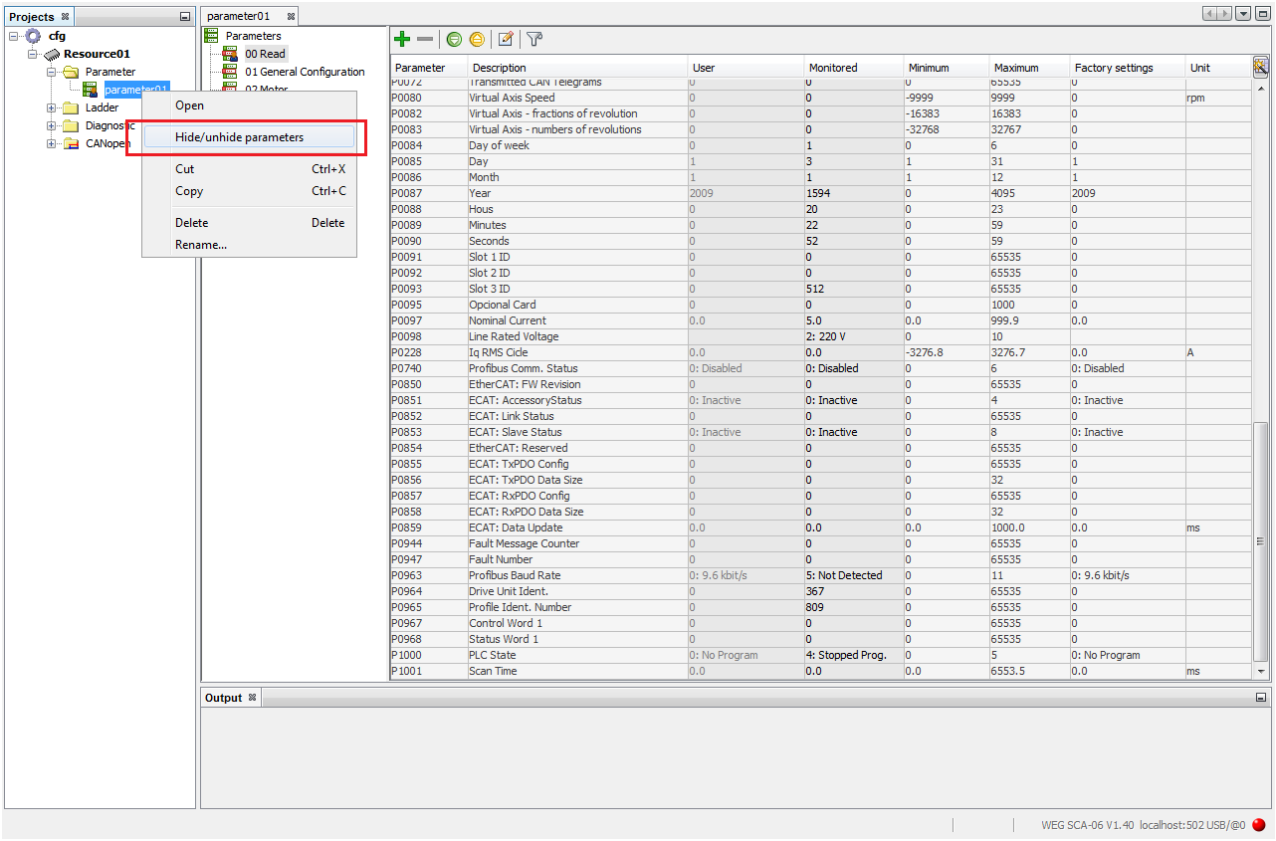


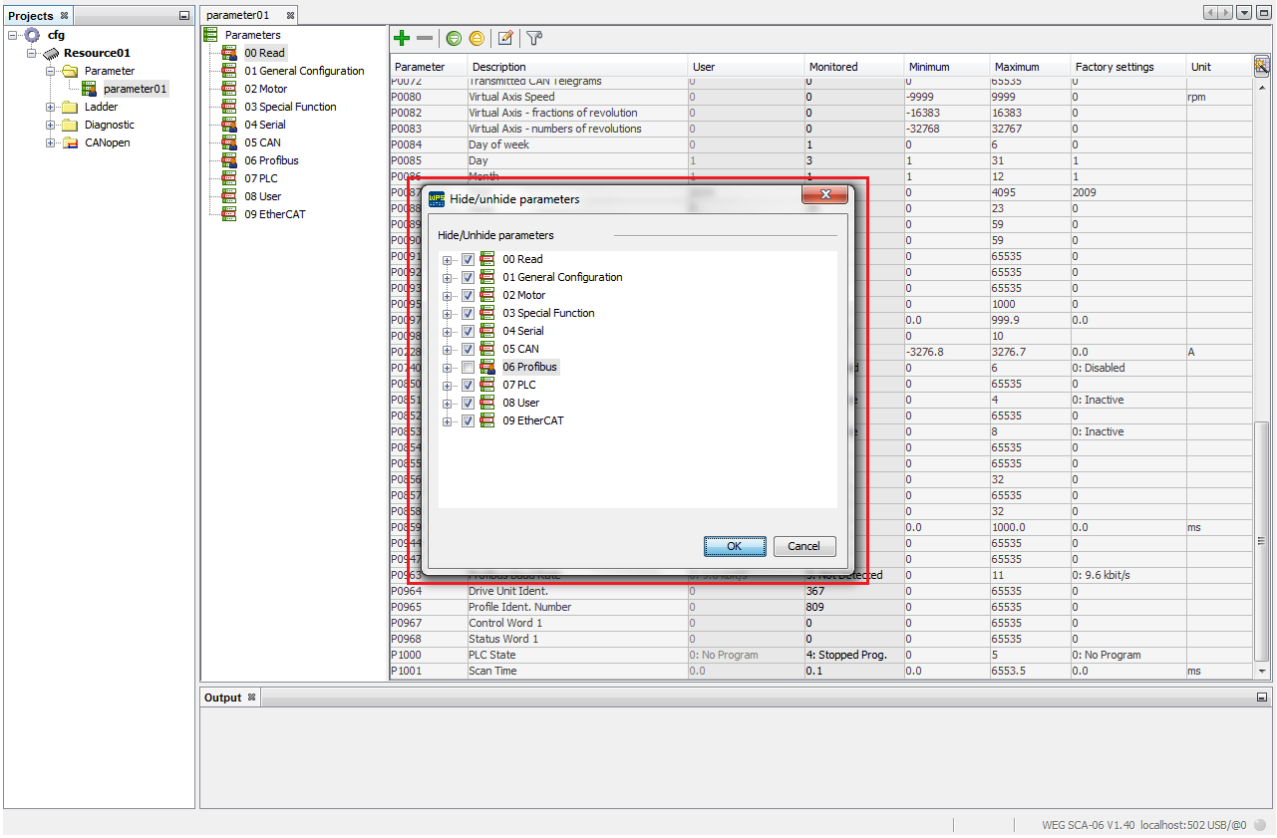
Para...	Description	User	Monitored	Minimum	Maxi...	Factory settings	Unit
P0002	Motor Speed	0	0	-9999	9999	0	rpm
P0003	Motor Current	0.0	**0.0**	-999.9	999.9	0.0	A
P0004	DC Link Voltage (Ud)	0	308	0	999	0	V
P0006	Drive Status	0: Ready	0: Ready	0	5	0: Ready	
P0008	DI3 to DI1 Status	0	0	0	7	0	
P0009	DI101 to DI106 Status	0	0	0	63	0	
P0010	DI107 to DI112 Status	0	0	0	63	0	
P0011	DI201 to DI206 S			0	63	0	
P0012	DI207 to DI212 S			0	63	0	
P0013	DI301 to DI306 S			0	63	0	
P0014	DI307 to DI312 S			0	63	0	
P0015	DO1 Status			0	1	0	
P0016	DO101 to DO106			0	63	0	
P0017	DO201 to DO206			0	63	0	
P0018	DO301 to DO306			0	63	0	
P0021	Internal Air Temp		1	0	1000	0	°C
P0022	Dissipator Temper		3	0	1000	0	°C
P0023	Software Version		.40	0.00	655.35	9.99	
P0024	Bootloader Versio		0.03	0.00	655.35	0.00	
P0025	FPGA Project Ver		.03	0.00	655.35	0.00	
P0030	Present Alarm			0	2000	0	
P0031	Last Alarm		26	0	2000	0	
P0032	Last alarm Day.M		.01	0.00	31.12	0.00	
P0033	Last alarm year		586	0	4096	0	
P0034	Last alarm Hour.M		7.01	0.00	23.59	0.00	
P0035	Present Fault			0	2000	0	
P0036	Last Fault			0	2000	0	
P0037	Last Fault Day.M		.01	0.00	31.12	0.00	
P0038	Last fault year		594	0	4096	0	
P0039	Last fault Hour.M		7.13	0.00	23.59	0.00	
P0040	Second fault		2	0	2000	0	
P0041	Second fault Day.Month	0.00	3.01	0.00	31.12	0.00	
P0042	Second fault year	0	1594	0	4096	0	
P0043	Second fault Hour.Min	0.00	17.12	0.00	23.59	0.00	
P0044	Third fault	0	32	0	2000	0	
P0045	Third fault Day.Month	0.00	1.01	0.00	31.12	0.00	
P0046	Third fault year	0	1577	0	4096	0	
P0047	Third fault Hour.Min	0.00		0.00	23.59	0.00	



Para...	Description	User	M...	Minimum	Maxim...	Factory settings	Unit
P0002	Motor Speed	0		-9999	9999	0	rpm
P0003	Motor Current	0.0		-999.9	999.9	0.0	A
P0004	DC Link Voltage (Ud)	0		0	999	0	V
P0006	Drive Status	0: Ready		0	5	0: Ready	
P0008	DI3 to DI1 Status	0		0	7	0	
P0009	DI101 to DI106 Status	0		0	63	0	
P0010	DI107 to DI112 Status	0		0	63	0	
P0011	DI201 to DI206 Status	0		0	63	0	
P0012	DI207 to DI212 Status	0		0	63	0	
P0013	DI301 to DI306 Status	0		0	63	0	
P0014	DI307 to DI312 Status	0		0	63	0	
P0015	DO1 Status	0		0	1	0	
P0016	DO101 to DO106 Status	0		0	63	0	
P0017	DO201 to DO206 Status	0		0	63	0	
P0018	DO301 to DO306 Status	0		0	63	0	
P0021	Internal Air Temperature	0		0	1000	0	°C
P0022	Dissipator Temperature	0		0	1000	0	°C
P0023	Software Version	9.99		0.00	655.35	9.99	
P0024	Bootloader Version	0.00		0.00	655.35	0.00	
P0025	FPGA Project Version	0.00		0.00	655.35	0.00	
P0030	Present Alarm	0		0	2000	0	
P0031	Last Alarm	0		0	2000	0	
P0032	Last alarm Day.Month	0.00		0.00	31.12	0.00	
P0033	Last alarm year	0		0	4096	0	
P0034	Last alarm Hour.Min	0.00		0.00	23.59	0.00	
P0035	Present Fault	0		0	2000	0	
P0036	Last Fault	0		0	2000	0	
P0037	Last Fault Day.Month	0.00		0.00	31.12	0.00	
P0038	Last fault year	0		0	4096	0	
P0039	Last fault Hour.Min	0.00		0.00	23.59	0.00	
P0040	Second fault	0		0	2000	0	
P0041	Second fault Day.Month	0.00		0.00	31.12	0.00	
P0042	Second fault year	0		0	4096	0	
P0043	Second fault Hour.Min	0.00		0.00	23.59	0.00	
P0044	Third fault	0		0	2000	0	
P0045	Third fault Day.Month	0.00		0.00	31.12	0.00	
P0046	Third fault year	0		0	4096	0	

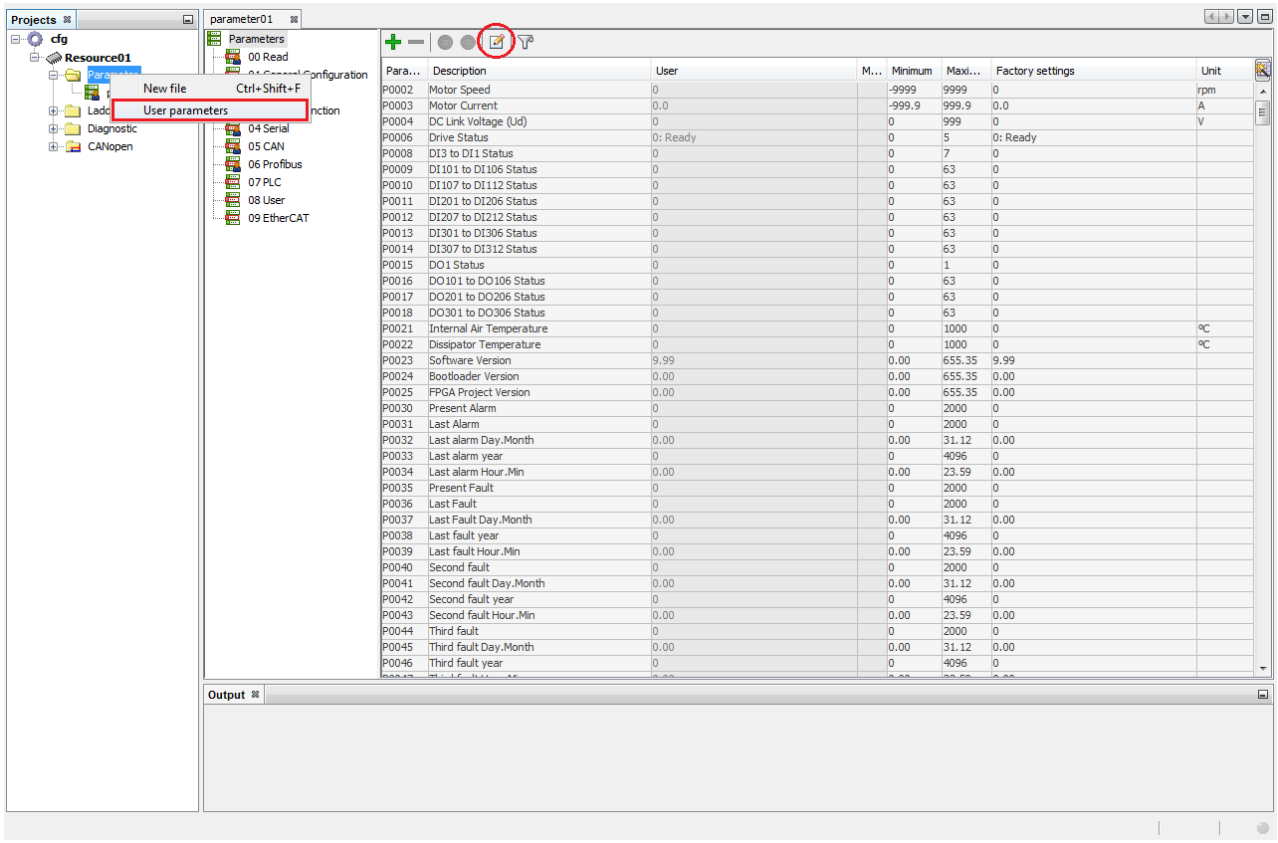
5. **Hide and Show Parameters and Groups of Parameters**. By means of this option, you have full control of the parameters and groups of parameters. It is possible to hide and unhide individual parameters, multiple parameters, individual groups and multiple groups in the same action.


Projects
cfg
Resource01
Parameter
parameter01
Ladder
Diagnostic
CANopen
Open
Hide/unhide parameters
Cut Ctrl+X
Copy Ctrl+C
Delete Delete
Rename...
parameter01
Parameters
00 Read
01 General Configuration
Parameter
Description
User
Monitored
Minimum
Maximum
Factory settings
Unit
P0080 Virtual Axis Speed
P0082 Virtual Axis - fractions of revolution
P0083 Virtual Axis - numbers of revolutions
P0084 Day of week
P0085 Day
P0086 Month
P0087 Year
P0088 Hous
P0089 Minutes
P0090 Seconds
P0091 Slot 1 ID
P0092 Slot 2 ID
P0093 Slot 3 ID
P0095 Opcional Card
P0097 Nominal Current
P0098 Line Rated Voltage
P0228 Iq RMS Cicle
P0740 Profibus Comm. Status
P0850 EtherCAT: FW Revision
P0851 ECAT: AccessoryStatus
P0852 ECAT: Link Status
P0853 ECAT: Slave Status
P0854 EtherCAT: Reserved
P0855 ECAT: TxPDO Config
P0856 ECAT: TxPDO Data Size
P0857 ECAT: RxPDO Config
P0858 ECAT: RxPDO Data Size
P0859 ECAT: Data Update
P0944 Fault Message Counter
P0947 Fault Number
P0963 Profibus Baud Rate
P0964 Drive Unit Ident.
P0965 Profile Ident. Number
P0967 Control Word 1
P0968 Status Word 1
P1000 PLC State
P1001 Scan Time
Output
WEG SCA-06 V1.40 localhost:502 USB/@0


Projects
cfg
Resource01
Parameter
parameter01
Ladder
Diagnostic
CANopen
parameter01
Parameters
00 Read
01 General Configuration
02 Motor
03 Special Function
04 Serial
05 CAN
06 Profibus
07 PLC
08 User
09 EtherCAT
Hide/unhide parameters
Hide/Unhide parameters
OK
Cancel
Output
WEG SCA-06 V1.40 localhost:502 USB/@0

11.14.6.5 User Parameters

In order to open the configuration screen of the user parameters, just click on the **User Parameters** option on the Parameter node of the project tree or click on the icon indicated on the tool bar of the parameter file.



Configuration Table.

On the user parameter configuration table, it is possible to define several attributes to the parameters, such as description, minimum and maximum values, unit, digits, data type, etc.

NOTE!
These settings will be automatically displayed in the parameter table. However, to be sent to the device, you need to *download* the resource.

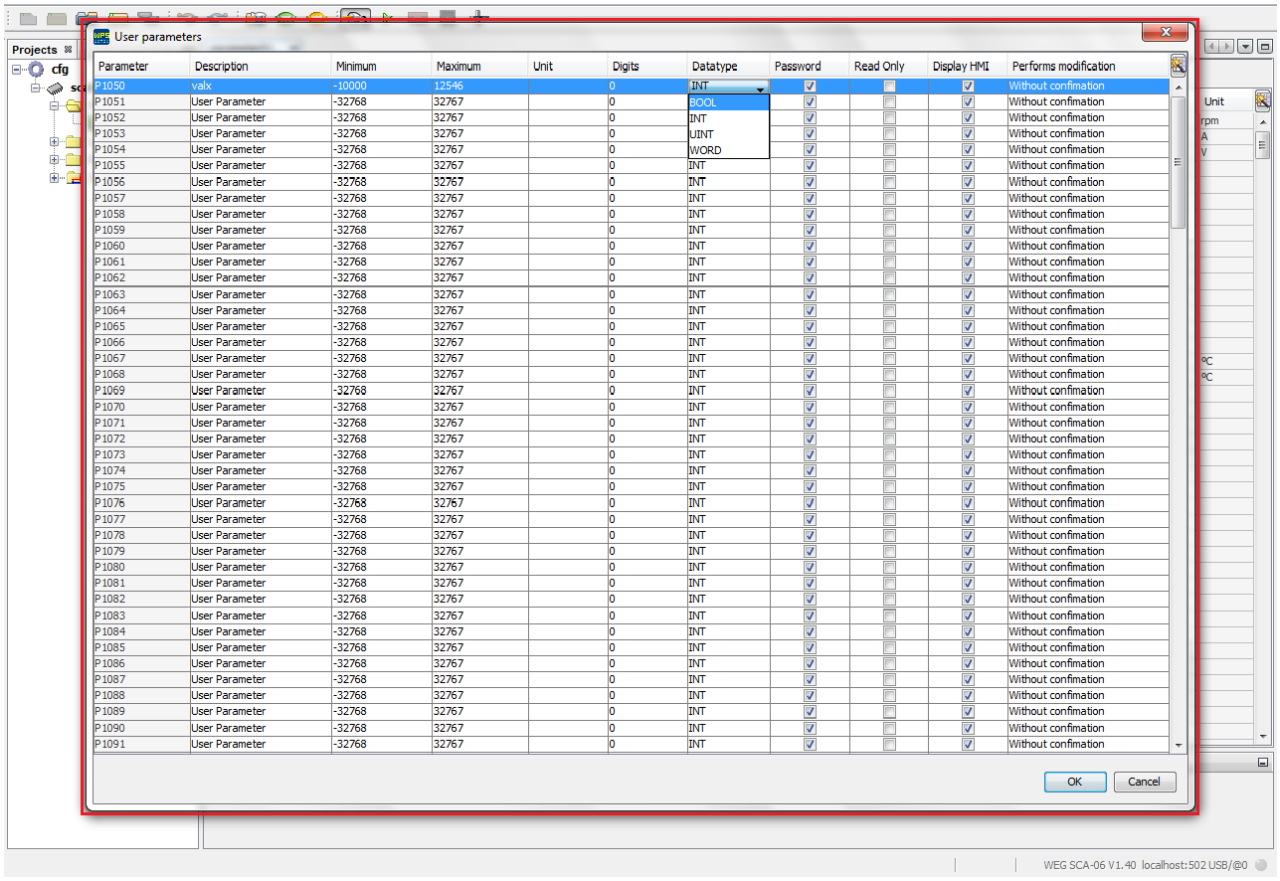


Table fields:

- Parameter: User parameter identification.
- Description: Description of the user parameter in the parameter table. On devices that have text-based HMIs, the description is sent to the machine and displayed on the HMI.
- Minimum: Minimum input value for parameter.
- Maximum: Maximum input value for parameter.
- Unit: Unit displayed on the device's HMI.
- Default: Value loaded when restore factory default is selected.
- Retentive: Retain value after rebooting devices.
- Hexadecimal: Displays the value in hexadecimal.
- Digits: Number of decimal digits for displaying value.
- Datatype: Parameter datatype used by the ladder application.
- Password: Enables password request by changing parameter value.

- Read only: It does not allow the writing of values in the parameter by the communication network or the HMI. Writing is done only by the ladder application.

- Display HMI: Displays the parameter in the HMI.

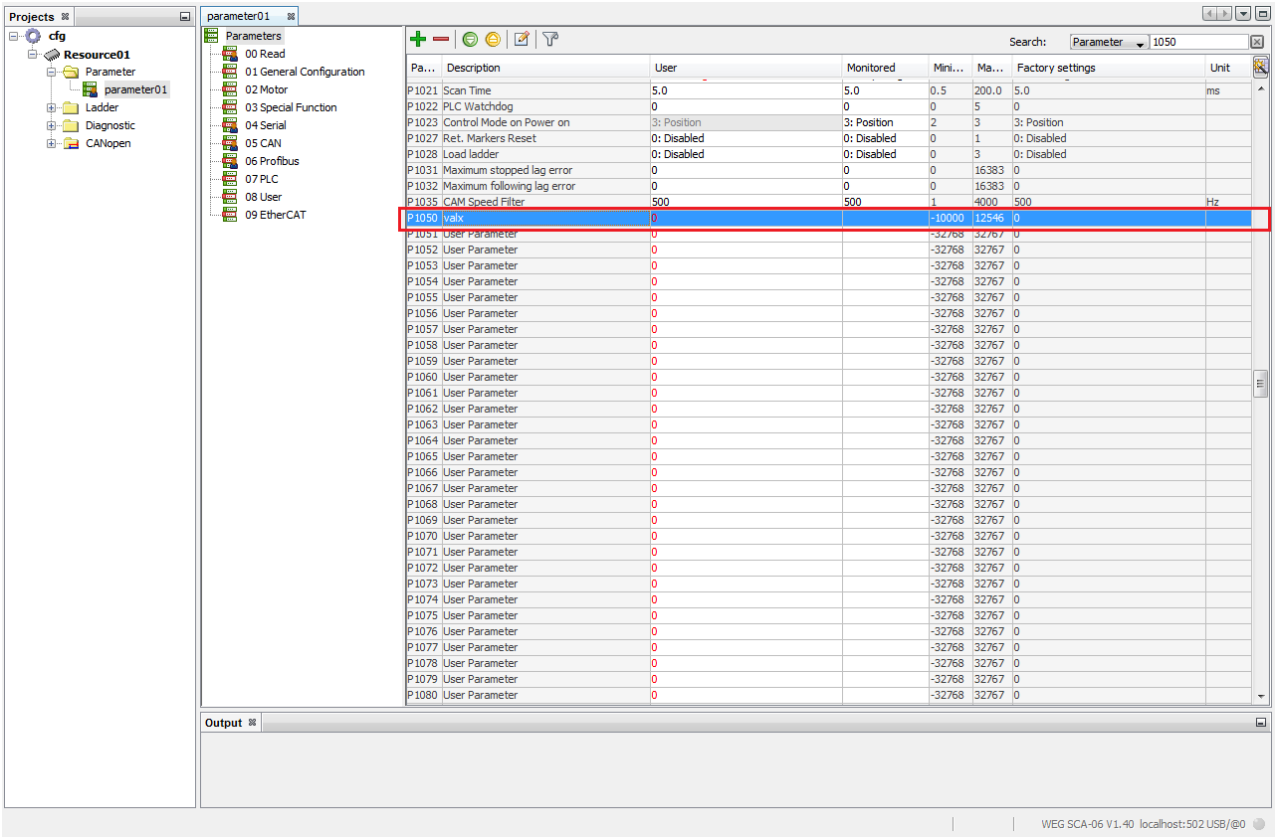
- Performs modification: Confirmation options when changing the parameter:
o No confirmation: Does not prompt for confirmation when changing parameter.
o With confirmation and engine stopped: Request confirmation and allow change only with engine stopped.
o With confirmation: Prompt for confirmation when changing parameter.

- Stopped motor: Perform change only with motor stopped.

- Help: On devices that have text-based HMI, you can edit a help text for the parameter.

View the user parameter

In the parameter table, the user parameters will be shown as they are configured on the configuration screen.



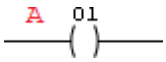
11.14.7Ladder

11.14.7.1 Coil

11.14.7.1.1 DIRECTCOIL

Logical block used to assign direct values of the output variables.

Ladder Representation



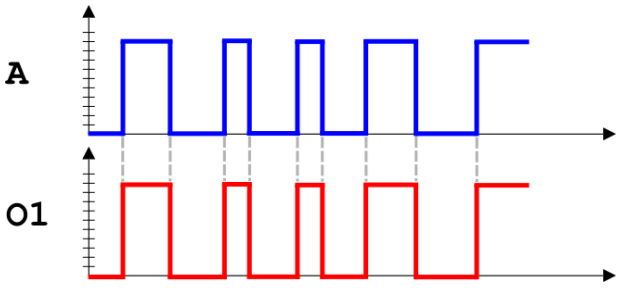
Block Structure

Variable Type	Name	Data Type	Description
VAR_OUTPUT	O1	BOOL	Block log output

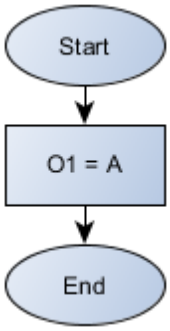
Operation

The block transfers the value of A for the memory address corresponding to O1.

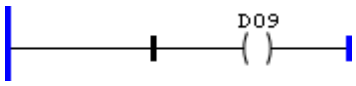
Diagram



Block Flowchart



Example

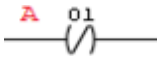


The above example keeps the digital output DO9 permanently connected, because the value of A in this case is the value of the left bus which is always considered high logic level (TRUE).

11.14.7.1.2 INVERTEDCOIL

Logical block used for assigning values denied to output variables.

Ladder Representation



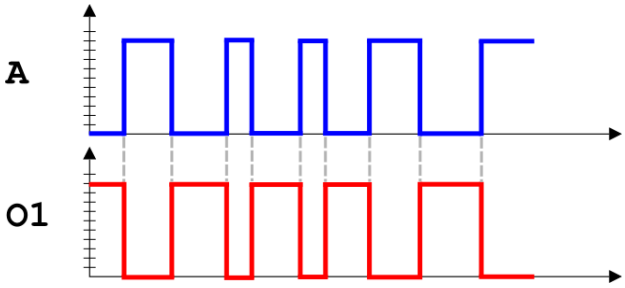
Block Structure

Variable Type	Name	Data Type	Description
VAR_OUTPUT	O1	BOOL	Block log output

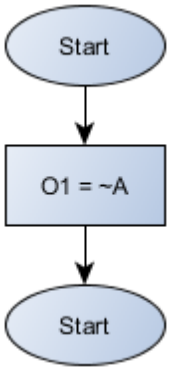
Operation

The block transfers the denied value of A for the memory address corresponding to O1.

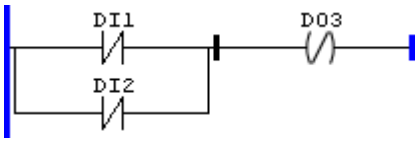
Diagram



Block Flowchart



Example

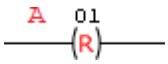


The above example disables the digital output DO3 when some of the digital inputs DI1 and DI2 are with FALSE value. When both inputs are with a TRUE value, DO3 activates.

11.14.7.1.3 RESETCOIL

Logical block used for indefinite disabling of output variables.

Ladder Representation



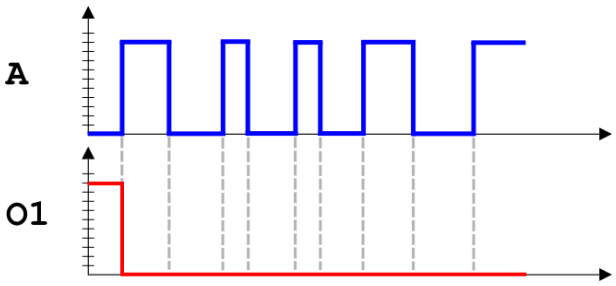
Block Structure

Variable Type	Name	Data Type	Description
VAR_OUTPUT	O1	BOOL	Block log output

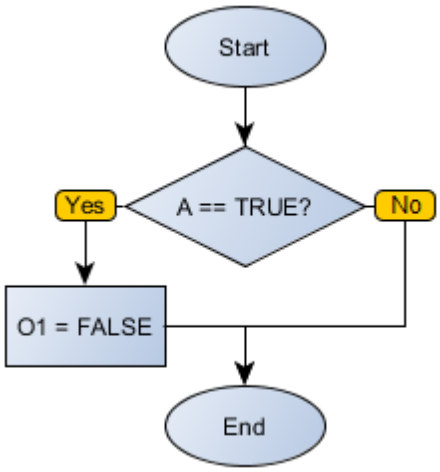
Operation

When identifying a TRUE value in A, this block transfers a FALSE value to the memory address corresponding to O1.
When identifying a FALSE value in A, this block performs no operation.

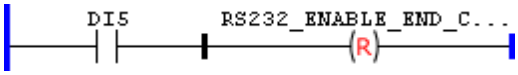
Diagram



Block Flowchart



Example

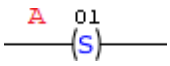


The example above activates permanently the system control marker that enables end-of-message character in RS232 communication to identify a TRUE level at the digital input DI5.

11.14.7.1.4 SETCOIL

Logical block used for indefinite enabling of output variables.

Ladder Representation



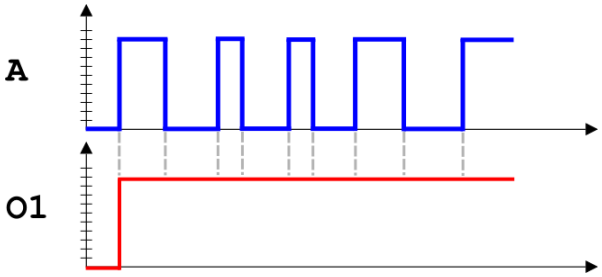
Block Structure

Variable Type	Name	Data Type	Description
VAR_OUTPUT	O1	BOOL	Block log output

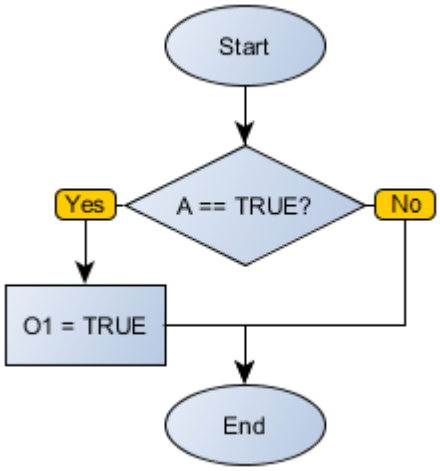
Operation

When identifying a TRUE value in A, this block transfers the value of A for the memory address corresponding to O1.
When identifying a FALSE value in A, this block performs no operation.

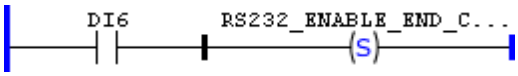
Diagram



Block Flowchart



Example

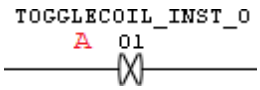


The example above activates permanently the system control marker that enables end-of-message character in RS232 communication to identify a TRUE level at the digital input DI6.

11.14.7.1.5 TOGGLECOIL

Logical block used for output variables alternance.

Ladder Representation



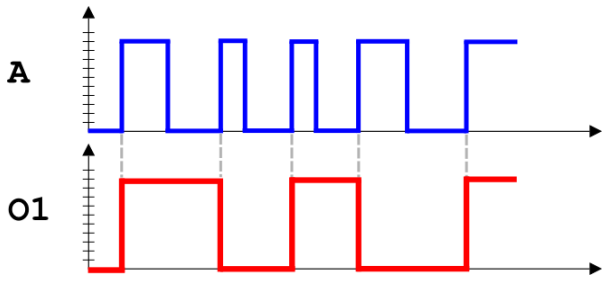
Block Structure

Variable Type	Name	Data Type	Description
VAR_OUTPUT	O1	BOOL	Block log output
VAR	TOGGLECOIL_INST_0	TOGGLECOIL	Instance of access to block structure

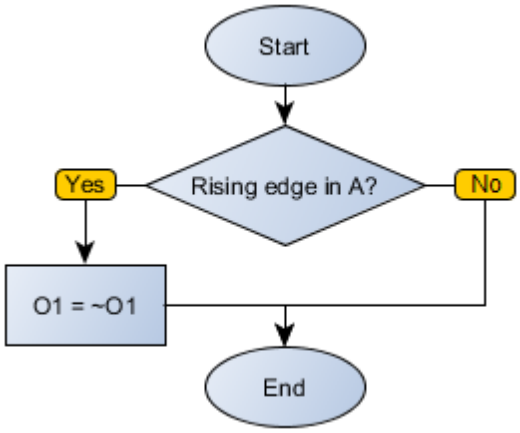
Operation

When identifying a transition from FALSE to TRUE (leading edge) on A, the block reverses the status of O1.

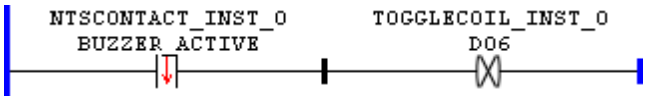
Diagram



Block Flowchart



Example



The above example inverts the state of the digital output DO6 to each disabling the internal buzzer.

11.14.7.2 Communication Network

11.14.7.2.1 Modbus RTU

11.14.7.2.1.1 Modbus RTU Overview

Operation in the Modbus RTU Network - Master Mode

The CFW300 allows operation as a master for the Modbus RTU network. For this operation, it is necessary to observe the following points:

- Only interface RS485 allows operation as a network master.
- It is necessary to program, in product configurations, the operation mode as "Master", besides the communication rate, parity, and stop bits, which must be the same for the whole equipment in the network.
- The Modbus RTU network master does not have an address, so the address configured in the CFW300 is not used.
- Sending and receiving telegrams via RS485 interface using the Modbus RTU is programmed by using blocks in Ladder programming language. It is necessary to know the available blocks and the Ladder programming software in order to be able to program the network master.

- The following functions are available for the sending of requisitions by the Modbus master:
  - Function 01: Read Coils
  - Function 02: Read Discrete Inputs
  - Function 03: Read Holding Registers
  - Function 04: Read Input Registers
  - Function 05: Write Single Coil
  - Function 06: Write Single Register
  - Function 15: Write Multiple Coils
  - Function 16: Write Multiple Registers

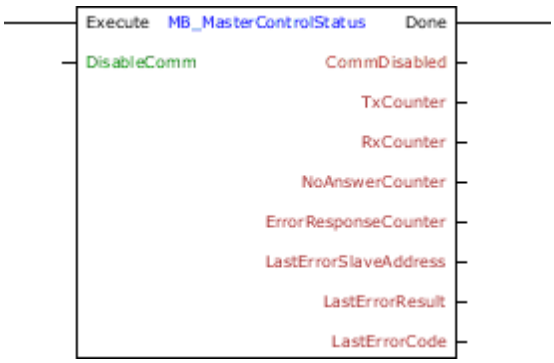
Blocks to program the master

In order to control and monitor the Modbus RTU communication using the CFW300, the following blocks were developed, and they must be used when programming in Ladder.

11.14.7.2.1.2 MB_MasterControlStatus

Block that allows monitoring various statuses of the Modbus RTU network master.

Ladder Representation



Block Structure

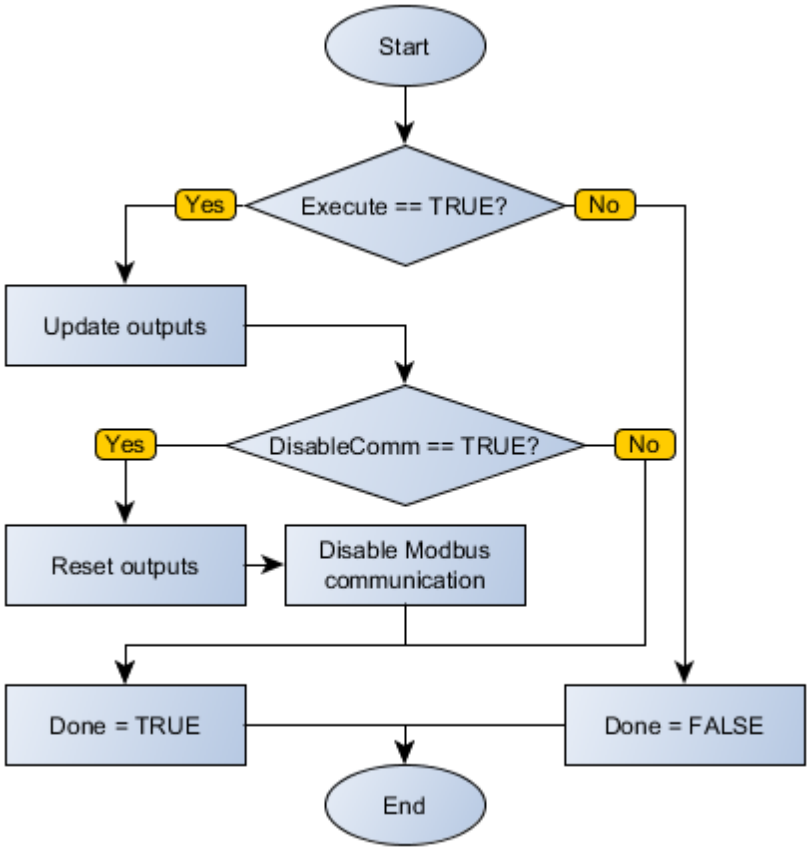
Variable Type	Name	Data Type	Description
VAR_INPUT	Execute	BOOL	Block enabling
	DisableComm	BOOL	Disables Modbus RTU communication
VAR_OUTPUT	Done	BOOL	Output enabling
	CommDisabled	BOOL	Disabled communication flag
	TxCounter	WORD UINT	Counter of requests sent
	RxCounter	WORD UINT	Counter of telegrams received
	NoAnswerCounter	WORD UINT	Counter of requests not answered
	ErrorResponseCounter	WORD UINT	Counter of responses received with error information
	LastErrorSlaveAddress	BYTE USINT	Slave address in which the last communication error was detected
	LastErrorResult	BYTE USINT	Operation result of the last communication error received (0 = No error) (4 – Response Timeout) (5 = Slave returned error)
	LastErrorCode	BYTE USINT	Code of the last communication error received

Operation

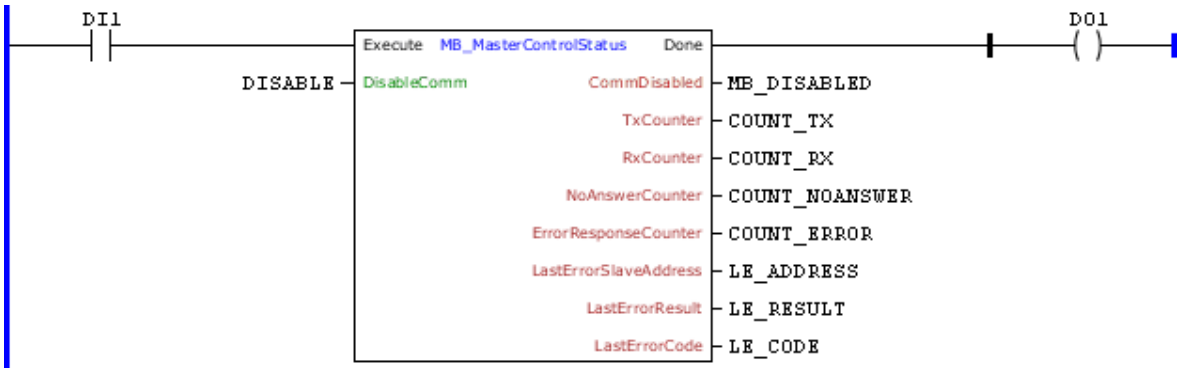
This block remains active while Execute is at TRUE level, updating its outputs according to the monitoring of the master and input requests. When Execute receives FALSE level, the inputs are ignored and the outputs are zeroed. The Done output receives TRUE level when Execute has TRUE level and block finished its execution.

A TRUE level DisableComm disables the Modbus RTU communication and resets the status counters and markers of the master. These markers and counters are displayed in the output block each having some data corresponding to its description. Their values are also cleared at shutdown of the master.

Block Flowchart



Example

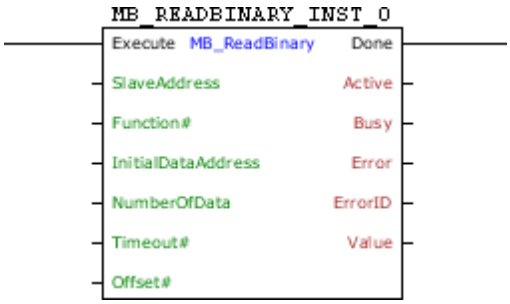


The example above requests status data of the Modbus RTU network master, and allows disabling communication through DISABLE. The block ends successfully, Done output is activated.

11.14.7.2.1.3 MB_ReadBinary

Block that performs a reading of up to 128 binary data (via Read Coils or Read Discrete Inputs) of a slave on the Modbus RTU network.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	Execute	BOOL	Block enabling
	SlaveAddress	BYTE	Slave address
	Function#	BYTE	Reading function code
	InitialDataAddress	WORD	Initial bit address of the data to be read
	NumberOfData	BYTE	Number of bits to be read (1 to 128)
	Timeout#	WORD	Maximum w aiting time for the slave response [ms]
	Offset#	BOOL	Offset Indication in InitialDataAddress, i.e., need to subtract 1 from this number
VAR_OUTPUT	Done	BOOL	Output enabling
	Active	BOOL	Aw aiting response flag
	Busy	BOOL	Flag indicating the RS485 interface is busy w ith another request
	Error	BOOL	Error in the execution flag
	ErrorID	BYTE	Identifier of the occurred error
	Value	BOOL	Variable that stores the received data
VAR	MB_READBINARY_INST_0	MB_READBINARY	Instance of access to block structure

Operation

When this block detects a leading edge on Execute, it checks whether the Modbus slave RTU in specified address in SlaveAddress is free to send data (Busy variable at FALSE level). If so, it sends the reading request of a number of bits indicated by NumberOfData in InitialDataAddress address using chosen function in Function# and sets the Active output, resetting it when receiving the response from the slave. The received data is stored in the Value variable. If the slave is not free, the block waits Busy go to FALSE level to resubmit the request.

NOTE!
If Execute goes to FALSE level and Busy is still at TRUE level, the request is canceled.

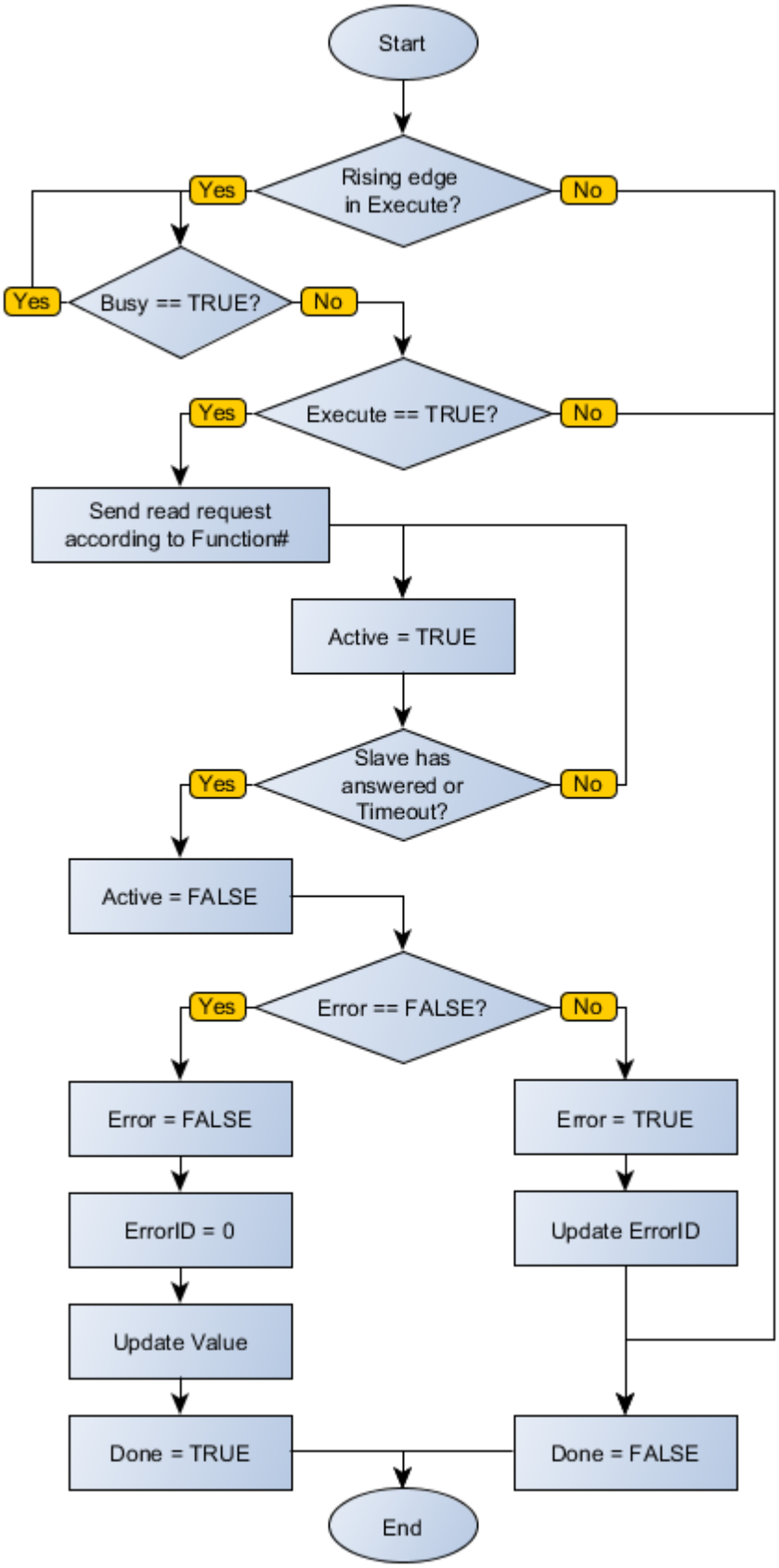
NOTE!
Value is an array of size equal to NumberOfData. It is important to check this compatibility not to generate errors in the block.

When Execute has FALSE value, Done remains FALSE. The Done output is only activated when the block finishes executing successfully, remaining at TRUE level until Execute receives FALSE.

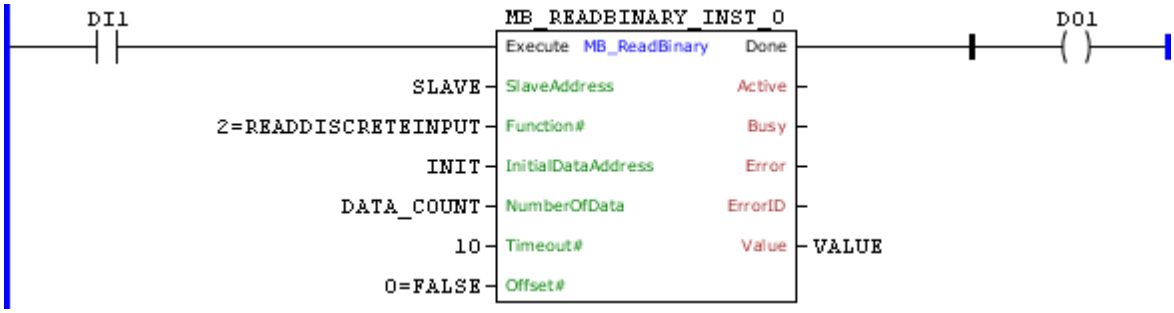
If there is any error in the execution, the Error output is enabled and ErrorID displays an error code according to the table below.

Code	Description
0	Executed successfully
1	Invalid input data
2	Master not enabled
4	Timeout in slave response
5	Slave returned error

Block Flowchart


Start
Rising edge in Execute?
Yes
No
Busy == TRUE?
Yes
No
Execute == TRUE?
Yes
No
Send read request according to Function#
Active = TRUE
Slave has answered or Timeout?
Yes
No
Active = FALSE
Error == FALSE?
Yes
No
Error = FALSE
ErrorID = 0
Update Value
Done = TRUE
Error = TRUE
Update ErrorID
Done = FALSE
End

Example

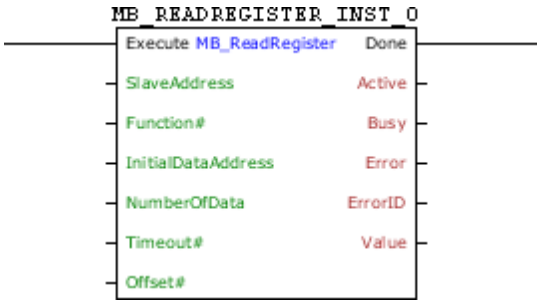


The above example requests reading of a number of binary data described by DATA_COUNT positioned in the INIT Modbus RTU slave of SLAVE address through the Read Discrete Input function. These data are forwarded to VALUE. The block ends successfully, Done output is activated.

11.14.7.2.1.4 MB_ReadRegister

Block that performs a reading of up to 64 16-bit registers (via Read Holding Registers or Read Input Registers) of a slave on the Modbus RTU network.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	Execute	BOOL	Block enabling
	SlaveAddress	BYTE	Slave address
	Function#	BYTE	Reading function code
	InitialDataAddress	WORD	Initial register address to be read
	NumberOfData	BYTE	Number of registers to be read (1 to 64)
	Timeout#	WORD	Maximum waiting time for the slave response [ms]
	Offset#	BOOL	Offset Indication in InitialDataAddress, i.e., need to subtract 1 from this number
VAR_OUTPUT	Done	BOOL	Output enabling
	Active	BOOL	Awaiting response flag
	Busy	BOOL	Flag indicating the RS485 interface is busy with another request
	Error	BOOL	Error in the execution flag
	ErrorID	BYTE	Identifier of the occurred error
	Value	BYTE SINT USINT WORD UINT INT DWORD UDINT DINT REAL	Variable that stores the received data
VAR	MB_READREGISTER_INST_0	MB_READREGISTER	Instance of access to block structure

Operation

When this block detects a leading edge on Execute, it checks whether the Modbus RTU slave in specified address in SlaveAddress is free to send data (Busy variable at FALSE level). If so, it sends the reading request of a number of registers indicated by NumberOfData in InitialDataAddress address using chosen function in Function# and sets the Active output, resetting them when receiving the response from the slave. The received data is stored in the Value variable. If the slave is not free, the block waits Busy go to FALSE level to resubmit the request.

NOTE!
If Execute goes to FALSE level and Busy is still at TRUE level, the request is canceled.

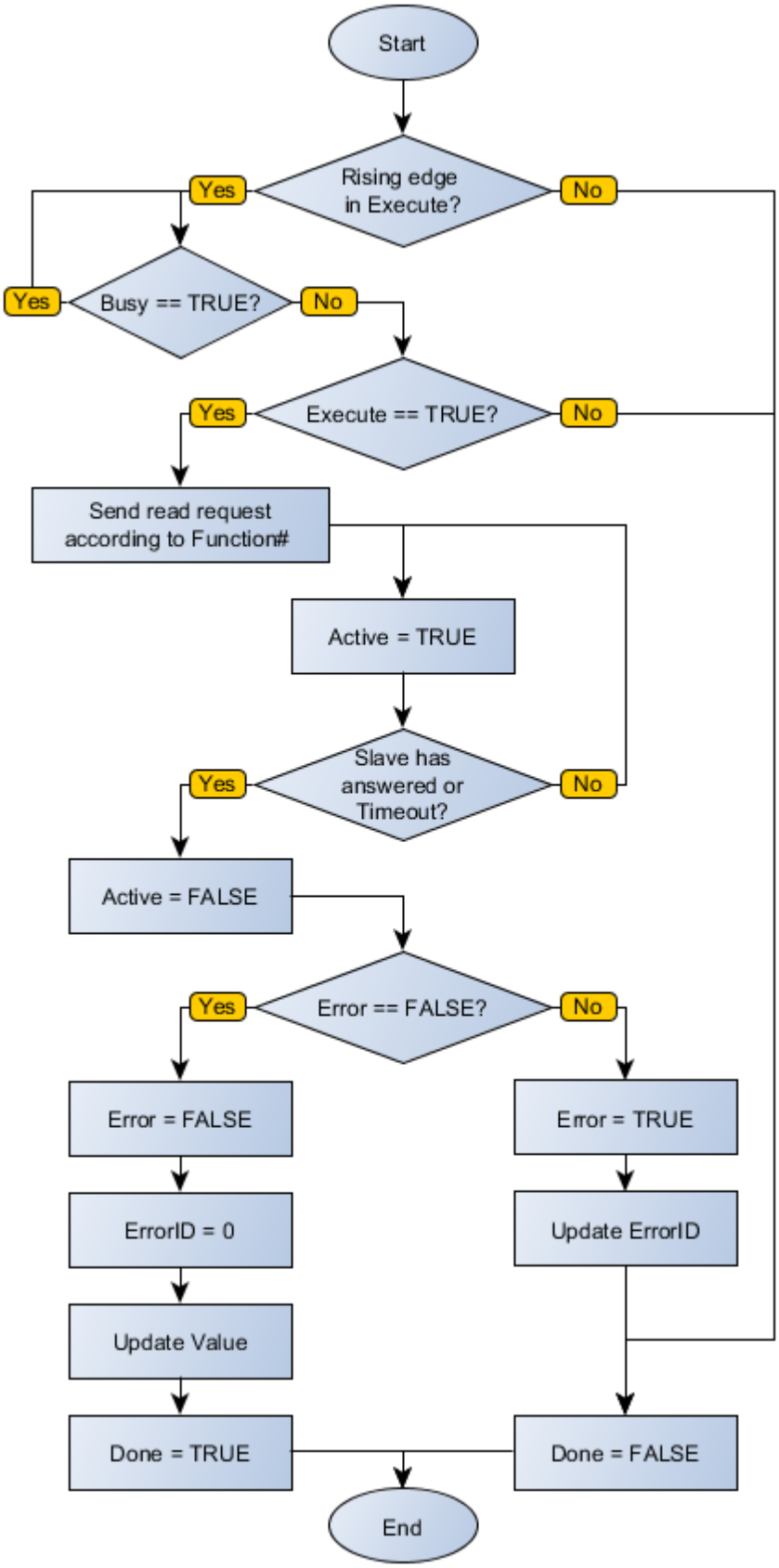
NOTE!
Value is an array of number of bits NumberOfData multiplied by 16. That is, if NumberOfData is 16, Value can be an array of 32 BYTE positions, 16 WORD positions or 8 DWORD positions. It is important to check this compatibility not to generate errors in the block.

When Execute has FALSE value, Done remains FALSE. The Done output is only activated when the block finishes executing successfully, remaining at TRUE level until Execute receives FALSE.

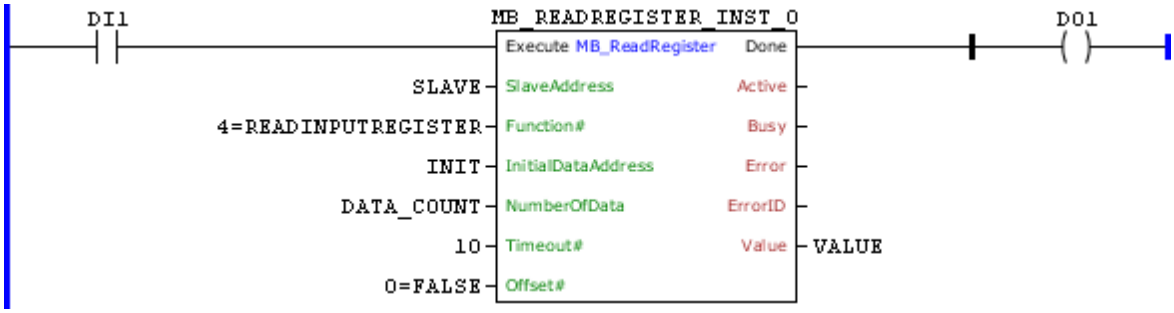
If there is any error in the execution, the Error output is enabled and ErrorID displays an error code according to the table below.

Code	Description
0	Executed successfully
1	Invalid input data
2	Master not enabled
4	Timeout in slave response
5	Slave returned error

Block Flowchart


Start
Rising edge in Execute?
Yes
No
Busy == TRUE?
Yes
No
Execute == TRUE?
Yes
No
Send read request according to Function#
Active = TRUE
Slave has answered or Timeout?
Yes
No
Active = FALSE
Error == FALSE?
Yes
No
Error = FALSE
ErrorID = 0
Update Value
Done = TRUE
Error = TRUE
Update ErrorID
Done = FALSE
End

Example



The above example requests reading of a number of binary data described by DATA_COUNT positioned in the INIT in the Modbus RTU slave of SLAVE address through the Read Input Register function. These data are forwarded to VALUE. The block ends successfully, Done output is activated.

11.14.7.2.1.5 MB_SlaveStatus

Block that allows monitoring the status of 4 slaves of the Modbus RTU network.

Ladder Representation

Block Structure

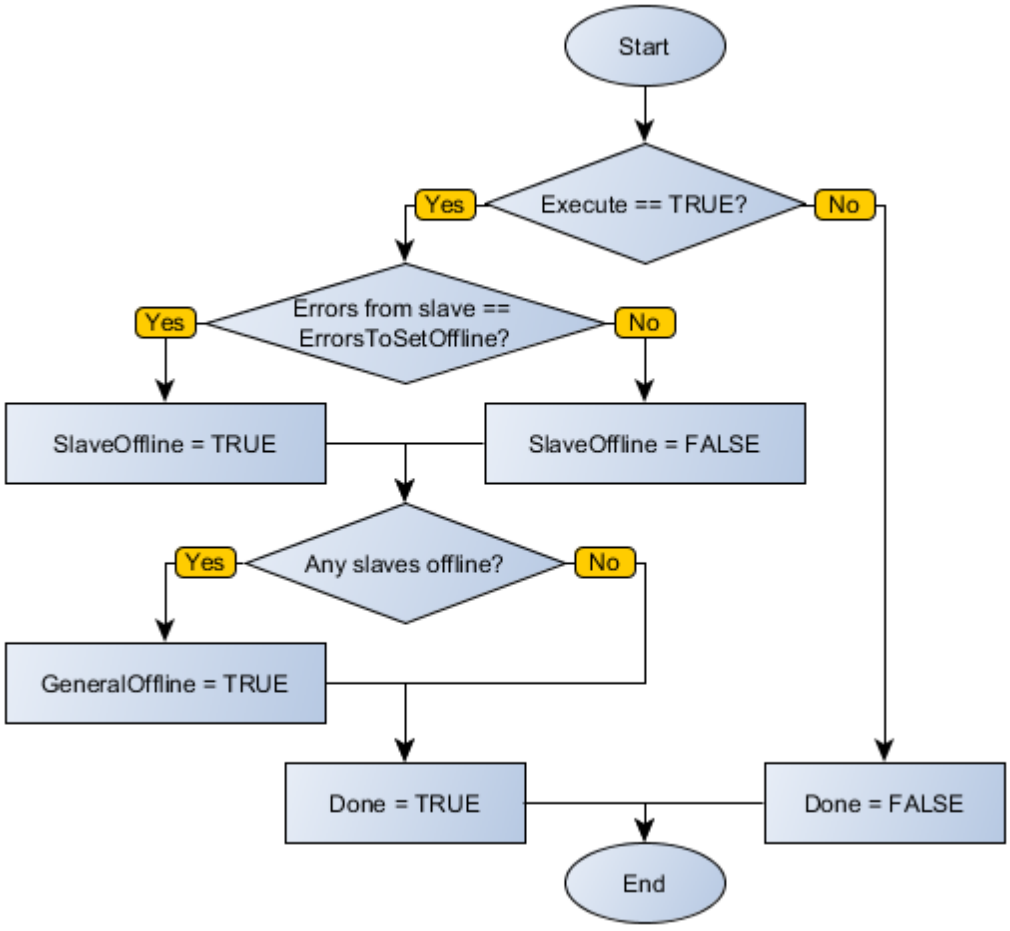
Variable Type	Name	Data Type	Description
VAR_INPUT	Execute	BOOL	Block enabling
	ErrorsToSetOffline#	BYTE	Amount of errors that the master must identify until it considers communication w ith an offline slave
	AddressSlave1#	BYTE	Slave address 1 to be monitored
	AddressSlave2#	BYTE	Slave address 2 to be monitored
	AddressSlave3#	BYTE	Slave address 3 to be monitored
	AddressSlave4#	BYTE	Slave address 4 to be monitored
VAR_OUTPUT	Done	BOOL	Output enabling
	GeneralOffline	BOOL	Flag indicating any one of the monitored communication is offline
	Slave1Offline	BOOL	Flag of offline status slave 1
	Slave2Offline	BOOL	Flag of offline status slave 2
	Slave3Offline	BOOL	Flag of offline status slave 3
	Slave4Offline	BOOL	Flag of offline status slave 4

Operation

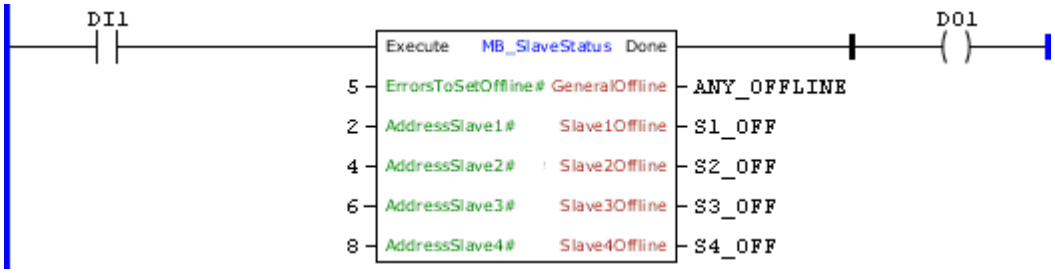
This block remains active while Execute is at TRUE level, updating its outputs according to the number of errors recorded for each slave. When Execute receives FALSE level, the inputs are ignored and the outputs are zeroed. The Done output receives TRUE level when Execute has TRUE level and block finished its execution.

The ErrorsToSetOffline # input allows registering the number of errors identified in a slave that will feature an offline communication. AddressSlave inputs allow inserting four slave addresses to be monitored. When this monitored slave reports the programmed number of errors, its corresponding SlaveOffline output is set to TRUE level. If any of SlaveOffline outputs is at TRUE level, GeneralOffline also receives TRUE level.

Block Flowchart



Example

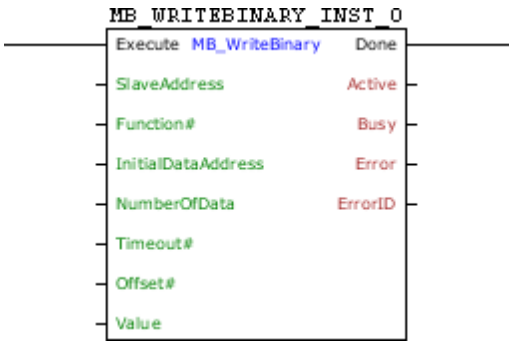


The above example checks the number of error responses sent by the slaves 2, 4, 6 and 8 of the Modbus RTU. If any of them is greater than 5, its SX_OFF status is led to TRUE level. The block ends successfully, Done output is activated.

11.14.7.2.1.6 MB_WriteBinary

Block that performs a writing of up to 128 binary data (via Write Single Coil or Write Multiple Coils) in a slave on the Modbus RTU network.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	Execute	BOOL	Block enabling
	SlaveAddress	BYTE	Slave address
	Function#	BYTE	Writing function code
	InitialDataAddress	WORD	Initial bit address where the data will be written
	NumberOfData	BYTE	Number of bits to be written (1 to 128)
	Timeout#	WORD	Maximum waiting time for the slave response [ms]
	Offset#	BOOL	Offset Indication in InitialDataAddress, i.e., need to subtract 1 from this number
	Value	BOOL	Variable that stores the data to be written
VAR_OUTPUT	Done	BOOL	Output enabling
	Active	BOOL	Awaiting response flag
	Busy	BOOL	Flag indicating the RS485 interface is busy with another request
	Error	BOOL	Error in the execution flag
	ErrorID	BYTE	Identifier of the occurred error
VAR	MB_WRITEBINARY_INST_0	MB_WRITEBINARY	Instance of access to block structure

Operation

When this block detects a leading edge on Execute, it checks whether the Modbus RTU slave in specified address in SlaveAddress is free to send data (Busy variable at FALSE level). If so, it sends the writing request of a number of bits indicated by NumberOfData in InitialDataAddress address using chosen function in Function# and sets the Active output, resetting it when receiving the response from the slave. If the slave is not free, the block waits Busy go to FALSE level to resubmit the request.

NOTE!
If Execute goes to FALSE level and Busy is still at TRUE level, the request is canceled.

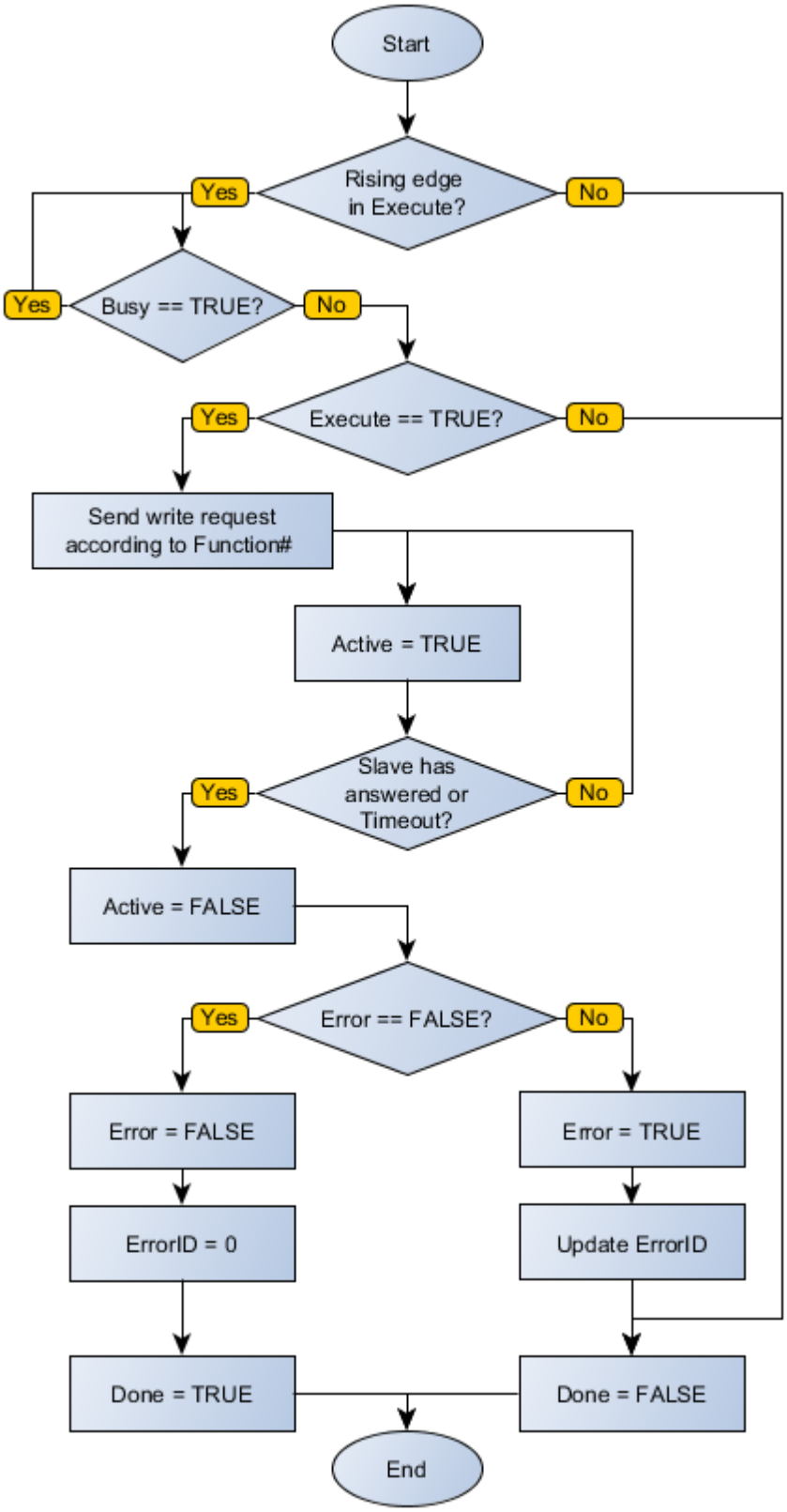
NOTE!
Value is an array of size equal to NumberOfData. It is important to check this compatibility not to generate errors in the block.

When Execute has FALSE value, Done remains FALSE. The Done output is only activated when the block finishes executing successfully, remaining at TRUE level until Execute receives FALSE.

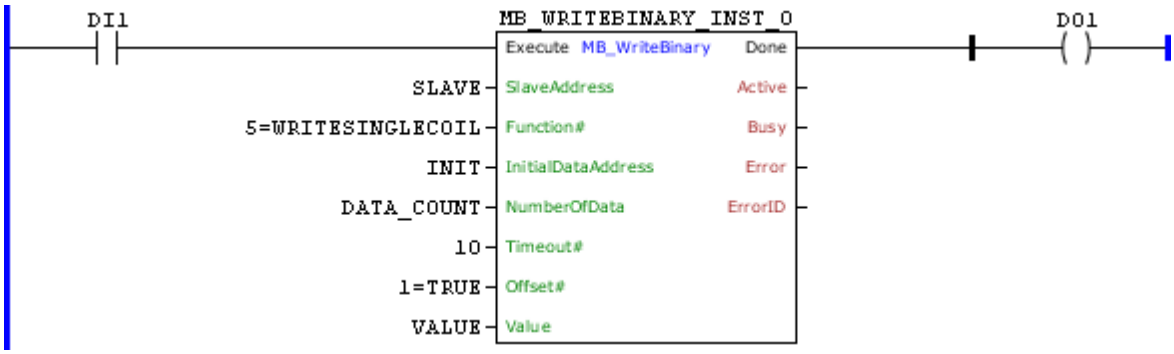
If there is any error in the execution, the Error output is enabled and ErrorID displays an error code according to the table below.

Code	Description
0	Executed successfully
1	Invalid input data
2	Master not enabled
4	Timeout in slave response
5	Slave returned error

Block Flowchart


Start
Rising edge in Execute?
Yes
No
Busy == TRUE?
Yes
No
Execute == TRUE?
Yes
No
Send write request according to Function#
Active = TRUE
Slave has answered or Timeout?
Yes
No
Active = FALSE
Error == FALSE?
Yes
No
Error = FALSE
Error = TRUE
ErrorID = 0
Update ErrorID
Done = TRUE
Done = FALSE
End

Example

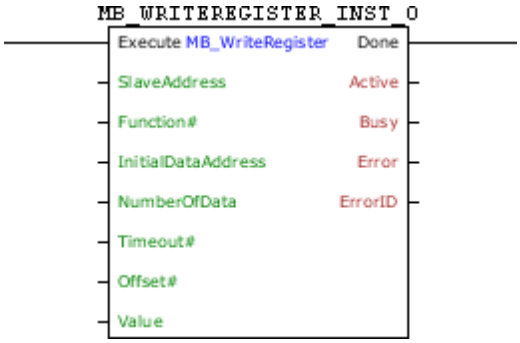


The example above requests written data contained in VALUE, with size described by DATA_COUNT, at addresses positioned from INIT on Modbus RTU slave at address SLAVE using the function Write Single Coil. The block ends successfully, Done output is activated.

11.14.7.2.1.7 MB_WriteRegister

Block that performs a reading of up to sixteen 16-bit registers (via Write Single Register or Write Multiple Registers) of a slave on the Modbus RTU network.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	Execute	BOOL	Block enabling
	SlaveAddress	BYTE	Slave address
	Function#	BYTE	Writing function code
	InitialDataAddress	WORD	Initial register address to be written
	NumberOfData	BYTE	Number of registers to be written (1 to 16)
	Timeout#	WORD	Maximum waiting time for the slave response [ms]
	Offset#	BOOL	Offset Indication in InitialDataAddress, i.e., need to subtract 1 from this number
	Value	BYTE SINT USINT WORD UINT INT DWORD UDINT DINT REAL	Variable that stores the data to be written
VAR_OUTPUT	Done	BOOL	Output enabling
	Active	BOOL	Awaiting response flag
	Busy	BOOL	Flag indicating the RS485 interface is busy with another request
	Error	BOOL	Error in the execution flag
	ErrorID	BYTE	Identifier of the occurred error
VAR	MB_WRITEREGISTER_INST_0	MB_WRITEREGISTER	Instance of access to block structure

Operation

When this block detects a leading edge on Execute, it checks whether the Modbus RTU slave in specified address in SlaveAddress is free to send data (Busy variable at FALSE level). If so, it sends the writing request of Value values in a number of registers indicated by NumberOfData in InitialDataAddress address using chosen function in Function# and sets the Active output, resetting it when receiving the response from the slave. If the slave is not free, the block waits Busy go to FALSE level to resubmit the request.

NOTE!
If Execute goes to FALSE level and Busy is still at TRUE level, the request is canceled.

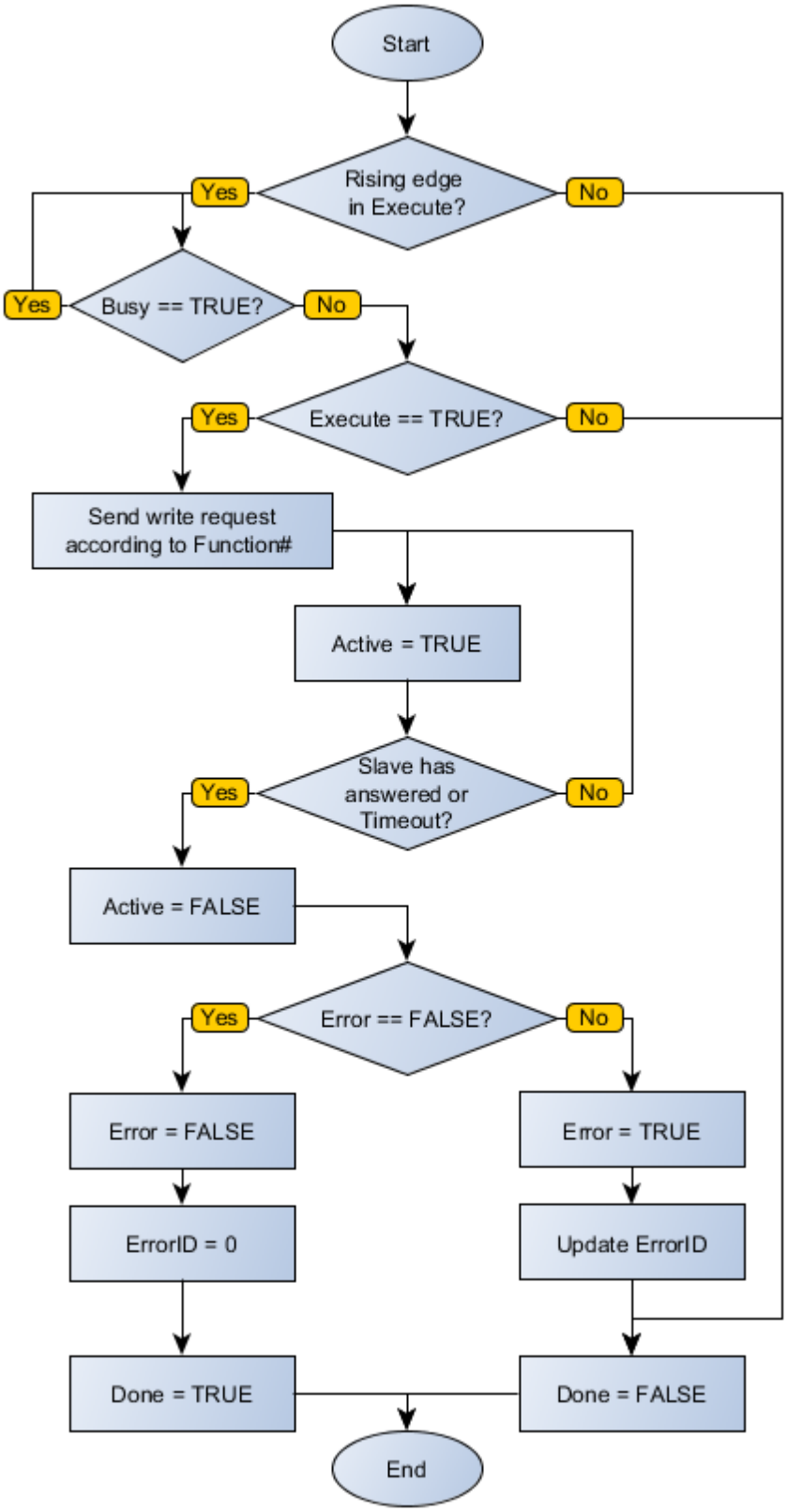
NOTE!
Value is an array of number of bits NumberOfData multiplied by 16. That is, if NumberOfData is 16, Value can be an array of 32 BYTE positions, 16 WORD positions or 8 DWORD positions. It is important to check this compatibility not to generate errors in the block.

When Execute has FALSE value, Done remains FALSE. The Done output is only activated when the block finishes executing successfully, remaining at TRUE level until Execute receives FALSE.

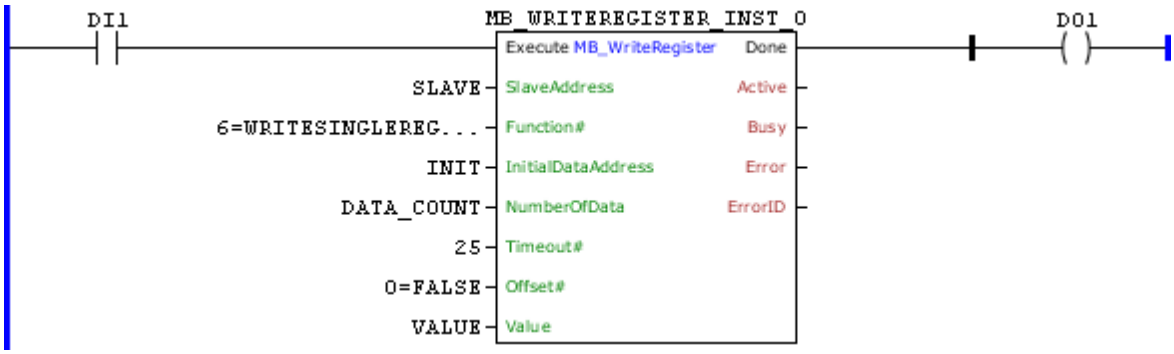
If there is any error in the execution, the Error output is enabled and ErrorID displays an error code according to the table below.

Code	Description
0	Executed successfully
1	Invalid input data
2	Master not enabled
4	Timeout in slave response
5	Slave returned error

Block Flowchart


Start
Rising edge in Execute?
Yes
No
Busy == TRUE?
Yes
No
Execute == TRUE?
Yes
No
Send write request according to Function#
Active = TRUE
Slave has answered or Timeout?
Yes
No
Active = FALSE
Error == FALSE?
Yes
No
Error = FALSE
Error = TRUE
ErrorID = 0
Update ErrorID
Done = TRUE
Done = FALSE
End

Example



The example above requests written data contained in VALUE, with size described by DATA_COUNT, at addresses positioned from INIT on Modbus RTU slave at address SLAVE using the function Write Single Register. The block ends successfully, Done output is activated.

11.14.7.3 Compare

11.14.7.3.1 COMP_EQ

Block that compares the values of Value1 and Value2, enabling the output Q if both are equal.

Ladder Representation

Block Structure

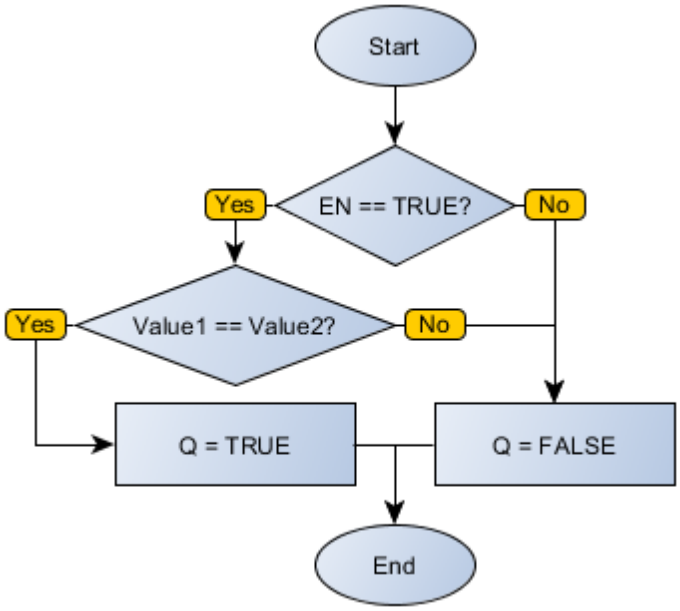
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	First value of comparison
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Second value of comparison
VAR_OUTPUT	Q	BOOL	Indicator of equality

Operation

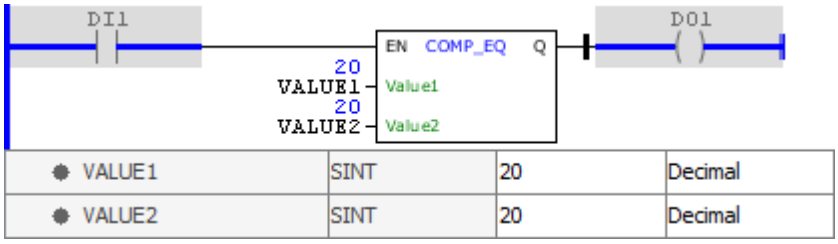
When this block has a TRUE value in EN, it sends to the output Q the TRUE value if Value1 and Value2 are the same. Otherwise, Q receives FALSE.

When EN has FALSE value, Q remains in FALSE.

Block Flowchart

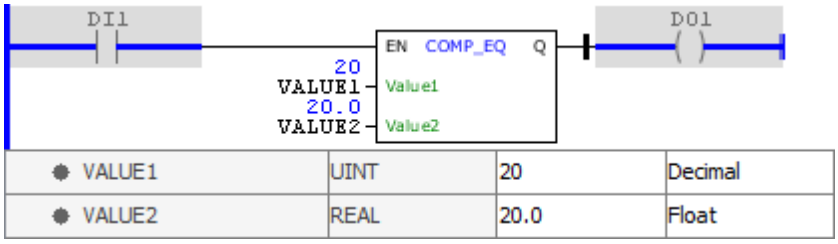


Example



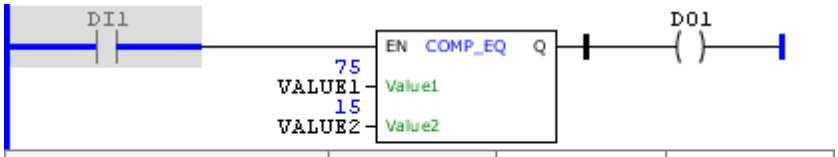
VALUE1	SINT	20	Decimal
VALUE2	SINT	20	Decimal

The example above checks equality between VALUE1 and VALUE2. Since both variables have the same value, the Q output is activated.



VALUE1	UINT	20	Decimal
VALUE2	REAL	20.0	Float

The example above checks equality between VALUE1 and VALUE2. Since both variables have the same value, the Q output is activated. Notice that the types of the input variables can be different without causing execution problems.



VALUE1	UINT	75	Decimal
VALUE2	UINT	15	Decimal

The example above checks equality between VALUE1 and VALUE2. Since both variables have different values, the Q output is disabled.

11.14.7.3.2 COMP_GE

Block that compares the values of Value1 and Value2, enabling the output Q if Value1 is higher than or equal to Value2.

Ladder Representation

Block Structure

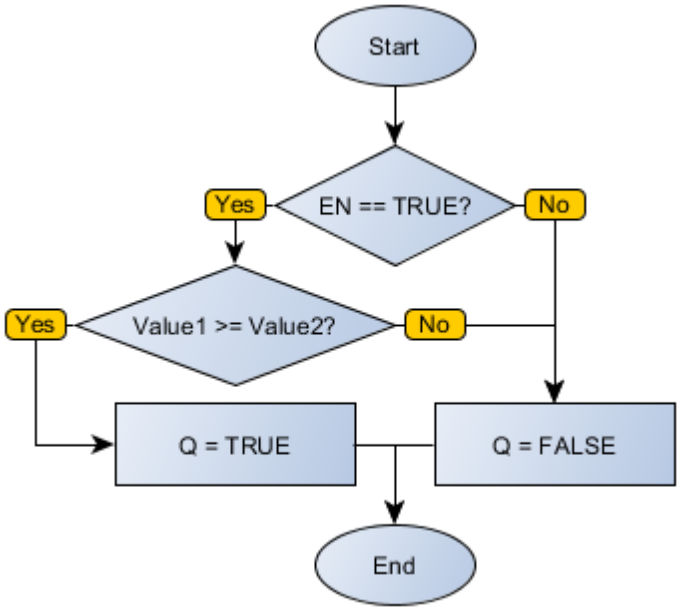
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	First value of comparison
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Second value of comparison
VAR_OUTPUT	Q	BOOL	Indicator of equality or majority of Value1

Operation

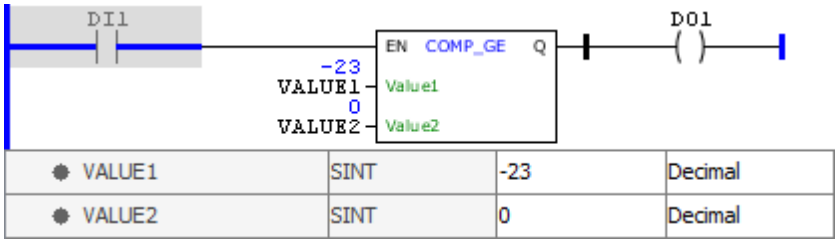
When this block has a TRUE value in EN it sends the Q output to the TRUE value if Value1 is higher than or equal to Value2. Otherwise, Q receives FALSE.

When EN has FALSE value, Q remains in FALSE.

Block Flowchart

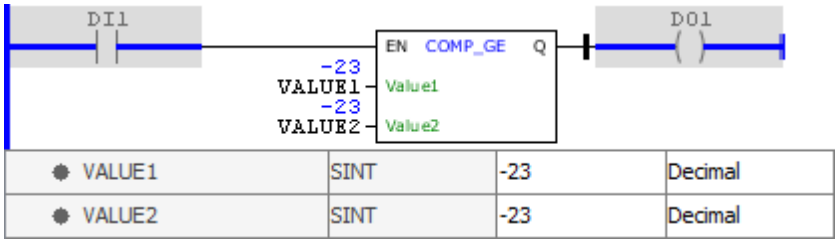


Example



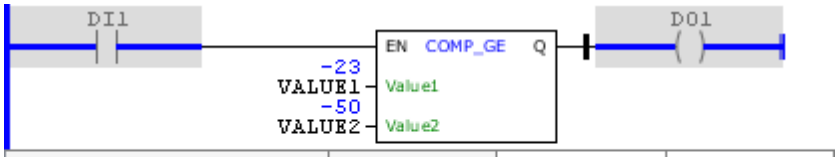
VALUE1	SINT	-23	Decimal
VALUE2	SINT	0	Decimal

The example above checks equality or majority of VALUE1 in relation to VALUE2. Since VALUE1 has lower value than VALUE2, the Q output is disabled.



VALUE1	SINT	-23	Decimal
VALUE2	SINT	-23	Decimal

The example above checks equality or majority of VALUE1 in relation to VALUE2. Since both variables have the same value, the Q output is activated.



VALUE1	SINT	-23	Decimal
VALUE2	SINT	-50	Decimal

The example above checks equality or majority of VALUE1 in relation to VALUE2. Since VALUE1 has higher value than VALUE2, the Q output is activated.

11.14.7.3.3 COMP_GT

Block that compares the values of Value1 and Value2, enabling the output Q if Value1 is higher than Value2.

Ladder Representation

Block Structure

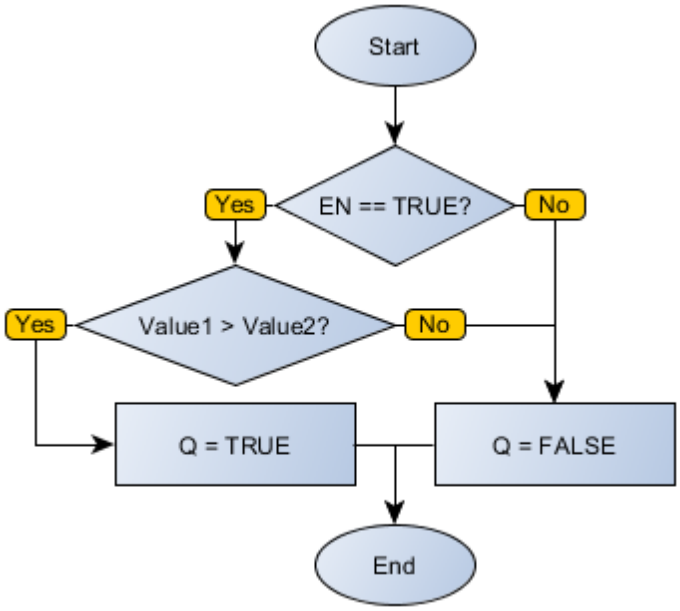
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	First value of comparison
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Second value of comparison
VAR_OUTPUT	Q	BOOL	Indicator of majority of Value1

Operation

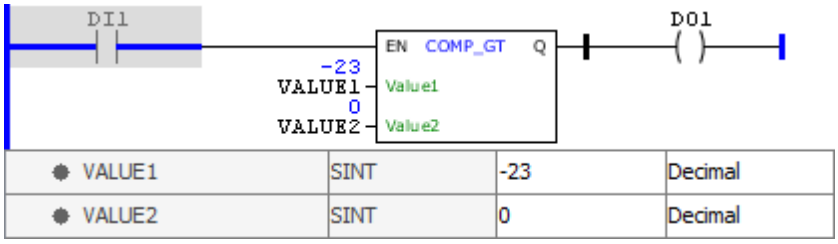
When this block has a TRUE value in EN, it sends to the Q output the TRUE value if Value1 is higher than Value2. Otherwise, Q receives FALSE.

When EN has FALSE value, Q remains in FALSE.

Block Flowchart

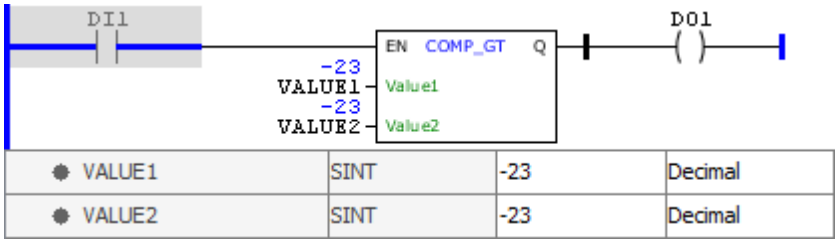


Example



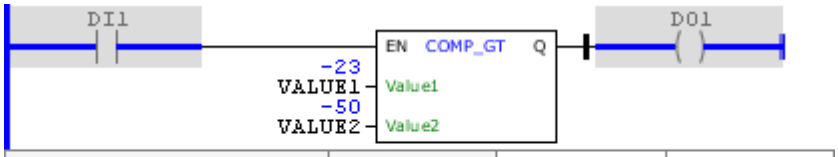
VALUE1	SINT	-23	Decimal
VALUE2	SINT	0	Decimal

The example above checks the majority of VALUE1 in relation to VALUE2. Since VALUE1 has lower value than VALUE2, the Q output is disabled.



VALUE1	SINT	-23	Decimal
VALUE2	SINT	-23	Decimal

The example above checks the majority of VALUE1 in relation to VALUE2. Since both variables have the same value, the Q output is disabled.



VALUE1	SINT	-23	Decimal
VALUE2	SINT	-50	Decimal

The example above checks the majority of VALUE1 in relation to VALUE2. Since VALUE1 has higher value than VALUE2, the Q output is activated.

11.14.7.3.4 COMP_LE

Block that compares the values of Value1 and Value2, enabling the output Q if Value1 is lower than or equal to Value2.

Ladder Representation

Block Structure

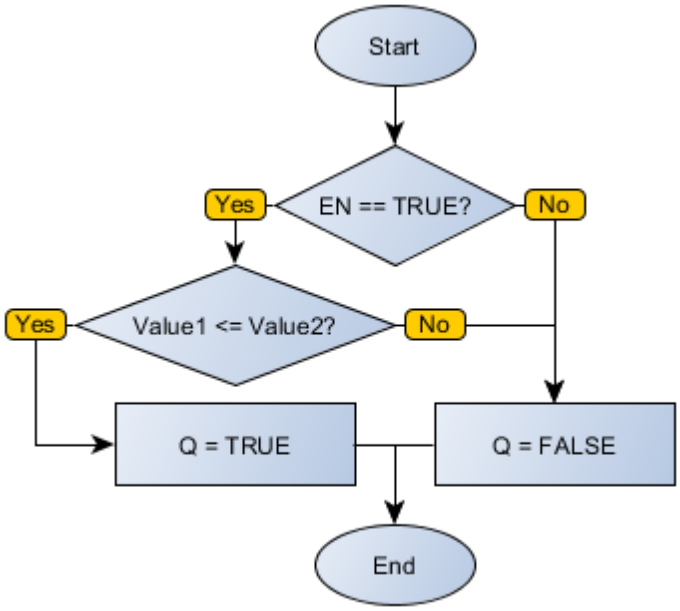
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	First value of comparison
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Second value of comparison
VAR_OUTPUT	Q	BOOL	Indicator of equality or minority of Value1

Operation

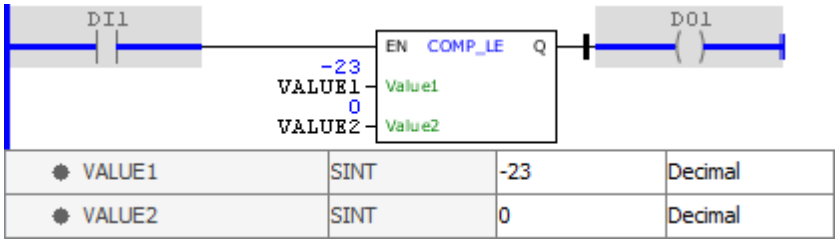
When this block has a TRUE value in EN, it sends to the Q output the TRUE value if Value1 is lower than or equal to Value2. Otherwise, Q receives FALSE.

When EN has FALSE value, Q remains in FALSE.

Block Flowchart

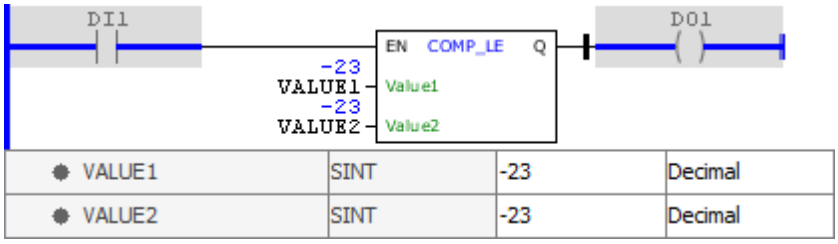


Example



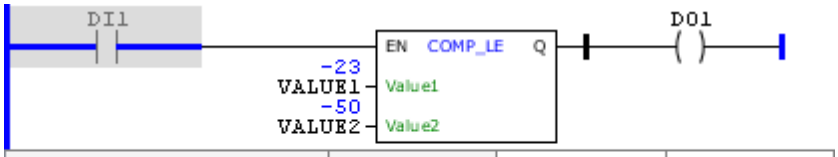
VALUE1	SINT	-23	Decimal
VALUE2	SINT	0	Decimal

The example above checks equality or minority of VALUE1 in relation to VALUE2. Since VALUE1 has lower value than VALUE2, the Q output is activated.



VALUE1	SINT	-23	Decimal
VALUE2	SINT	-23	Decimal

The example above checks equality or minority of VALUE1 in relation to VALUE2. Since both variables have the same value, the Q output is activated.



VALUE1	SINT	-23	Decimal
VALUE2	SINT	-50	Decimal

The example above checks equality or minority of VALUE1 in relation to VALUE2. Since VALUE1 has higher value than VALUE2, the Q output is disabled.

11.14.7.3.5 COMP_LT

Block that compares the values of Value1 and Value2, enabling the output Q if Value1 is lower than Value2.

Ladder Representation

Block Structure

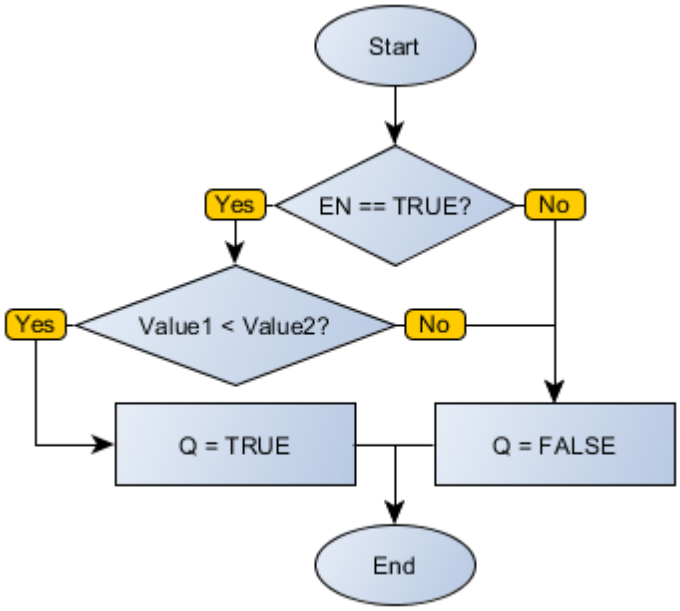
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	First value of comparison
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Second value of comparison
VAR_OUTPUT	Q	BOOL	Indicator of minority of Value1

Operation

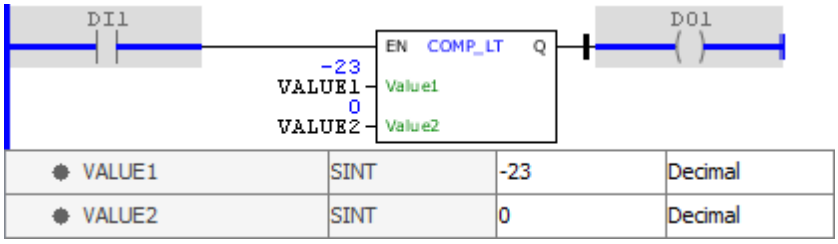
When this block has a TRUE value in EN, it sends to the Q output the TRUE value if Value1 is lower than or equal to Value2. Otherwise, Q receives FALSE.

When EN has FALSE value, Q remains in FALSE.

Block Flowchart

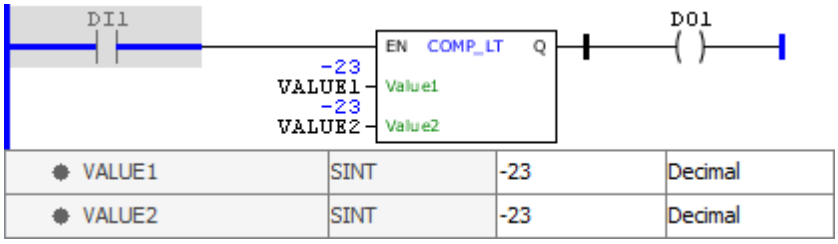


Example



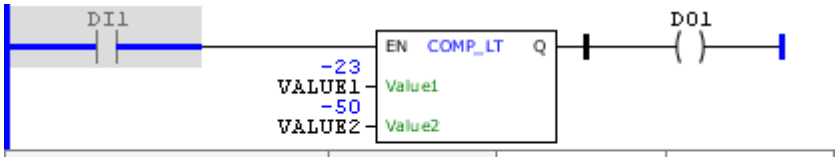
VALUE1	SINT	-23	Decimal
VALUE2	SINT	0	Decimal

The example above checks minority of VALUE1 in relation to VALUE2. Since VALUE1 has lower value than VALUE2, the Q output is activated.



VALUE1	SINT	-23	Decimal
VALUE2	SINT	-23	Decimal

The example above checks the minority of VALUE1 in relation to VALUE2. Since both variables have the same value, the Q output is disabled.



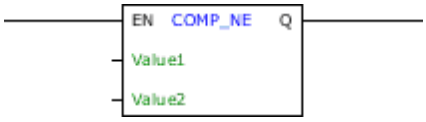
VALUE1	SINT	-23	Decimal
VALUE2	SINT	-50	Decimal

The example above checks the minority of VALUE1 in relation to VALUE2. Since VALUE1 has higher value than VALUE2, the Q output is disabled.

11.14.7.3.6 COMP_NE

Block that compares the values of Value1 and Value2, enabling the Q output if Value1 is different from Value2.

Ladder Representation



Block Structure

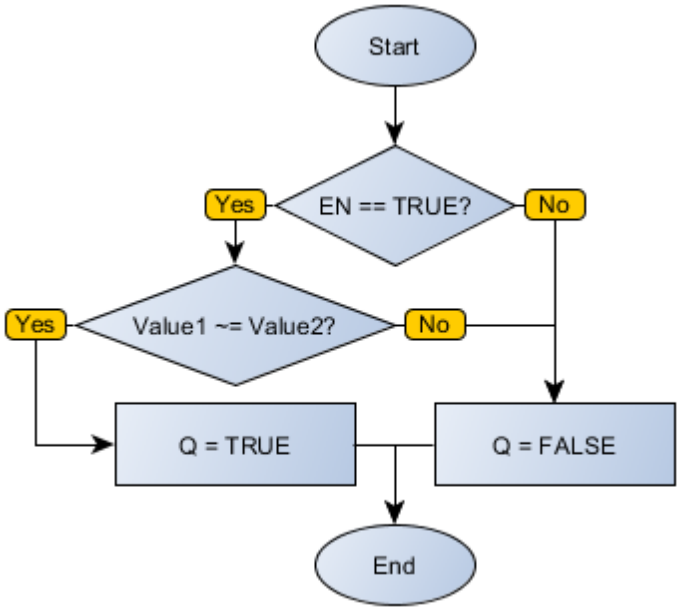
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	First value of comparison
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Second value of comparison
VAR_OUTPUT	Q	BOOL	Indicator of inequality

Operation

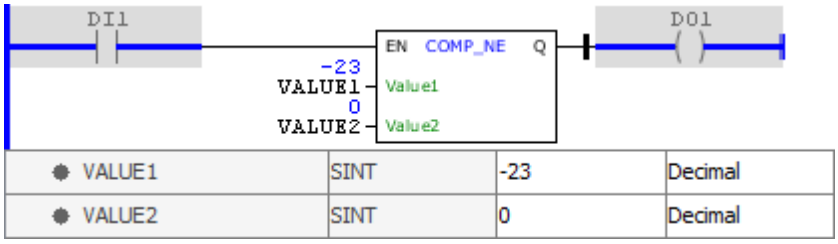
When this block has a TRUE value in EN, it sends to the Q output the TRUE value if Value1 is different from Value2. Otherwise, Q receives FALSE.

When EN has FALSE value, Q remains in FALSE.

Block Flowchart

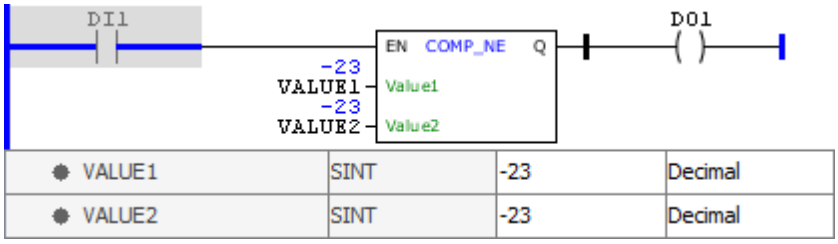


Example



VALUE1	SINT	-23	Decimal
VALUE2	SINT	0	Decimal

The example above checks inequality between VALUE1 and VALUE2. Since both variables have different values, the Q output is activated.



VALUE1	SINT	-23	Decimal
VALUE2	SINT	-23	Decimal

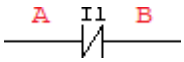
The example above checks equality between VALUE1 and VALUE2. Since both variables have the same value, the Q output is disabled.

11.14.7.4 Contact

11.14.7.4.1 NCCONTACT

Normally closed contact.

Ladder Representation



Block Structure

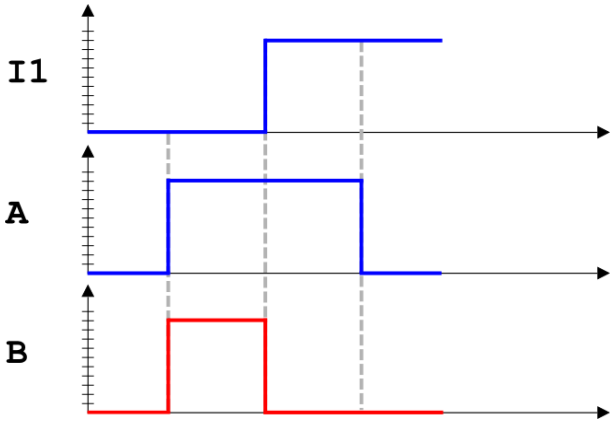
Variable Type	Name	Data Type	Description
VAR_OUTPUT	I1	BOOL	Block control input

Operation

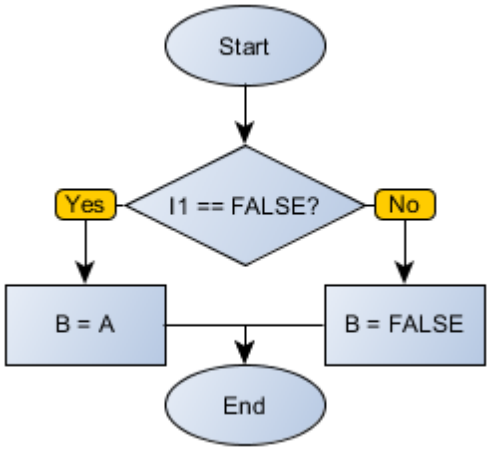
When variable I1 is with TRUE value, B receives FALSE.
When variable I1 is with FALSE value, B receives the value of A.

NOTE!
Watch out for series and parallel associations of contacts. Refer to section Contact Logic for further information.

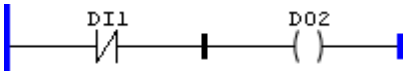
Diagram



Block Flowchart



Example

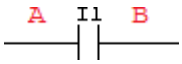


The above example performs the transfer of the opposite value of digital input DI1 to the digital output DO2.

11.14.7.4.2 NOCONTACT

Normally open contact.

Ladder Representation



Block Structure

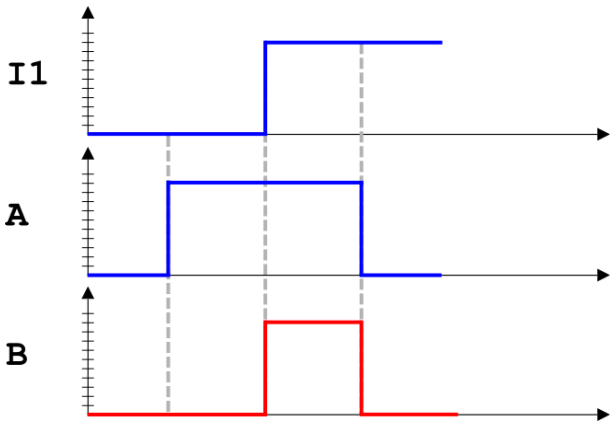
Variable Type	Name	Data Type	Description
VAR_OUTPUT	I1	BOOL	Block control input

Operation

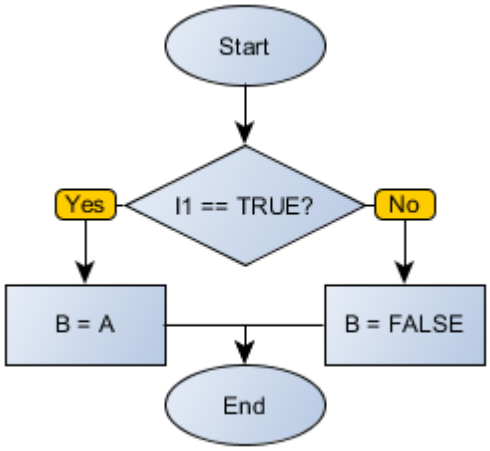
When variable I1 is with FALSE value, B receives FALSE.
When variable I1 is with TRUE value, B receives the value of A.

NOTE!
Watch out for series and parallel associations of contacts. Refer to section Contact Logic for further information.

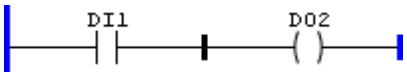
Diagram



Block Flowchart



Example



The above example performs the transfer of the value of digital input DI1 to the digital output DO2.

11.14.7.4.3 NTSCONTACT

Falling edge transition contact.

Ladder Representation

Block Structure

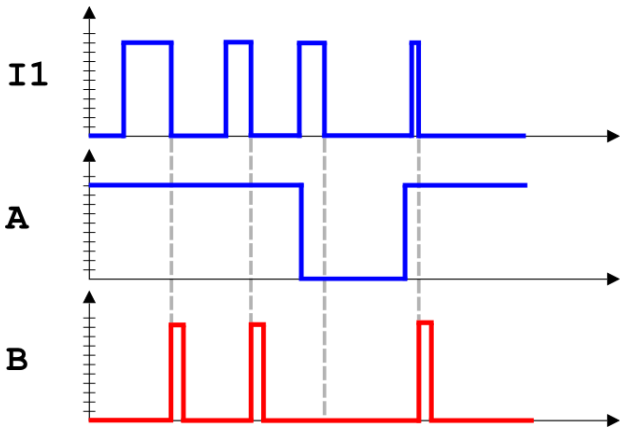
Variable Type	Name	Data Type	Description
VAR_INPUT	I1	BOOL	Block control input
VAR	NTSCONTACT_INST_0	NTSCONTACT	Instance of access to block structure

Operation

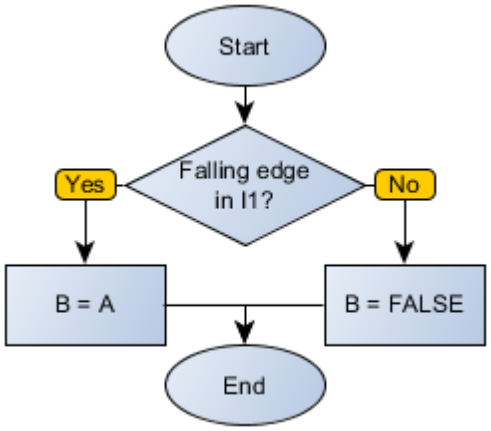
At the instant the variable I1 transitions from TRUE to FALSE (falling edge or negative edge transition), B receives the value of A for a scan cycle.
At all other times, B receives the FALSE value.

NOTE!
Watch out for series and parallel associations of contacts. Refer to section Contact Logic for further information.

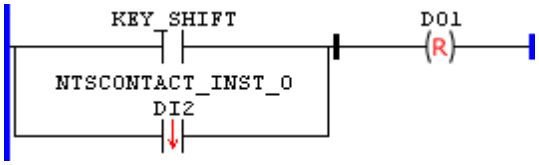
Diagram



Block Flowchart



Example

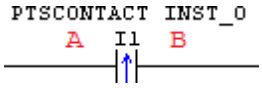


The above example resets the digital output DO1 if the SHIFT key is pressed or a positive pulse on the digital input DI2 is given.

11.14.7.4.4 PTSCONTACT

Leading edge transition contact.

Ladder Representation



Block Structure

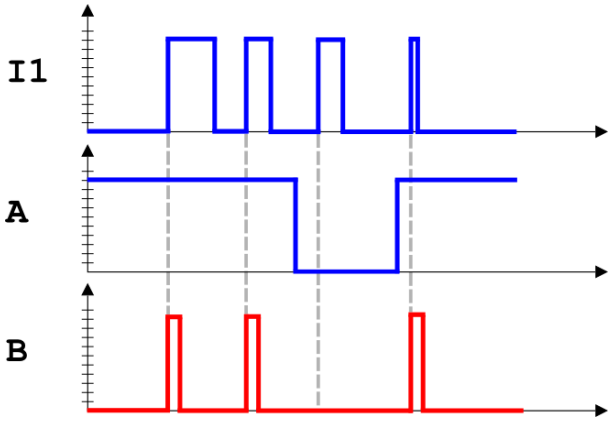
Variable Type	Name	Data Type	Description
VAR_INPUT	I1	BOOL	Block control input
VAR	PTSCONTACT_INST_0	PTSCONTACT	Instance of access to block structure

Operation

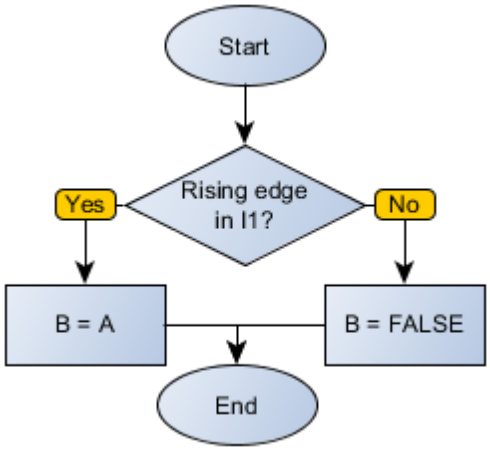
At the instant the variable I1 transitions from FALSE to TRUE (leading edge or positive edge transition), B receives the value of A for a scan cycle.
At all other times, B receives the FALSE value.

NOTE!
Watch out for series and parallel associations of contacts. Refer to section Contact Logic for further information.

Diagram



Block Flowchart



Example

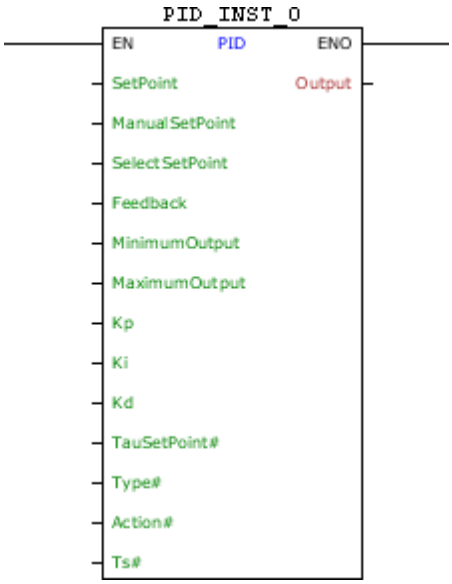
The above example resets the digital output DO1 if the SHIFT key is pressed and a positive pulse on the digital input DI2 is given.

11.14.7.5 Control

11.14.7.5.1 PID

Block that performs the function of a discrete PID controller. From the input variables, it calculates the corresponding controller output.

Ladder Representation



Block Structure

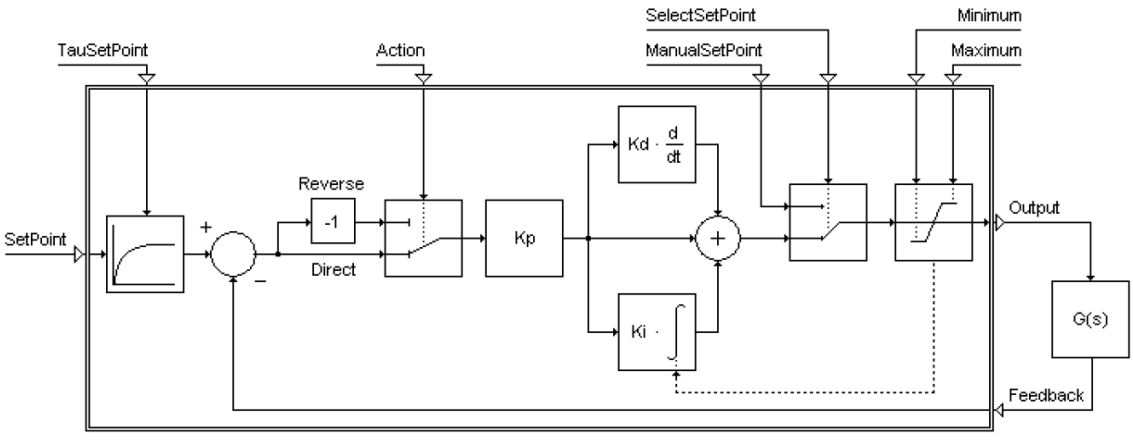
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	SetPoint	REAL	Automatic reference (pre-control)
	ManualSetPoint	REAL	Forced reference (post control)
	SelectSetPoint	BOOL	Selects w hich reference to use
	Feedback	REAL	Feedback loop variable
	MinimumOutput	REAL	Minimum value of the controller output
	MaximumOutput	REAL	Maximum value of the controller output
	Kp	REAL	Proportional gain
	Ki	REAL	Integral gain
	Kd	REAL	Derivative gain
	TauSetPoint#	REAL	Time constant of the automatic reference in put filter
	Type#	BYTE	Controller type
	Action#	BYTE	Control action
	Ts#	UINT	Sampling time [ms]
VAR_OUTPUT	ENO	BOOL	Output enabling
	Output	REAL	Controller output
VAR	PID_INST_0	PID	Instance of access to block structure

Operation

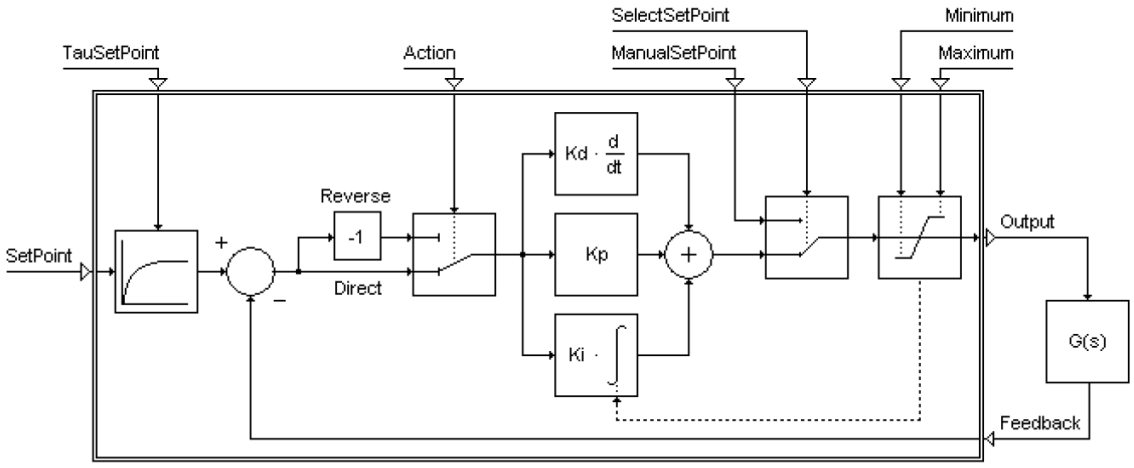
On the positive transition edge in EN, Output receives zero value, and the block executes its functionality as EN is at TRUE level.

When enabled, this block performs a routine PID control with the Kp, Ki and Kd parameters chosen. The PID topology used may be the Academic or Parallel, depending on what is chosen in Type#.

Academic Form:



Parallel Form:



The levels of the output signal of the controller are saturated at value MinimumOutput and MaximumOutput. The SelectSetPoint input level FALSE causes the SetPoint reference be adopted, allowing the controller maintains control over the process. When SelectSetPoint goes to TRUE level, the controller has no more domain, and ManualSetPoint becomes to be considered the output signal of the controller.

Action# will define the feedback operation. If Action# is DIRECT, the operation will be SetPoint – Feedback. If Action# is REVERSE, the operation will be Feedback – SetPoint.

Feedback receives the process variable considered as the plant output. Ts# receives the sampling time for the controller and # TauSetPoint receives the time constant for the input filter of the automatic reference.

When EN has FALSE value, Output remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

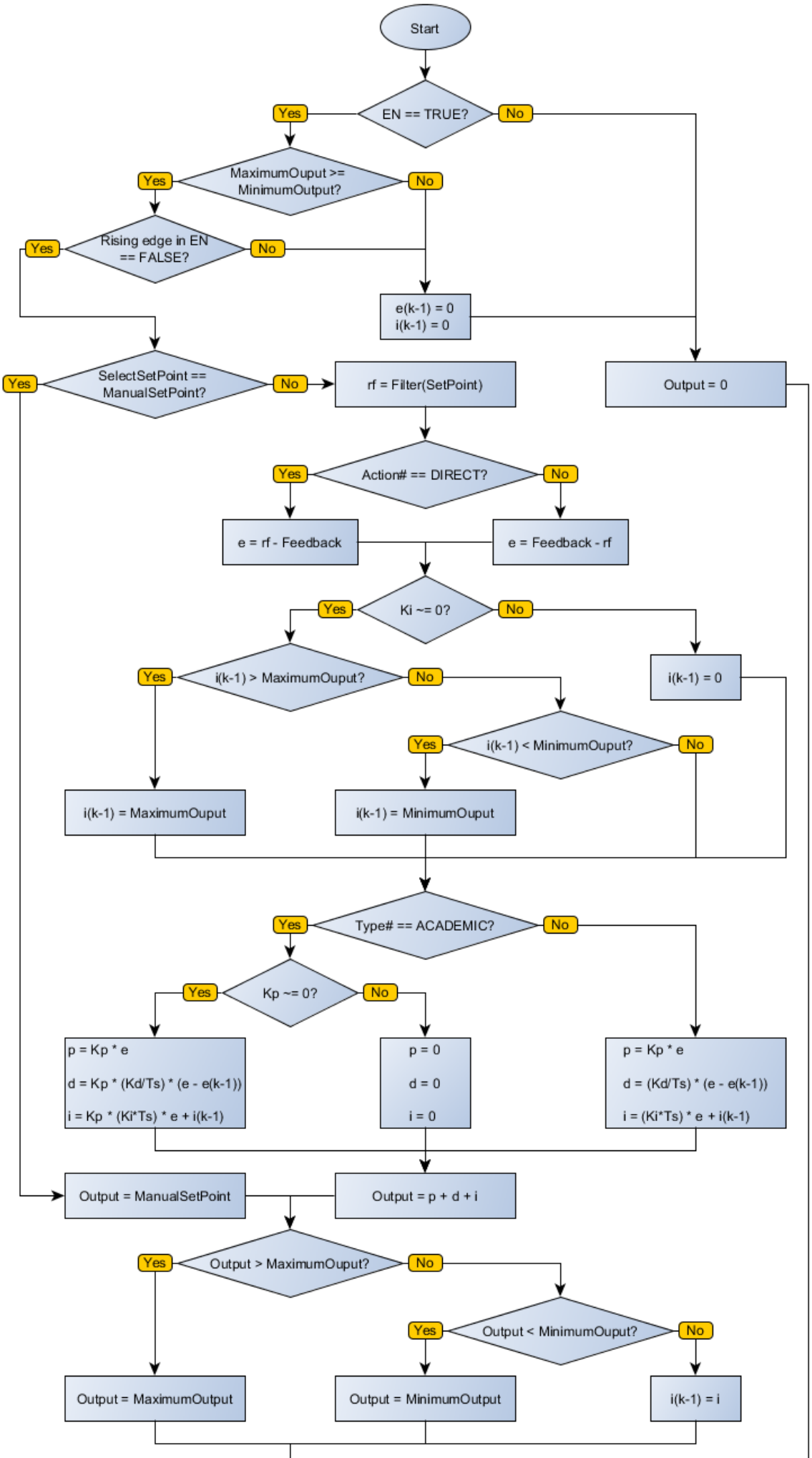
NOTE!
Effects of the alteration of gains on the process

- If Kp decreases, the process becomes slower; generally more stable or less oscillating; it has less overshoot.
- If Kp increases, the process responds faster; it may become more unstable or more oscillating; it has more overshoot.
- If Ki decreases, the process becomes slower, lagging to reach the "SetPoint"; it becomes more stable or less oscillating; it has less overshoot.
- If Ki increases, the process becomes faster, quickly reaching the "SetPoint"; it becomes more unstable or more oscillating; it has more overshoot.
- If Kd decreases, the process becomes slower; it has less overshoot.
- If Kd increases, it has more overshoot.

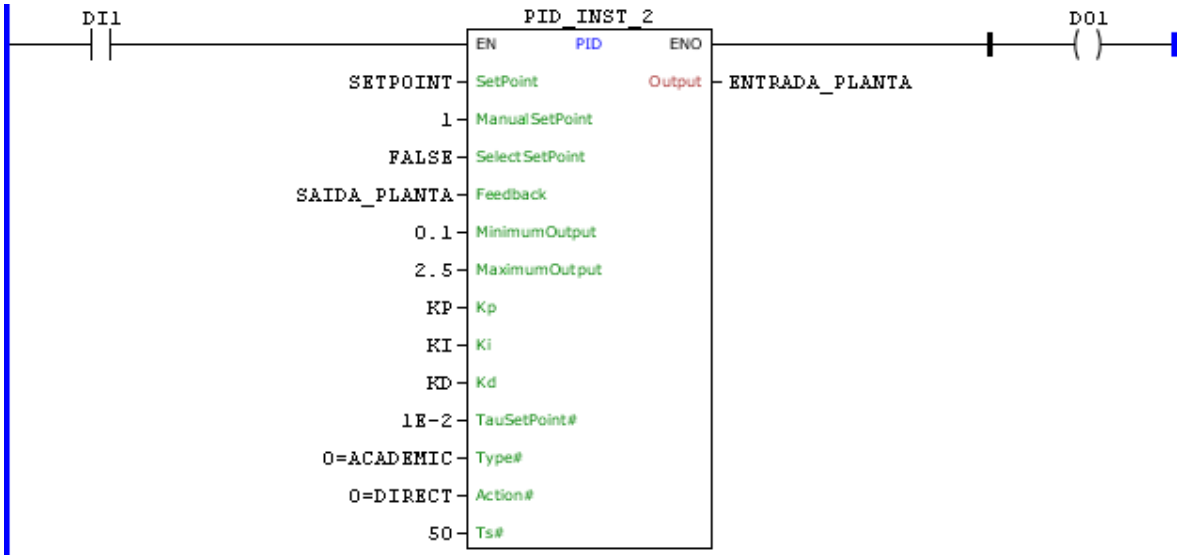
NOTE!
How to improve the performance of the process through the adjustment of gains (valid for the Academic PID)

- If the performance of the process is almost good, but the overshoot is a bit high, try to: (1) decrease Kp 20%, (2) decrease Ki 20% and/or (3) decrease Kd 50%.
- If the performance of the process is almost good, but it does not have overshoot and lags to reach the "SetPoint", try to: (1) increase Kp 20%, (2) increase Ki 20% and/or (3) increase Kd 50%.
- If the performance of the process is good, but the process output is varying too much, try to: (1) increase Kd 50%, (2) decrease Kp 20%.
- If the performance of the process is bad, i.e. after start up, the transitory lasts several periods of oscillation that reduce very slowly or never reduce at all, try to: (1) decrease Kp 50%.
- If the performance of the process is bad, i.e. after start up it slowly moves towards the "SetPoint" without overshoot, but is still very far and the process output is less than the rated value, try to: (1) increase Kp 50%, (2) increase Ki 50%, (3) increase Kd 70%.

Block Flowchart


Start
EN == TRUE?
Yes
No
MaximumOuput >= MinimumOutput?
Yes
No
Rising edge in EN == FALSE?
Yes
No
e(k-1) = 0
i(k-1) = 0
Output = 0
SelectSetPoint == ManualSetPoint?
Yes
No
rf = Filter(SetPoint)
Action# == DIRECT?
Yes
No
e = rf - Feedback
e = Feedback - rf
Ki ~= 0?
Yes
No
i(k-1) = 0
i(k-1) > MaximumOuput?
Yes
No
i(k-1) < MinimumOuput?
Yes
No
i(k-1) = MaximumOuput
i(k-1) = MinimumOuput
Type# == ACADEMIC?
Yes
No
Kp ~= 0?
Yes
No
p = Kp * e
d = Kp * (Kd/Ts) * (e - e(k-1))
i = Kp * (Ki*Ts) * e + i(k-1)
p = 0
d = 0
i = 0
p = Kp * e
d = (Kd/Ts) * (e - e(k-1))
i = (Ki*Ts) * e + i(k-1)
Output = ManualSetPoint
Output = p + d + i
Output > MaximumOuput?
Yes
No
Output < MinimumOuput?
Yes
No
Output = MaximumOutput
Output = MinimumOutput
i(k-1) = i

Example



The above example creates a loop of a digital PID form with sampling time 50 ms, using the constants KP, KI and KD for control. Automatic reference SETPOINT, filtered by a first order filter with time constant of 0:01 is used. The error signal is calculated as the difference between the filtered reference and variable SAIDA_PLANTA. The controller output is saturated between the values 0.1 and 2.5 and sent to the variable ENTRADA_PLANTA.

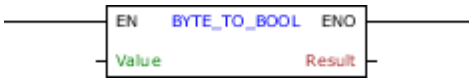
11.14.7.6 Conversion

11.14.7.6.1 BOOL

11.14.7.6.1.1 BYTE_TO_BOOL

Block that performs the conversion of a BYTE value into a BOOL value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT	Value in BYTE
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BOOL	Value in BOOL

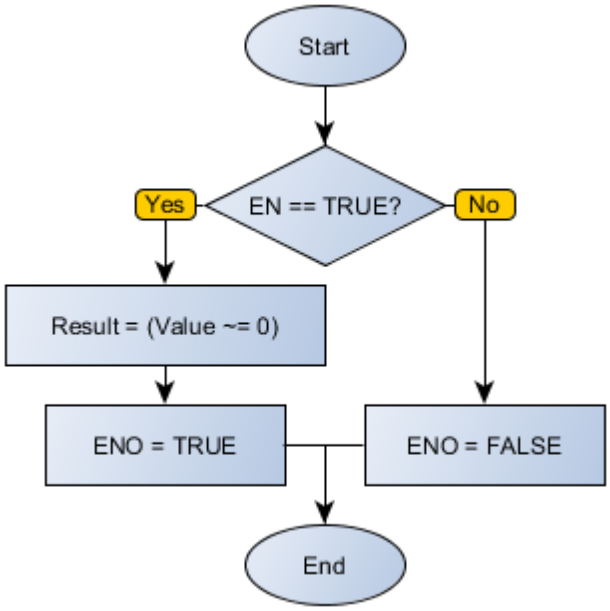
Operation

When this block has a TRUE value in EN, it interprets the Value value as BYTE and converts it into BOOL, storing in Result.

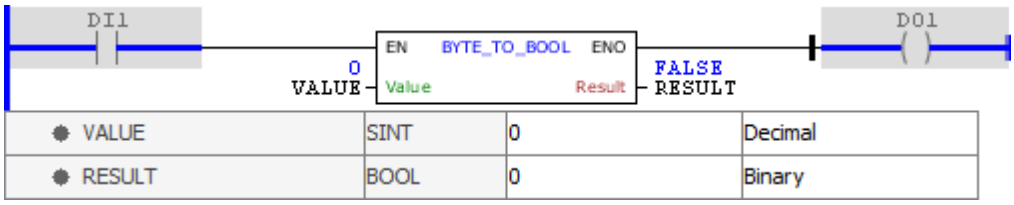
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

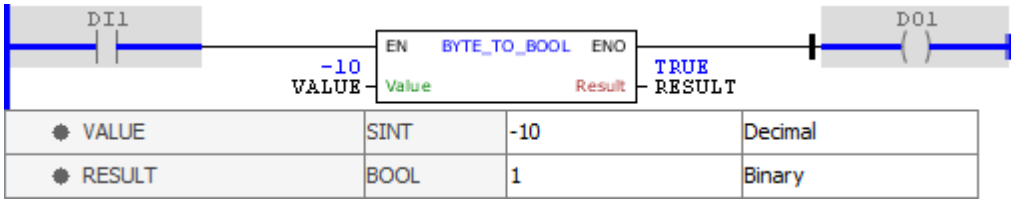
Block Flowchart



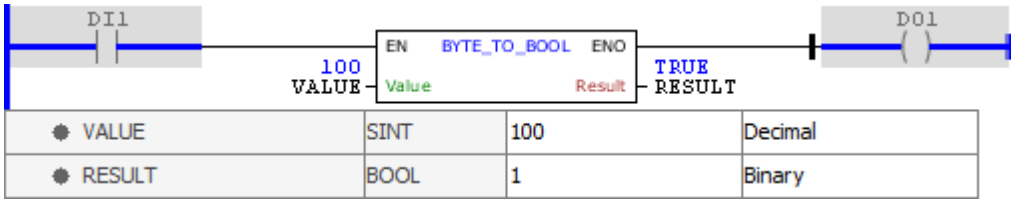
Example



VALUE	SINT	0	Decimal
RESULT	BOOL	0	Binary



VALUE	SINT	-10	Decimal
RESULT	BOOL	1	Binary



VALUE	SINT	100	Decimal
RESULT	BOOL	1	Binary

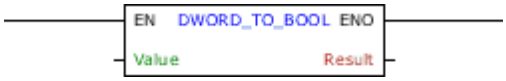
The examples above perform the conversion of VALUE variable, in BYTE, into a BOOL value storing

the final result in RESULT. The block ends with success and ENO output is activated.

11.14.7.6.1.2 DWORD_TO_BOOL

Block that performs the conversion of a DWORD value into a BOOL value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	DWORD UDINT DINT	Value in DWORD
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BOOL	Value in BOOL

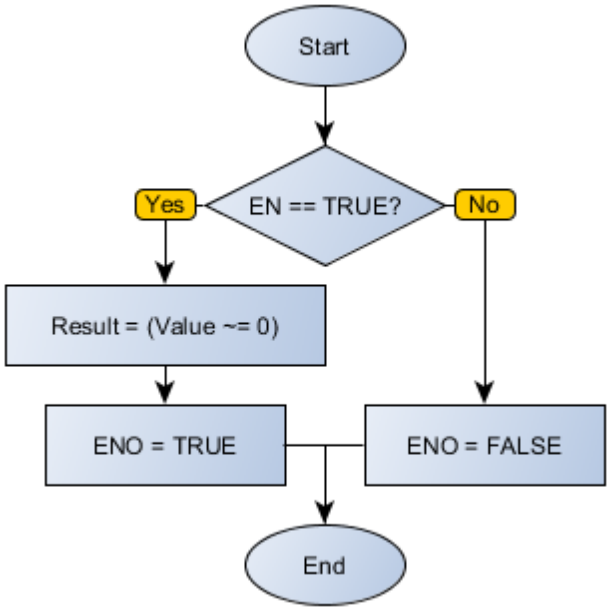
Operation

When this block has a TRUE value in EN, it interprets the Value value as DWORD and converts it into BOOL, storing in Result.

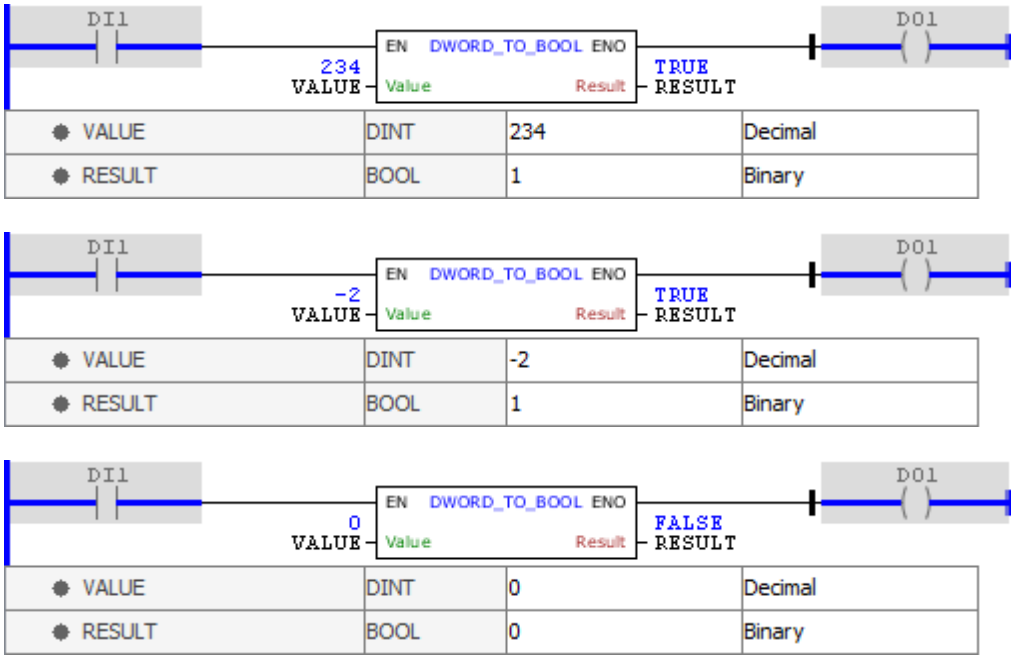
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example

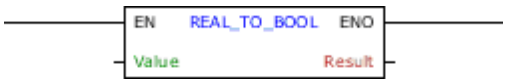


The examples above perform the conversion of VALUE variable, in DWORD, into a BOOL value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.14.7.6.1.3 REAL_TO_BOOL

Block that performs the conversion of a REAL value into a BOOL value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Value in REAL
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BOOL	Value in BOOL

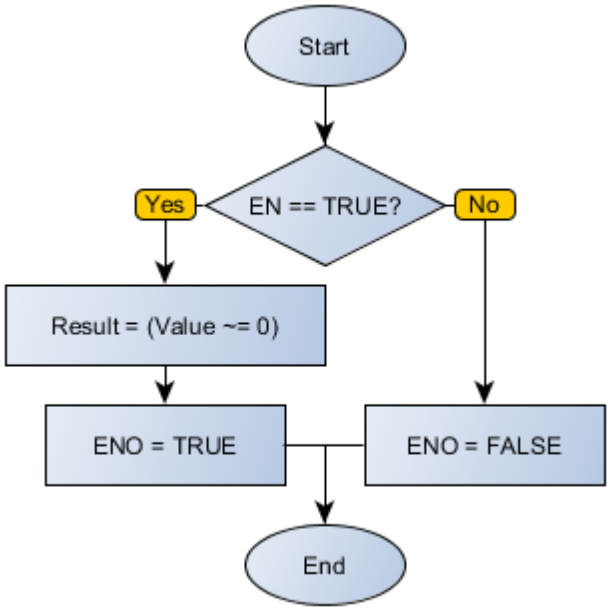
Operation

When this block has a TRUE value in EN, it interprets the Value value as REAL and converts it into BOOL, storing in Result.

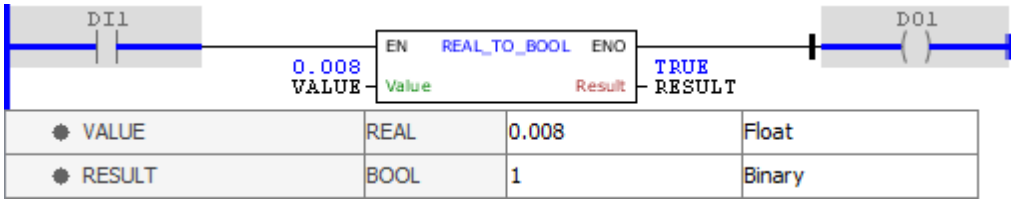
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

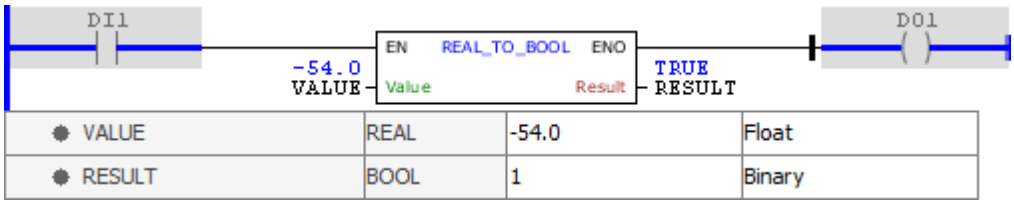
Block Flowchart



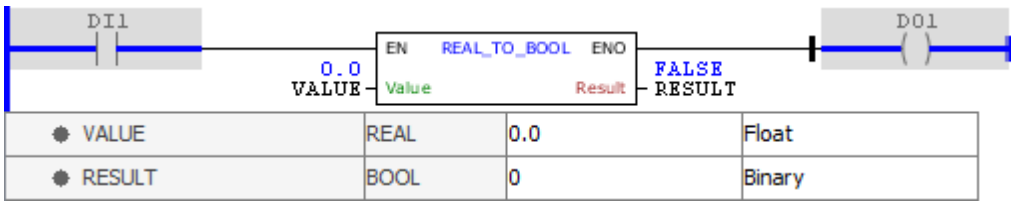
Example



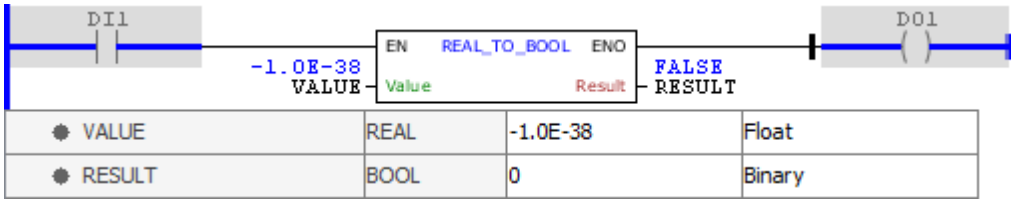
VALUE	REAL	0.008	Float
RESULT	BOOL	1	Binary



VALUE	REAL	-54.0	Float
RESULT	BOOL	1	Binary



VALUE	REAL	0.0	Float
RESULT	BOOL	0	Binary



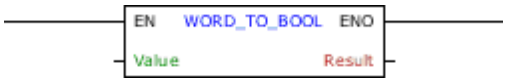
VALUE	REAL	-1.0E-38	Float
RESULT	BOOL	0	Binary

The examples above perform the conversion of VALUE variable, in REAL, into a BOOL value storing the final result in RESULT. The block ends with success and ENO output is activated. Notice in the last example that the values very close to the machine epsilon may result in an interpretation of the FALSE value.

11.14.7.6.1.4 WORD_TO_BOOL

Block that performs the conversion of a WORD value into a BOOL value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	WORD UINT INT	Value in WORD
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BOOL	Value in BOOL

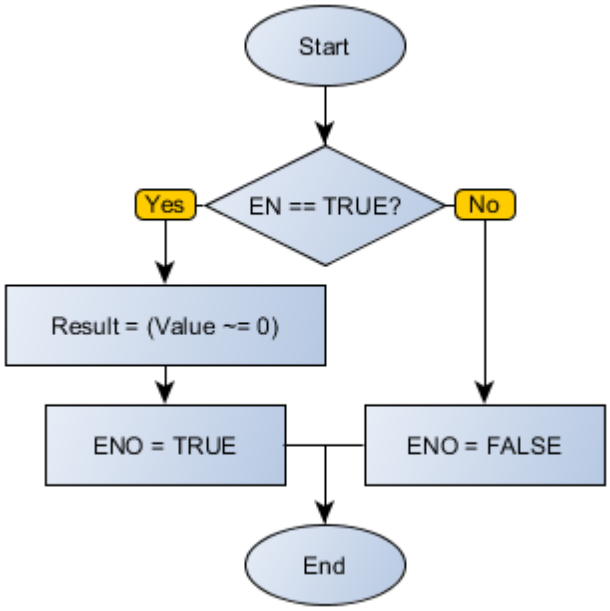
Operation

When this block has a TRUE value in EN, it interprets the Value value as WORD and converts it into BOOL, storing in Result.

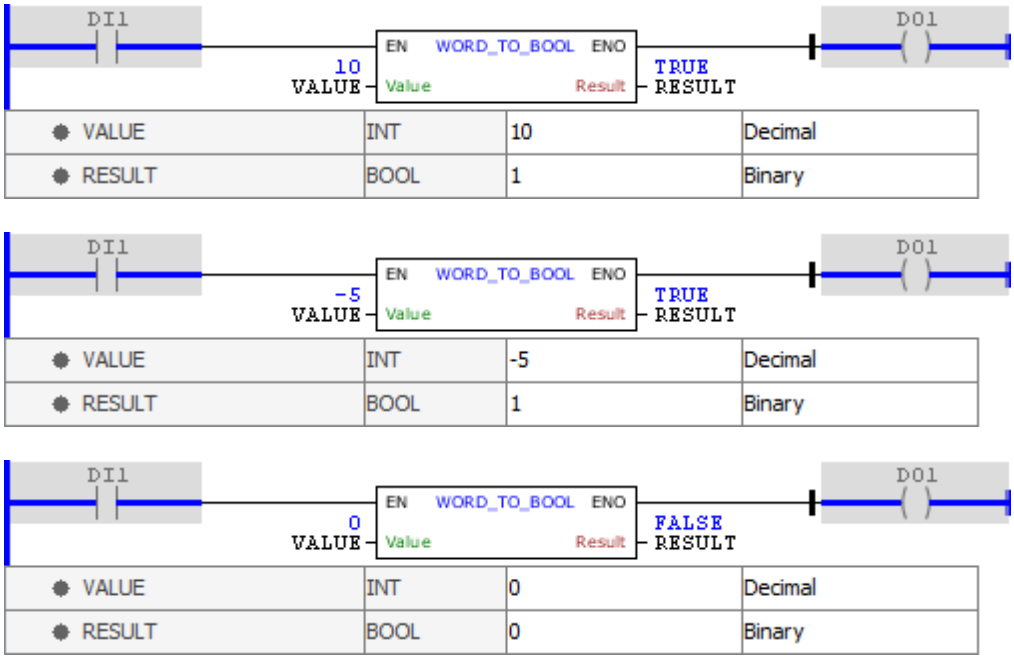
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



VALUE	INT	10	Decimal
RESULT	BOOL	1	Binary

VALUE	INT	-5	Decimal
RESULT	BOOL	1	Binary

VALUE	INT	0	Decimal
RESULT	BOOL	0	Binary

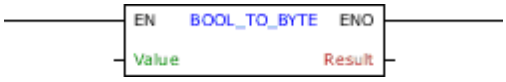
The examples above perform the conversion of VALUE variable, in WORD, into a BOOL value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.14.7.6.2 BYTE

11.14.7.6.2.1 BOOL_TO_BYTE

Block that performs the conversion of a BOOL value into a BYTE value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BOOL	Value in BOOL
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT	Value in BYTE

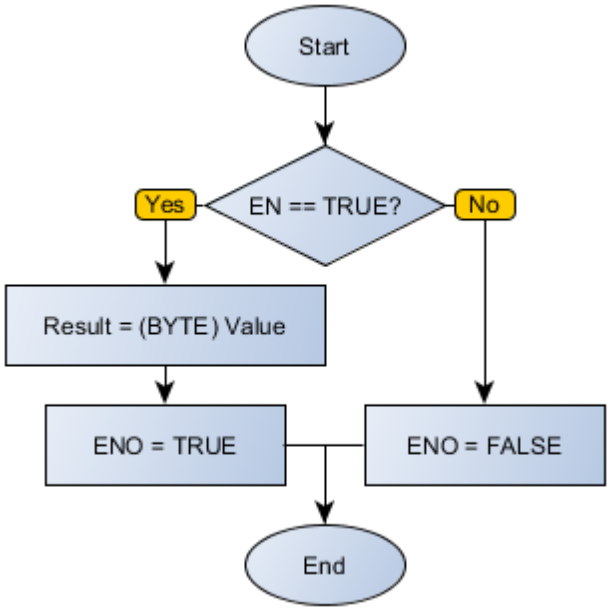
Operation

When this block has a TRUE value in EN, it interprets the Value value as BOOL and converts it into BYTE, storing in Result.

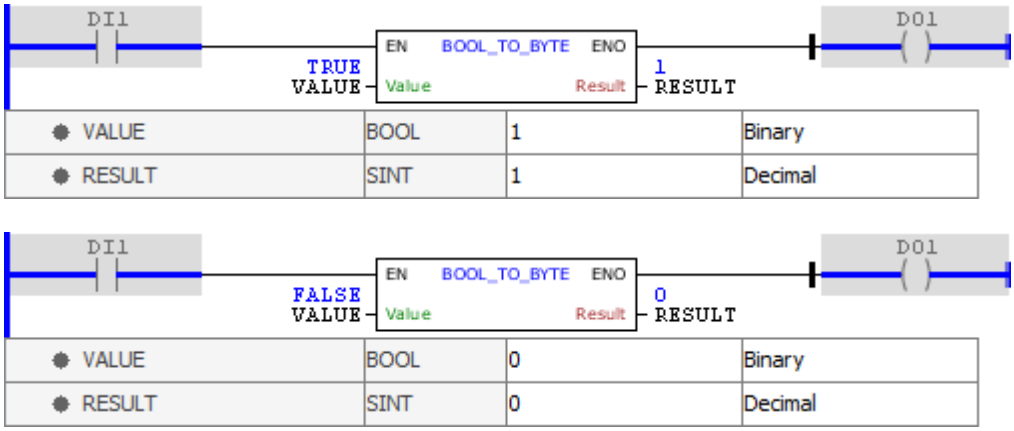
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



VALUE	BOOL	1	Binary
RESULT	SINT	1	Decimal

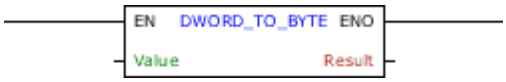
VALUE	BOOL	0	Binary
RESULT	SINT	0	Decimal

The examples above perform the conversion of variable VALUE, in BOOL, into a BYTE value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.14.7.6.2.2 DWORD_TO_BYTE

Block that performs the conversion of a DWORD value into a BYTE value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	DWORD UDINT DINT	Value in DWORD
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT	Value in BYTE

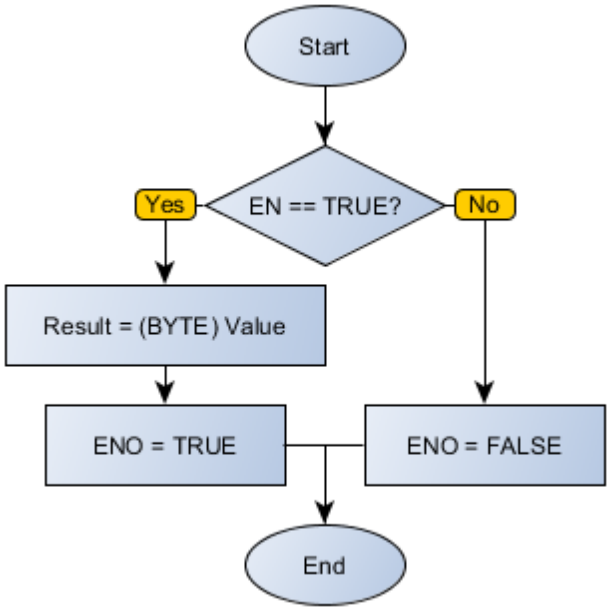
Operation

When this block has a TRUE value in EN, it interprets the Value value as DWORD and converts it into BYTE, storing in Result.

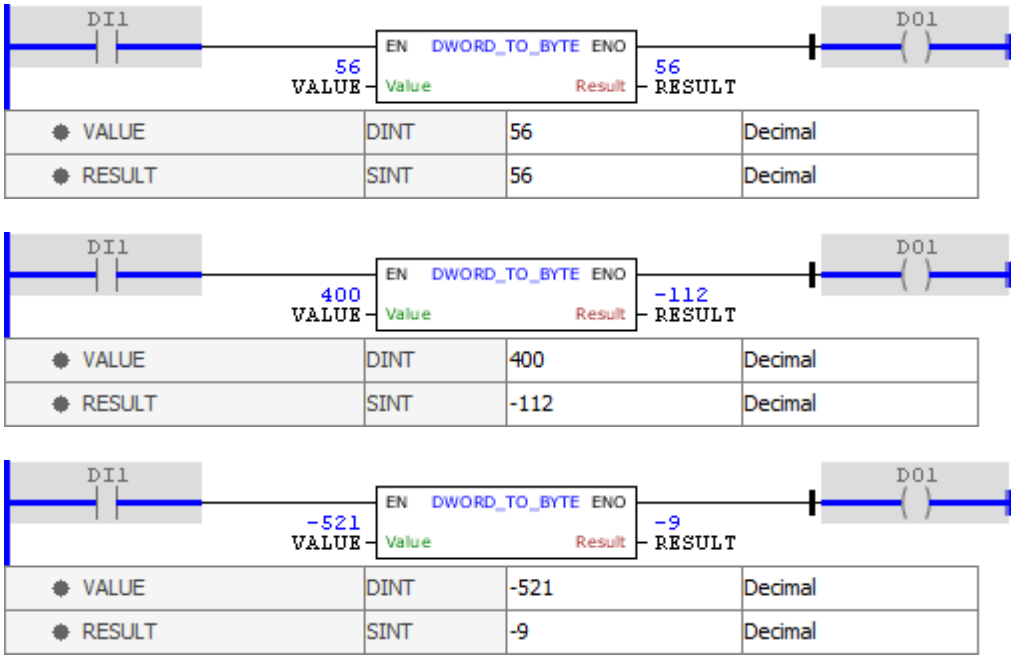
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example

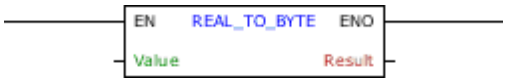


The examples above perform the conversion of variable VALUE, in DWORD, into a BYTE value storing the final result in RESULT. The block ends with success and ENO output is activated. Notice that only the eight least significant bits are taken into account.

11.14.7.6.2.3 REAL_TO_BYTE

Block that performs the conversion of a REAL value into a BYTE value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Value in REAL
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT	Value in BYTE

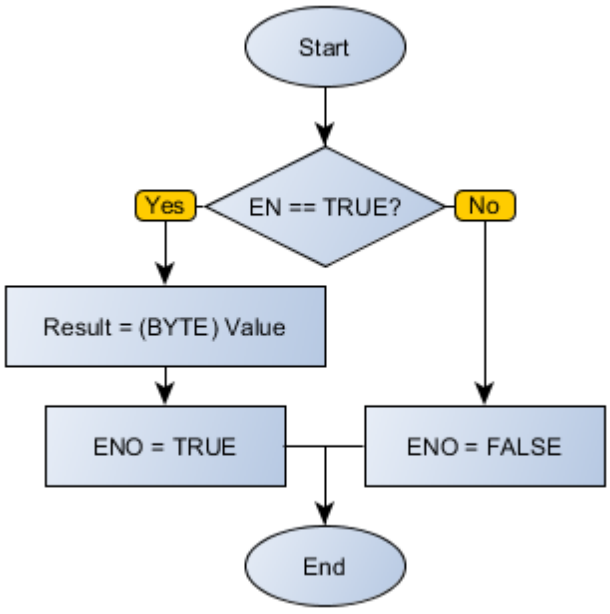
Operation

When this block has a TRUE value in EN, it interprets the Value value as REAL and converts it into BYTE, storing in Result.

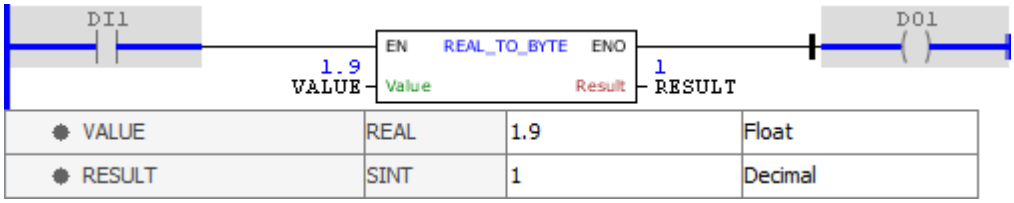
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

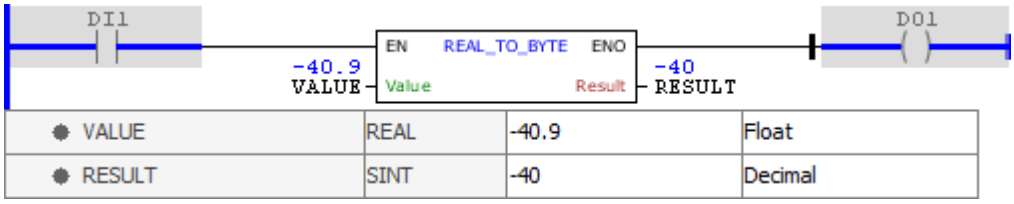
Block Flowchart



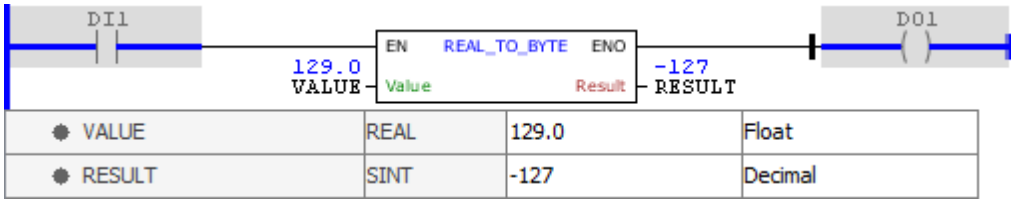
Example



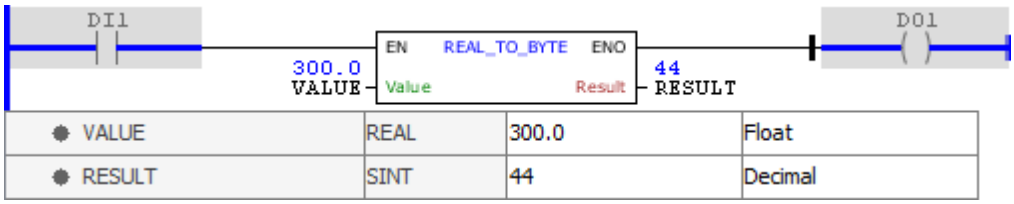
VALUE	REAL	1.9	Float
RESULT	SINT	1	Decimal



VALUE	REAL	-40.9	Float
RESULT	SINT	-40	Decimal



VALUE	REAL	129.0	Float
RESULT	SINT	-127	Decimal



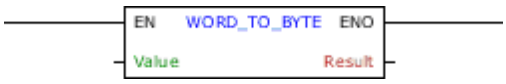
VALUE	REAL	300.0	Float
RESULT	SINT	44	Decimal

The examples above perform the conversion of variable VALUE, in REAL, into a BYTE value storing the final result in RESULT. The block ends with success and ENO output is activated. Notice that the results are truncated in decimal and only the eight least significant bits are taken into account.

11.14.7.6.2.4 WORD_TO_BYTE

Block that performs the conversion of a WORD value into a BYTE value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	WORD UINT INT	Value in WORD
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT	Value in BYTE

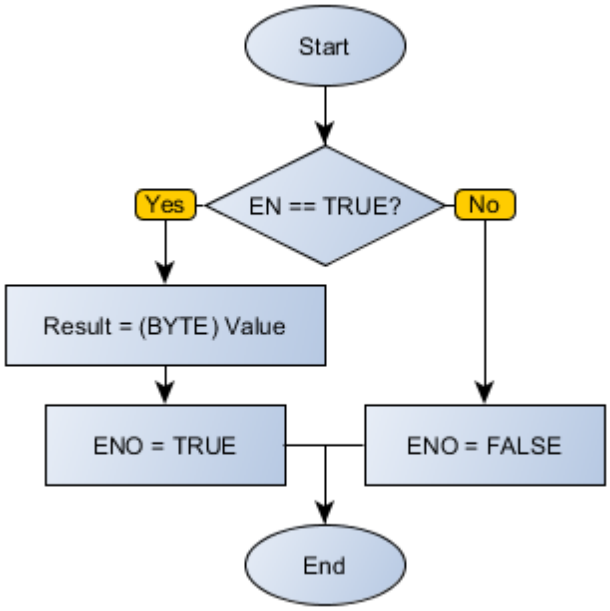
Operation

When this block has a TRUE value in EN, it interprets the Value value as WORD and converts it into BYTE, storing in Result.

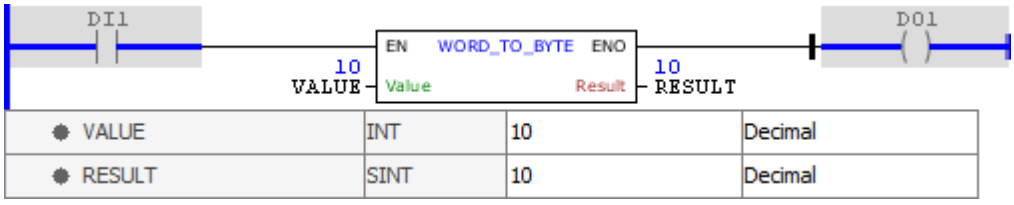
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

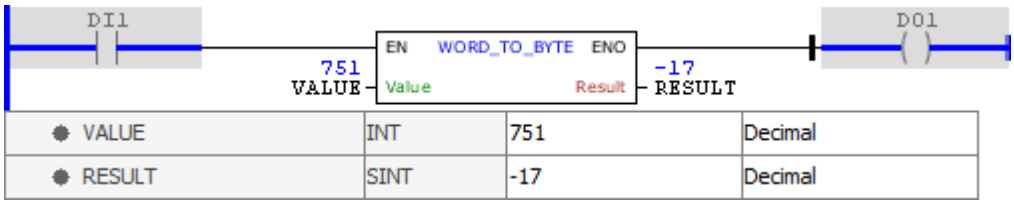
Block Flowchart



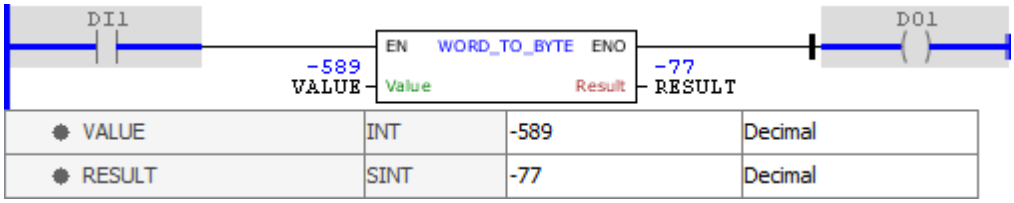
Example



VALUE	INT	10	Decimal
RESULT	SINT	10	Decimal



VALUE	INT	751	Decimal
RESULT	SINT	-17	Decimal



VALUE	INT	-589	Decimal
RESULT	SINT	-77	Decimal

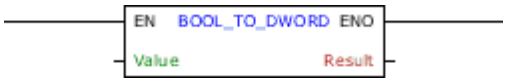
The examples above perform the conversion of variable VALUE, in WORD, into a BYTE value storing the final result in RESULT. The block ends with success and ENO output is activated. Notice that only the eight least significant bits are taken into account.

11.14.7.6.3 DWORD

11.14.7.6.3.1 BOOL_TO_DWORD

Block that performs the conversion of a BOOL value into a DWORD value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BOOL	Value in BOOL
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	DWORD UDINT DINT	Value in DWORD

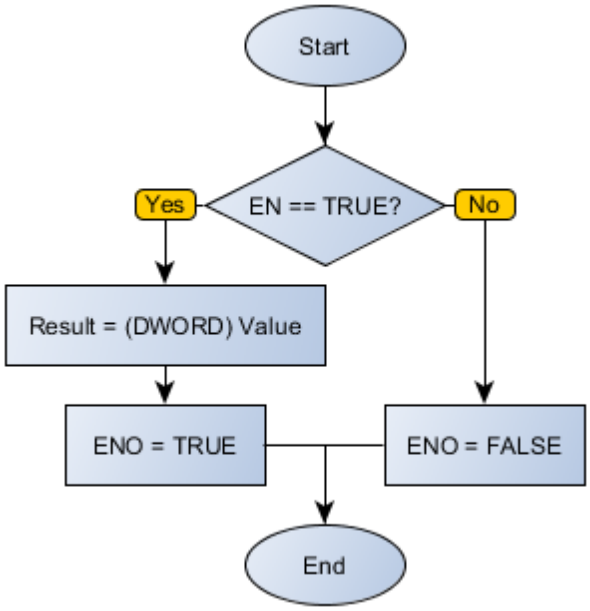
Operation

When this block has a TRUE value in EN, it interprets the Value value as BOOL and converts it into DWORD, storing in Result.

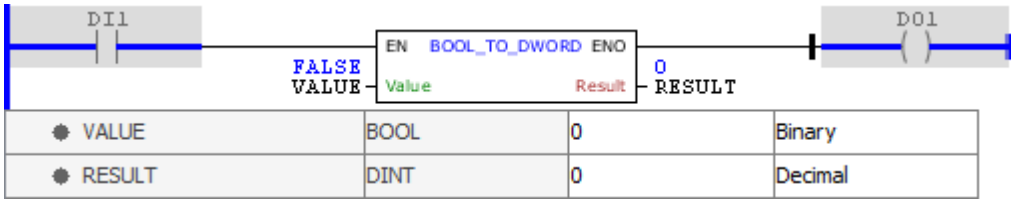
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

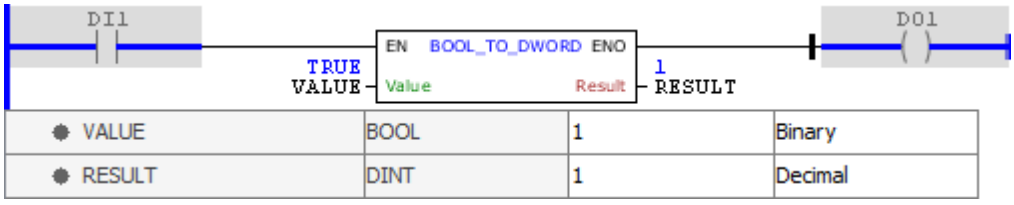
Block Flowchart



Example



VALUE	BOOL	0	Binary
RESULT	DINT	0	Decimal



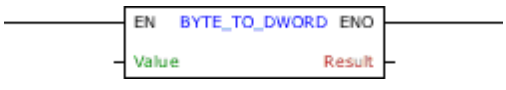
VALUE	BOOL	1	Binary
RESULT	DINT	1	Decimal

The examples above perform the conversion of VALUE variable, in BOOL, into a DWORD value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.14.7.6.3.2 BYTE_TO_DWORD

Block that performs the conversion of a BYTE value into a DWORD value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT	Value in BYTE
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	DWORD UDINT DINT	Value in DWORD

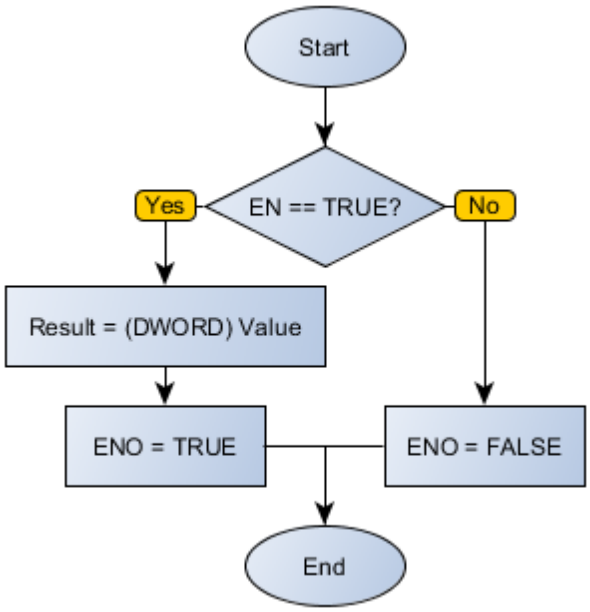
Operation

When this block has a TRUE value in EN, it interprets the Value value as BYTE and converts it into DWORD, storing in Result.

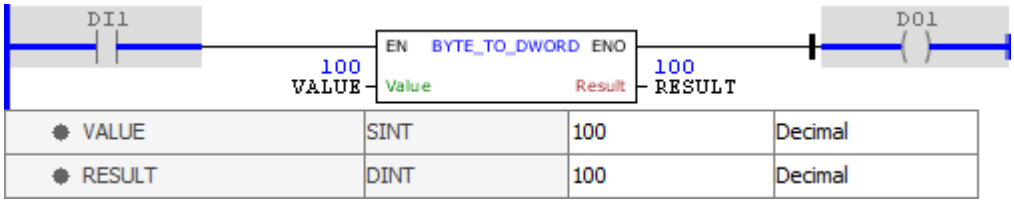
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

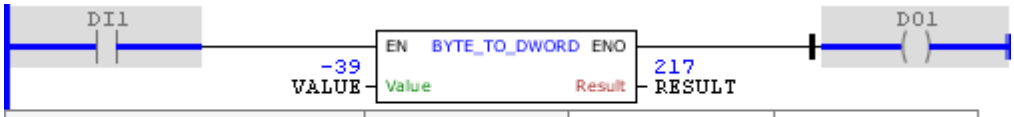
Block Flowchart



Example



VALUE	SINT	100	Decimal
RESULT	DINT	100	Decimal



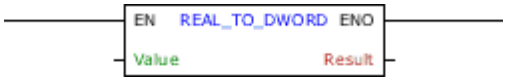
VALUE	SINT	-39	Decimal
RESULT	DINT	217	Decimal

The examples above perform the conversion of variable VALUE, in BYTE, into a DWORD value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.14.7.6.3.3 REAL_TO_DWORD

Block that performs the conversion of a REAL value into a DWORD value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Value in REAL
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	DWORD UDINT DINT	Value in DWORD

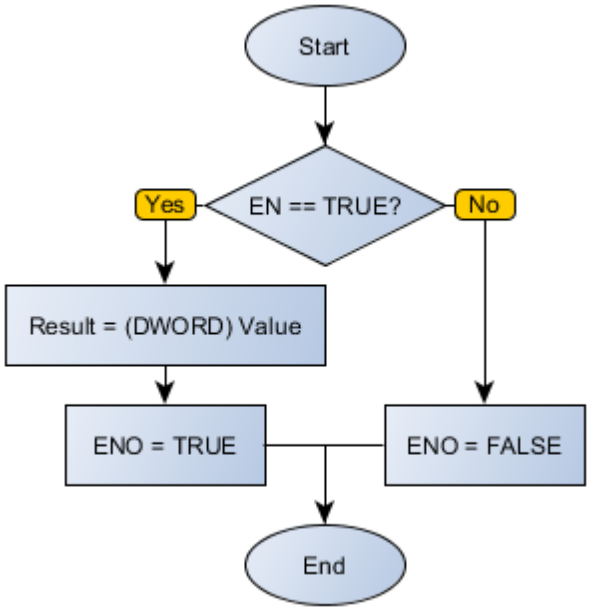
Operation

When this block has a TRUE value in EN, it interprets the Value value as REAL and converts it into DWORD, storing in Result.

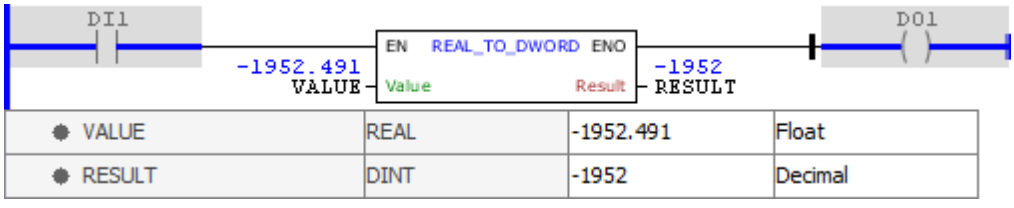
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

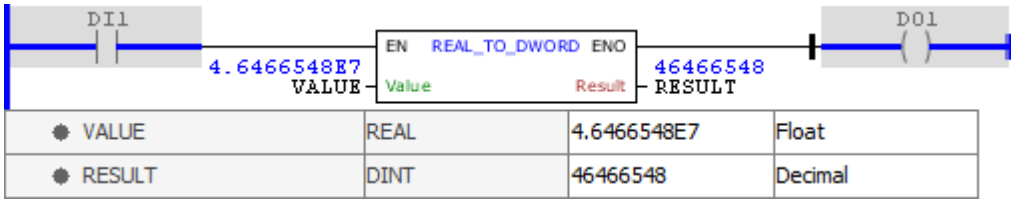
Block Flowchart



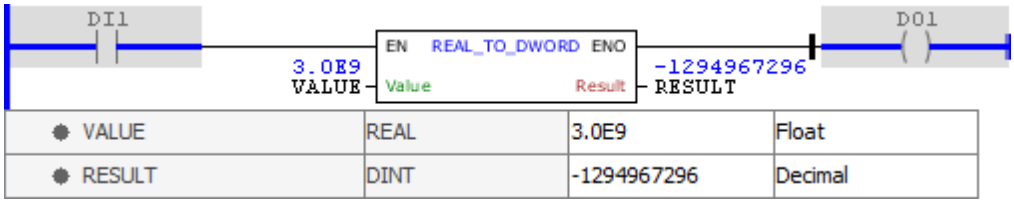
Example



VALUE	REAL	-1952.491	Float
RESULT	DINT	-1952	Decimal



VALUE	REAL	4.6466548E7	Float
RESULT	DINT	46466548	Decimal



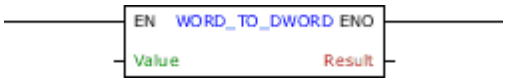
VALUE	REAL	3.0E9	Float
RESULT	DINT	-1294967296	Decimal

The examples above perform the conversion of variable VALUE, in REAL, into a DWORD value storing the final result in RESULT. The block ends with success and ENO output is activated. Note that the results are truncated in decimal and only the thirty-two least significant bits are taken into account.

11.14.7.6.3.4 WORD_TO_DWORD

Block that performs the conversion of a WORD value into a DWORD value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	WORD UINT INT	Value in WORD
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	DWORD UDINT DINT	Value in DWORD

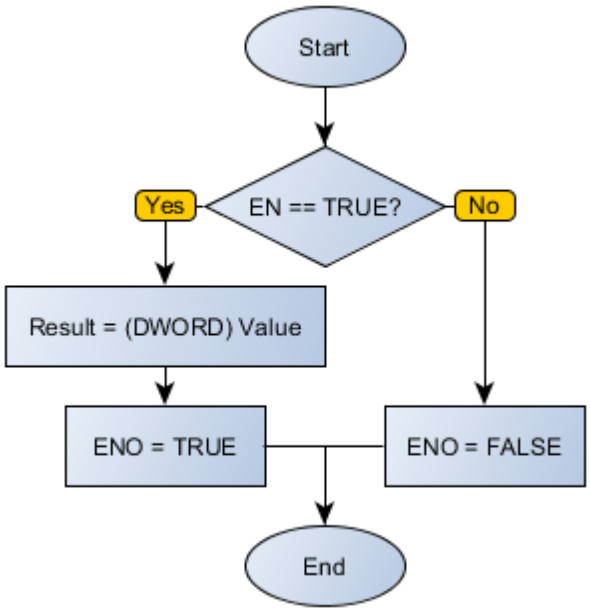
Operation

When this block has a TRUE value in EN, it interprets the Value value as WORD and converts it into DWORD, storing in Result.

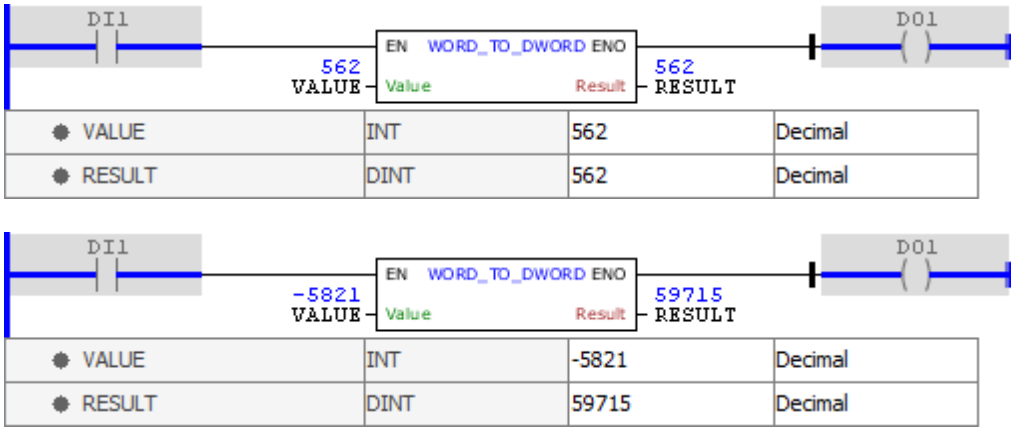
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



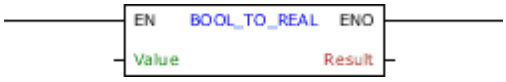
The examples above convert the VALUE variable, in WORD, into a DWORD value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.14.7.6.4 REAL

11.14.7.6.4.1 BOOL_TO_REAL

Block that performs the conversion of a BOOL value into a REAL value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BOOL	Value in BOOL
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	REAL	Value in REAL

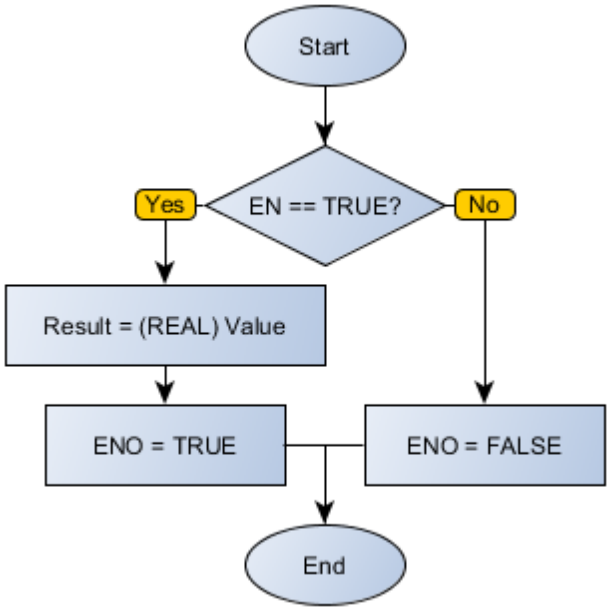
Operation

When this block has a TRUE value in EN, it interprets the Value value as BOOL and converts it into REAL, storing in Result.

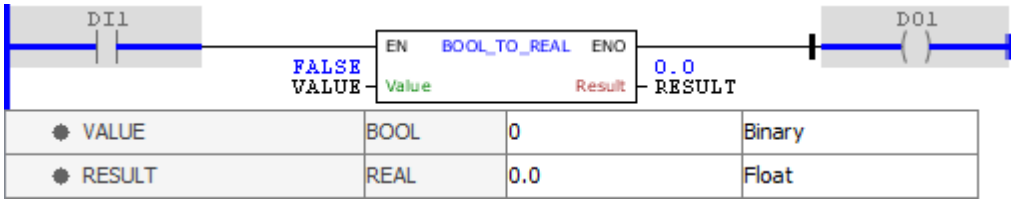
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

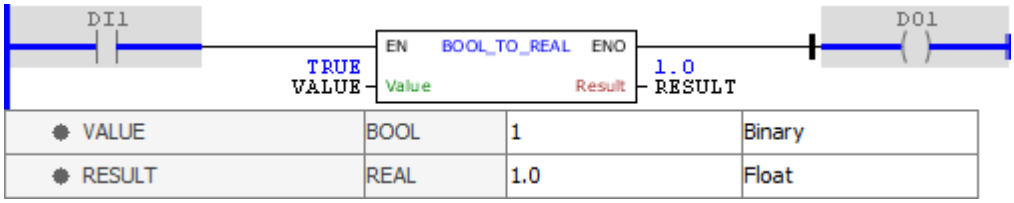
Block Flowchart



Example



VALUE	BOOL	0	Binary
RESULT	REAL	0.0	Float



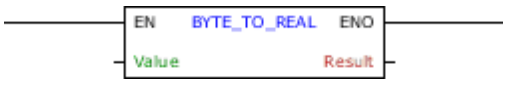
VALUE	BOOL	1	Binary
RESULT	REAL	1.0	Float

The examples above perform the conversion of variable VALUE, in BOOL, into a REAL value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.14.7.6.4.2 BYTE_TO_REAL

Block that performs the conversion of a BYTE value into a REAL value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT	Value in BYTE
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	REAL	Value in REAL

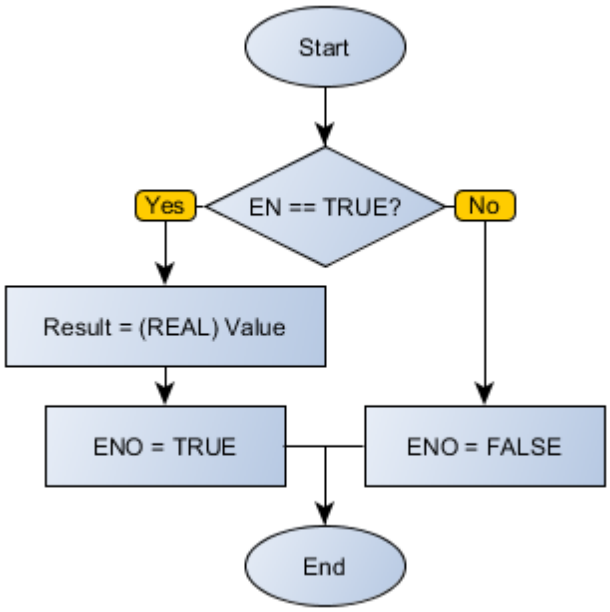
Operation

When this block has a TRUE value in EN, it interprets the Value value as BYTE and converts it into REAL, storing in Result.

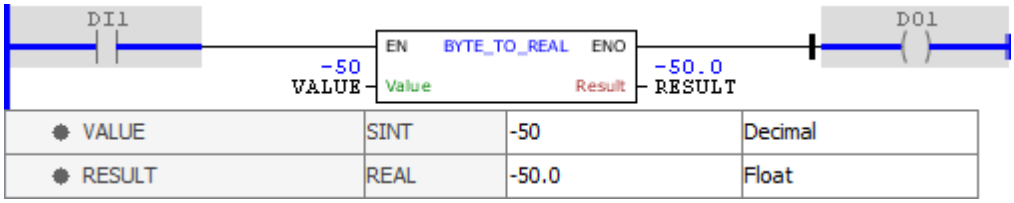
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

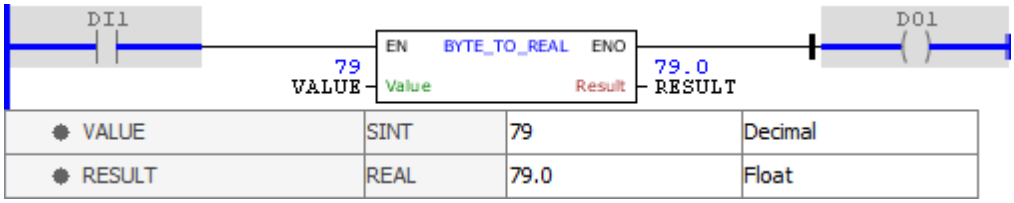
Block Flowchart



Example



VALUE	SINT	-50	Decimal
RESULT	REAL	-50.0	Float



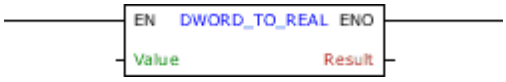
VALUE	SINT	79	Decimal
RESULT	REAL	79.0	Float

The examples above perform the conversion of variable VALUE, in BYTE, into a REAL value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.14.7.6.4.3 DWORD_TO_REAL

Block that performs the conversion of a DWORD value into a REAL value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	DWORD UDINT DINT	Value in DWORD
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	REAL	Value in REAL

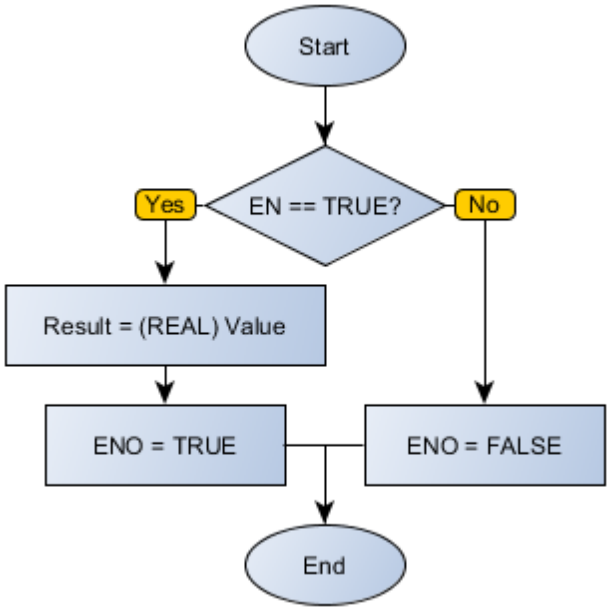
Operation

When this block has a TRUE value in EN, it interprets the Value value as DWORD and converts it into REAL, storing in Result.

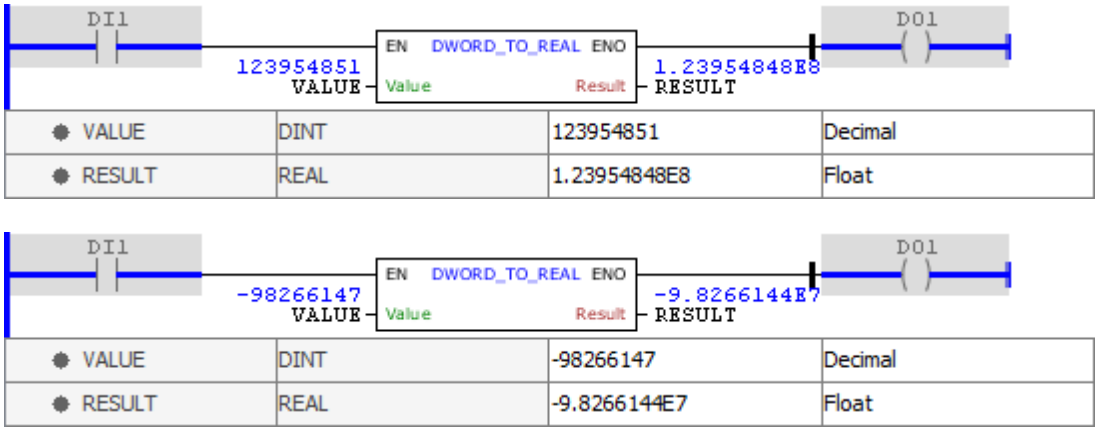
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



VALUE	DINT	123954851	Decimal
RESULT	REAL	1.23954848E8	Float

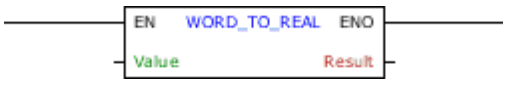
VALUE	DINT	-98266147	Decimal
RESULT	REAL	-9.8266144E7	Float

The examples above perform the conversion of variable VALUE, in DWORD, into a REAL value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.14.7.6.4.4 WORD_TO_REAL

Block that performs the conversion of a WORD value into a REAL value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	WORD UINT INT	Value in WORD
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	REAL	Value in REAL

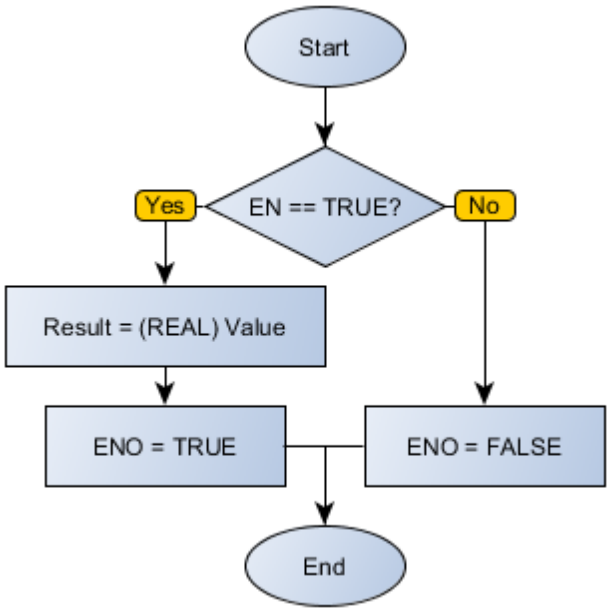
Operation

When this block has a TRUE value in EN, it interprets the Value value as WORD and converts it into REAL, storing in Result.

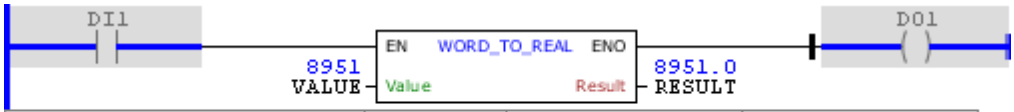
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

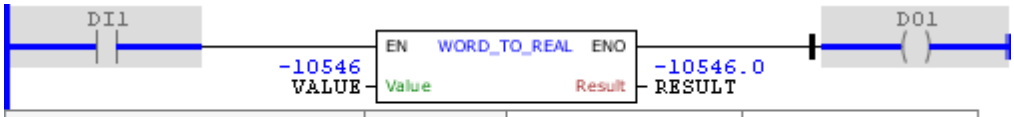
Block Flowchart



Example



VALUE	INT	8951	Decimal
RESULT	REAL	8951.0	Float



VALUE	INT	-10546	Decimal
RESULT	REAL	-10546.0	Float

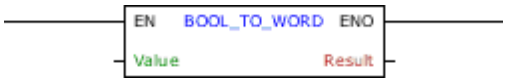
The examples above perform the conversion of variable VALUE, in WORD, into a REAL value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.14.7.6.5 WORD

11.14.7.6.5.1 BOOL_TO_WORD

Block that performs the conversion of a BOOL value into a WORD value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BOOL	Value in BOOL
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	WORD UINT INT	Value in WORD

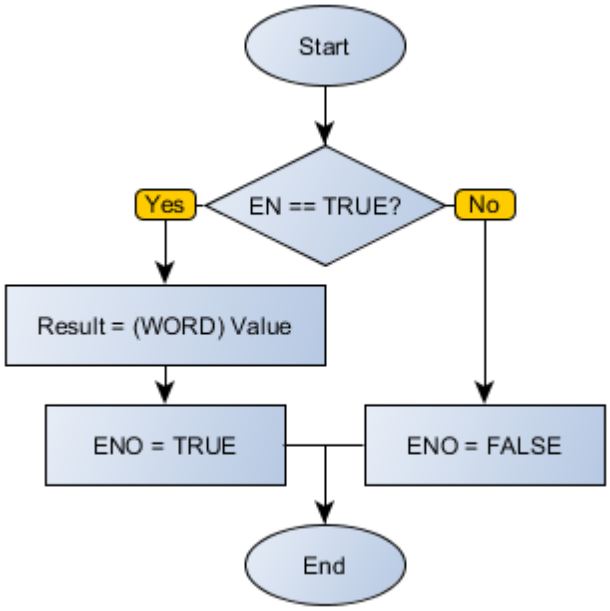
Operation

When this block has a TRUE value in EN, it interprets the Value value as BOOL and converts it into WORD, storing in Result.

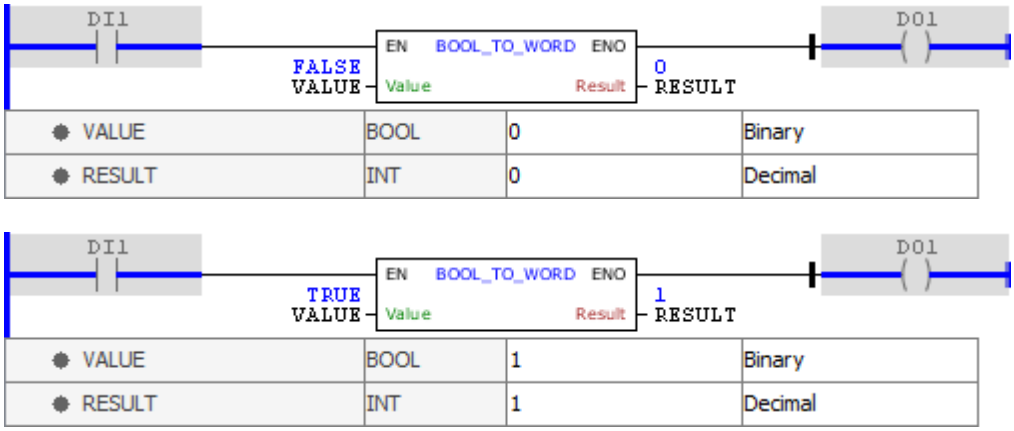
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example

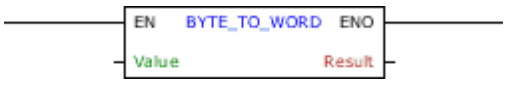


The examples above perform the conversion of VALUE variable, in BOOL, into a WORD value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.14.7.6.5.2 BYTE_TO_WORD

Block that performs the conversion of a BYTE value into a WORD value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT	Value in BYTE
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	WORD UINT INT	Value in WORD

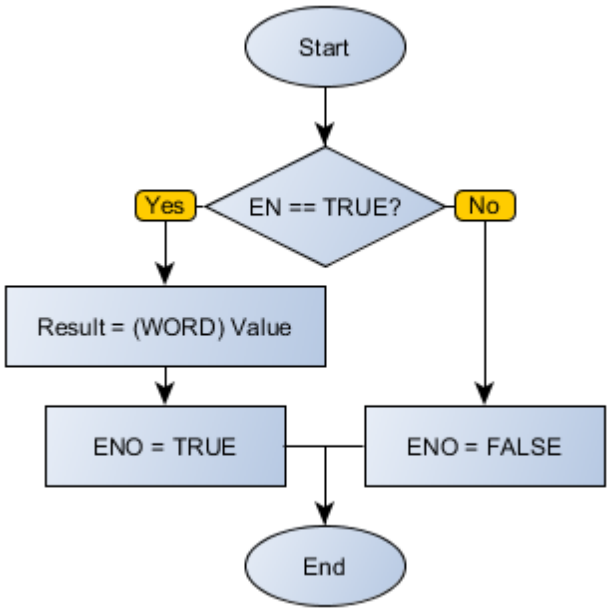
Operation

When this block has a TRUE value in EN, it interprets the Value value as BYTE and converts it into WORD, storing in Result.

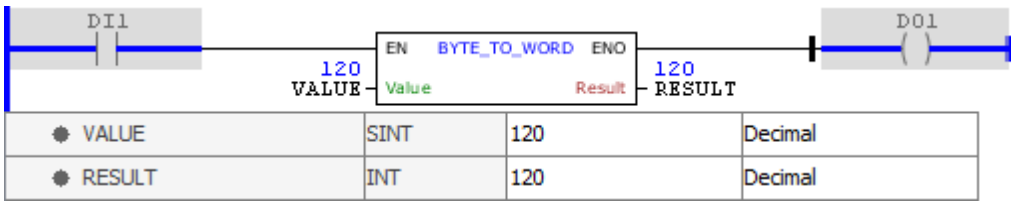
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

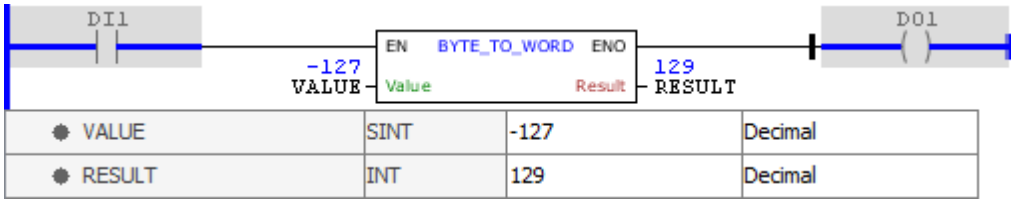
Block Flowchart



Example



VALUE	SINT	120	Decimal
RESULT	INT	120	Decimal



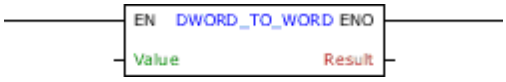
VALUE	SINT	-127	Decimal
RESULT	INT	129	Decimal

The examples above perform the conversion of variable VALUE, in BYTE, into a WORD value storing the final result in RESULT. The block ends with success and ENO output is activated.

11.14.7.6.5.3 DWORD_TO_WORD

Block that performs the conversion of a DWORD value into a WORD value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	DWORD UDINT DINT	Value in DWORD
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	WORD UINT INT	Value in WORD

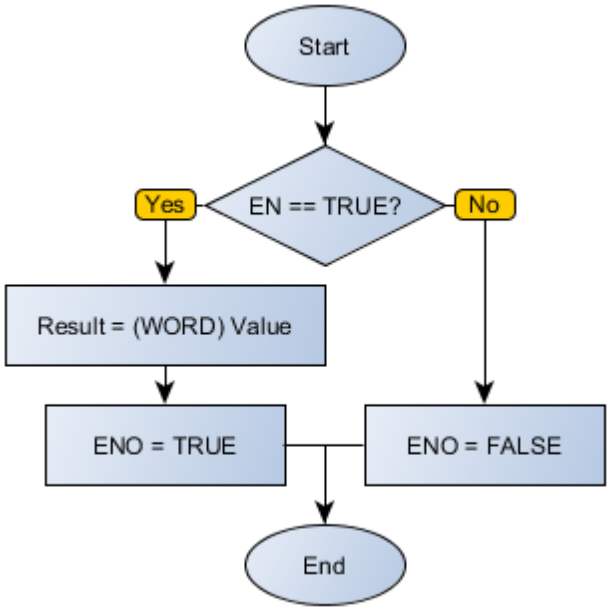
Operation

When this block has a TRUE value in EN, it interprets the Value value as DWORD and converts it into WORD, storing in Result.

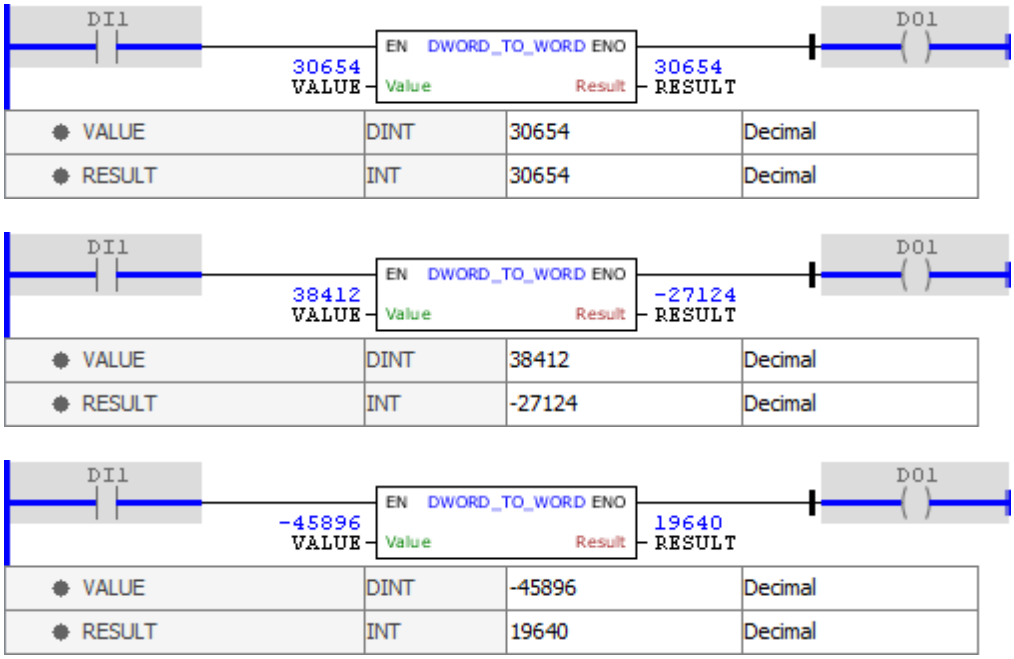
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example

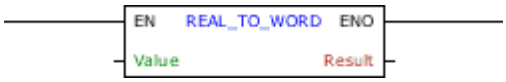


The examples above convert the VALUE variable, in DWORD, into a WORD value storing the final result in RESULT. The block ends with success and ENO output is activated. Notice that only the sixteen least significant bits are taken into account.

11.14.7.6.5.4 REAL_TO_WORD

Block that performs the conversion of a REAL value into a WORD value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Value in REAL
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	WORD UINT INT	Value in WORD

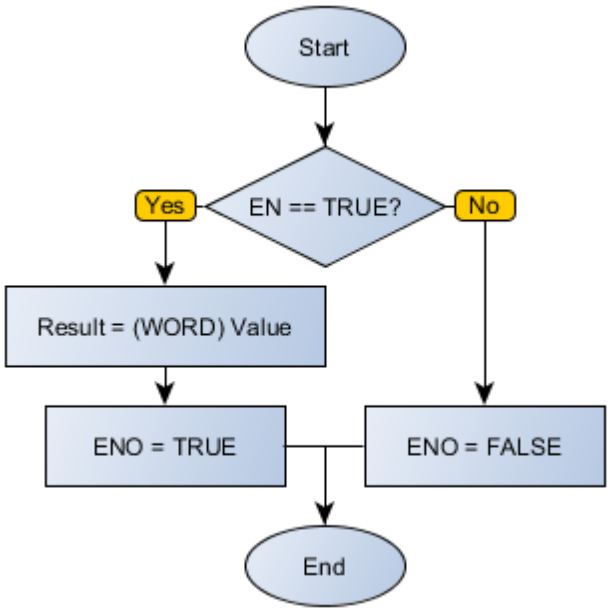
Operation

When this block has a TRUE value in EN, it interprets the Value value as REAL and converts it into WORD, storing in Result.

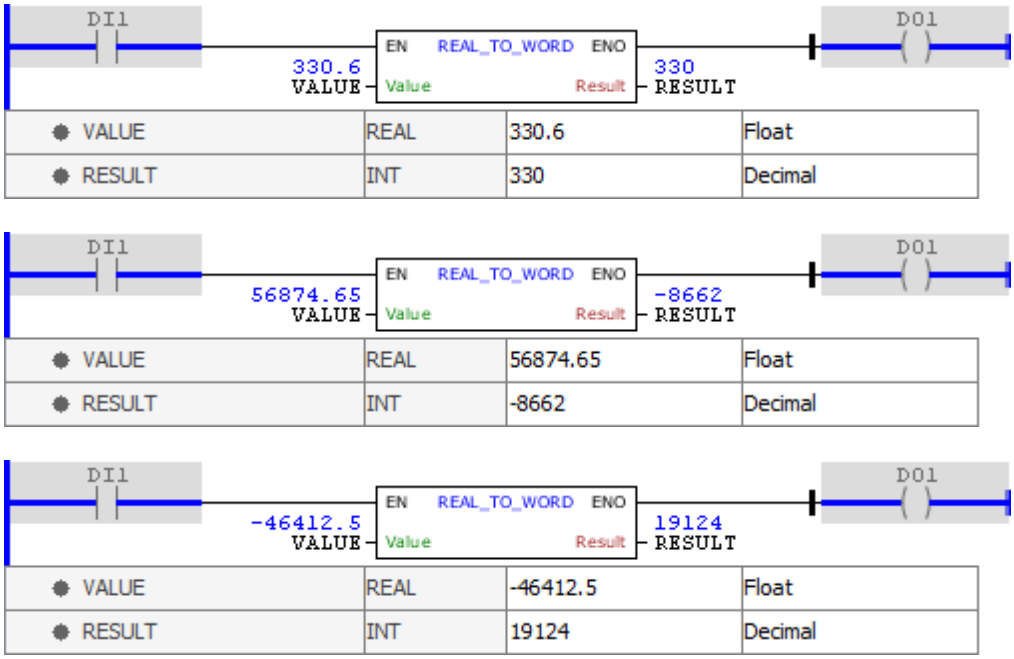
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



VALUE	REAL	330.6	Float
RESULT	INT	330	Decimal

VALUE	REAL	56874.65	Float
RESULT	INT	-8662	Decimal

VALUE	REAL	-46412.5	Float
RESULT	INT	19124	Decimal

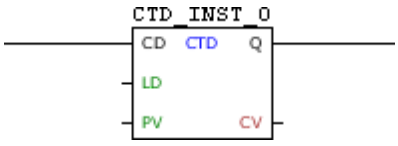
The examples above convert the VALUE variable, in DWORD, into a WORD value storing the final result in RESULT. The block ends with success and ENO output is activated. Note that the results are truncated in decimal and only the sixteen least significant bits are taken into account.

11.14.7.7 Counter

11.14.7.7.1 CTD

Countdown block of input pulses.

Ladder Representation



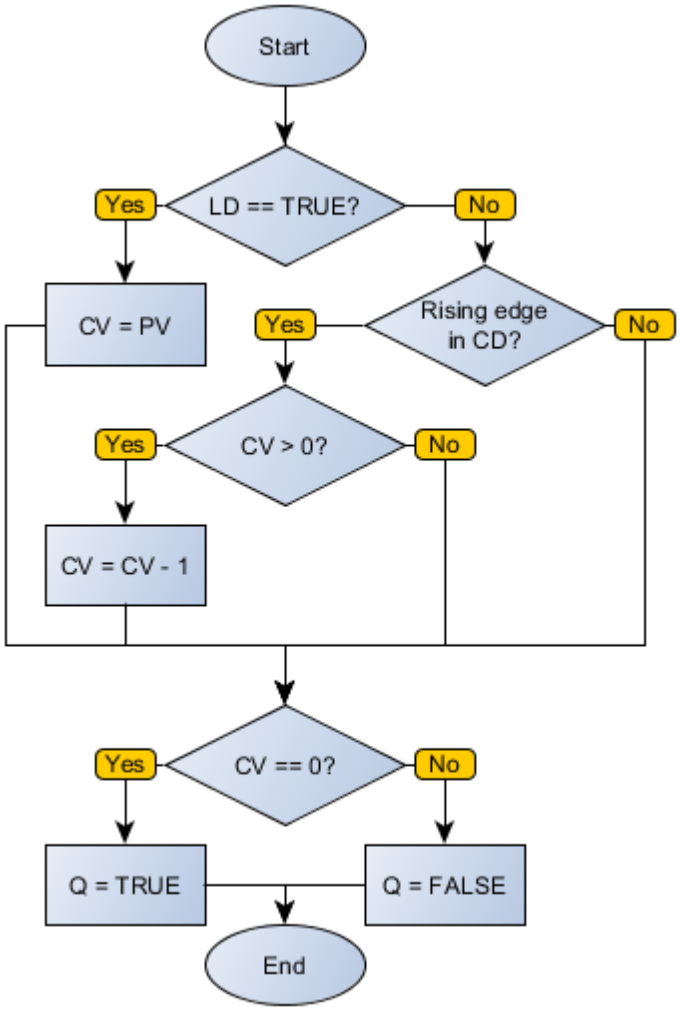
Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	CD	BOOL	Pulse identifier
	LD	BOOL	Loads the value of PV in CV
	PV	WORD UINT	Value of initial configuration
VAR_OUTPUT	Q	BOOL	Counter zeroed flag
	CV	WORD UINT	Current count value
VAR	CTD_INST_0	CTD	Instance of access to block structure

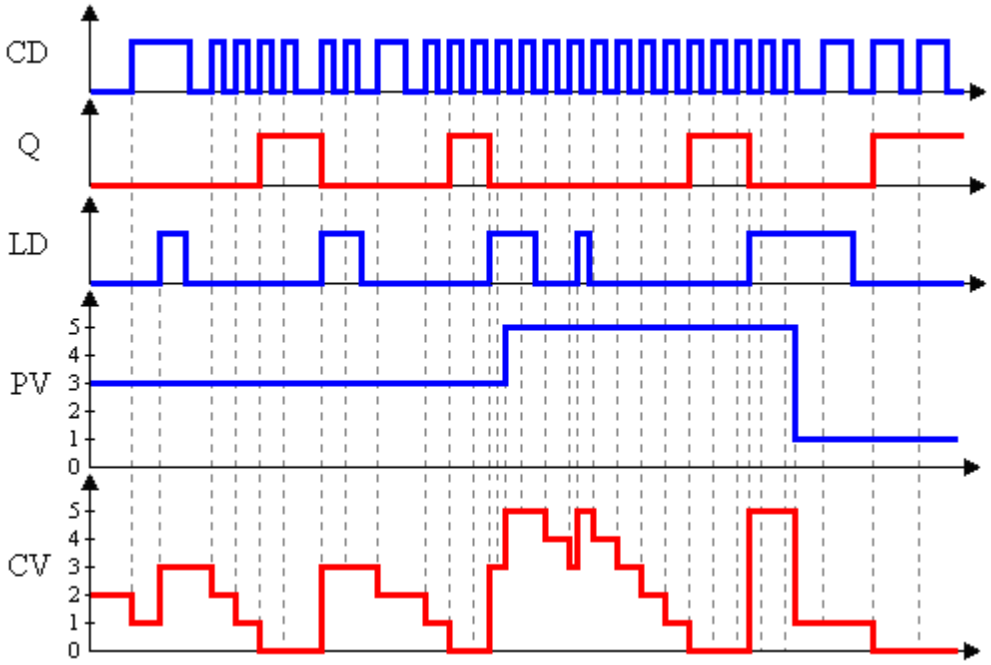
Operation

When this block identifies a leading edge in CD, it decrements the CV variable until it is zero. While CV equals zero, the output Q remains at TRUE level. By detecting high-level LD, the block loads the PV value in CV.

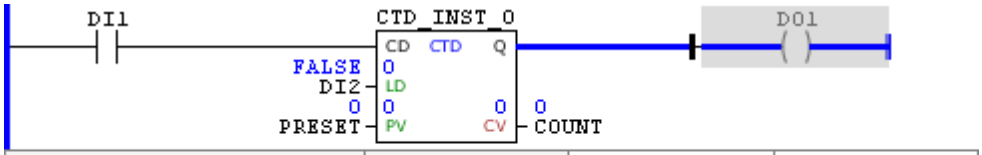
Block Flowchart



Operation Diagram

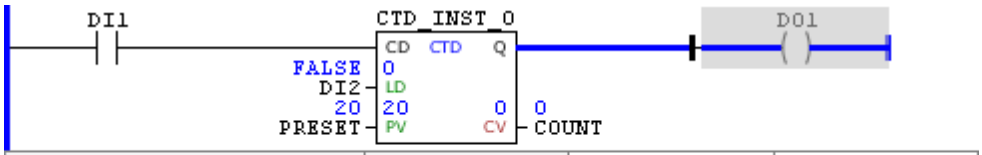


Example



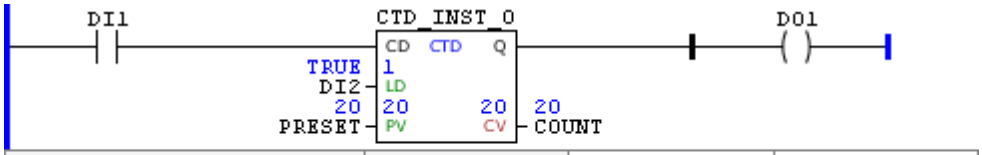
PRESET	UINT	0	Decimal
COUNT	UINT	0	Decimal

The above example shows the initial conditions of routine. As CV has a value of zero, the Q output is enabled.



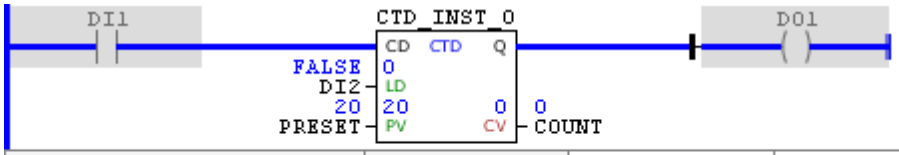
PRESET	UINT	20	Decimal
COUNT	UINT	0	Decimal

The value of the PV variable was changed to 20, but not yet loaded.



PRESET	UINT	20	Decimal
COUNT	UINT	20	Decimal

By identifying TRUE level in LD, the block loads the PV value to CV. Since this value is greater than zero, the Q output is disabled.



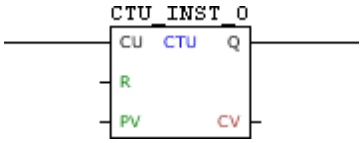
PRESET	UINT	20	Decimal
COUNT	UINT	0	Decimal

At each leading edge identified in CD, the value of COUNT is decremented until it reaches zero, when the Q output is enabled.

11.14.7.7.2 CTU

Block for gradual count of input pulses.

Ladder Representation



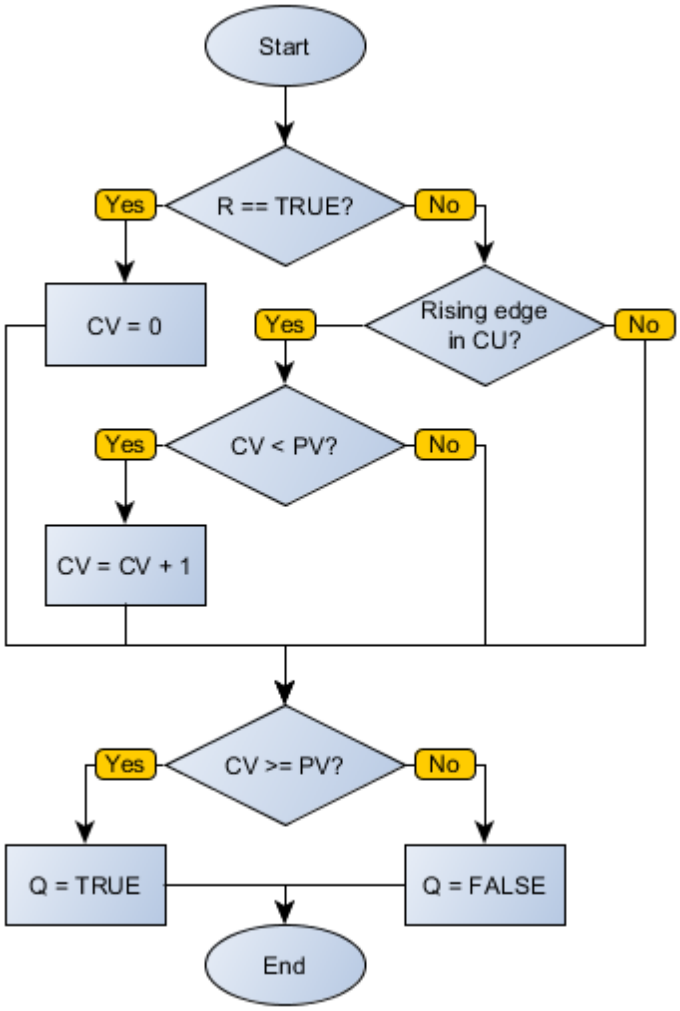
Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	CU	BOOL	Pulse identifier
	R	BOOL	Loads the zero value in CV
	PV	WORD UINT	Maximum count value
VAR_OUTPUT	Q	BOOL	Counter overrun flag
	CV	WORD UINT	Current count value
VAR	CTU_INST_0	CTU	Instance of access to block structure

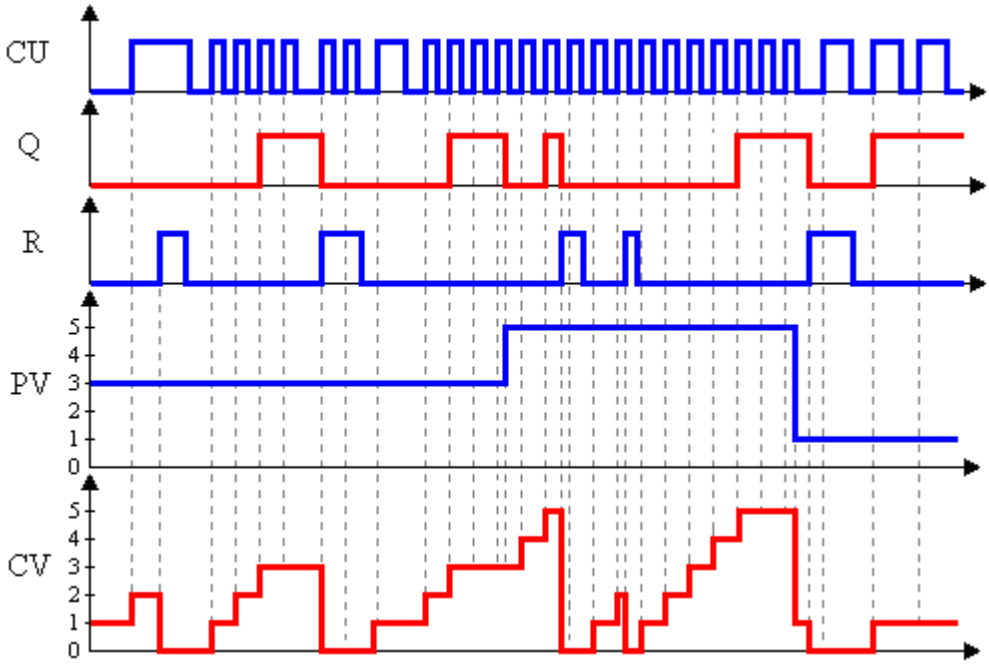
Operation

When this block identifies a leading edge in CD, it increments the CV variable until it is equal to PV. While CV equals PV, the output Q remains at TRUE level. By detecting high-level R, the block loads the zero value in CV.

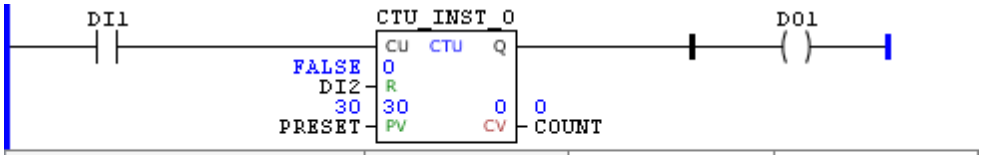
Block Flowchart



Operation Diagram

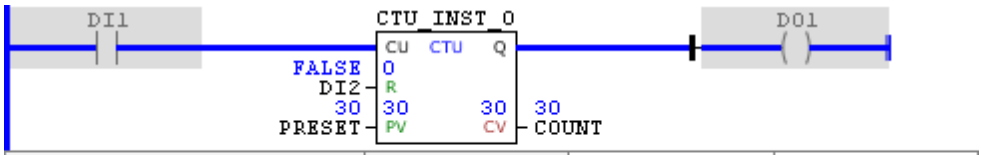


Example



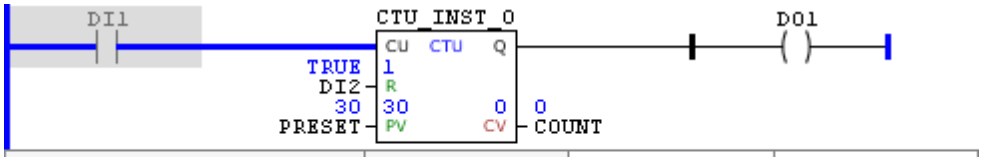
PRESET	UINT	30	Decimal
COUNT	UINT	0	Decimal

The above example shows the initial conditions of routine. Since CV has a lower value than of PV, the Q output is disabled.



PRESET	UINT	30	Decimal
COUNT	UINT	30	Decimal

At each leading edge identified in CU, the value of CV is incremented until it reaches the PV value, when the Q output is enabled.



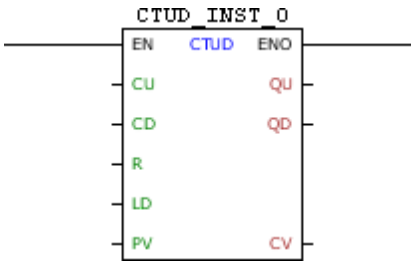
PRESET	UINT	30	Decimal
COUNT	UINT	0	Decimal

By identifying TRUE level in R, the block loads the zero value to CV. Since this value is lower than of PV, the Q output is disabled.

11.14.7.7.3 CTUD

Block for gradual count and countdown of input pulses.

Ladder Representation



Block Structure

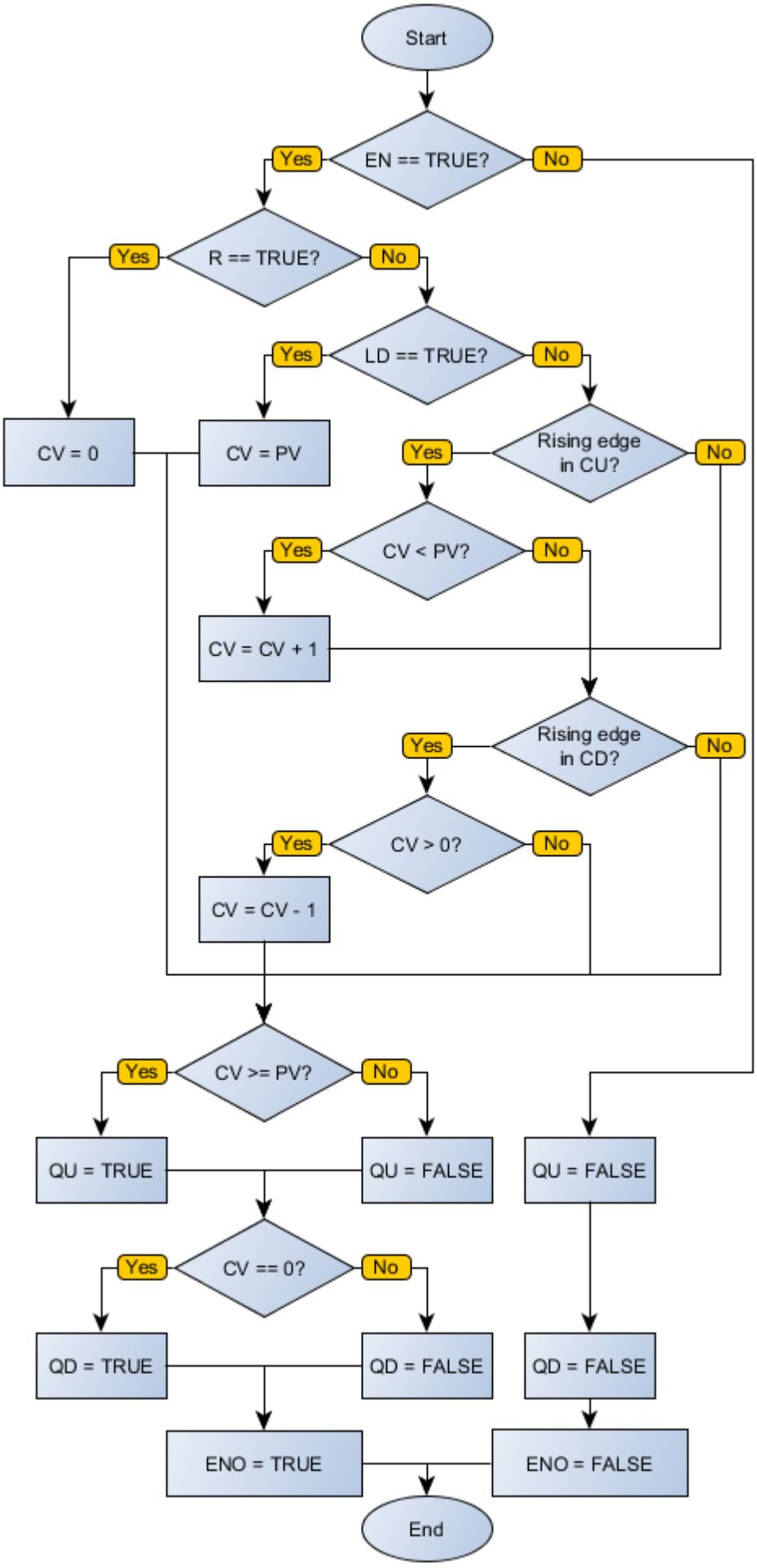
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	CU	BOOL	Pulse identifier for incremental
	CD	BOOL	Pulse identifier for decremental
	R	BOOL	Loads the zero value in CV
	LD	BOOL	Loads the value of PV in CV
	PV	WORD UINT	Reference value
VAR_OUTPUT	ENO	BOOL	Output enabling
	QU	BOOL	Counter overrun flag
	QD	BOOL	Counter zeroed flag
	CV	WORD UINT	Current count value
VAR	CTUD_INST_0	CTUD	Instance of access to block structure

Operation

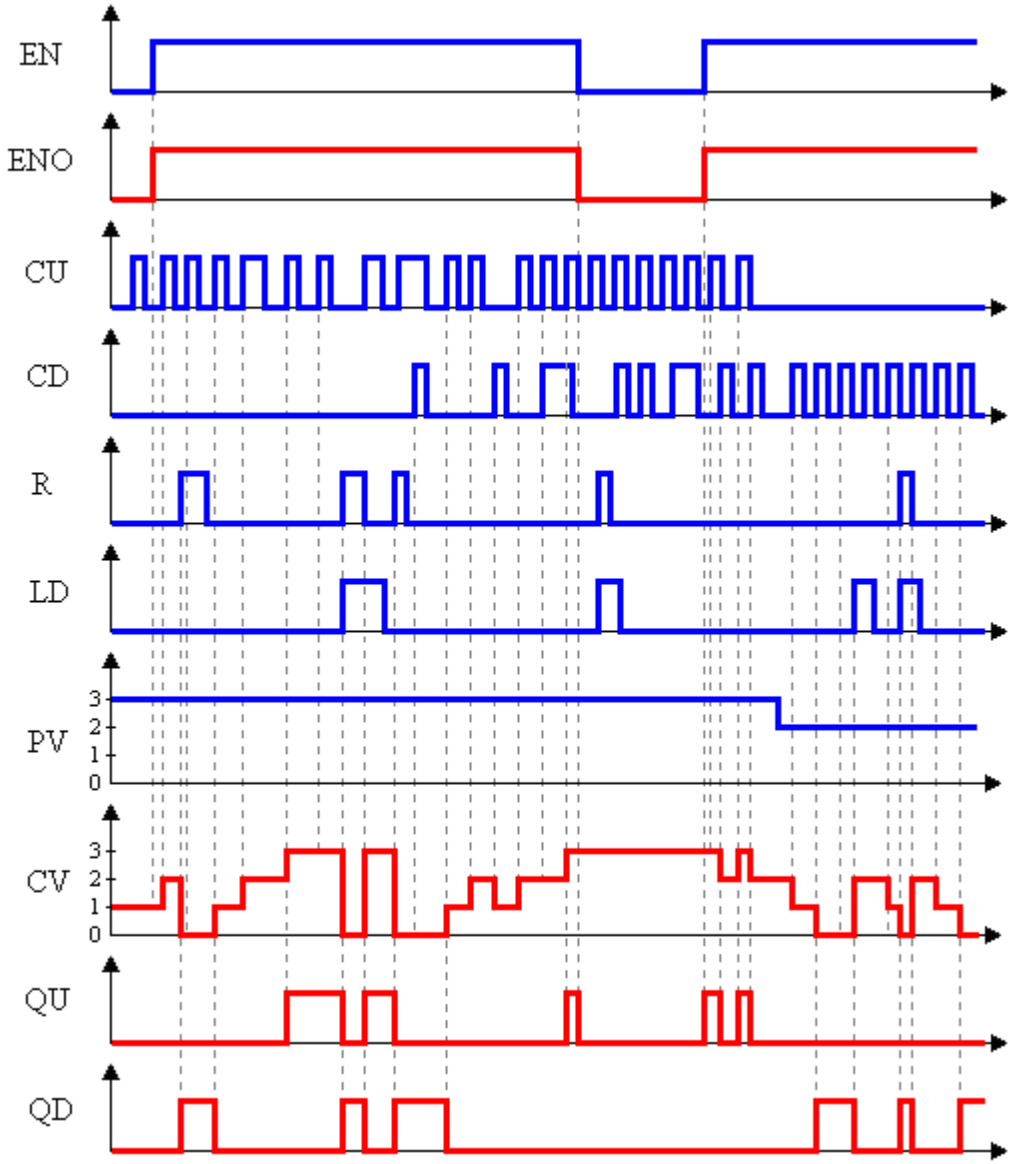
When this block has a TRUE value in EN, it acts as a CTD block and block CTU at the same time acting on the same CV counter. That is: increments CV until it reaches PV to the leading edges in CU and decrements CV until it reaches zero to the leading edges in CD. A positive transition in R carries zero in CV, while a leading edge in LD loads the PV value in CV. If CV has zero value, QD receives TRUE, and if CV has value equal to PV, QU receives TRUE.

The ENO value forwards to the next Ladder block the EN value.

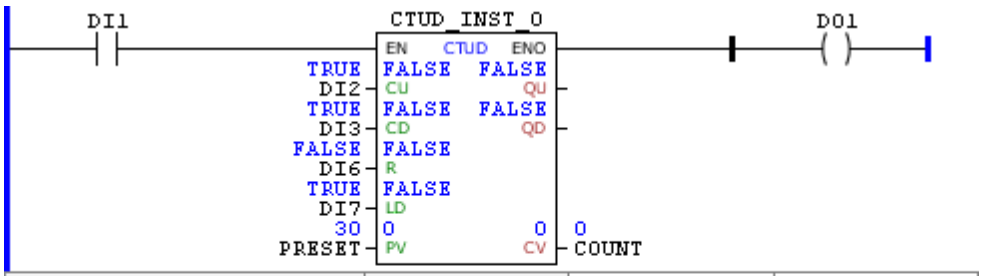
Block Flowchart


Start
EN == TRUE?
Yes
No
R == TRUE?
Yes
No
LD == TRUE?
Yes
No
CV = 0
CV = PV
Rising edge in CU?
Yes
No
CV < PV?
Yes
No
CV = CV + 1
Rising edge in CD?
Yes
No
CV > 0?
Yes
No
CV = CV - 1
CV >= PV?
Yes
No
QU = TRUE
QU = FALSE
QU = FALSE
CV == 0?
Yes
No
QD = TRUE
QD = FALSE
QD = FALSE
ENO = TRUE
ENO = FALSE
End

Operation Diagram

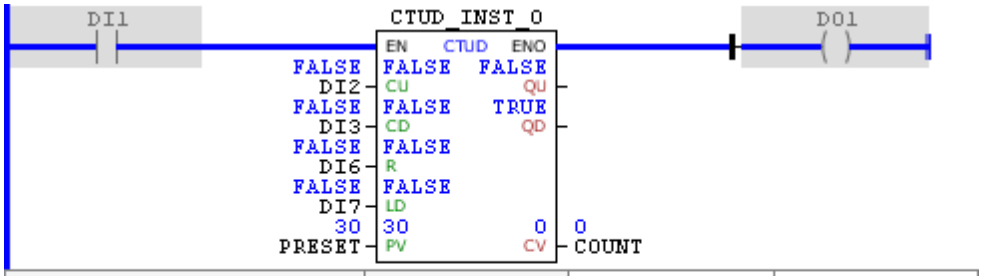


Example



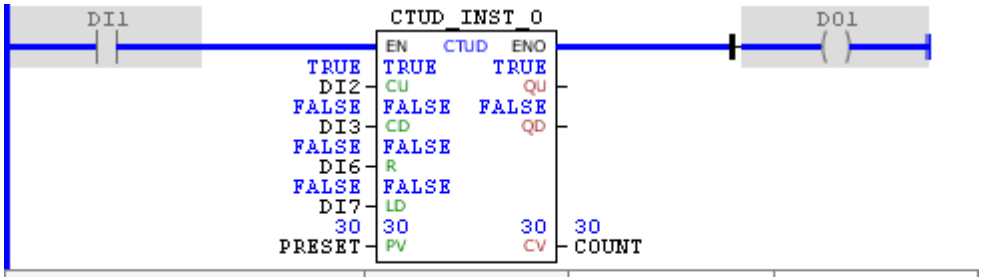
PRESET	UINT	30	Decimal
COUNT	UINT	0	Decimal

The example above shows the disabled block, with all its internal variables zeroed. Although the external controls are activated, these values are not forwarded to the instance of the block.



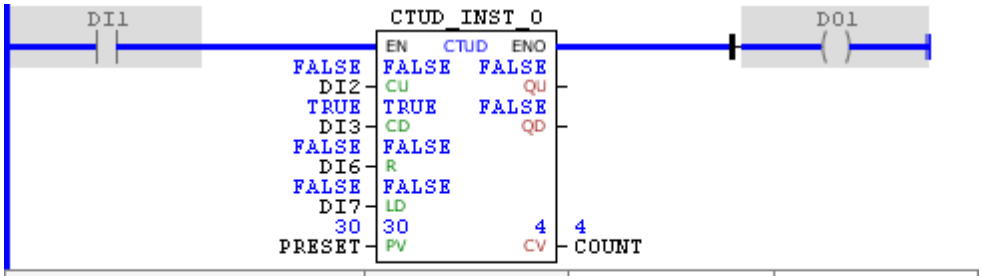
PRESET	UINT	30	Decimal
COUNT	UINT	0	Decimal

When activated, the block identifies the value of PRESET, loading it in PV, and identifies that the output is at zero, enabling the QD output. When execution is completed, the ENO output is activated.



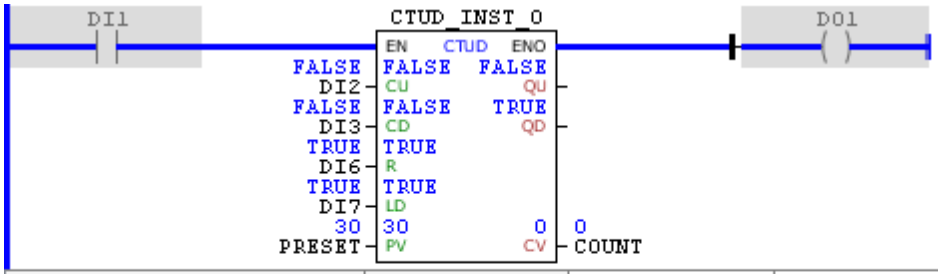
PRESET	UINT	30	Decimal
COUNT	UINT	30	Decimal

At each leading edge identified in CU, the value of CV is incremented until it reaches the PV value, when the QU output is enabled. When execution is completed, the ENO output is activated.



PRESET	UINT	30	Decimal
COUNT	UINT	4	Decimal

At each leading edge detected in CD, the CV value is decremented. When CV is a value between zero and PV, both QD and QU outputs are deactivated. When execution is completed, the ENO output is activated.



PRESET	UINT	30	Decimal
COUNT	UINT	0	Decimal

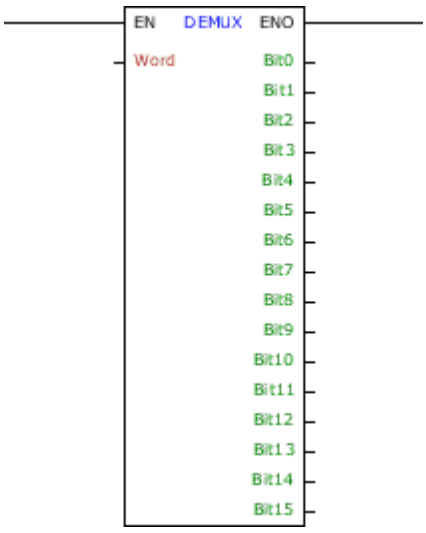
A TRUE value in R resets CV, while a TRUE value in LD loads the value of PV to CV. As we can see, R prevails over LD, leaving CV and enabling the QD output. When execution is completed, the ENO output is activated.

11.14.7.8 Data Transfer

11.14.7.8.1 DEMUX

Block that creates 16 new BOOL variables from the decomposition of a WORD variable.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Word	WORD UINT INT	Input variable of 15 bits
VAR_OUTPUT	ENO	BOOL	End of operation
	Bit0 – Bit15	BOOL	Bit of the corresponding position of Word

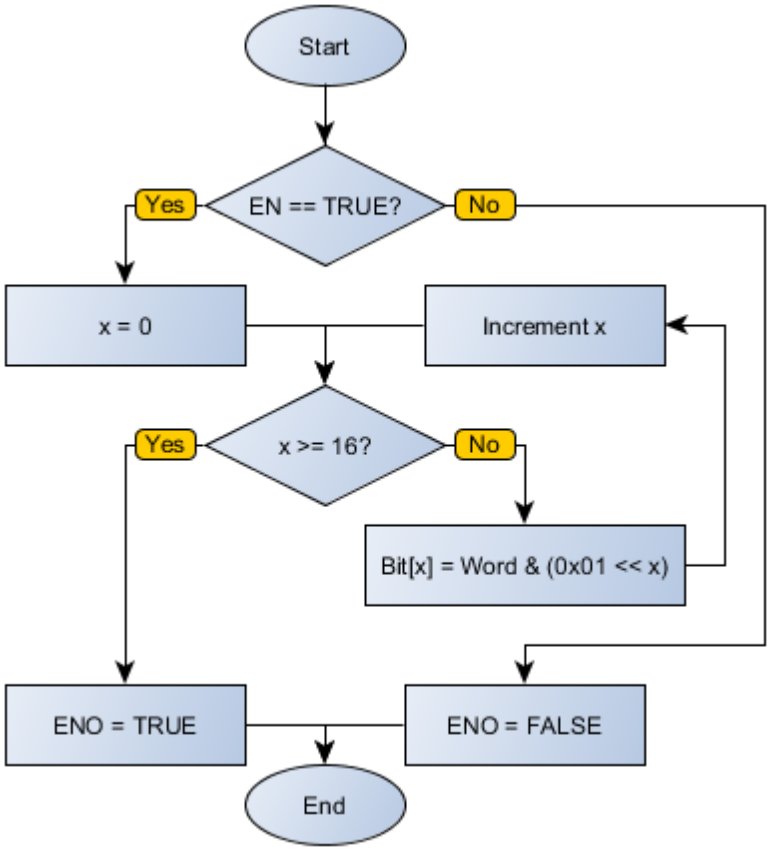
Operation

When this block has a TRUE value in EN, it decomposes the input variable in Word 15 Boolean values stored in Bit0 to Bit15 variables. Bit0 corresponds to the LSB (least significant bit) and Bit15 corresponds to the MSB (most significant bit).

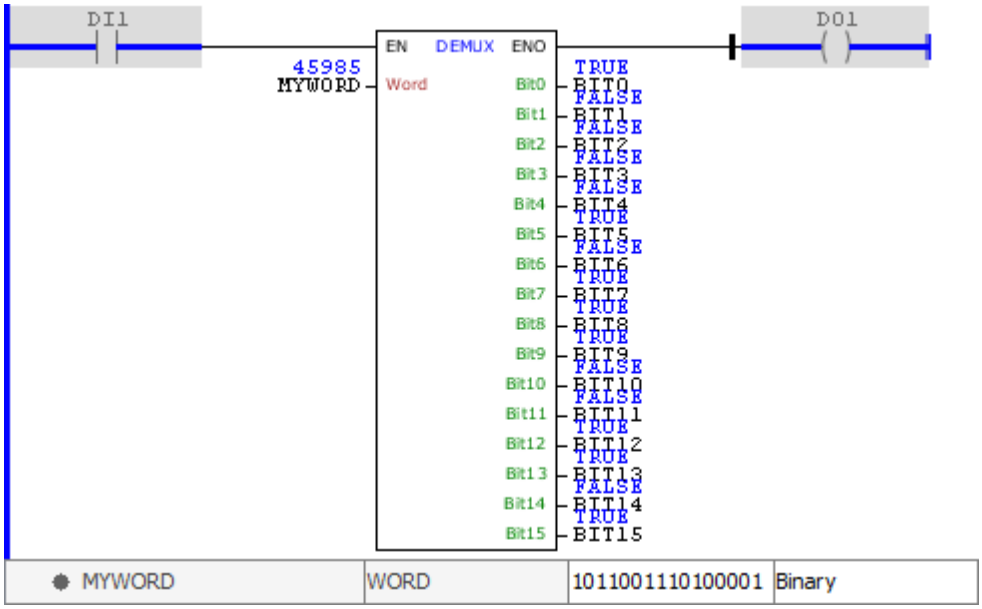
When EN has FALSE value, output variables remain unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example

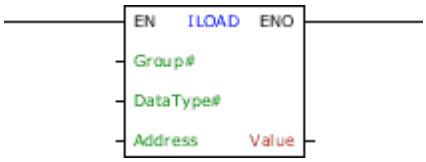


The example above decomposes the value of MYWORD in Boolean values, which are stored in the output variables BIT0 to Bit15. The block ends successfully and the ENO output is activated.

11.14.7.8.2 ILOAD

Block which indirectly loads the value of a variable and transfers it to Value.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Group#	BYTE	Group w here the variable is stored
	DataType#	BYTE	Data type of the selected variable
	Address	DWORD UDINT DINT	Address of the global variable, as its group
VAR_OUTPUT	ENO	BOOL	End of operation
	Value	As selected in DataType#	Value of the selected variable

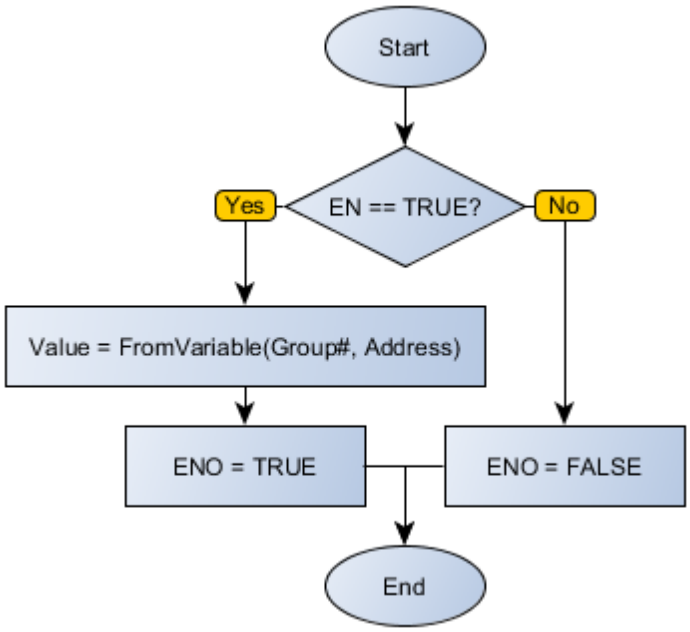
Operation

When this block has a TRUE value in EN, it loads, in Value, the of the Address variable belonging to the Group# group, as the selected DataType#.

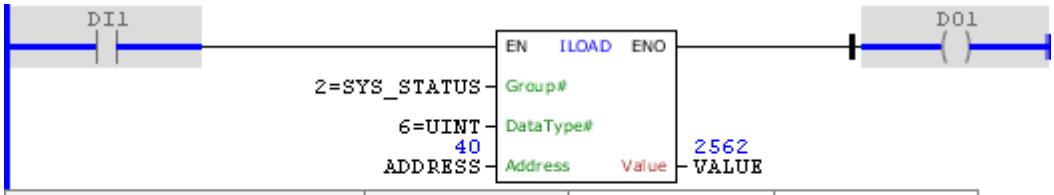
When EN has FALSE value, Value remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



ADDRESS	DINT	40	Decimal
VALUE	UINT	2562	Decimal

The above example loads the value of the address 40 of group 2 (GLOBAL_SYSTEM%S), which represents the status of ESC key in UINT format for the VALUE variable. The block ends with success and ENO output is activated.

11.14.7.8.3 ILOADBOOL

Block that indirectly loads the value of a bit in a global variable address.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Group#	BYTE	Group w here the variable is stored
	Address	DWORD UDINT DINT	Address of the global variable, as its group
	Bit	BYTE USINT SINT	Position of the bit to be checked
VAR_OUTPUT	ENO	BOOL	End of operation
	Value	BOOL	Value of the bit selected by the input arguments

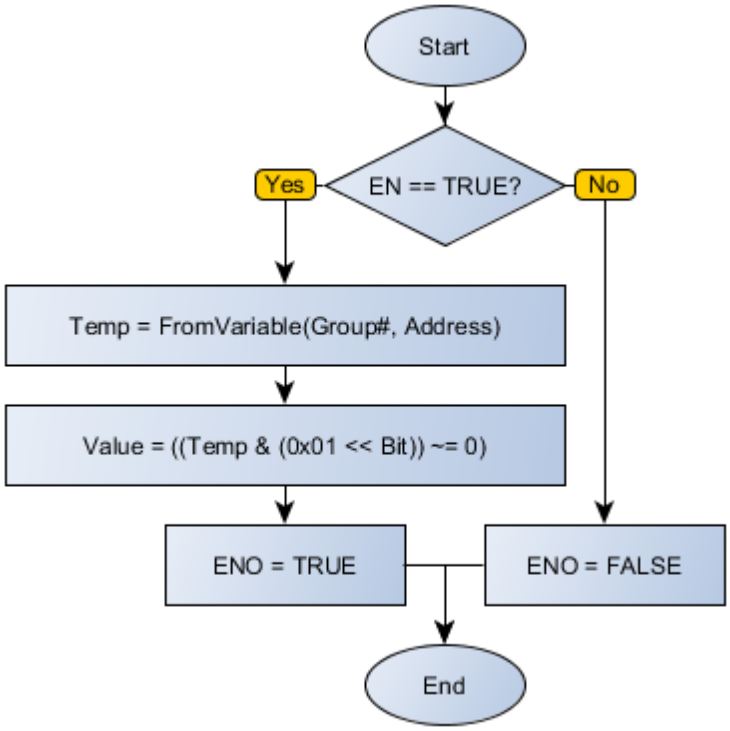
Operation

When this block has a TRUE value in EN, it loads, in Value, the Bit contents of the Address variable belonging to the Group# group.

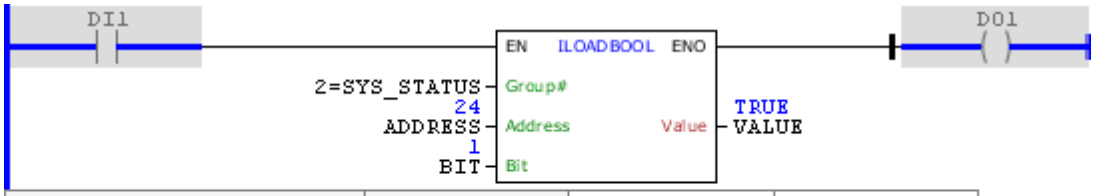
When EN has FALSE value, Value remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



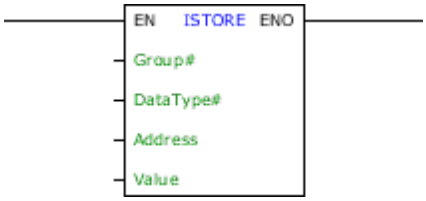
ADDRESS	DINT	24	Decimal
BIT	BYTE	1	Decimal
VALUE	BOOL	1	Binary

The above example loads the value of bit 1 of the address 24 of group 2 (S GLOBAL_SYSTEM%), which represents the status of ESC key for the VALUE variable. The block ends with success and ENO output is activated.

11.14.7.8.4 ISTORE

Block that indirectly loads the Value value in a variable.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Group#	BYTE	Group where the variable is stored
	DataType#	BYTE	Data type of the selected variable
	Address	DWORD UDINT DINT	Address of the global variable, as its group
	Value	Depending on the selection of the DataType#	Value to be written in the selected variable
VAR_OUTPUT	ENO	BOOL	End of operation

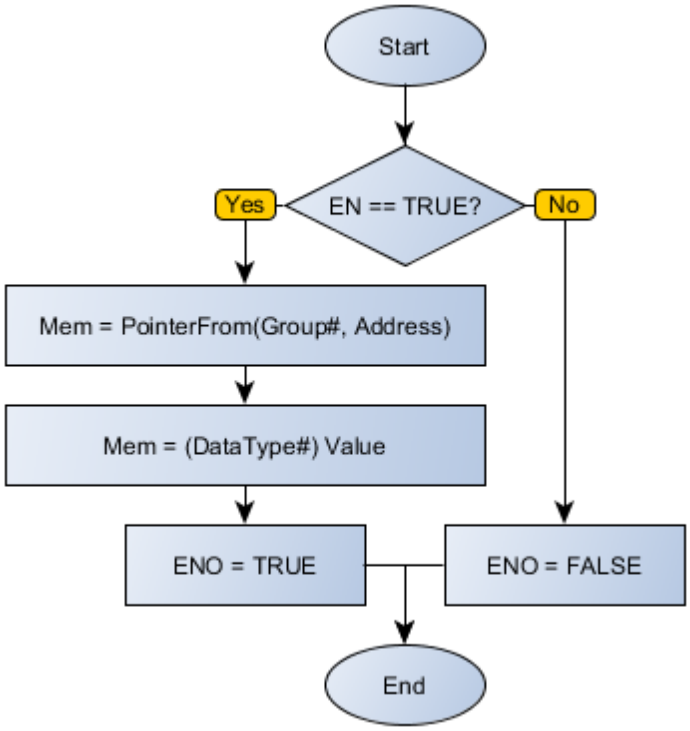
Operation

When this block has a TRUE value in EN, it loads the Value value in the contents of the Address variable belonging to the Group# group, as the selected DataType#.

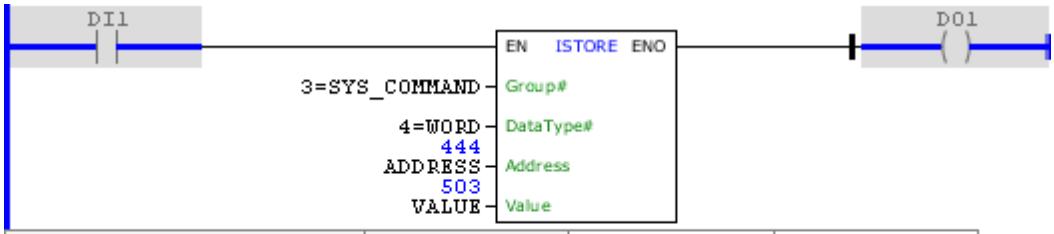
When EN has FALSE value, Value remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



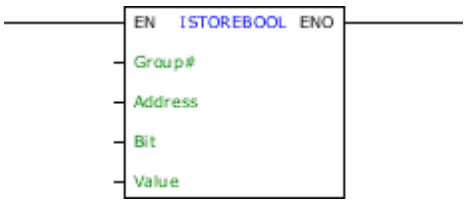
ADDRESS	DINT	444	Decimal
VALUE	WORD	503	Decimal

The example above stores the VALUE value in WORD format in address 444 of group 3 (GLOBAL_SYSTEM% C), which represents the index of the communication port Modbus TCP. The block ends with success and ENO output is activated.

11.14.7.8.5 ISTOREBOOL

Block that indirectly loads the Value value in a bit in a global variable address.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Group#	BYTE	Group where the variable is stored
	Address	DWORD UDINT DINT	Address of the global variable, as its group
	Bit	BYTE USINT SINT	Position of the bit to be modified
	Value	BOOL	New value of the selected bit
VAR_OUTPUT	ENO	BOOL	End of operation

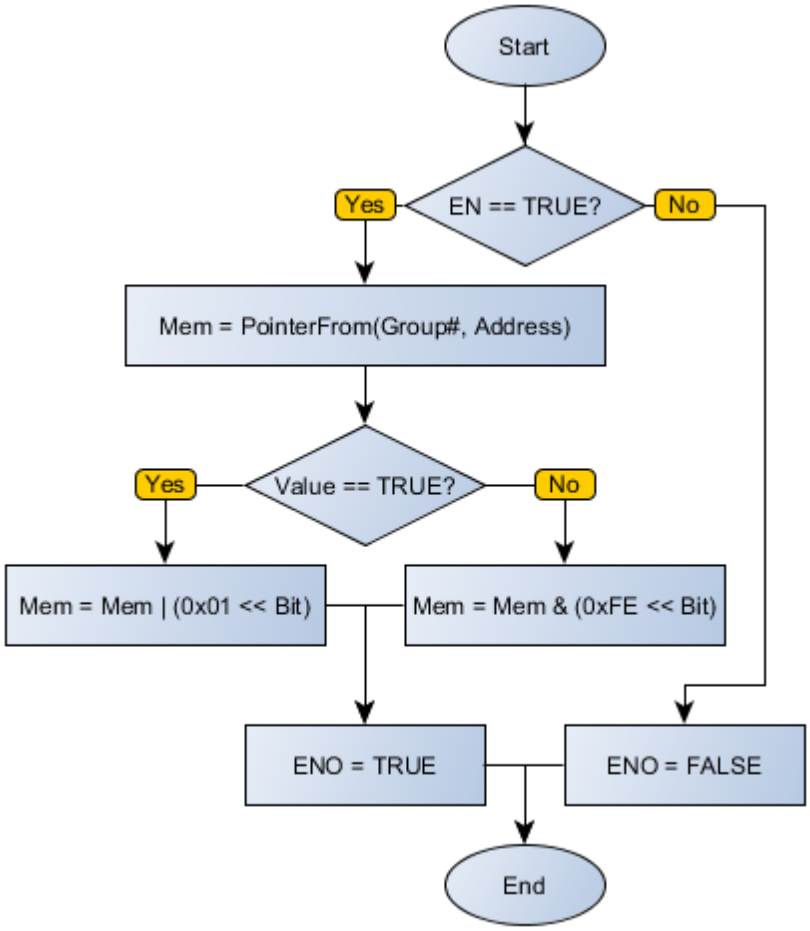
Operation

When this block has a TRUE value in EN, it loads the Value value in the Bit contents of the Address variable belonging to the Group# group.

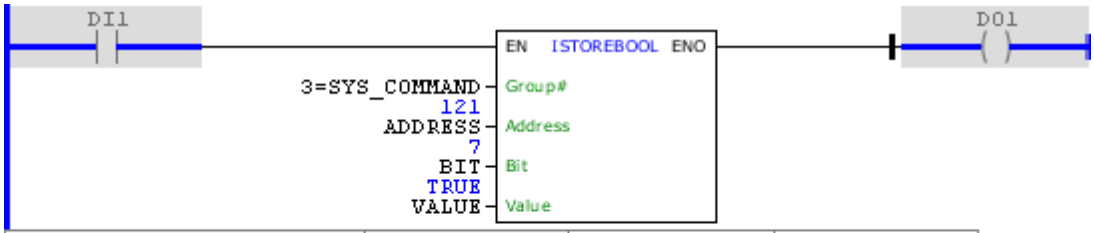
When EN has FALSE value, Value remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



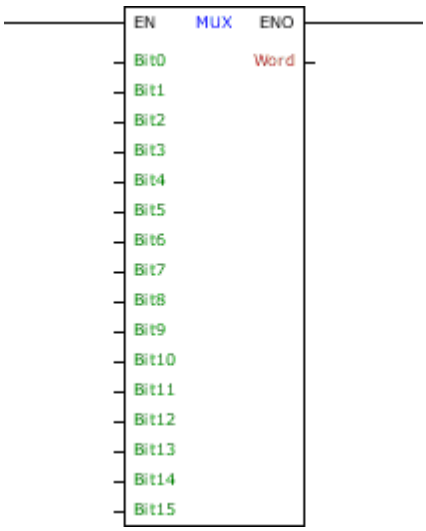
ADDRESS	DINT	121	Decimal
BIT	BYTE	7	Decimal
VALUE	BOOL	1	Binary

The example above stores the value of VALUE in bit 7 of the address 121 in group 3 (GLOBAL_SYSTEM% C), which represents the disable command of CANopen communication. The block ends with success and ENO output is activated.

11.14.7.8.6 MUX

Block that creates a new WORD variable from the concatenation of 16 BOOL variables.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Bit0 – Bit15	BOOL	Bit of the corresponding position in the new word
VAR_OUTPUT	ENO	BOOL	End of operation
	Word	WORD UINT INT	New word formed from the input bits

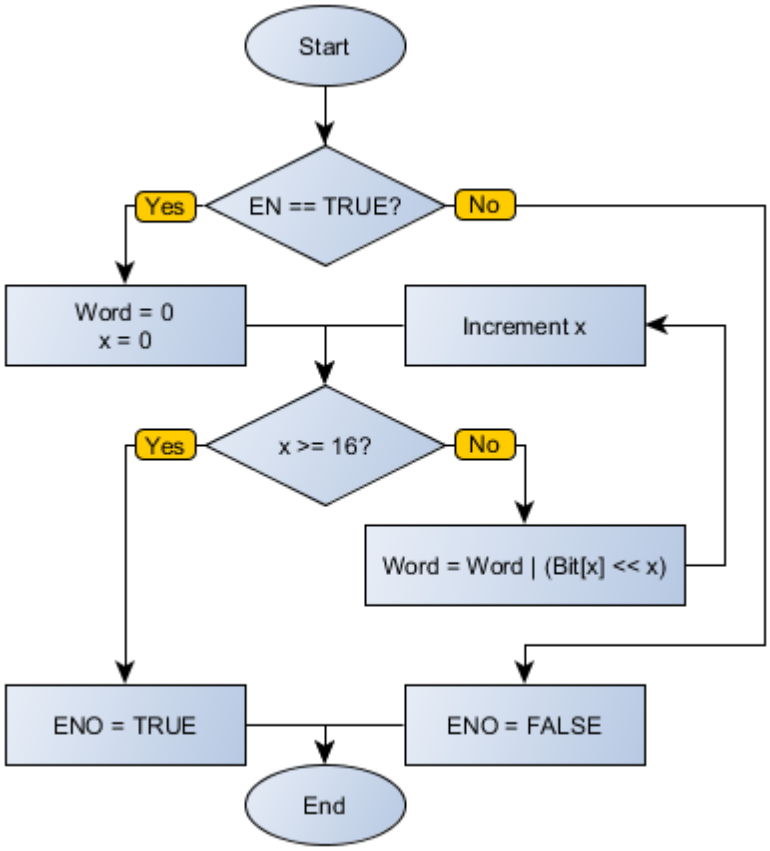
Operation

When this block has a TRUE value in EN, it concatenates Boolean values of the input variables and stores this value in the variable Word. Bit0 corresponds to LSB (least significant bit) and Bit15 corresponds to the MSB (most significant bit).

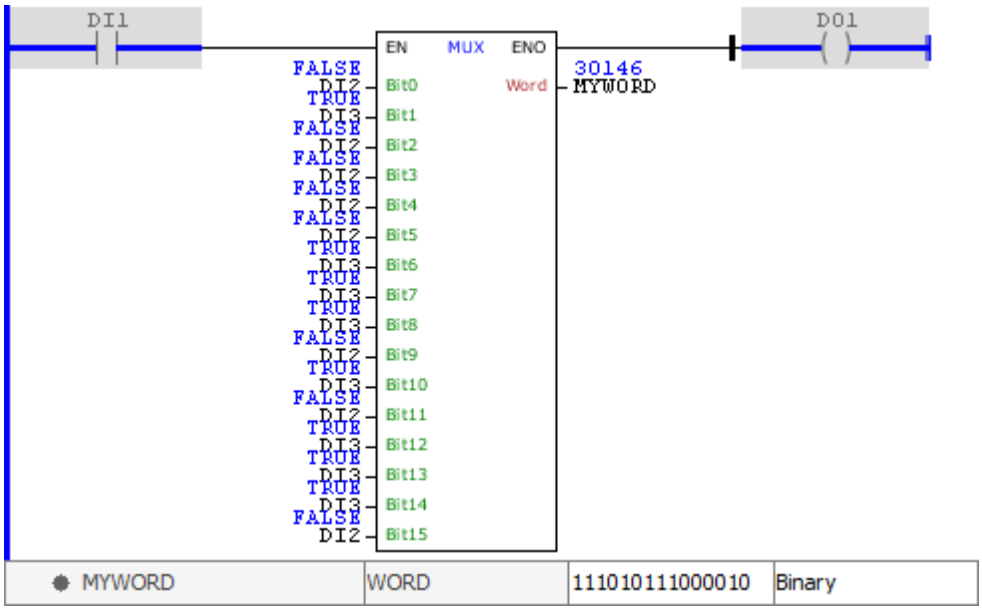
When EN has FALSE value, Word remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example

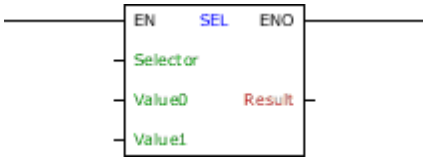


The above example concatenates the Boolean values of the input bits of the block to form the output word stored in MYWORD. The block ends with success and ENO output is activated.

11.14.7.8.7 SEL

Block that replicates to the output the value of an input variable (Value0 or Value1) according to the Selector selection.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Selector	BOOL	Variable that selects the input
	Value0	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Multiplexed input number 1
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Multiplexed input number 2
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Output value selected

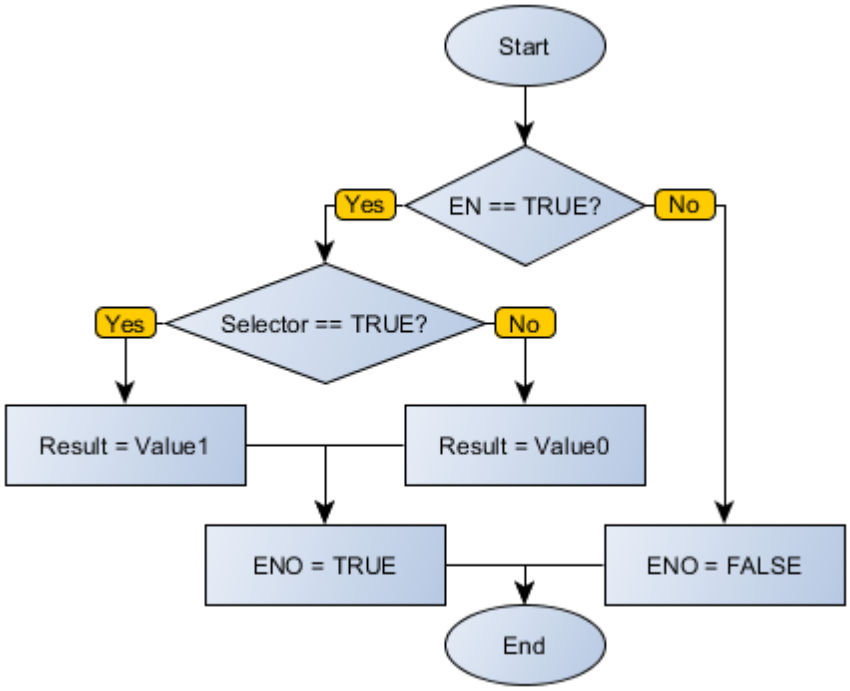
Operation

When this block has a TRUE value in EN, it replicates to the Result variable the Value0 value if selector is FALSE or the Value1 value if Selector is TRUE.

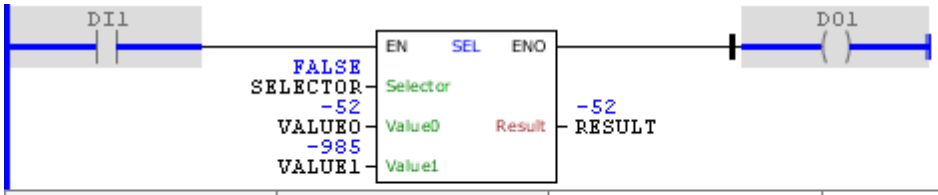
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart

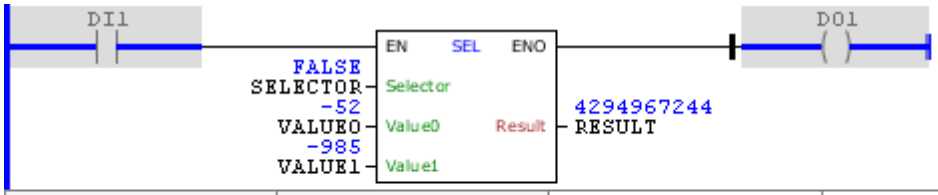


Example



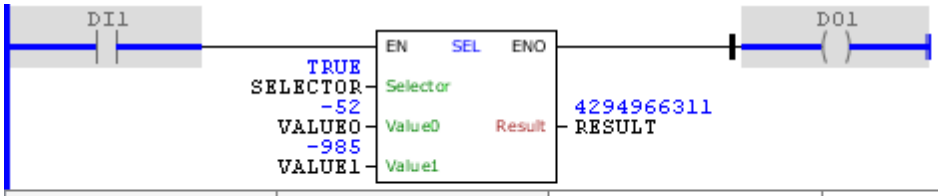
SELECTOR	BOOL	0	Binary
VALUE0	SINT	-52	Decimal
VALUE1	INT	-985	Decimal
RESULT	INT	-52	Decimal

The above example uses the SELECTOR variable as multiplexing channel selector. When it is at FALSE level, the RESULT output gets the value of VALUE0. The block ends successfully and the ENO output is activated.



SELECTOR	BOOL	0	Binary
VALUE0	SINT	-52	Decimal
VALUE1	INT	-985	Decimal
RESULT	DWORD	4294967244	Decimal

The above example uses the SELECTOR variable as multiplexing channel selector. When it is at FALSE level, the RESULT output gets the value of VALUE0. The block ends successfully and the ENO output is activated. Note that the binary pattern has been maintained even though the decimal representation is corrupted, since DWORD does not accept negative values.



SELECTOR	BOOL	1	Binary
VALUE0	SINT	-52	Decimal
VALUE1	INT	-985	Decimal
RESULT	DWORD	4294966311	Decimal

The above example uses the SELECTOR variable as multiplexing channel selector. When it is at TRUE level, the RESULT output gets the value of VALUE1. The block ends successfully and the ENO output is activated. Note that the binary pattern has been maintained even though the decimal representation is corrupted, since DWORD does not accept negative values.

11.14.7.8.8 STORE

Block that performs direct storage of data from a source to a destination.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	SRC	BYTE USINT SINT WORD UINT INT DWORD DINT DINT REAL	Data source
VAR_OUTPUT	ENO	BOOL	End of operation
	DST	BYTE USINT SINT WORD UINT INT DWORD DINT DINT REAL	Data destination

Operation

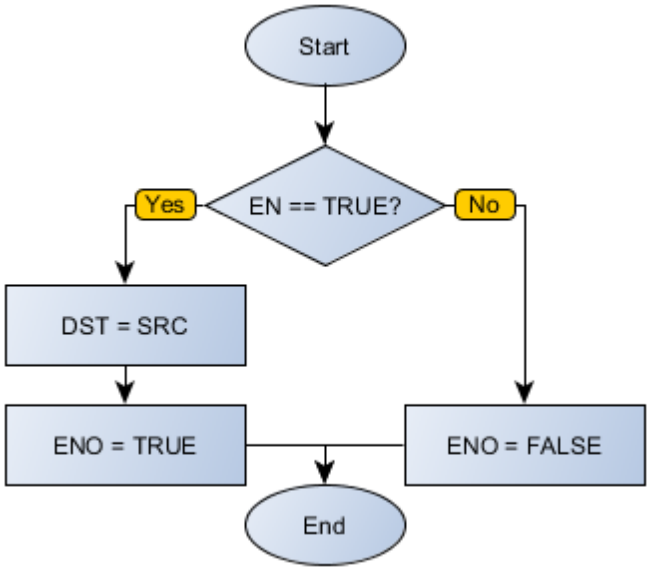
When this block has a TRUE value in EN, it stores the contents from SRC into DST.

NOTE!
SRC and DST must have data types of the same size.

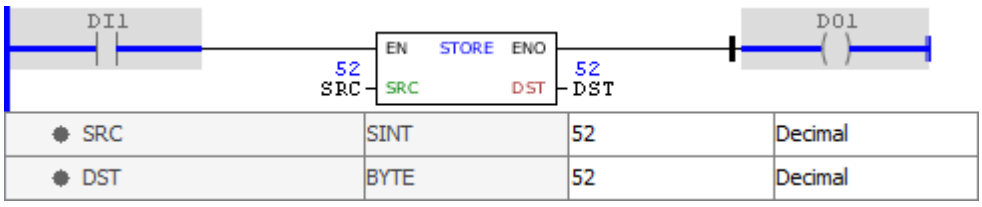
When EN has FALSE value, DST remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



The example above stores the value of the variable SRC in DST. The block ends with success and ENO output is activated.

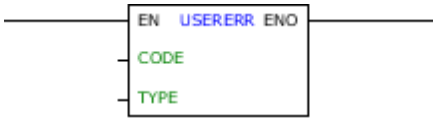
SRC	SINT	-15	Decimal
DST	BYTE	241	Decimal

The example above stores the value of the variable SRC in DST. The block ends with success and ENO output is activated. Note that the binary pattern is maintained between variables of different types.

11.14.7.8.9 USERERR

Block that generates an alarm or fault with the number programmed by the user.

Ladder Representation



Block Structure

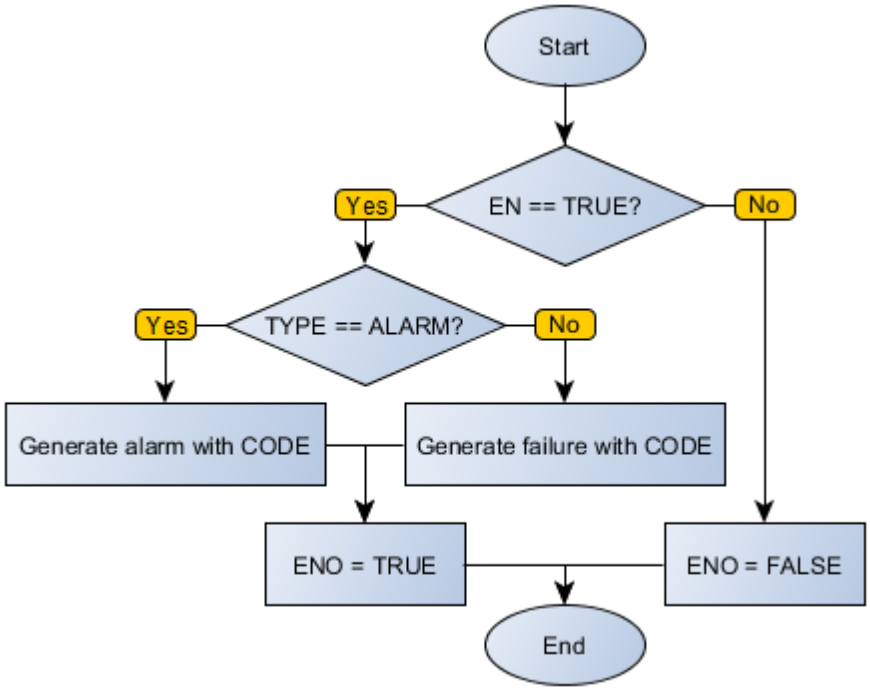
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	CODE	WORD UINT	Error code generated (750 - 799)
	TYPE	BYTE	Error type generated (0 - Alarm) (1 - Fault)
VAR_OUTPUT	ENO	BOOL	Success in the generation of error
VAR	USERERR_INST_0	USERERR	Instance of access to block structure

Operation

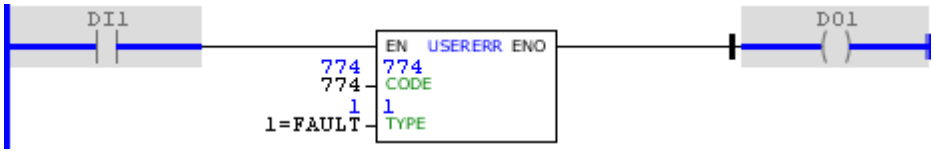
When this block has a TRUE value in EN, it generates an alarm or equipment failure, depending on the type defined in TYPE with CODE code.

The value of ENO informs if the generation of alarm or fault has been executed successfully.

Block Flowchart



Example



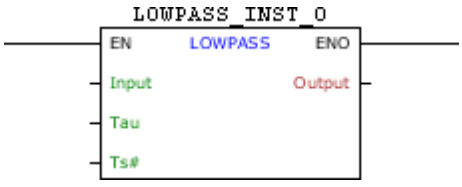
The above example, when identifying TRUE level in DI1, generates a fault with the code 774 and sets the DO1 output.

11.14.7.9 Filter

11.14.7.9.1 LOWPASS

Block that filters the input using a low pass filter of first order and inserts the result in the output.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Input	REAL	Input signal
	Tau	REAL	Filter time constant
	Ts#	UINT	Sampling time [ms]
VAR_OUTPUT	ENO	BOOL	Output enabling
	Output	REAL	Filter output
VAR	LOWPASS_INST_0	LOWPASS	Instance of access to block structure

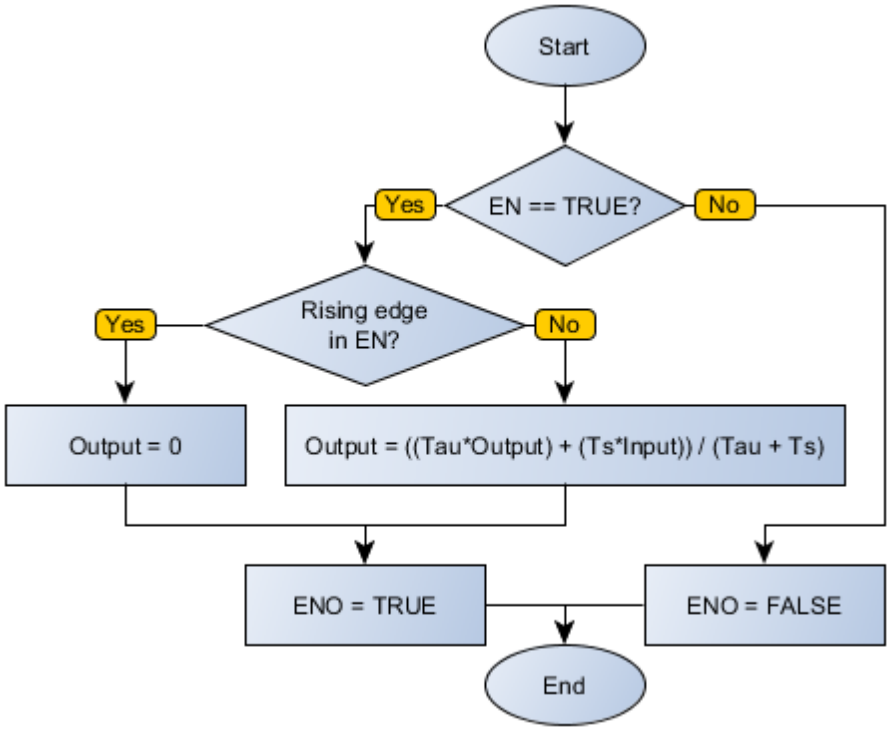
Operation

When this block has a TRUE value in EN, filters the input value of Input using a low pass first order filter described by Tau and Ts#, inserting the result in Output. On the leading edge of EN, Output receives zero.

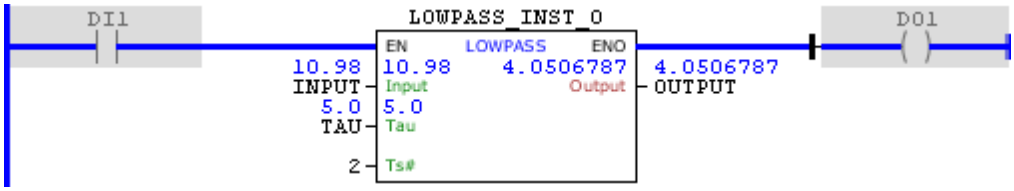
When EN has FALSE value, Output remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

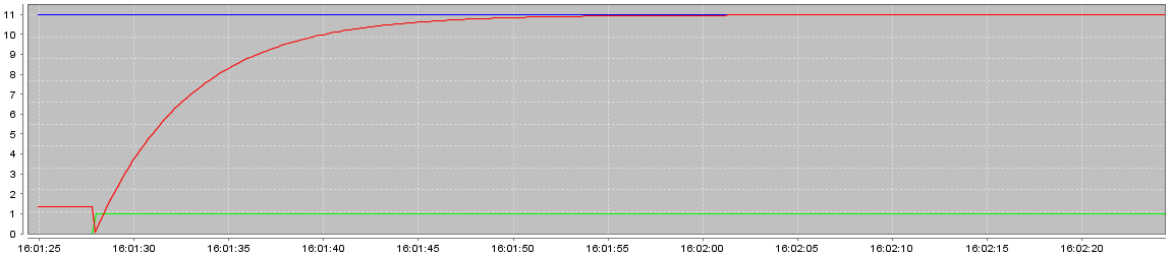
Block Flowchart



Example



The above example causes OUTPUT, by identifying a leading edge in EN, to display a behavior of first order with time constant equal to Tau and the sampling time of 2 ms, in order to achieve the reference set to INPUT. At each calculation completed successfully, the ENO output is activated.



11.14.7.10 Logic

11.14.7.10.1 Logic Bit

11.14.7.10.1.1 RESETBIT

Logical block used to perform reset of a specific bit in a field.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_IN_OUT	Data	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable w hose bit w ill be changed
VAR_INPUT	EN	BOOL	Block enabling
	Position	BYTE USINT	Position of the bit that w ill be changed
VAR_OUTPUT	DONE	BOOL	Operation successful

Operation

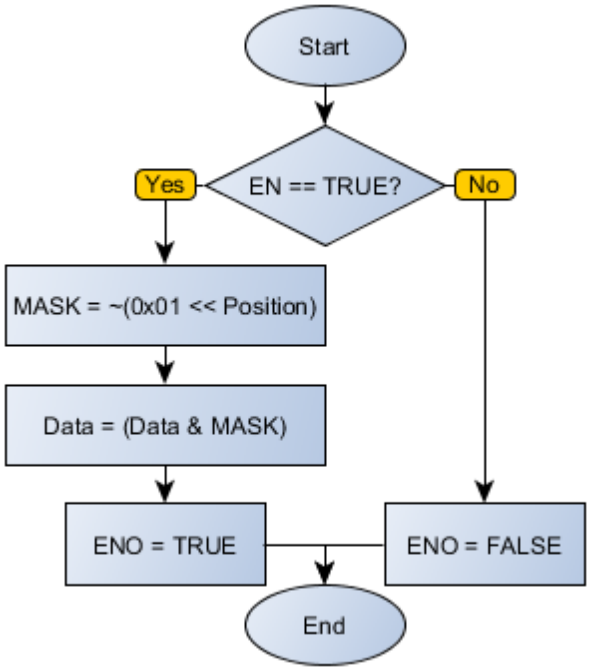
This block when it has a TRUE value in EN, resets the bit indicated in Position in the Data variable that is forwarded to the output already with its updated value.

When EN has FALSE value, Data remains unchanged.

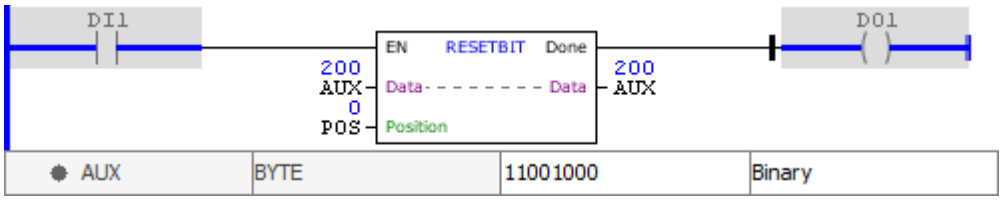
The DONE variable receives the same EN value, except when there is an error in the reset of the bit, then getting a FALSE value.

NOTE!
It is important to notice that Position is within the range of values of bits corresponding to variable type in Data. For example: if Data is a BYTE, it has 8 bits, and Position must contain a value between 0 and 7.

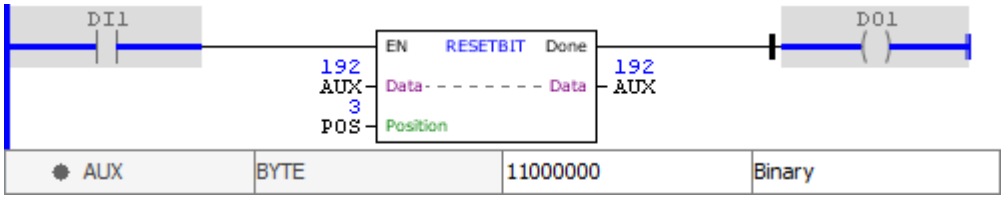
Block Flowchart



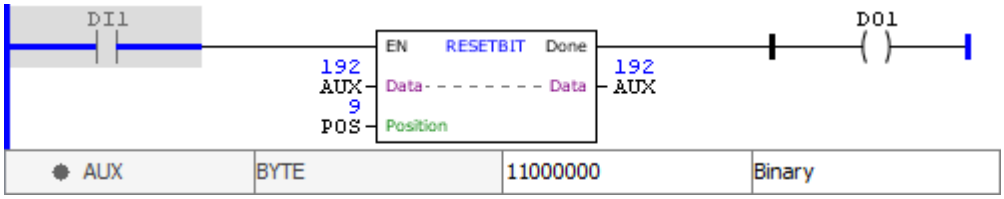
Example



The example above resets the bit of AUX zero position, whose initial value is 200 (1100 1000, in binary). Since this bit already had FALSE value, nothing has changed.



The example above resets the bit in position three of AUX by changing its binary value and, therefore, its decimal representation.



The example above resets the bit in position nine of AUX. Since AUX is a variable BYTE type, it has only eight bits. Thus, the example above creates a runtime error in the block and therefore the output is not enabled.

11.14.7.10.1.2 SETBIT

Logical block used to perform the set of a specific bit in a field.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_IN_OUT	Data	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable w hose bit w ill be changed
VAR_INPUT	EN	BOOL	Block enabling
	Position	BYTE USINT	Position of the bit that w ill be changed
VAR_OUTPUT	DONE	BOOL	Operation successful

Operation

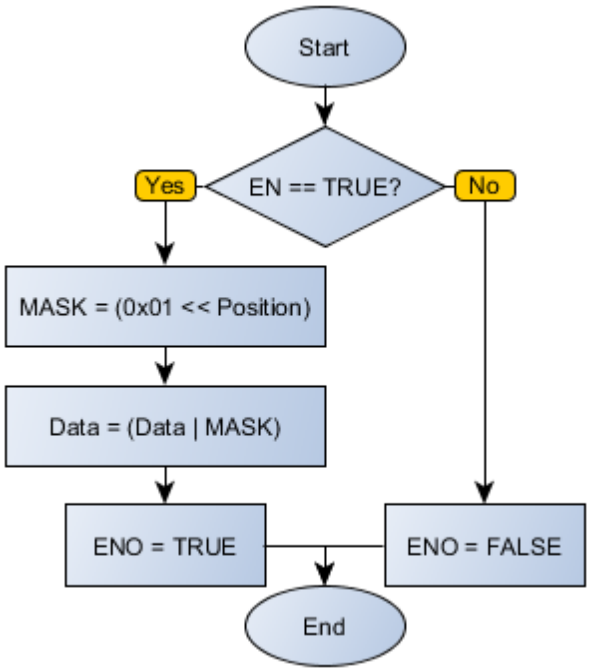
This block when it has a TRUE value in EN, sets the bit indicated in Position in the Data variable that is forwarded to the output already with its updated value.

When EN has FALSE value, Data remains unchanged.

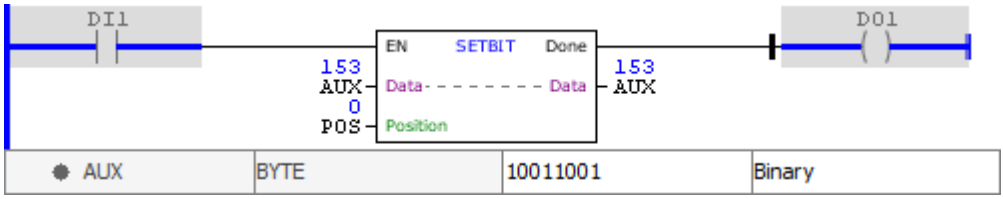
The DONE variable receives the same EN value, except when there is an error in the set of the bit, then getting a FALSE value.

NOTE!
It is important to notice that Position is within the range of values of bits corresponding to variable type in Data. For example: if Data is a BYTE, it has 8 bits, and Position must contain a value between 0 and 7.

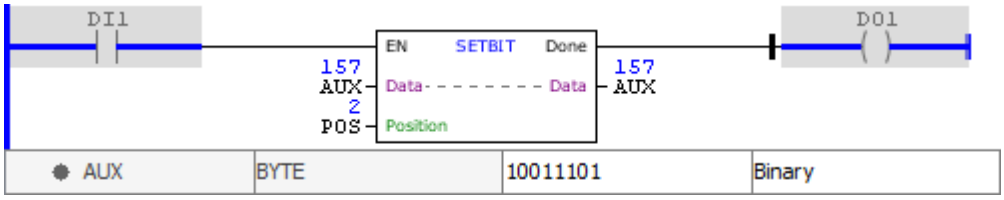
Block Flowchart



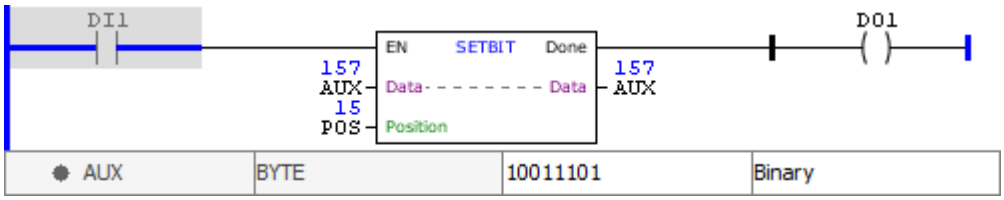
Example



The example above sets the bit of AUX zero position, whose initial value is 153 (1001 1001, in binary). Since this bit already had TRUE value, nothing has changed.



The example above sets the bit in position three of AUX by changing its binary value and, therefore, its decimal representation.



The example above sets the bit in position fifteen of AUX. Since AUX is a variable BYTE type, it has only eight bits. Thus, the example above creates a runtime error in the block and therefore the output is not enabled.

11.14.7.10.1.3 TESTBIT

Logical block that revolutions the value of a specific bit in a field.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	Data	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable w hose bit w ill be tested
	EN	BOOL	Block enabling
	Position	BYTE USINT	Position of the bit that w ill be changed
VAR_OUTPUT	Q	BOOL	Value of the tested bit

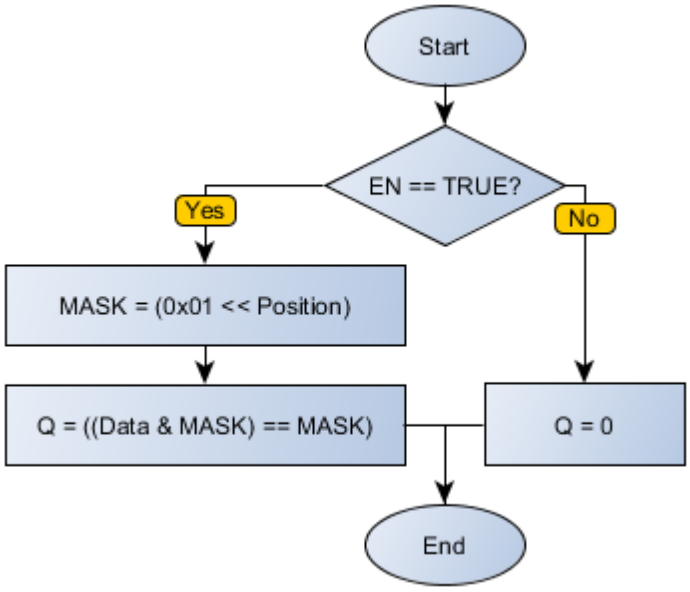
Operation

This block when it has a TRUE value in EN, sends to the output Q the bit value indicated in Position in the Data variable.

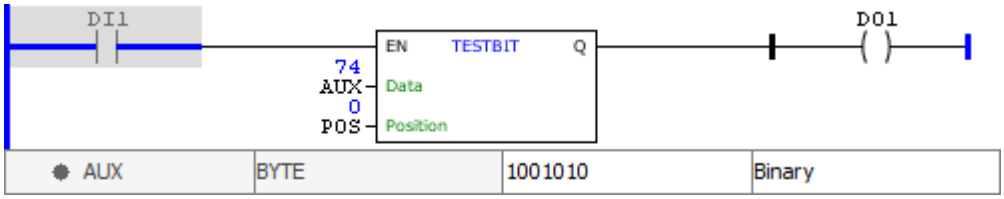
When EN has FALSE value, Q also receives FALSE.

NOTE!
It is important to notice that Position is within the range of values of bits corresponding to variable type in Data. For example: if Data is a BYTE, it has 8 bits, and Position must contain a value between 0 and 7.

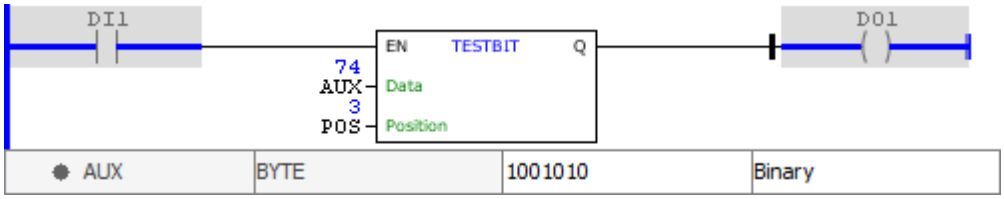
Block Flowchart



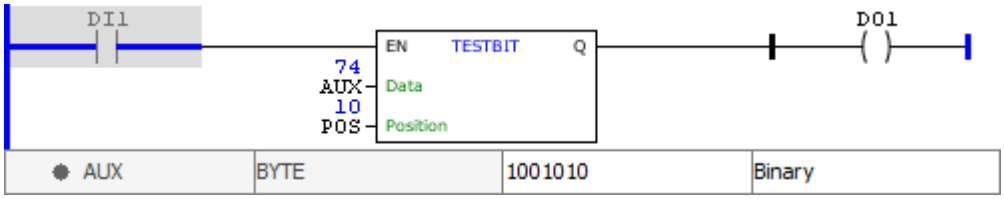
Example



The example above sets the bit value of zero position of AUX, whose initial value is 74 (0100 1010 in binary) to the output Q. Since this bit has value 0, the output is disabled.



The example above sets the value of the bit of position three of AUX to the output Q. Since this bit has value 1, the output is enabled.



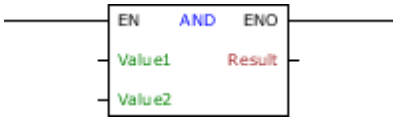
The example above sets the bit value of position ten of AUX to output Q. Since AUX is a variable of BYTE type, it has only eight bits. Thus, the example above creates a runtime error in the block and therefore the output is disabled.

11.14.7.10.2 Logic Boolean

11.14.7.10.2.1 AND

Logical block that performs an boolean "and" operation between two variables, storing the result in a third one.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable for the operation
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable for the operation
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable that stores the result of the operation

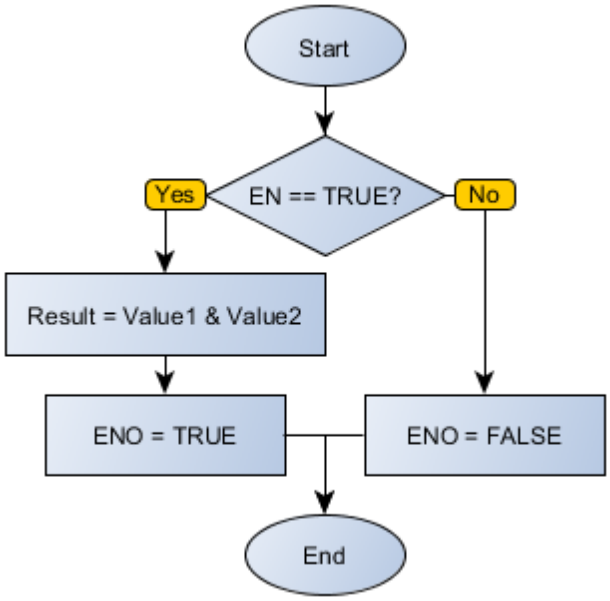
Operation

When this block has a TRUE value in EN, it sends to the Result output the “and” Boolean operation of input variables Value1 and Value2.

When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example

AUX	BYTE	11001000	Binary
AUX2	BYTE	10111001	Binary
AUX3	BYTE	10001000	Binary

The example above performs an "and" Boolean operation between AUX and AUX2, storing the result in AUX3.

11.14.7.10.2.2 NOT

Block that performs a logical operation of boolean "not" in a variable, storing the result in another.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Reference variable for the operation
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable that stores the result of the operation

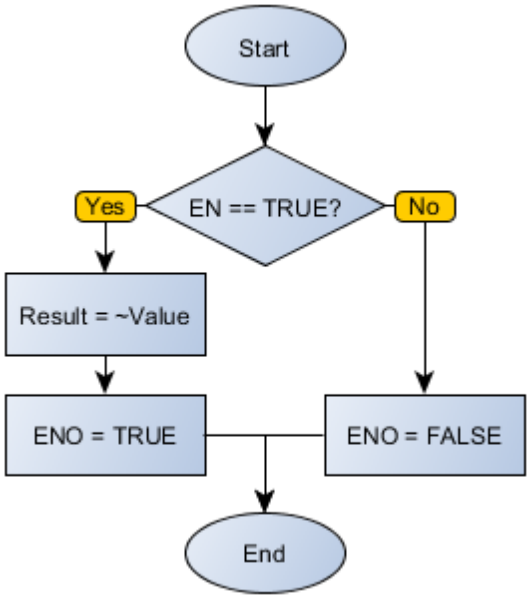
Operation

When this block has a TRUE value in EN, it sends to the Result output the denied Boolean value of the Value input variable.

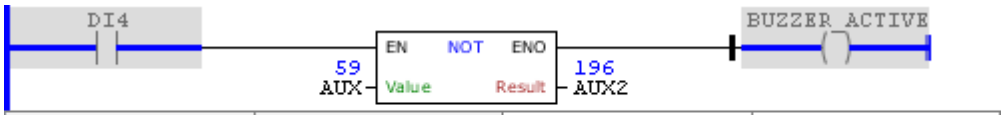
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



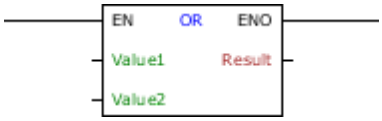
AUX	BYTE	111011	Binary
AUX2	BYTE	11000100	Binary

The example above performs a boolean "not" operation in AUX, storing the result in AUX2.

11.14.7.10.2.3 OR

Logical block that performs an Boolean "or" operation between two variables, storing the result in a third one.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable for the operation
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable for the operation
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable that stores the result of the operation

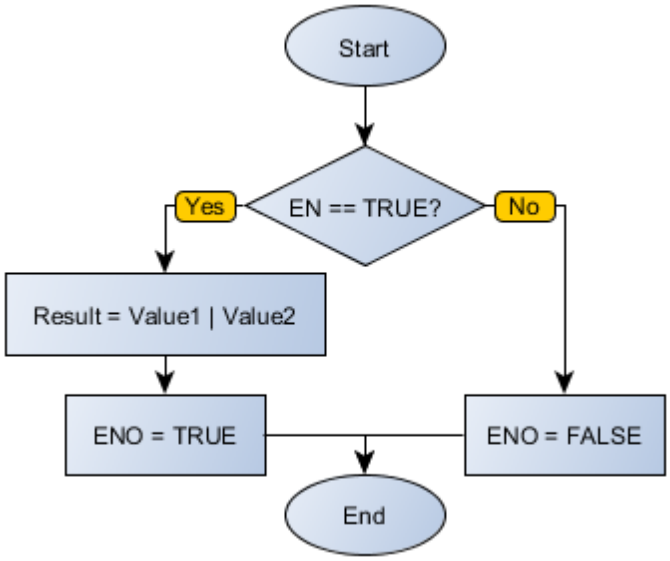
Operation

When this block has a TRUE value in EN, it sends to the Result output the “or” Boolean operation of input variables Value1 and Value2.

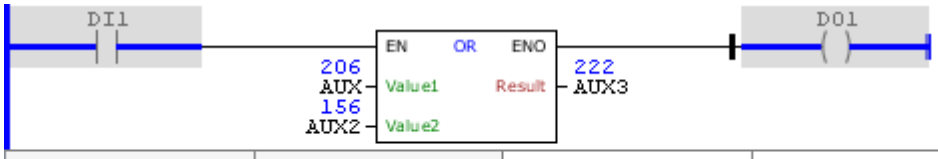
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



AUX	BYTE	11001110	Binary
AUX2	BYTE	10011100	Binary
AUX3	BYTE	11011110	Binary

The example above performs an "or" Boolean operation between AUX and AUX2, storing the result in AUX3.

11.14.7.10.2.4 XNOR

Logical block that performs an Boolean "not exclusive or" operation between two variables, storing the result in a third one.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable for the operation
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable for the operation
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable that stores the result of the operation

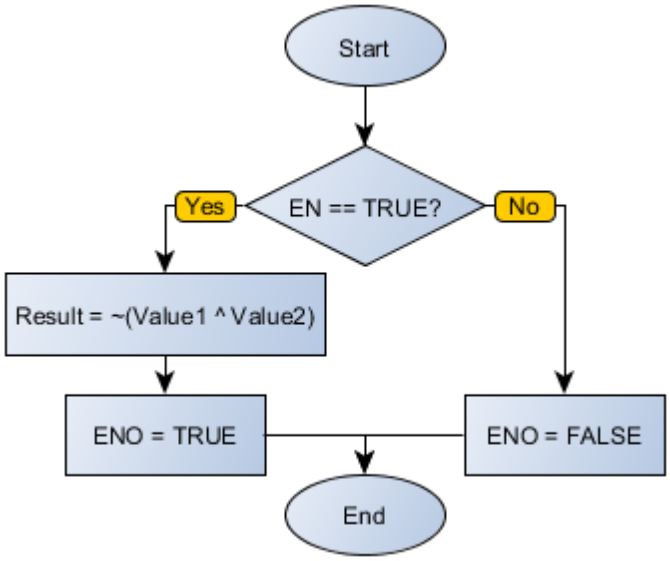
Operation

When this block has a TRUE value in EN, it sends to the Result output the “denied exclusive or” Boolean operation of input variables Value1 and Value2.

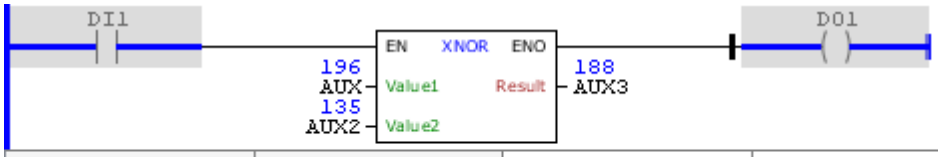
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



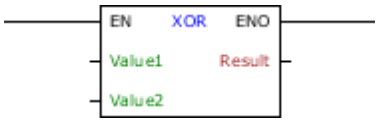
AUX	BYTE	11000100	Binary
AUX2	BYTE	10000111	Binary
AUX3	BYTE	10111100	Binary

The example above performs a "denied exclusive or" Boolean operation between AUX and AUX2, storing the result in AUX3.

11.14.7.10.2.5 XOR

Logical block that performs an Boolean "exclusive or" operation between two variables, storing the result in a third one.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable for the operation
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable for the operation
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable that stores the result of the operation

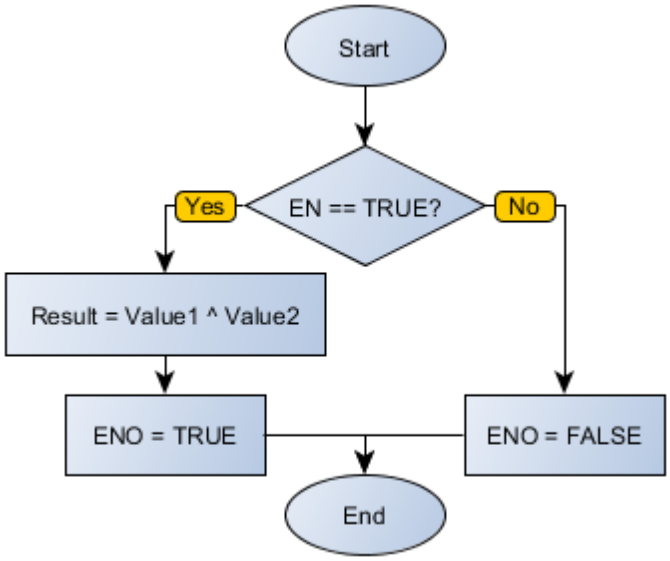
Operation

When this block has a TRUE value in EN, it sends to the Result output the “xor” Boolean operation of input variables Value1 and Value2.

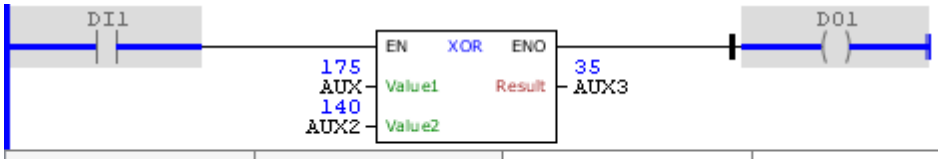
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart



Example



AUX	BYTE	10101111	Binary
AUX2	BYTE	10001100	Binary
AUX3	BYTE	100011	Binary

The example above performs a "xor" Boolean operation between AUX and AUX2, storing the result in AUX3.

11.14.7.10.3 Logic Rotate

11.14.7.10.3.1 ROL

Block that performs a logical left rotation operation in a value passed by Value, storing the result in Result.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable to undergo rotation
	Shift	BYTE USINT	Shift index
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable that stores the result of the operation

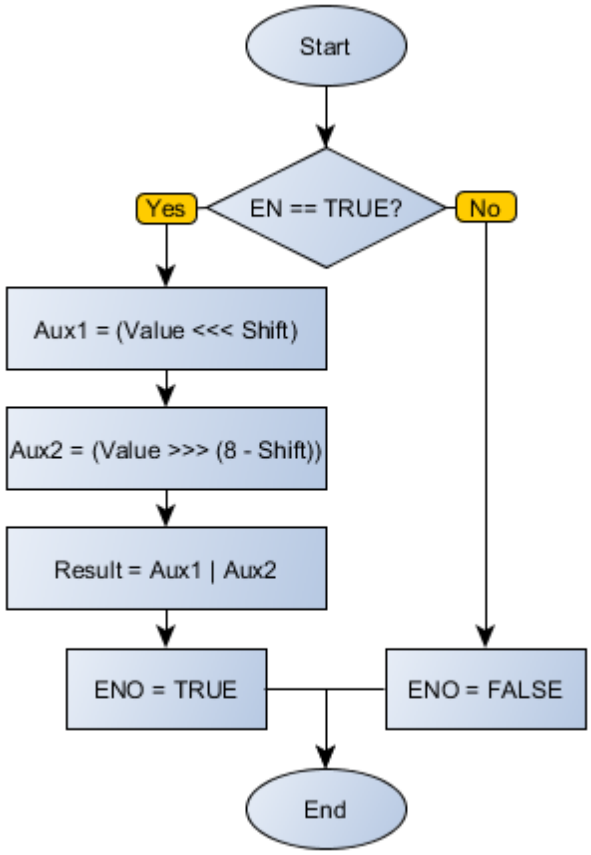
Operation

When this block has a TRUE value in EN, it sends to the Result output the value of the Value variable after performing a number of logical left shifts, according to the Shift value. The most significant bits that are being discarded are returned to the least significant bits, characterizing the rotation.

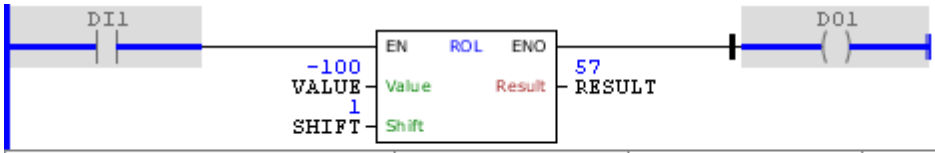
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart

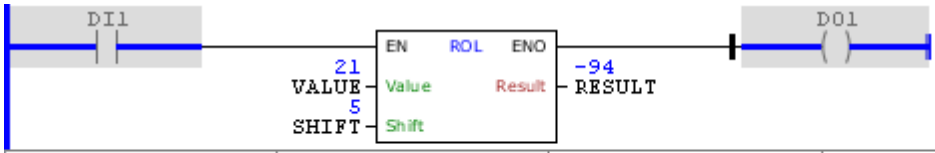


Example



VALUE	SINT	10011100	Binary
SHIFT	BYTE	1	Decimal
RESULT	SINT	111001	Binary

The above example performs a logical left shift by one position in the VALUE variable whose initial value is -100 (1001 1100 in binary). The discarded bits on the left are reinserted on the right. The final result (0011 1001 in binary) is stored in RESULT.



VALUE	SINT	10101	Binary
SHIFT	BYTE	5	Decimal
RESULT	SINT	10100010	Binary

The above example performs a logical left rotation by five positions in the VALUE variable whose initial value is 21 (0001 0101 in binary). The discarded bits on the left are reinserted on the right. The final result (1010 0010 in binary) is stored in RESULT.

11.14.7.10.3.2 ROR

Block that performs a logical right rotation operation in a value passed by Value, storing the result in Result.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable to undergo rotation
	Shift	BYTE USINT	Shift index
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable that stores the result of the operation

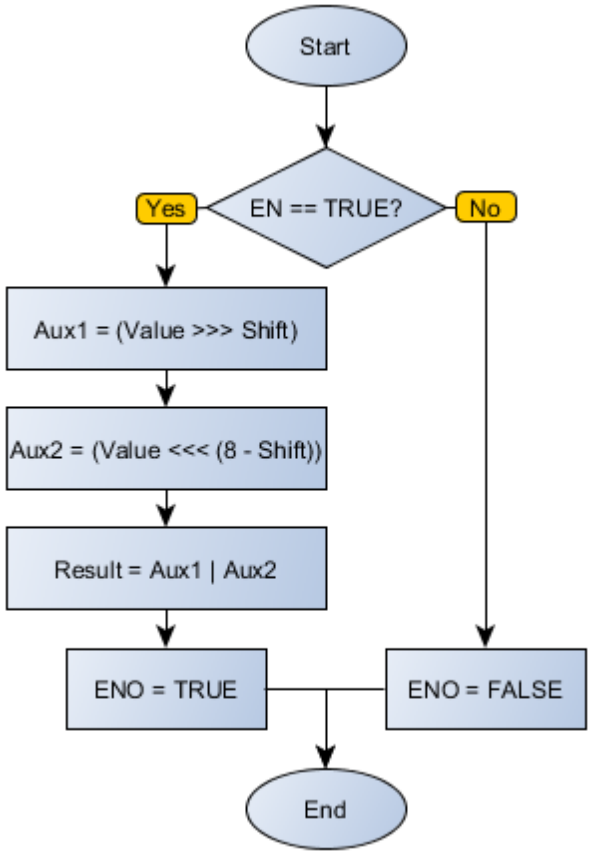
Operation

When this block has a TRUE value in EN, it sends to the Result output the value of the Value variable after performing a number of logical right shifts, according to the Shift value. The least significant bits that are being discarded are returned to the most significant bits, characterizing the rotation.

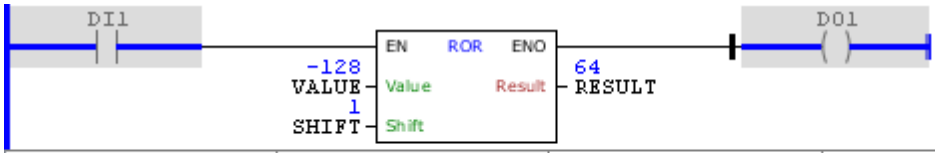
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart

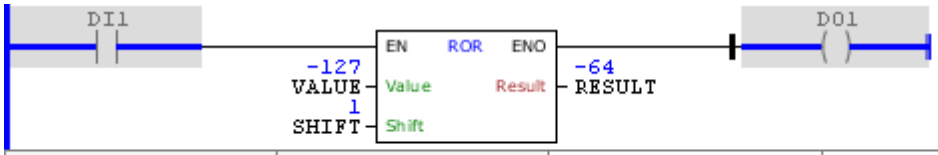


Example



VALUE	SINT	10000000	Binary
SHIFT	BYTE	1	Decimal
RESULT	SINT	1000000	Binary

The above example performs a logic right shift by one position in the VALUE variable whose initial value is -128 (1000 0000 in binary). The discarded bits on the right are reinserted on the left. The final result (0100 0000 in binary) is stored in RESULT. Notice that the sign is not preserved in this operation.



VALUE	SINT	10000001	Binary
SHIFT	BYTE	1	Decimal
RESULT	SINT	11000000	Binary

The above example performs a logical right rotation by one position in the VALUE variable whose initial value is -127 (1000 0001 in binary). The discarded bits on the right are reinserted on the left. The final result (1100 0000 in binary) is stored in RESULT.

11.14.7.10.4 Logic Shift

11.14.7.10.4.1 ASHL

Block that performs a binary left shift operation in a value passed by Value, storing the result in Result.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	SINT INT DINT	Variable to undergo shift
	Shift	BYTE USINT	Shift index
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	SINT INT DINT	Variable that stores the result of the operation

Operation

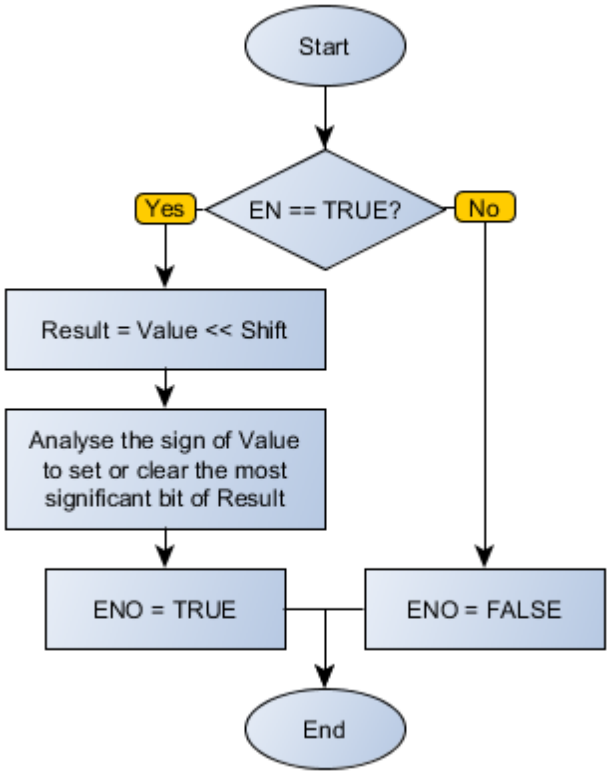
When this block has a TRUE value in EN, it sends to the Result output the value of the Value variable after performing a number of arithmetic left shifts, according to the Shift value.

NOTE!
All arithmetic shifts implemented maintain the sign of the variable.

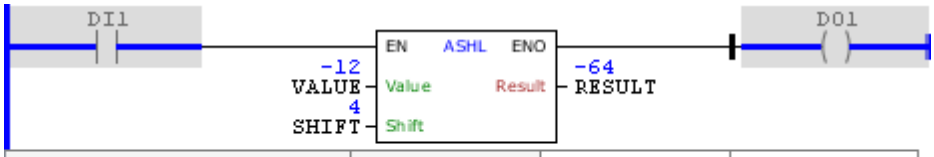
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart

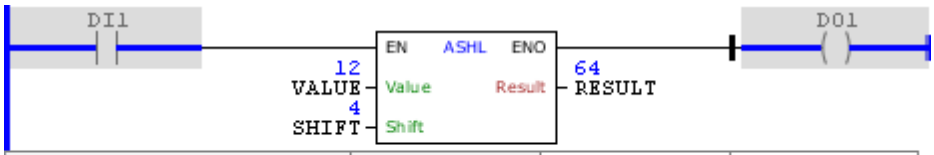


Example



VALUE	SINT	11110100	Binary
SHIFT	USINT	100	Binary
RESULT	SINT	11000000	Binary

Description of exemple.



VALUE	SINT	1100	Binary
SHIFT	USINT	100	Binary
RESULT	SINT	1000000	Binary

Description of exemple.

11.14.7.10.4.2 ASHR

Block that performs arithmetic left shift operation in a value passed by Value, storing the result in Result.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	SINT INT DINT	Variable to undergo shift
	Shift	BYTE USINT	Shift index
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	SINT INT DINT	Variable that stores the result of the operation

Operation

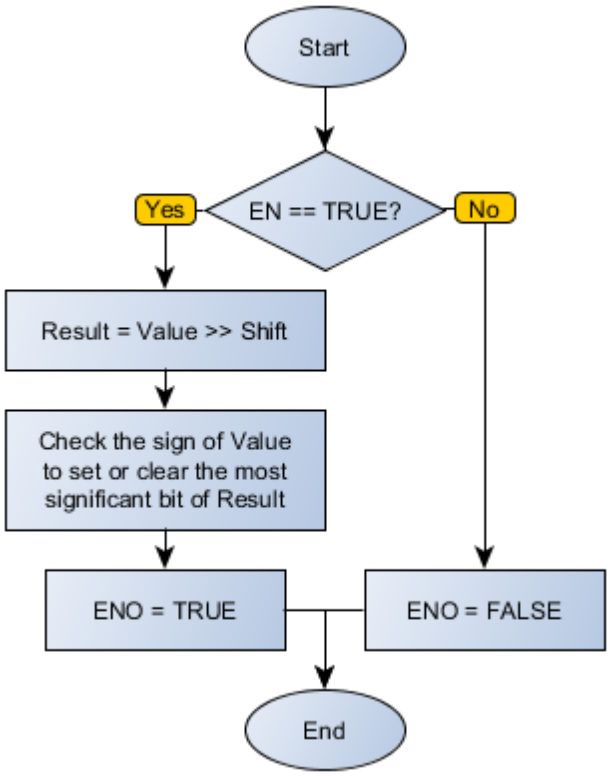
When this block has a TRUE value in EN, it sends to the Result output the value of the Value variable after performing a number of arithmetic right shifts, according to the Shift value.

NOTE!
All arithmetic shifts implemented maintain the sign of the variable.

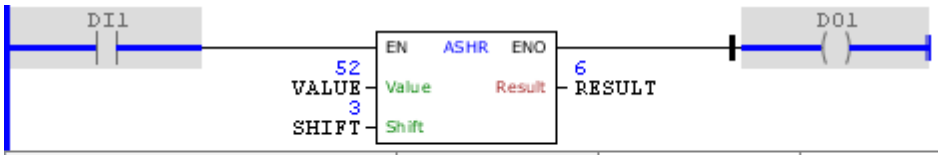
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart

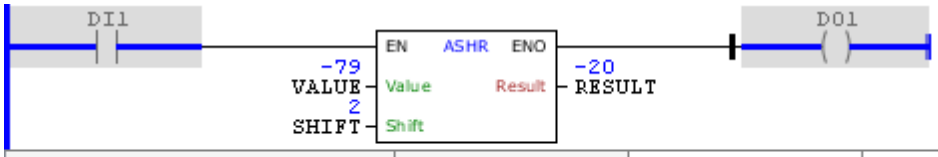


Example



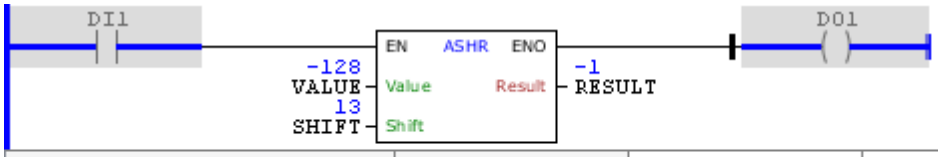
VALUE	SINT	110100	Binary
SHIFT	BYTE	3	Decimal
RESULT	INT	110	Binary

The above example performs an arithmetic right shift by three positions in the VALUE variable whose initial value is 52 (0011 0100 in binary). The bits on the right are being discarded, and on the left new zeros are inserted. The final result (0000 0110 in binary) is stored in RESULT.



VALUE	SINT	10110001	Binary
SHIFT	BYTE	2	Decimal
RESULT	INT	1111111111101100	Binary

The above example performs an arithmetic right shift by two positions in the VALUE variable whose initial value is -79 (1011 0001 in binary). The bits on the right will be discarded and new ones on the left are inserted, since the arithmetic right shifts preserve the sign of the variable. The final result (1111 0110 in binary) is stored in RESULT.



VALUE	SINT	10000000	Binary
SHIFT	BYTE	13	Decimal
RESULT	INT	1111111111111111	Binary

The above example performs an arithmetic right shift by thirteen positions in the VALUE variable whose initial value is -128 (1000 0000 in binary). The bits on the right are being discarded, and on the left new ones are inserted. The final result (1111 1111 in binary) is stored in RESULT.

11.14.7.10.4.3 SHL

Block that performs a binary logical left shift operation in a value passed by Value, storing the result in Result.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable to undergo shift
	Shift	BYTE USINT	Shift index
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable that stores the result of the operation

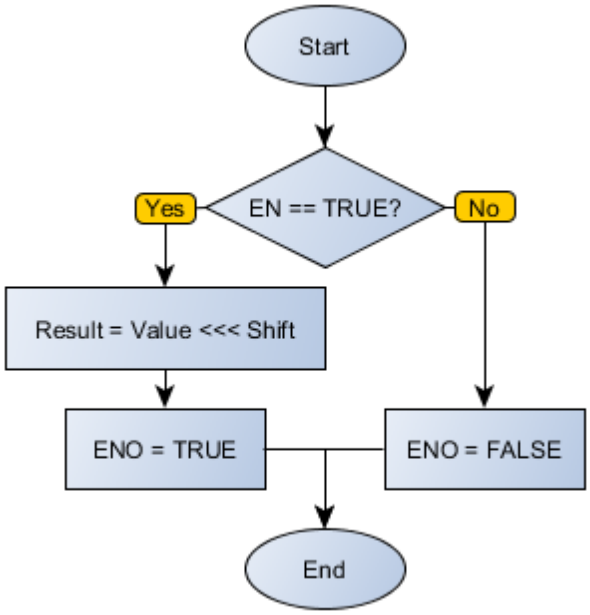
Operation

When this block has a TRUE value in EN, it sends to the Result output the value of the Value variable after performing a number of logical shifts left, according to the Shift value.

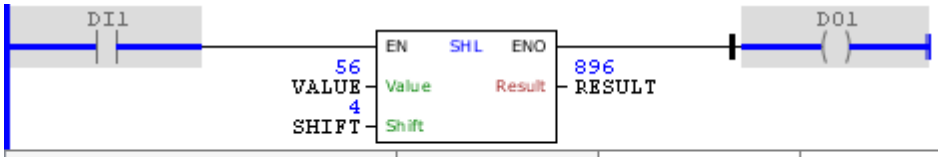
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart

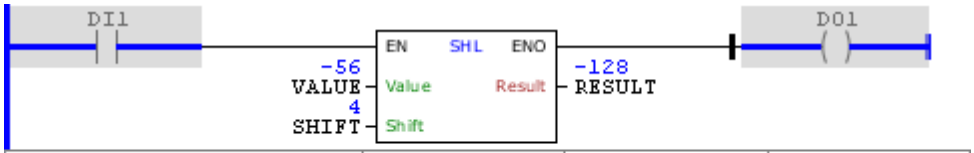


Example



VALUE	INT	111000	Binary
SHIFT	BYTE	4	Decimal
RESULT	INT	1110000000	Binary

The above example performs a logical right shift by four positions in the VALUE variable whose initial value is 56 (0011 1000 in binary). The bits on the left are being discarded, and on the left new zeros are inserted. The final result (0011 1000 0000 in binary) is stored in RESULT.



VALUE	SINT	11001000	Binary
SHIFT	USINT	4	Decimal
RESULT	SINT	10000000	Binary

The above example performs a logical right shift by four positions in the VALUE variable whose initial value is -56 (1100 1000 in binary). The bits on the left are being discarded, and on the left new zeros are inserted. The final result (1100 1000 0000 in binary) is stored in RESULT. Since RESULT is SINT type, it only accepts the first eight bits (1000 0000).

11.14.7.10.4.4 SHR

Block that performs a binary logical right shift operation in a value passed by Value, storing the result in Result.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable to undergo shift
	Shift	BYTE USINT	Shift index
VAR_OUTPUT	ENO	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable that stores the result of the operation

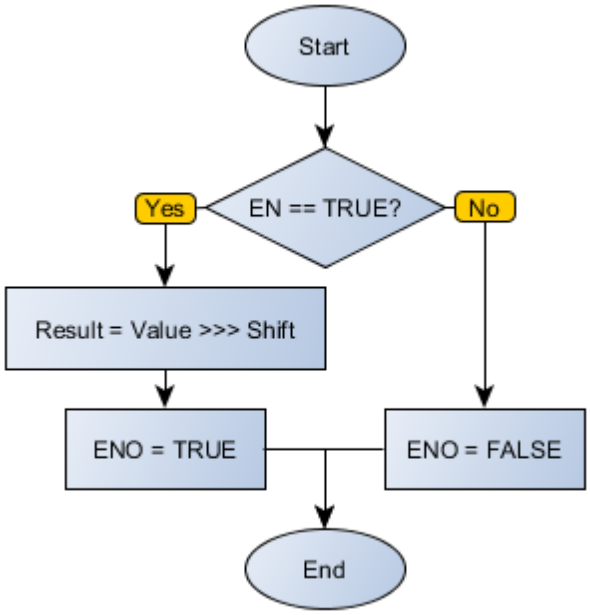
Operation

When this block has a TRUE value in EN, it sends to the Result output the value of the Value variable after performing a number of logical shifts right, according to the Shift value.

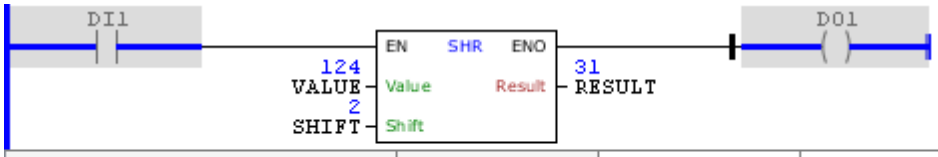
When EN has FALSE value, Result remains unchanged.

The ENO value forwards to the next Ladder block the EN value after the operation is completed.

Block Flowchart

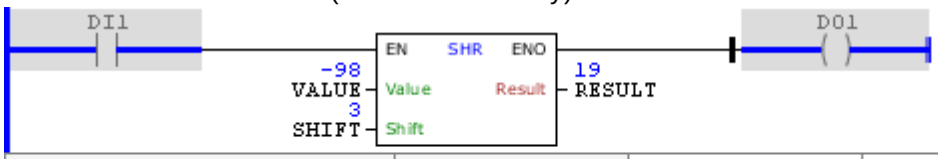


Example



VALUE	SINT	1111100	Binary
SHIFT	BYTE	2	Decimal
RESULT	SINT	11111	Binary

The above example performs a logical right shift by two positions in the VALUE variable whose initial value is 124 (0111 1100 in binary). The bits on the right are being discarded, and on the left new zeros are inserted. The final result (0001 1111 in binary) is stored in RESULT.



VALUE	SINT	10011110	Binary
SHIFT	BYTE	3	Decimal
RESULT	INT	10011	Binary

The above example performs a logical right shift by three positions in the VALUE variable whose initial value is -98 (1001 1110 in binary). The bits on the right are being discarded, and on the left new zeros are inserted. The final result (0001 0011 in binary) is stored in RESULT.

11.14.7.11 Math

11.14.7.11.1 Math Basic

11.14.7.11.1.1 ABS

Block that calculates the Value module, storing the result in Result.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Reference variable for the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Variable that stores the result of the operation

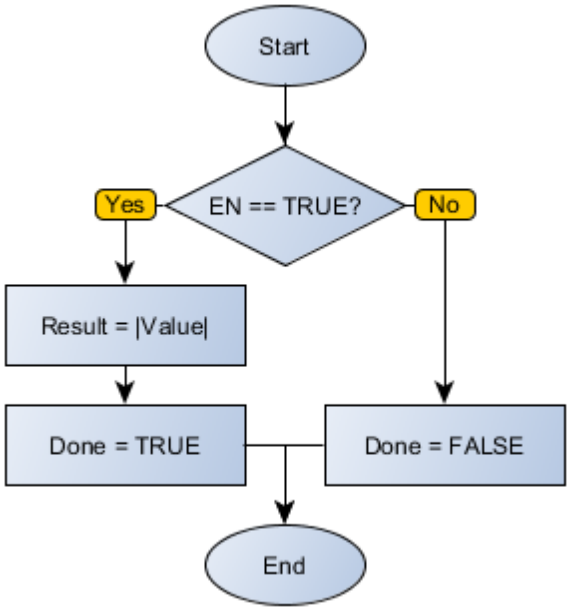
Operation

When this block has a TRUE value in EN, it sends to the Result output the absolute value of the

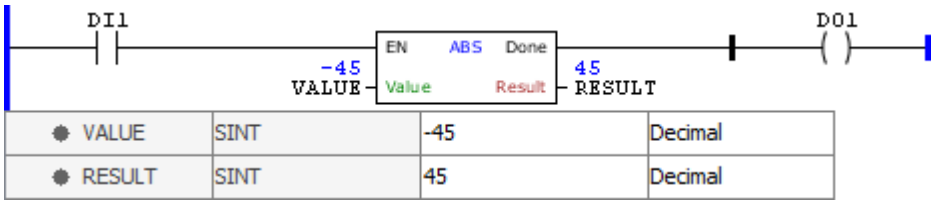
Value variable. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

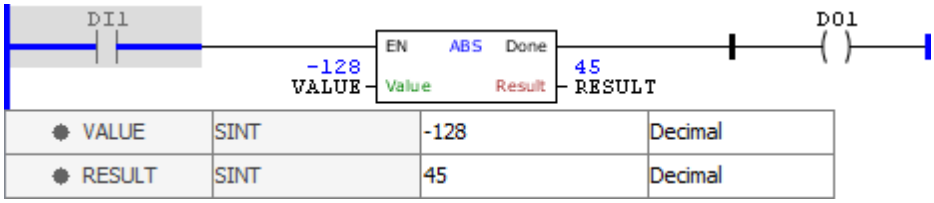


Example



VALUE	SINT	-45	Decimal
RESULT	SINT	45	Decimal

The above example calculates the absolute value of the VALUE variable whose initial value is -45, storing the final result, 45, in RESULT.



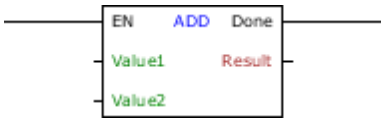
VALUE	SINT	-128	Decimal
RESULT	SINT	45	Decimal

The above example calculates the absolute value of the VALUE variable whose initial value is -45. The final result, 128, cannot be stored in RESULT, because it is outside the limits of accepted values by SINT type. Therefore, RESULT remains unchanged and the output is disabled.

11.14.7.11.1.2 ADD

Block that calculates the sum of the values of Value1 and Value2, storing the result in Result.

Ladder Representation



Block Structure

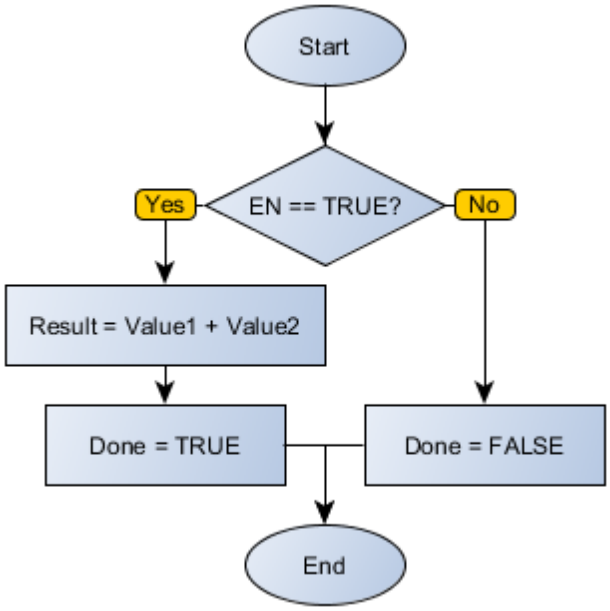
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	First addend of the operation
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Second addend of the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Variable that stores the result of the operation

Operation

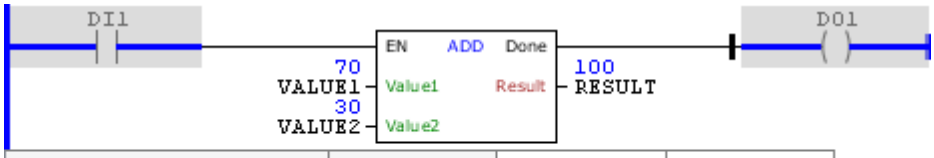
When this block has a TRUE value in EN, it sends to the Result output the sum of Value1 and Value2 variables. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

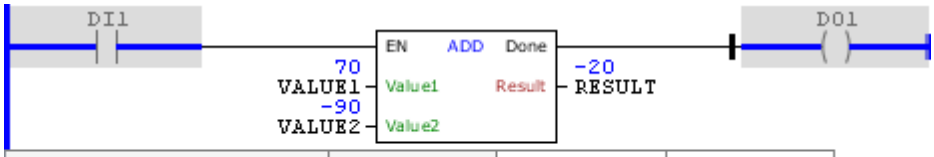


Example



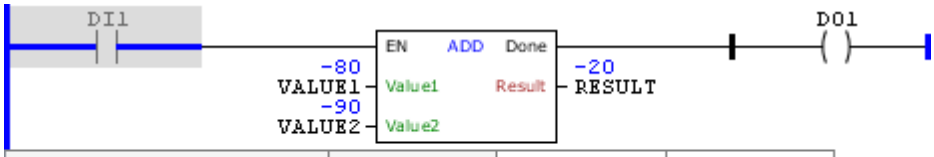
VALUE1	SINT	70	Decimal
VALUE2	SINT	30	Decimal
RESULT	SINT	100	Decimal

The above example calculates the sum of VALUE 1 and VALUE2 variables, storing the final result in RESULT.



VALUE1	SINT	70	Decimal
VALUE2	SINT	-90	Decimal
RESULT	SINT	-20	Decimal

The above example calculates the sum of VALUE 1 and VALUE2 variables, storing the final result in RESULT. Notice that the block accepts arguments of both signs.



VALUE1	SINT	-80	Decimal
VALUE2	SINT	-90	Decimal
RESULT	SINT	-20	Decimal

The above example calculates the sum of VALUE1 and VALUE2 variables. The final result -170 cannot be stored in RESULT, because it is outside the limits of accepted values by SINT type. Therefore, RESULT remains unchanged and the output is disabled.

11.14.7.11.1.3 DIV

Block that calculates the division of the values of Value1 and Value2, storing the result in Result.

Ladder Representation

Block Structure

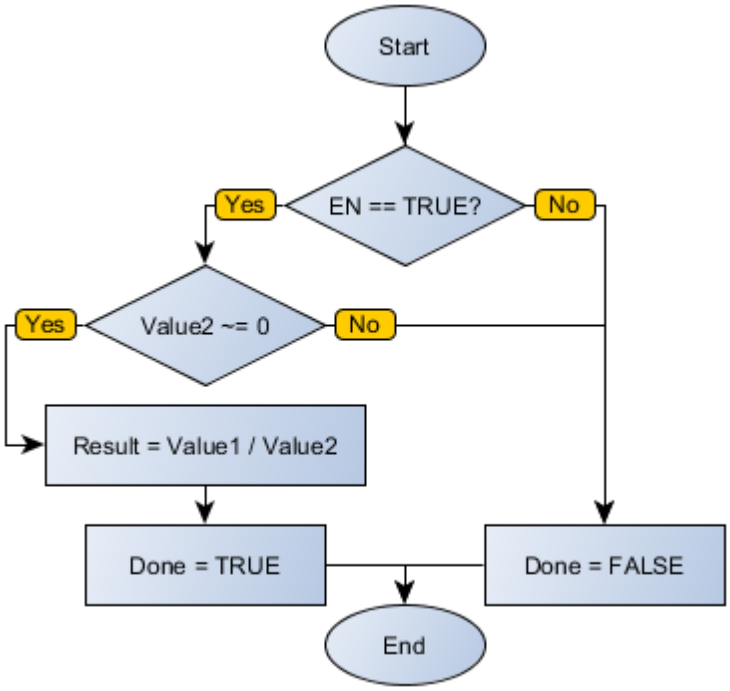
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Dividend of the operation
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Divisor of the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Variable that stores the result of the operation

Operation

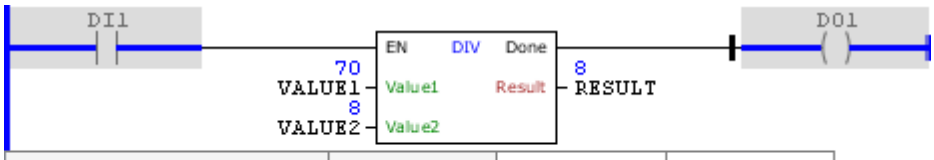
When this block has a TRUE value in EN, it sends to the Result output the division of Value1 and Value2 variables. The value stored will be the exact division if Result is REAL, or, in other cases, only the quotient. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

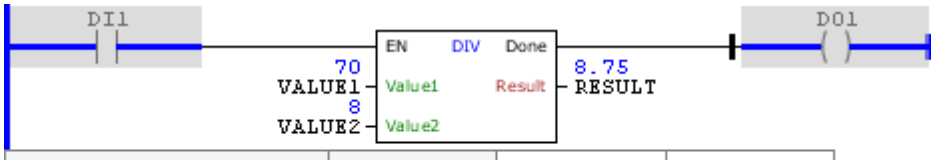


Example



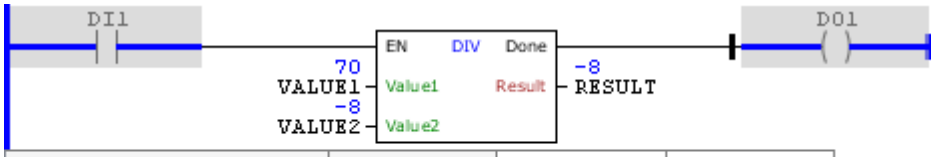
VALUE1	SINT	70	Decimal
VALUE2	SINT	8	Decimal
RESULT	SINT	8	Decimal

The above example calculates the division of VALUE 1 and VALUE2 variables, storing the final result in RESULT. Since RESULT is SINT type, only the quotient is stored in it.



VALUE1	SINT	70	Decimal
VALUE2	SINT	8	Decimal
RESULT	REAL	8.75	Float

The above example calculates the division of VALUE 1 and VALUE2 variables, storing the final result in RESULT. Since RESULT is of REAL type, the exact value of the division is stored in it.



VALUE1	SINT	70	Decimal
VALUE2	SINT	-8	Decimal
RESULT	SINT	-8	Decimal

The above example calculates the division of VALUE 1 and VALUE2 variables, storing the final result in RESULT. Since RESULT is SINT type, only the quotient is stored in it. Notice that the block accepts arguments of both signs.

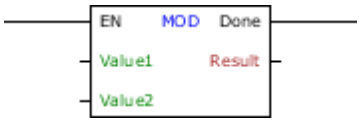
VALUE1	SINT	70	Decimal
VALUE2	SINT	0	Decimal
RESULT	SINT	-8	Decimal

The above example calculates the division of VALUE 1 and VALUE2 variables, storing the final result in RESULT. Since VALUE2 is zero, the block generates a runtime error, RESULT remains unchanged and the output is disabled.

11.14.7.11.1.4 MOD

Block that calculates the remainder of the values of Value1 and Value2, storing the result in Result.

Ladder Representation



Block Structure

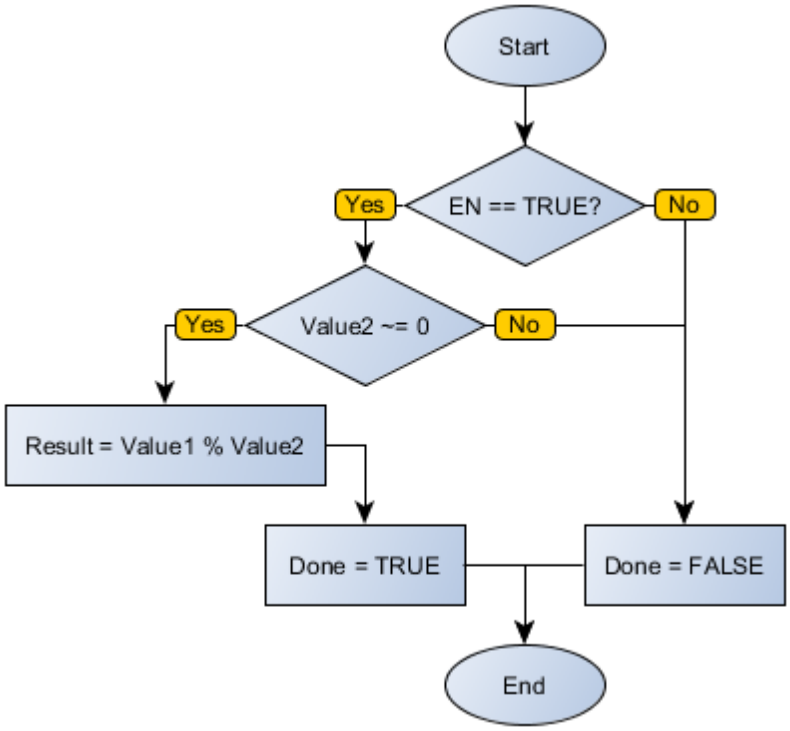
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Dividend of the operation
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Divisor of the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT	Variable that stores the result of the operation

Operation

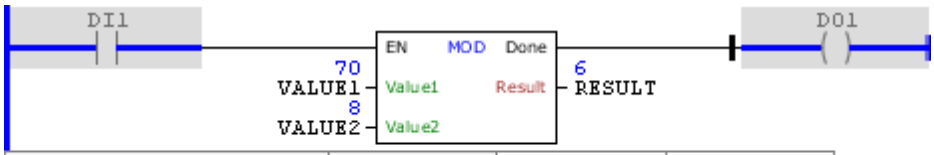
When this block has a TRUE value in EN, it sends to the Result output the remainder of Value1 and Value2 variables. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

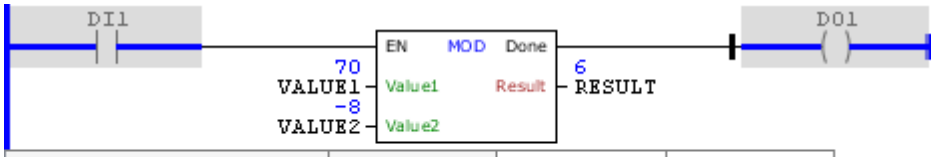


Example



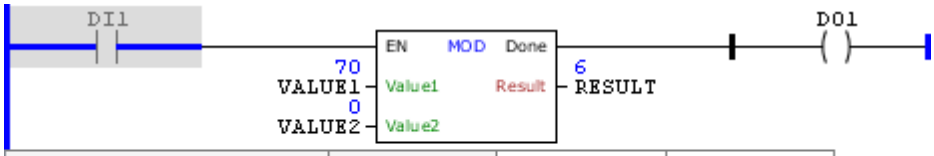
VALUE1	SINT	70	Decimal
VALUE2	SINT	8	Decimal
RESULT	SINT	6	Decimal

The above example calculates the remainder of VALUE 1 and VALUE2 variables, storing the final result in RESULT.



VALUE1	SINT	70	Decimal
VALUE2	SINT	-8	Decimal
RESULT	SINT	6	Decimal

The above example calculates the remainder of VALUE 1 and VALUE2 variables, storing the final result in RESULT. Notice that the block accepts arguments of both signs.



VALUE1	SINT	70	Decimal
VALUE2	SINT	0	Decimal
RESULT	SINT	6	Decimal

The above example calculates the remainder of VALUE 1 and VALUE2 variables, storing the final result in RESULT. Since VALUE2 is zero, the block generates a runtime error, RESULT remains unchanged and the output is disabled.

11.14.7.11.1.5 MUL

Block that calculates the multiplication of the values of Value1 and Value2, storing the result in Result.

Ladder Representation

Block Structure

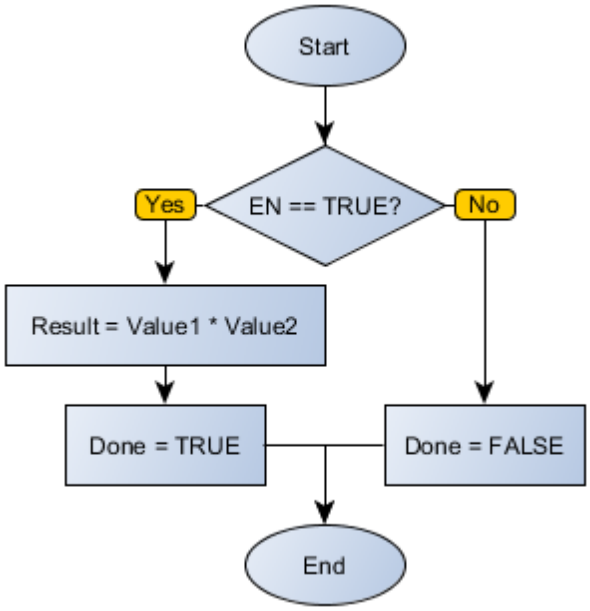
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	First factor of the operation
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Second factor of the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Variable that stores the result of the operation

Operation

When this block has a TRUE value in EN, it sends to the Result output the multiplication of Value1 and Value2 variables. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

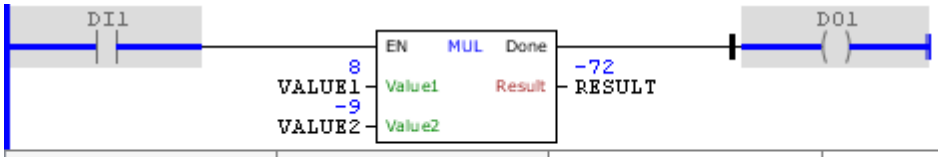
Block Flowchart



Example

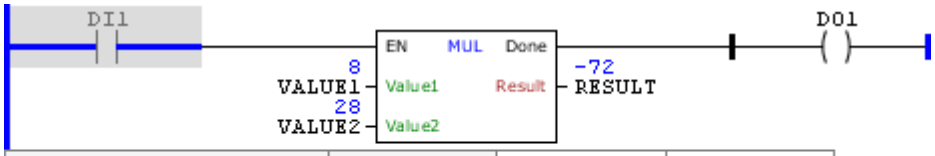
VALUE1	SINT	8	Decimal
VALUE2	SINT	12	Decimal
RESULT	SINT	96	Decimal

The above example calculates the product of VALUE 1 and VALUE2 variables, storing the final result in RESULT.



VALUE1	SINT	8	Decimal
VALUE2	SINT	-9	Decimal
RESULT	SINT	-72	Decimal

The above example calculates the product of VALUE 1 and VALUE2 variables, storing the final result in RESULT. Notice that the block accepts arguments of both signs.



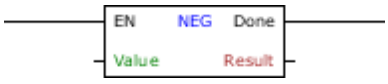
VALUE1	SINT	8	Decimal
VALUE2	SINT	28	Decimal
RESULT	SINT	-72	Decimal

The above example calculates the product of VALUE1 and VALUE2 variables. The final result 224 cannot be stored in RESULT, because it is outside the limits of accepted values by SINT type. Therefore, RESULT remains unchanged and the output is disabled.

11.14.7.11.1.6 NEG

Block that calculates the opposite (i.e., the product with -1) of a value passed by Value, storing the result in Result.

Ladder Representation



Block Structure

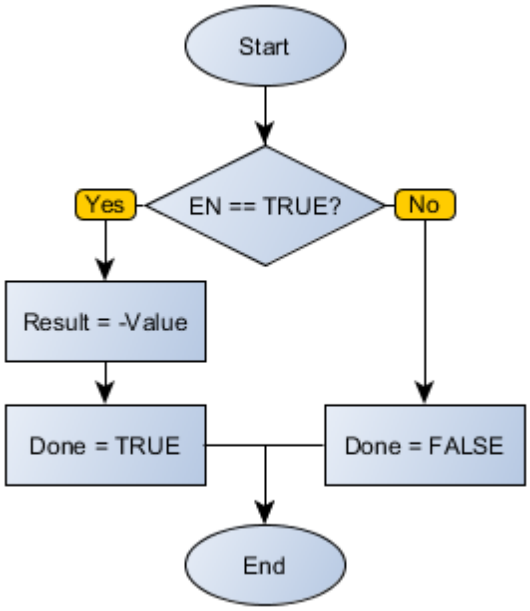
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Reference variable for the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Variable that stores the result of the operation

Operation

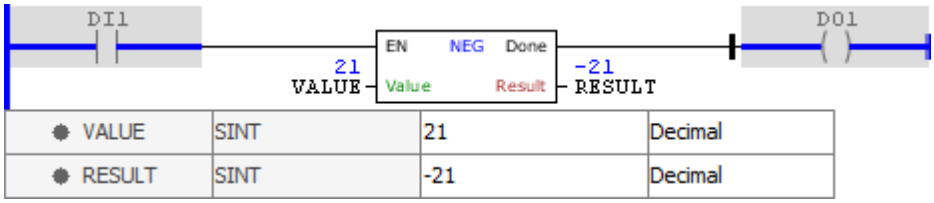
When this block has a TRUE value in EN, it sends to the Result output the opposite of the Value variable. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

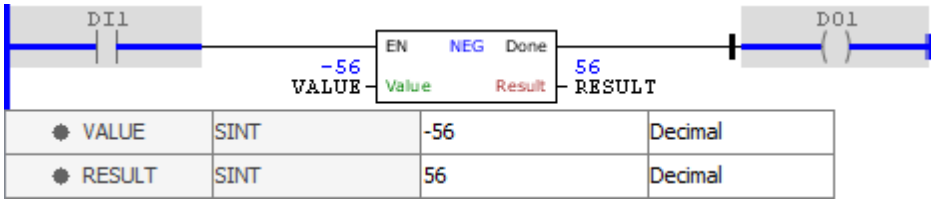


Example



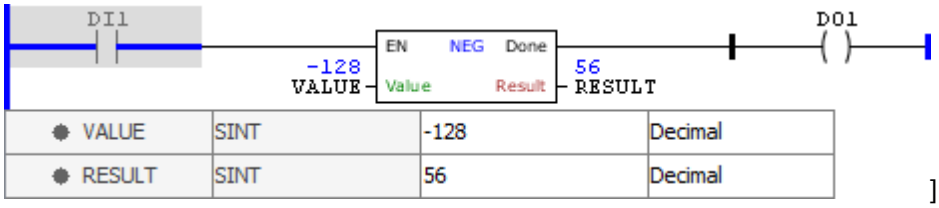
VALUE	SINT	21	Decimal
RESULT	SINT	-21	Decimal

The above example calculates the opposite of the VALUE variable whose initial value is 21, storing the final result, -21, in RESULT.



VALUE	SINT	-56	Decimal
RESULT	SINT	56	Decimal

The above example calculates the opposite of the VALUE variable whose initial value is -56, storing the final result, 56, in RESULT.



VALUE	SINT	-128	Decimal
RESULT	SINT	56	Decimal

]

The above example calculates the opposite of the VALUE variable whose initial value is -128. The final result, 128, cannot be stored in RESULT, because it is outside the limits of accepted values by SINT type. Therefore, RESULT remains unchanged and the output is disabled.

11.14.7.11.1.7 SUB

Block that calculates the subtraction between the Value1 and Value2 values, storing the result in Result.

Ladder Representation

Block Structure

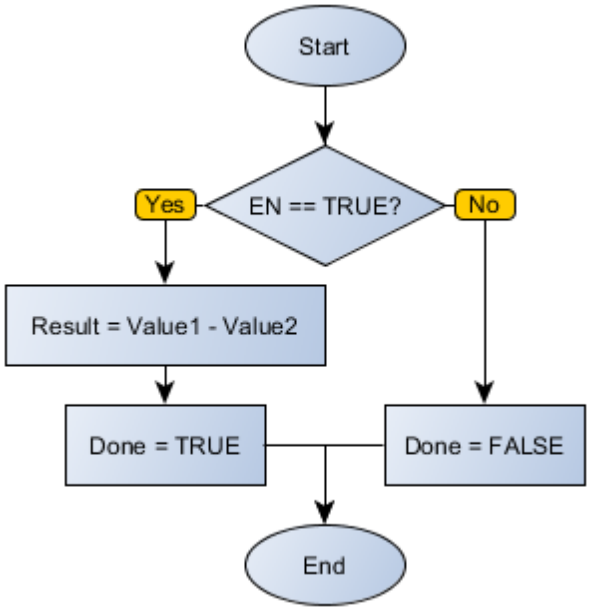
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Minuend of operation
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Subtrahend of operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Variable that stores the result of the operation

Operation

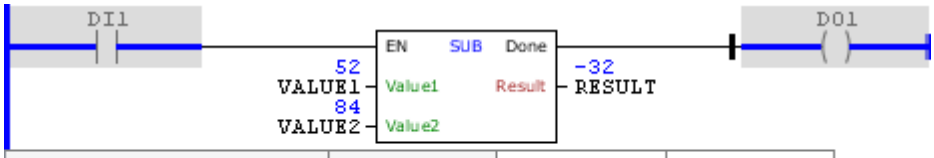
When this block has a TRUE value in EN, it sends to the Result output the subtraction of Value1 and Value2 variables. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

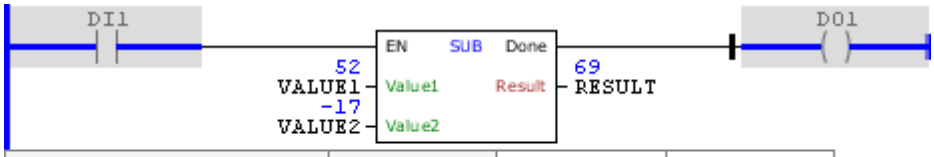


Example



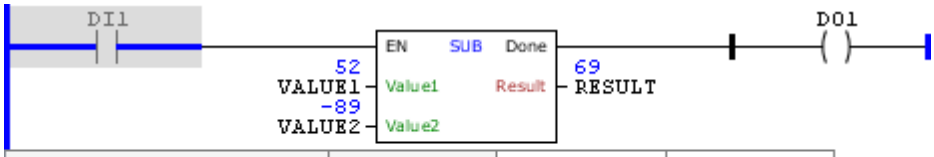
VALUE1	SINT	52	Decimal
VALUE2	SINT	84	Decimal
RESULT	SINT	-32	Decimal

The above example calculates the subtraction of VALUE 1 and VALUE2 variables, storing the final result in RESULT.



VALUE1	SINT	52	Decimal
VALUE2	SINT	-17	Decimal
RESULT	SINT	69	Decimal

The above example calculates the subtraction of VALUE 1 and VALUE2 variables, storing the final result in RESULT. Notice that the block accepts arguments of both signs.



VALUE1	SINT	52	Decimal
VALUE2	SINT	-89	Decimal
RESULT	SINT	69	Decimal

The above example calculates the subtraction of VALUE1 and VALUE2 variables. The final result 141 cannot be stored in RESULT, because it is outside the limits of accepted values by SINT type. Therefore, RESULT remains unchanged and the output is disabled.

11.14.7.11.2 Math Extended

11.14.7.11.2.1 ALOG10

Block that calculates the antilogarithm (exponent with base 10) of the Value value, storing the result in Result.

Ladder Representation

Block Structure

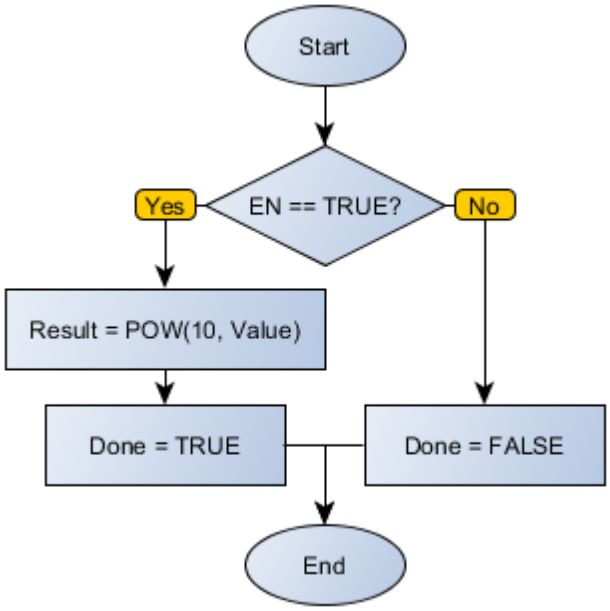
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Reference variable for the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	REAL	Variable that stores the result of the operation

Operation

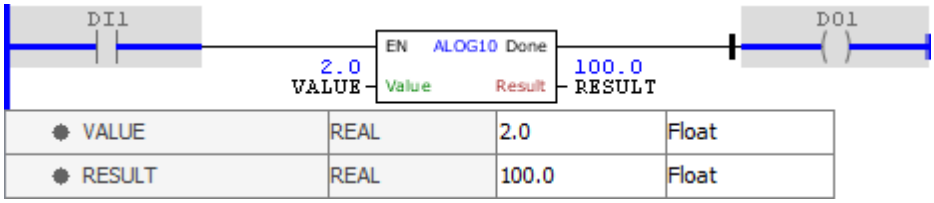
When this block has a TRUE value in EN, it sends to the Result output the antilogarithm of the Value variable. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

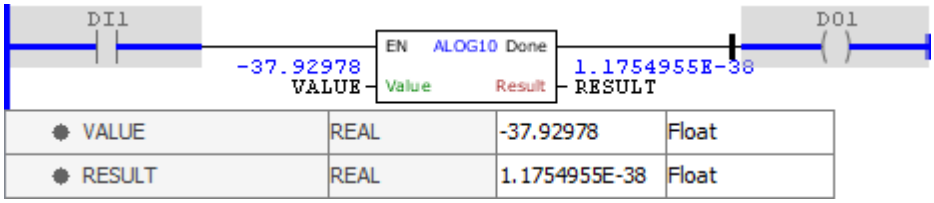


Example



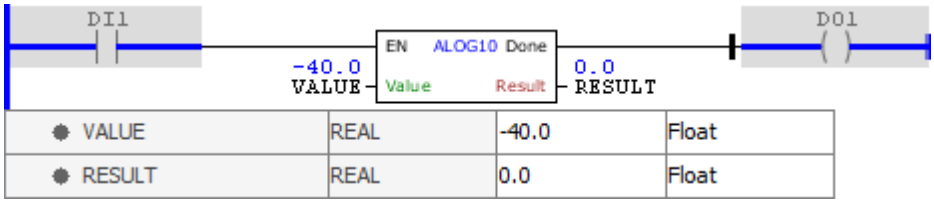
VALUE	REAL	2.0	Float
RESULT	REAL	100.0	Float

The above example calculates the antilogarithm of the VALUE variable, storing the final result in RESULT. The block ends with success and Done output is activated.



VALUE	REAL	-37.92978	Float
RESULT	REAL	1.1754955E-38	Float

The above example calculates the antilogarithm of the VALUE variable, storing the final result in RESULT. The indicated value is the minimum input value for which the block revolutions a nonzero result. The block ends with success and Done output is activated.



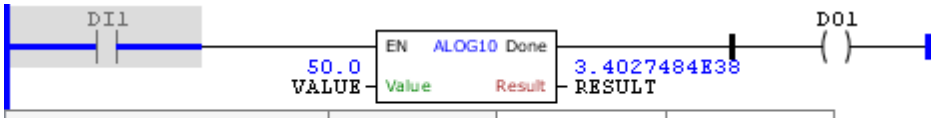
VALUE	REAL	-40.0	Float
RESULT	REAL	0.0	Float

The above example calculates the antilogarithm of the VALUE variable, storing the final result in

RESULT. Below the minimum values cause the block to return a null value. The block ends with success and Done output is activated.

VALUE	REAL	38.53183	Float
RESULT	REAL	3.4027484E38	Float

The above example calculates the antilogarithm of the VALUE variable, storing the final result in RESULT. The indicated value is the maximum input value for which the block revolutions a valid result. The block ends with success and Done output is activated.



VALUE	REAL	50.0	Float
RESULT	REAL	3.4027484E38	Float

The above example calculates the antilogarithm of the VALUE variable, storing the final result in RESULT. Values higher than the maximum cause the block to generate an error, the RESULT output remains unchanged and Done output is disabled.

11.14.7.11.2.2 EXP

Block that calculates the exponential of the Euler number "and" raised to the value of Value, storing the result in Result.

Ladder Representation

Block Structure

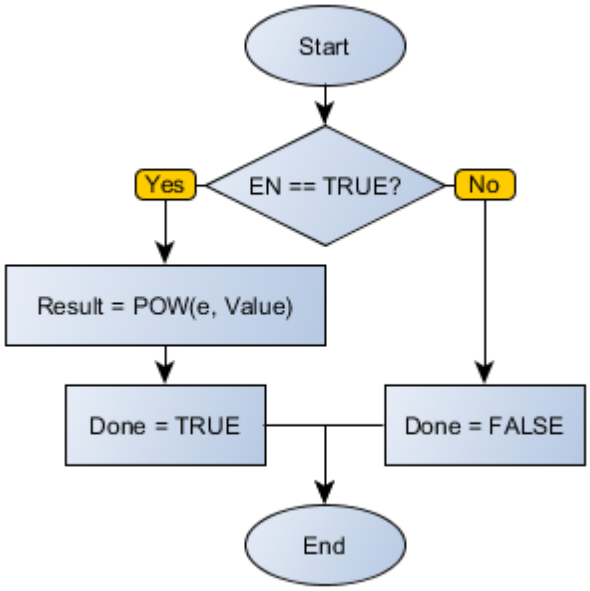
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Reference variable for the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	REAL	Variable that stores the result of the operation

Operation

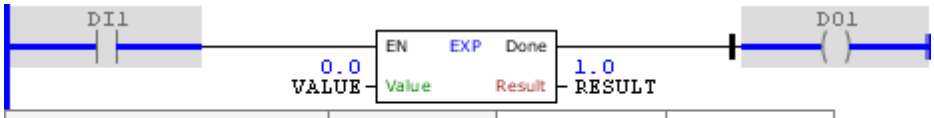
When this block has a TRUE value in EN, it sends to the Result output the exponent of the Euler number "and" raised to the Value variable. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart



Example



VALUE	REAL	0.0	Float
RESULT	REAL	1.0	Float

The above example calculates the exponent of the VALUE variable, storing the final result in RESULT. The block ends with success and Done output is activated.

VALUE	REAL	-87.33654	Float
RESULT	REAL	1.1754997E-38	Float

The above example calculates the exponent of the VALUE variable, storing the final result in RESULT. The indicated value is the minimum input value for which the block revolutions a nonzero result. The block ends with success and Done output is activated.

VALUE	REAL	-50000.0	Float
RESULT	REAL	0.0	Float

The above example calculates the exponent of the VALUE variable, storing the final result in RESULT. Values below the minimum cause the block to return to a null value. The block ends with success and Done output is activated.

VALUE	REAL	88.72283	Float
RESULT	REAL	3.4027985E38	Float

The above example calculates the exponent of the VALUE variable, storing the final result in RESULT. The indicated value is the maximum input value for which the block revolutions a valid result. The block ends with success and Done output is activated.

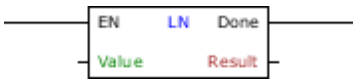
VALUE	REAL	90.0	Float
RESULT	REAL	3.4027985E38	Float

The above example calculates the exponent of the VALUE variable, storing the final result in RESULT. Values higher than the maximum cause the block to generate an error, the RESULT output remains unchanged and Done output is disabled.

11.14.7.11.2.3 LN

Block that calculates the natural logarithm of the Value value, storing the result in Result.

Ladder Representation



Block Structure

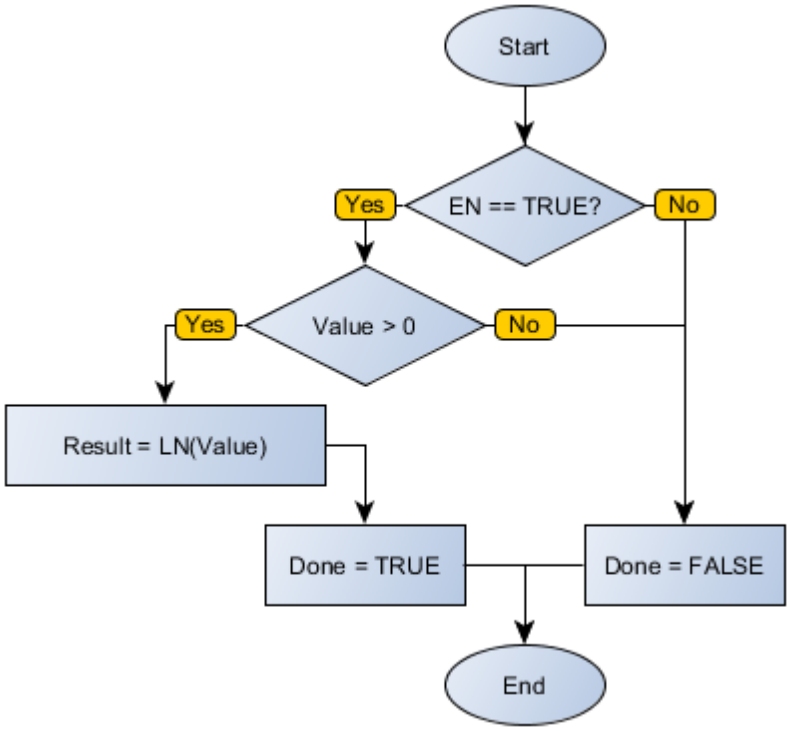
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Reference variable for the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	REAL	Variable that stores the result of the operation

Operation

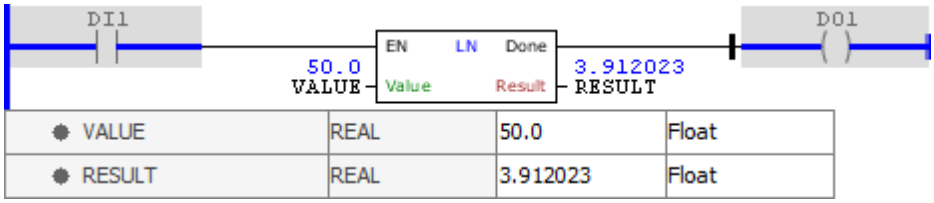
When this block has a TRUE value in EN, it sends to the Result output the natural logarithm of the Value variable. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart



Example



VALUE	REAL	50.0	Float
RESULT	REAL	3.912023	Float

The above example calculates the natural logarithm of the VALUE variable, storing the final result in RESULT. The block ends with success and Done output is activated.

VALUE	REAL	0.0	Float
RESULT	REAL	3.912023	Float

The above example calculates the natural logarithm of the VALUE variable, storing the final result in RESULT. The block generates a runtime error, since VALUE has value zero, and Done output is disabled.

11.14.7.11.2.4 LOG10

Block that calculates the common logarithm (base 10) of the Value value, storing the result in Result.

Ladder Representation

Block Structure

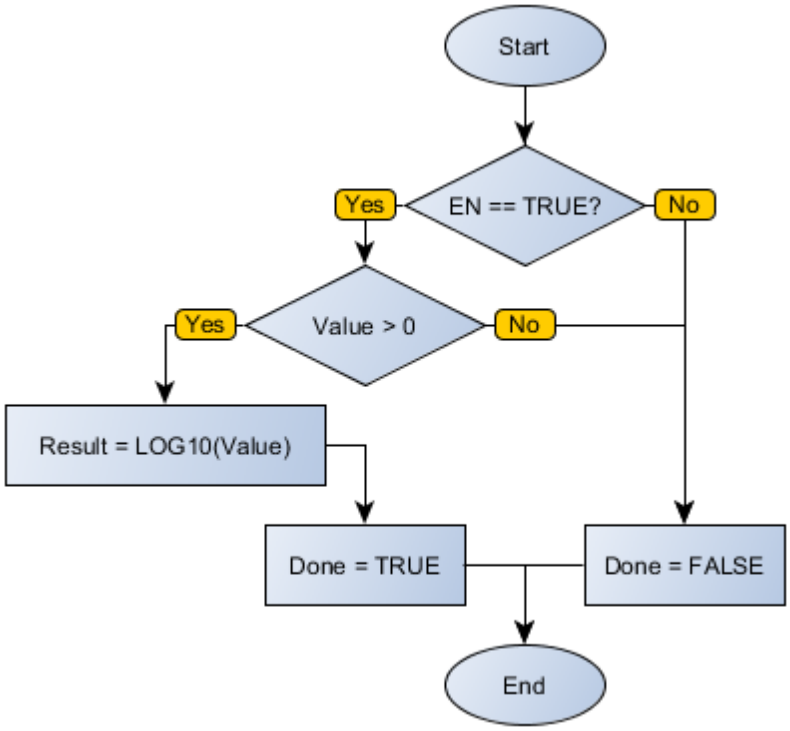
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Reference variable for the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	REAL	Variable that stores the result of the operation

Operation

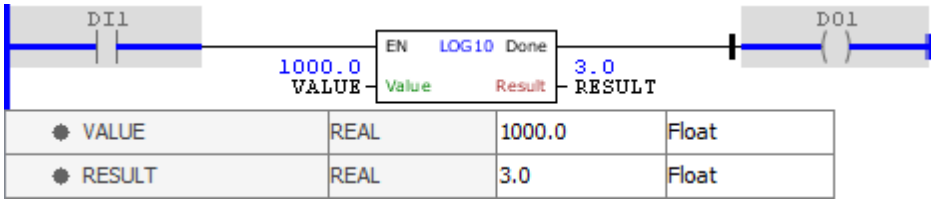
When this block has a TRUE value in EN, it sends to the Result output the common logarithm of the Value variable. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

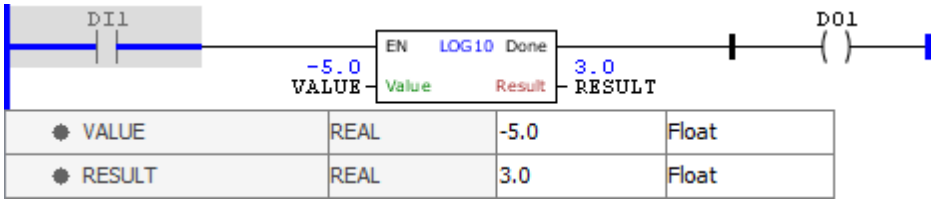


Example



VALUE	REAL	1000.0	Float
RESULT	REAL	3.0	Float

The above example calculates the common logarithm of the VALUE variable, storing the final result in RESULT. The block ends with success and Done output is activated.



VALUE	REAL	-5.0	Float
RESULT	REAL	3.0	Float

The above example calculates the common logarithm of the VALUE variable, storing the final result in RESULT. The block generates a runtime error, since VALUE has negative value, and Done output is disabled.

11.14.7.11.2.5 POW

Block that calculates the value of Value raised to the exponent Power, storing the result in Result.

Ladder Representation

Block Structure

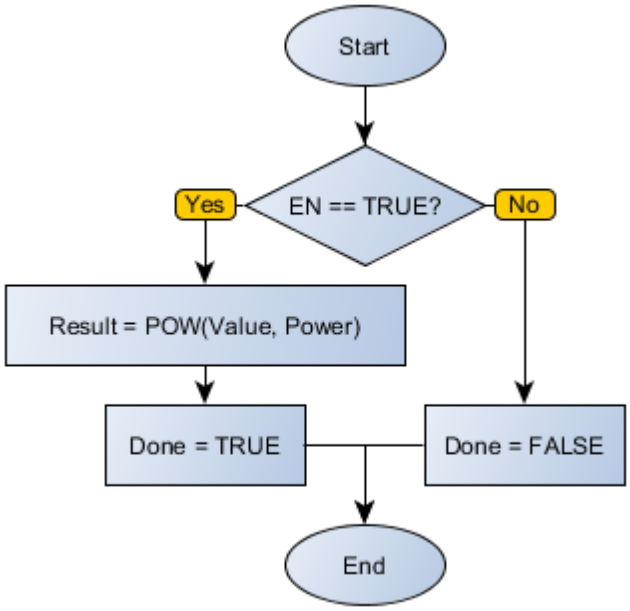
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Base of the operation
	Power	REAL	Exponent of the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	REAL	Variable that stores the result of the operation

Operation

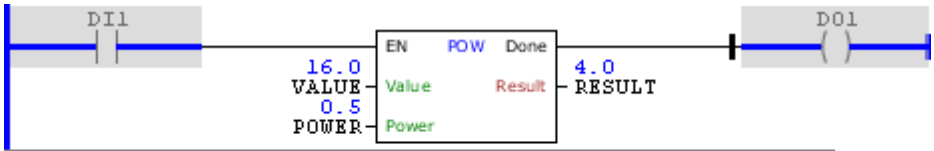
When this block has a TRUE value in EN, it sends to the Result output the value of Value raised to the exponent Power. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

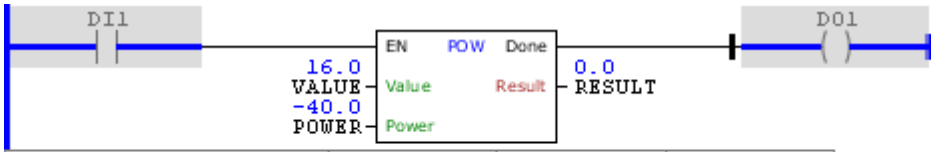


Example



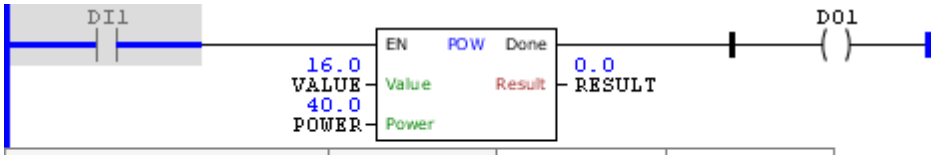
VALUE	REAL	16.0	Float
POWER	REAL	0.5	Float
RESULT	REAL	4.0	Float

The above example calculates the value of VALUE raised to the POWER variable, storing the final result in RESULT. The block ends with success and Done output is activated.



VALUE	REAL	16.0	Float
POWER	REAL	-40.0	Float
RESULT	REAL	0.0	Float

The above example calculates the value of VALUE raised to the POWER variable, storing the final result in RESULT. The block ends with success and Done output is activated.



VALUE	REAL	16.0	Float
POWER	REAL	40.0	Float
RESULT	REAL	0.0	Float

The above example calculates the value of VALUE raised to the POWER variable, storing the final result in RESULT. Since the result is higher than the maximum supported by REAL type, the block generates an error and Done output is disabled.

11.14.7.11.2.6 ROUND

Block that rounds the value of Value, storing the result in Result.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Reference variable for the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	REAL	Variable that stores the result of the operation

Operation

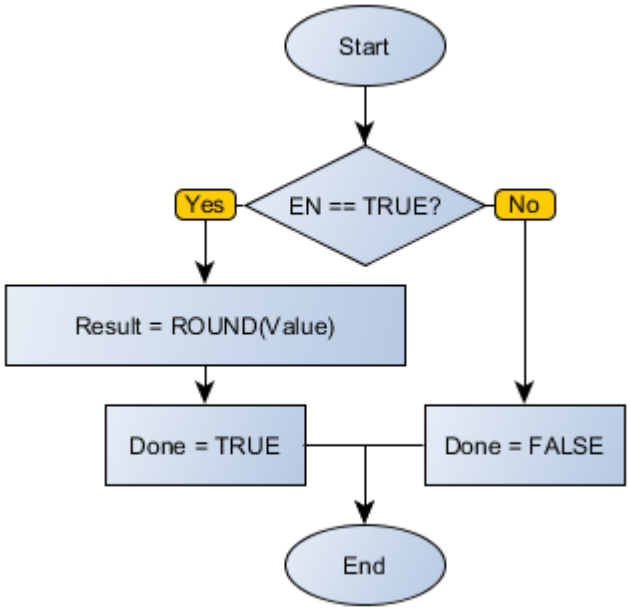
When this block has a TRUE value in EN, it sends to the Result output the rounded value of Value. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

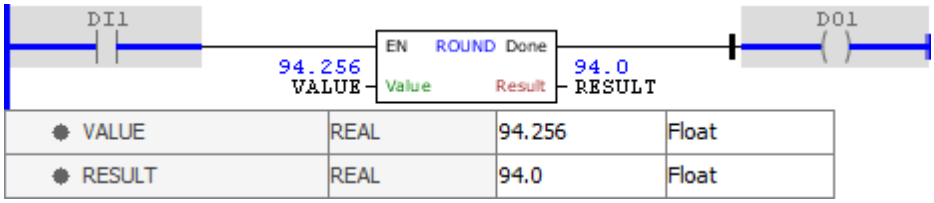
Compatibility

Device	Version
PLC300	2.10 or higher
SCA06	2.00 or higher

Block Flowchart

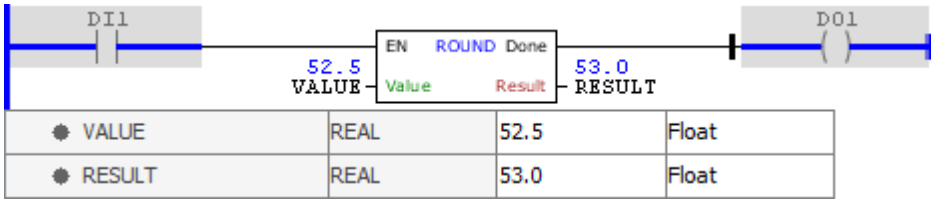


Example



VALUE	REAL	94.256	Float
RESULT	REAL	94.0	Float

The above example rounds the value of the VALUE variable, storing the final result in RESULT. Decimals less than 0.5 are discarded. The block ends with success and Done output is activated.



VALUE	REAL	52.5	Float
RESULT	REAL	53.0	Float

The above example rounds the value of the VALUE variable, storing the final result in RESULT. Decimals greater than or equal to 0.5 promote unity value immediately above. The block ends with success and Done output is activated.

11.14.7.11.2.7 SQRT

Block that calculates the square root value of Value, storing the result in Result.

Ladder Representation

Block Structure

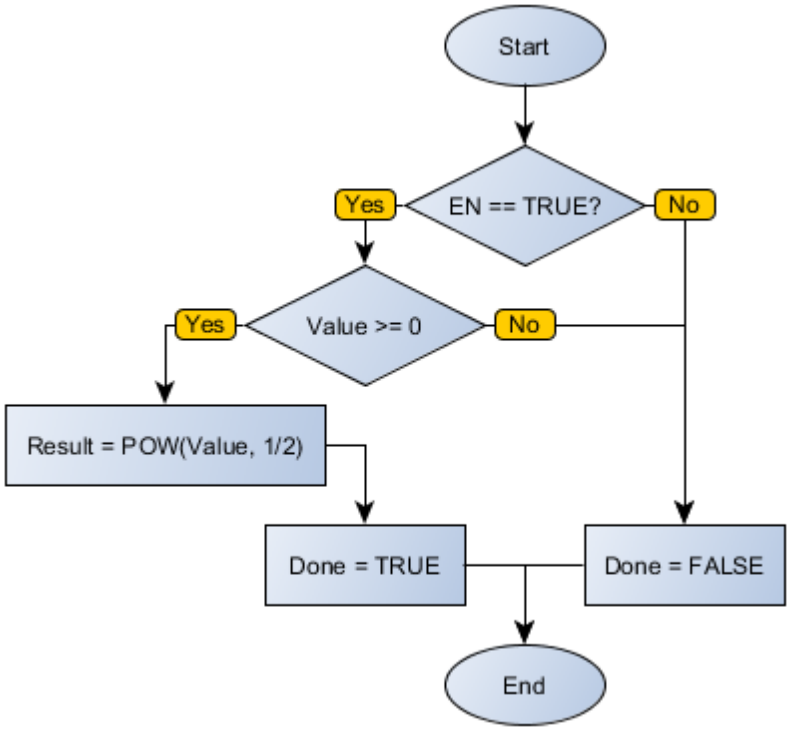
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Reference variable for the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	REAL	Variable that stores the result of the operation

Operation

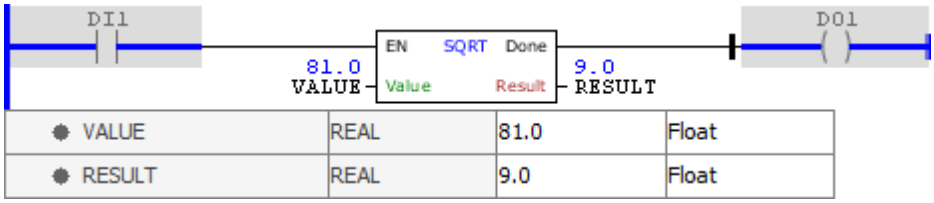
When this block has a TRUE value in EN, it sends to the Result output the square root value of Value. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

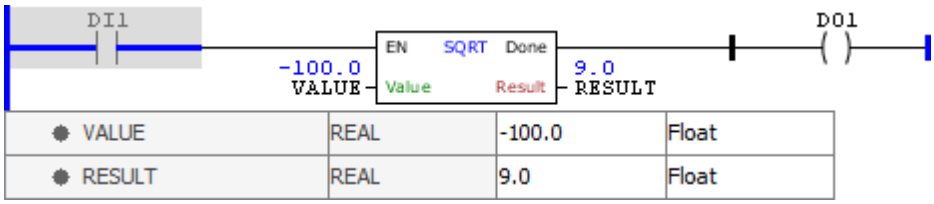


Example



VALUE	REAL	81.0	Float
RESULT	REAL	9.0	Float

The above example calculates the square root value of the VALUE variable, storing the final result in RESULT. The block ends with success and Done output is activated.



VALUE	REAL	-100.0	Float
RESULT	REAL	9.0	Float

The above example calculates the square root value of the VALUE variable, storing the final result in RESULT. The block generates a runtime error, since VALUE has negative value, and Done output is disabled.

11.14.7.11.2.8 TRUNC

Block that truncates the value of Value, storing the result in Result.

Ladder Representation

Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Reference variable for the operation
VAR_OUTPUT	Done	BOOL	End of operation
	Result	REAL	Variable that stores the result of the operation

Operation

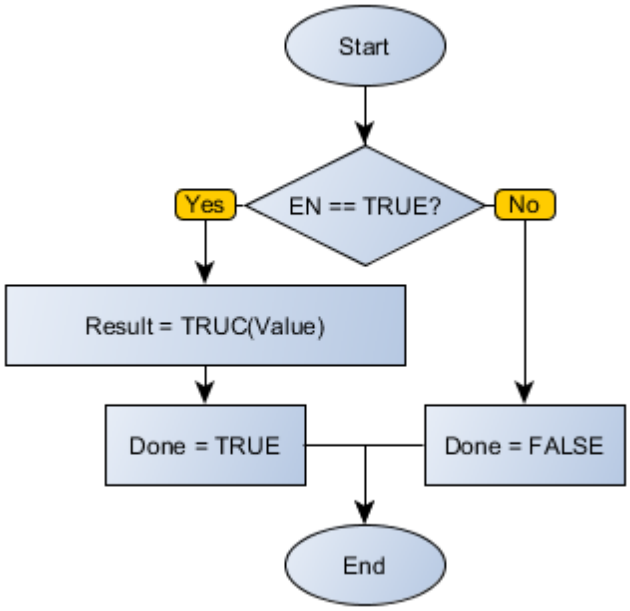
When this block has a TRUE value in EN, it sends to the Result output the truncated value of Value. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

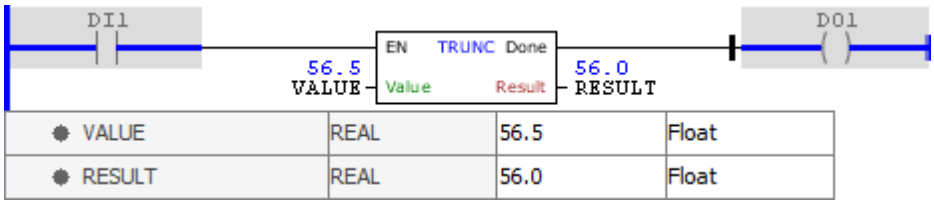
Compatibility

Device	Version
PLC300	2.10 or higher
SCA06	2.00 or higher

Block Flowchart



Example



VALUE	REAL	56.5	Float
RESULT	REAL	56.0	Float

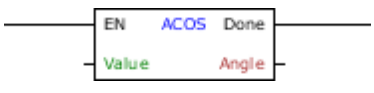
The above example truncates the value of the VALUE variable, storing the final result in RESULT. Decimals are discarded. The block ends with success and Done output is activated.

11.14.7.11.3 Math Trigonometry

11.14.7.11.3.1 ACOS

Block that calculates the arccosine of Value, storing the result in Angle.

Ladder Representation



Block Structure

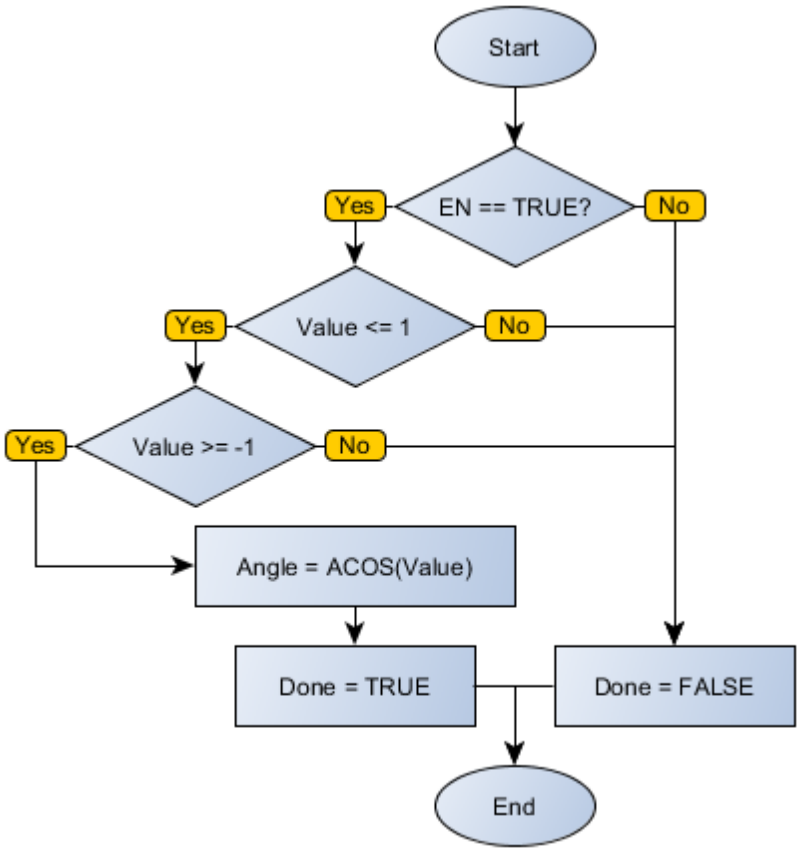
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Value of cosine
VAR_OUTPUT	Done	BOOL	End of operation
	Angle	REAL	Value of the angle w hose cosine is equal to Value (in radians)

Operation

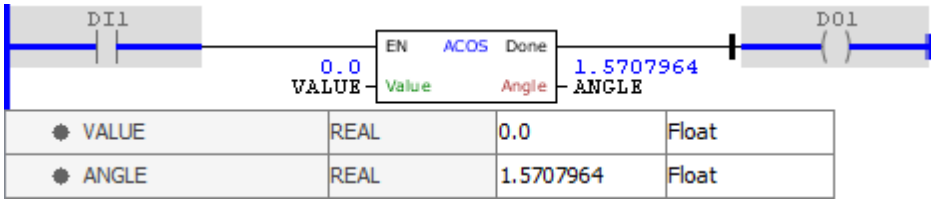
When this block has a TRUE value in EN, it sends to the Angle output the arccosine of Value. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Angle remains with its value unchanged.

When EN has FALSE value, Angle remains unchanged and Done remains in FALSE.

Block Flowchart

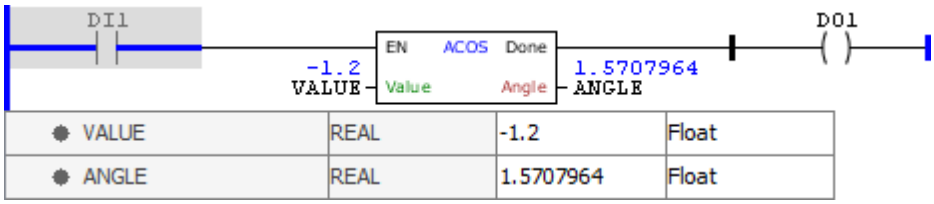


Example



VALUE	REAL	0.0	Float
ANGLE	REAL	1.5707964	Float

The above example calculates the arc, in radians, whose cosine is the VALUE variable, storing the final result in RESULT. The block ends with success and Done output is activated.



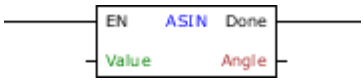
VALUE	REAL	-1.2	Float
ANGLE	REAL	1.5707964	Float

The above example calculates the arc, in radians, whose cosine is the VALUE variable, storing the final result in RESULT. The block generates a runtime error, since VALUE has value inferior to 1, and Done output is disabled.

11.14.7.11.3.2 ASIN

Block that calculates the arcsine of Value, storing the result in Angle.

Ladder Representation



Block Structure

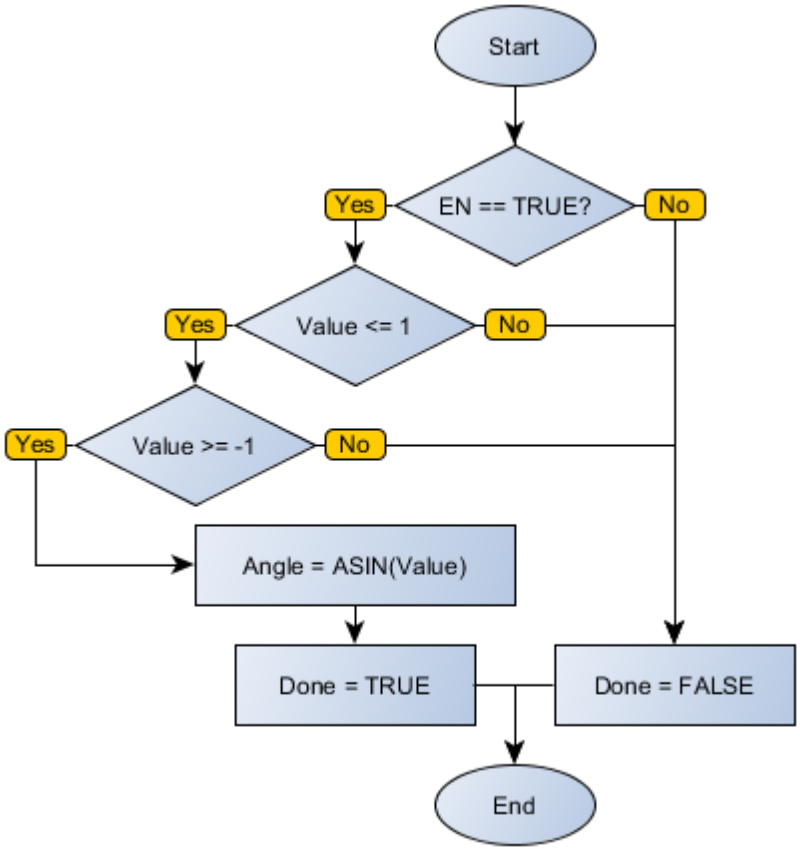
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Value of sine
VAR_OUTPUT	Done	BOOL	End of operation
	Angle	REAL	Value of the angle whose sine is equal to Value (in radians)

Operation

When this block has a TRUE value in EN, it sends to the Angle output the arcsine of Value. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Angle remains with its value unchanged.

When EN has FALSE value, Angle remains unchanged and Done remains in FALSE.

Block Flowchart



Example

VALUE	REAL	-1.0	Float
ANGLE	REAL	-1.5707964	Float

The above example calculates the arc, in radians, whose sine is the VALUE variable, storing the final result in RESULT. The block ends with success and Done output is activated.

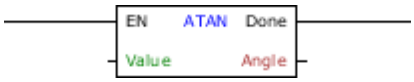
VALUE	REAL	2.0	Float
ANGLE	REAL	-1.5707964	Float

The above example calculates the arc, in radians, whose sine is the VALUE variable, storing the final result in RESULT. The block generates a runtime error, since VALUE has value superior to 1, and Done output is disabled.

11.14.7.11.3.3 ATAN

Block that calculates the arctangent of Value, storing the result in Angle.

Ladder Representation



Block Structure

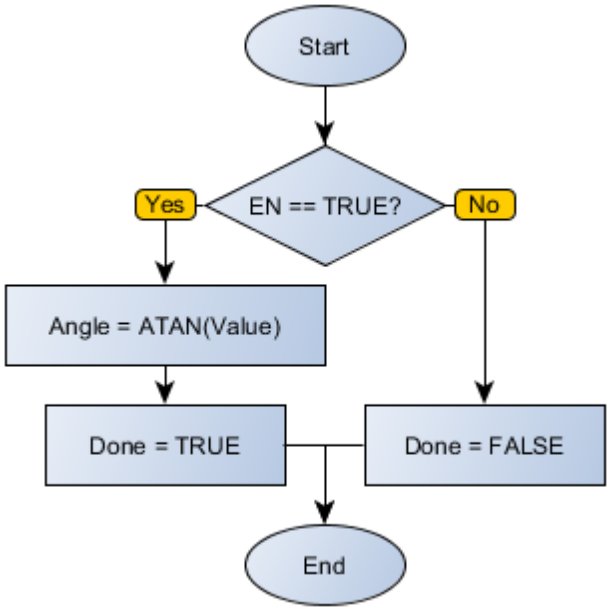
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	REAL	Value of tangent
VAR_OUTPUT	Done	BOOL	End of operation
	Angle	REAL	Value of the angle whose tangent is equal to Value (in radians)

Operation

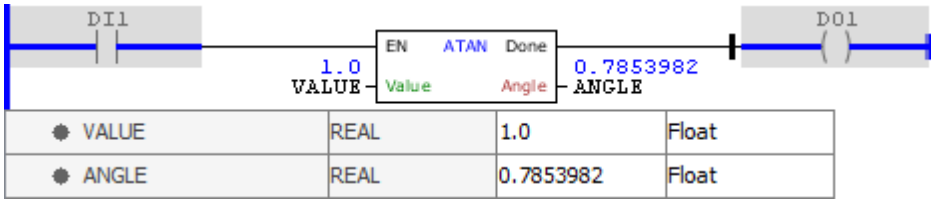
When this block has a TRUE value in EN, it sends to the Angle output the arctangent of Value. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Angle remains with its value unchanged.

When EN has FALSE value, Angle remains unchanged and Done remains in FALSE.

Block Flowchart

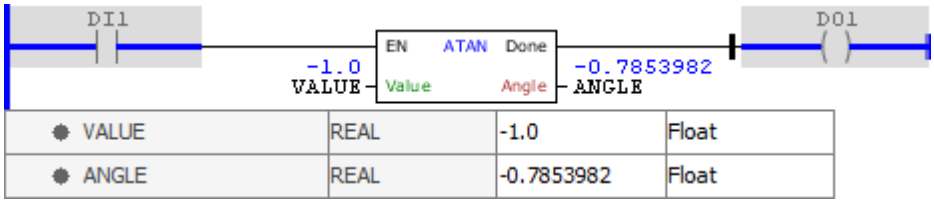


Example



VALUE	REAL	1.0	Float
ANGLE	REAL	0.7853982	Float

The above example calculates the arc, in radians, whose tangent is the VALUE variable, storing the final result in RESULT. The arc, for positive values, is always in the first quadrant. The block ends with success and Done output is activated.



VALUE	REAL	-1.0	Float
ANGLE	REAL	-0.7853982	Float

The above example calculates the arc, in radians, whose tangent is the VALUE variable, storing the final result in RESULT. The arc, for negative values, is always in the fourth quadrant. The block ends with success and Done output is activated.

11.14.7.11.3.4 ATAN2

Block that calculates the arctangent of Y/X, storing the result in Angle.

Ladder Representation

Block Structure

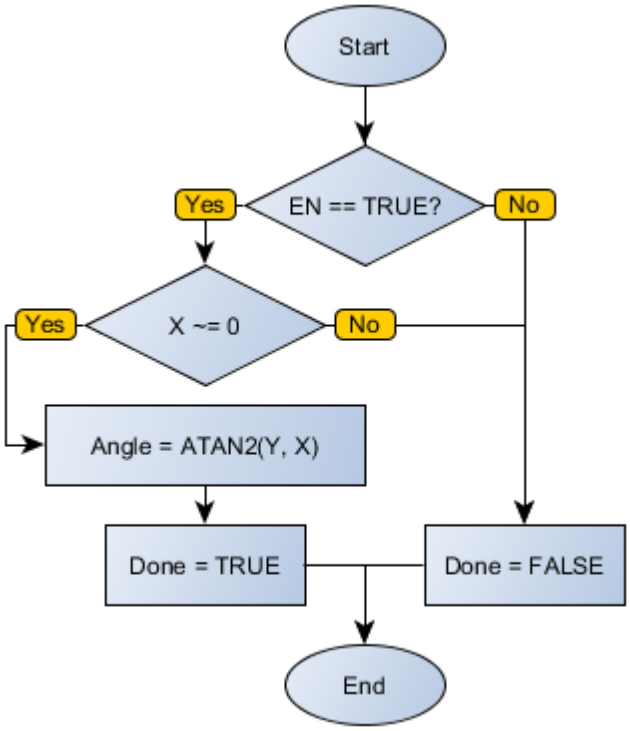
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	X	REAL	Parameter X of the function
	Y	REAL	Parameter Y of the function
VAR_OUTPUT	Done	BOOL	End of operation
	Angle	REAL	Value of the angle whose tangent is equal to (Y/X) (in radians)

Operation

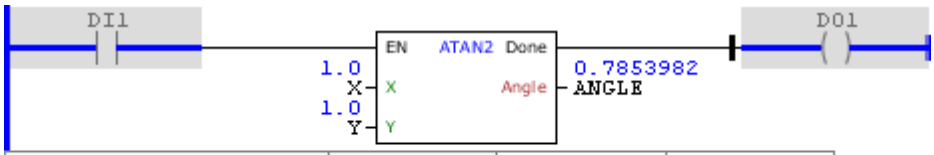
When this block has a TRUE value in EN, it sends to the Angle output the arctangent of Y/X. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Angle remains with its value unchanged.

When EN has FALSE value, Angle remains unchanged and Done remains in FALSE.

Block Flowchart



Example



X	REAL	1.0	Float
Y	REAL	1.0	Float
ANGLE	REAL	0.7853982	Float

The above example calculates the arc, in radians, whose tangent is the Y/X variable, storing the final result in RESULT. The arc, for positive values of X and Y, is always in the first quadrant. The block ends with success and Done output is activated.

X	REAL	-1.0	Float
Y	REAL	1.0	Float
ANGLE	REAL	2.3561945	Float

The above example calculates the arc, in radians, whose tangent is the Y/X variable, storing the final

result in RESULT. The arc, for negative values of X and positive values of Y, is always in the second quadrant. The block ends with success and Done output is activated.

X	REAL	-1.0	Float
Y	REAL	-1.0	Float
ANGLE	REAL	-2.3561945	Float

The above example calculates the arc, in radians, whose tangent is the Y/X variable, storing the final result in RESULT. The arc, for negative values of X and Y, is always in the third quadrant. The block ends with success and Done output is activated.

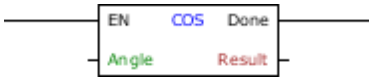
X	REAL	1.0	Float
Y	REAL	-1.0	Float
ANGLE	REAL	-0.7853982	Float

The above example calculates the arc, in radians, whose tangent is the Y/X variable, storing the final result in RESULT. The arc, for positive values of X and negative values of Y, is always in the fourth quadrant. The block ends with success and Done output is activated.

11.14.7.11.3.5 COS

Block that calculates the cosine of Angle, storing the result in Result.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Angle	REAL	Angle (in radians)
VAR_OUTPUT	Done	BOOL	End of operation
	Result	REAL	Variable that stores the result of the operation

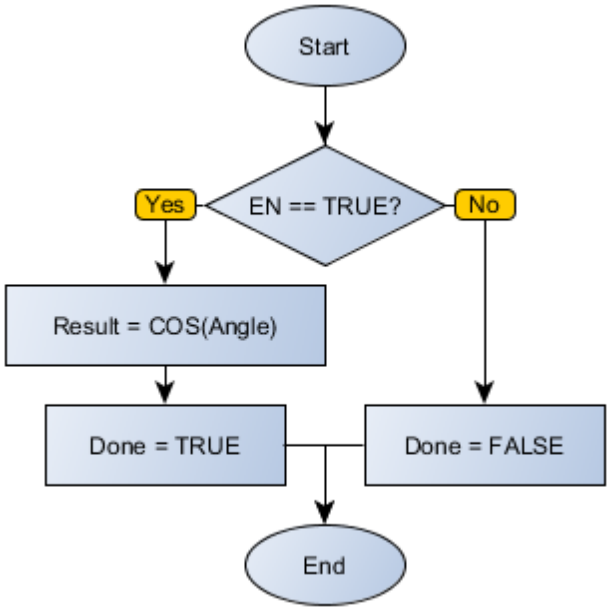
Operation

When this block has a TRUE value in EN, it sends to the Result output the cosine of Angle. If no

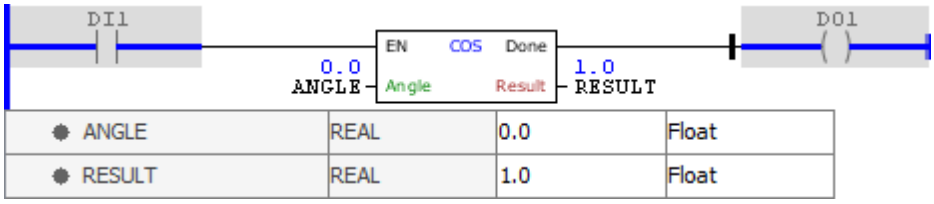
errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

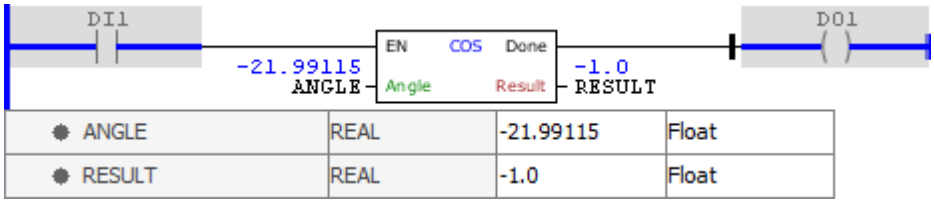


Example



ANGLE	REAL	0.0	Float
RESULT	REAL	1.0	Float

The above example calculates the cosine of the VALUE variable, interpreted in radians, storing the final result in RESULT. The block ends with success and Done output is activated.



ANGLE	REAL	-21.99115	Float
RESULT	REAL	-1.0	Float

The above example calculates the cosine of the VALUE variable, interpreted in radians, storing the final result in RESULT. The block ends with success and Done output is activated. Notice that the block accepts negative input values and greater than one turn.

11.14.7.11.3.6 SIN

Block that calculates the sine of Angle, storing the result in Result.

Ladder Representation

Block Structure

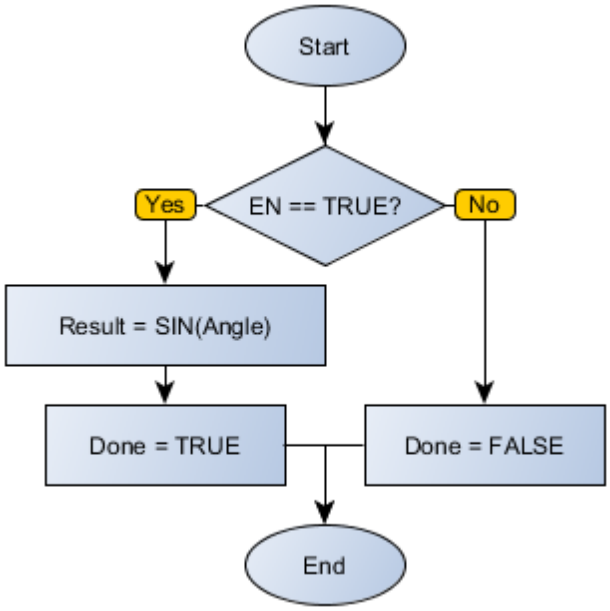
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Angle	REAL	Angle (in radians)
VAR_OUTPUT	Done	BOOL	End of operation
	Result	REAL	Variable that stores the result of the operation

Operation

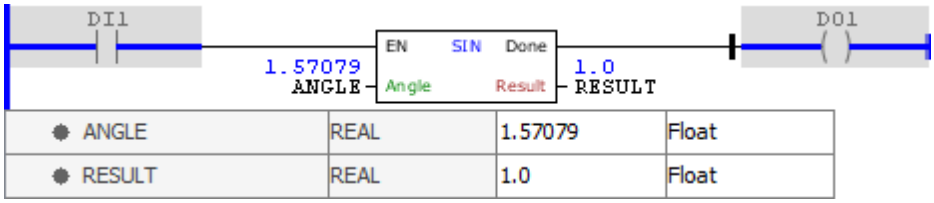
When this block has a TRUE value in EN, it sends to the Result output the sine of Angle. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

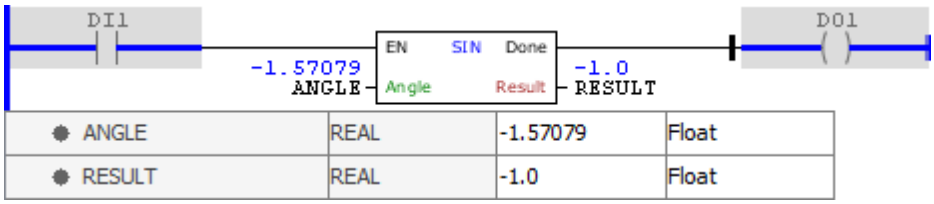


Example



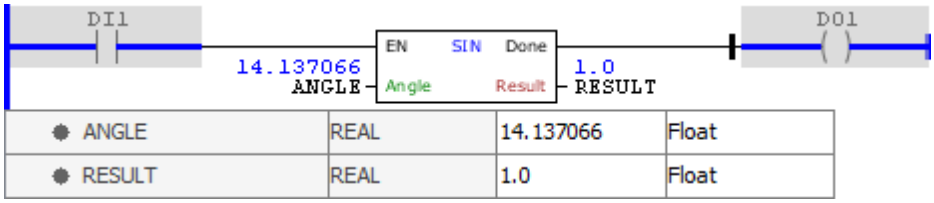
ANGLE	REAL	1.57079	Float
RESULT	REAL	1.0	Float

The above example calculates the sine of the VALUE variable, interpreted in radians, storing the final result in RESULT. The block ends with success and Done output is activated.



ANGLE	REAL	-1.57079	Float
RESULT	REAL	-1.0	Float

The above example calculates the sine of the VALUE variable, interpreted in radians, storing the final result in RESULT. The block ends with success and Done output is activated. Notice that the block accepts negative input values.



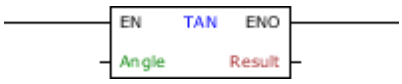
ANGLE	REAL	14.137066	Float
RESULT	REAL	1.0	Float

The above example calculates the sine of the VALUE variable, interpreted in radians, storing the final result in RESULT. The block ends with success and Done output is activated. Notice that the block accepts values greater than one full turn.

11.14.7.11.3.7 TAN

Block that calculates the tangent of Angle, storing the result in Result.

Ladder Representation



Block Structure

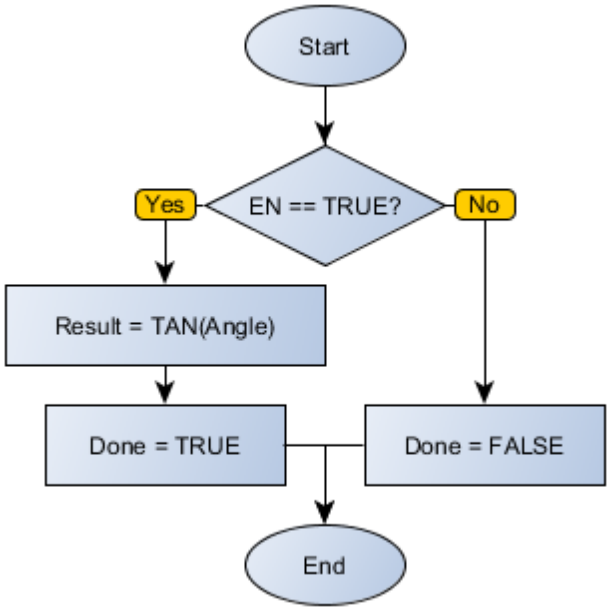
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Angle	REAL	Angle (in radians)
VAR_OUTPUT	Done	BOOL	End of operation
	Result	REAL	Variable that stores the result of the operation

Operation

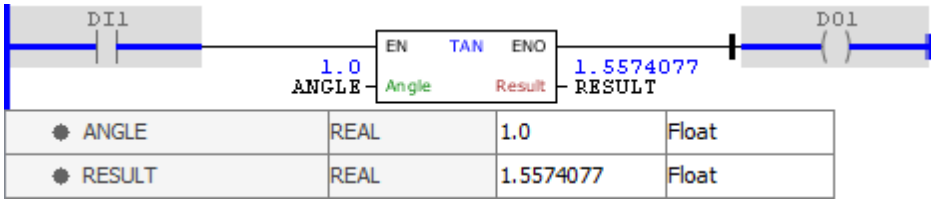
When this block has a TRUE value in EN, it sends to the Result output the tangent of Angle. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

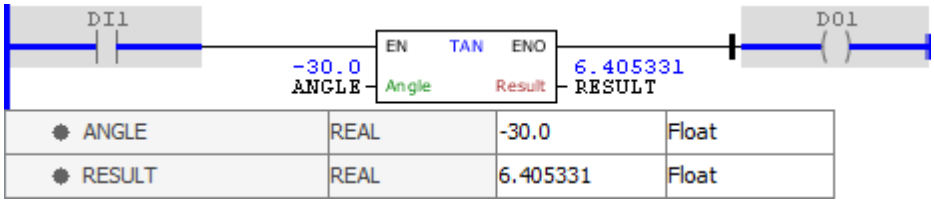


Example



ANGLE	REAL	1.0	Float
RESULT	REAL	1.5574077	Float

The above example calculates the tangent of the VALUE variable, interpreted in radians, storing the final result in RESULT. The block ends with success and Done output is activated.



ANGLE	REAL	-30.0	Float
RESULT	REAL	6.405331	Float

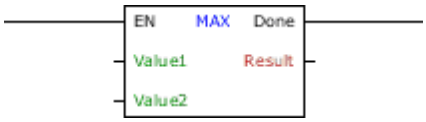
The above example calculates the tangent of the VALUE variable, interpreted in radians, storing the final result in RESULT. The block ends with success and Done output is activated. Notice that the block accepts negative input values and greater than one turn.

11.14.7.11.4 Math Util

11.14.7.11.4.1 MAX

Block that compares the values of Value1 and Value2 and stores the highest of them in Result.

Ladder Representation



Block Structure

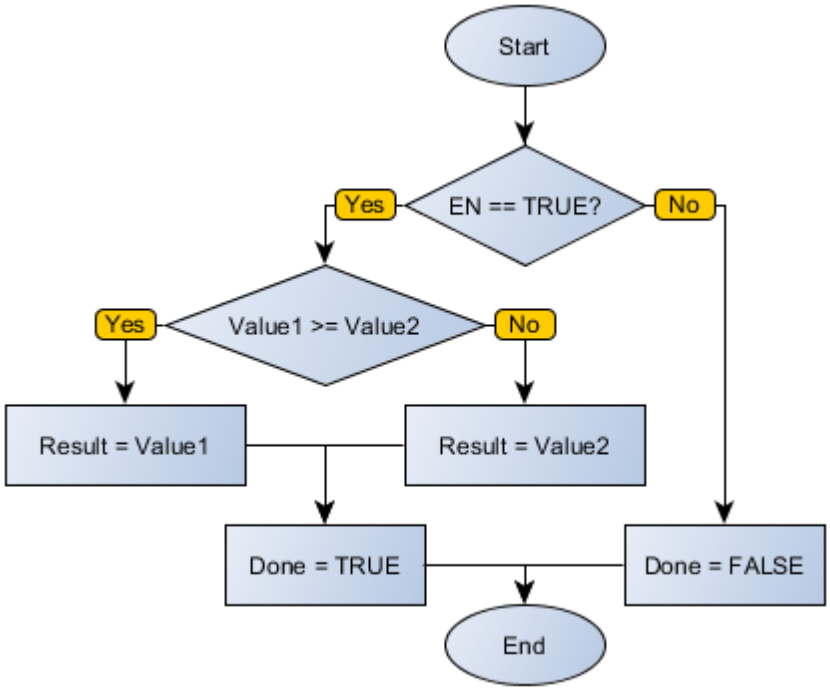
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	First value of comparison
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Second value of comparison
VAR_OUTPUT	Done	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Highest of the values compared

Operation

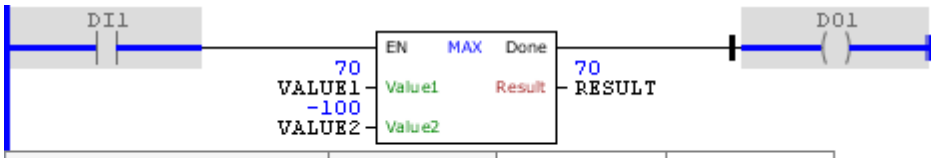
When this block has a TRUE value in EN, it sends to the Result output the highest value in the comparison between Value1 and Value2. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

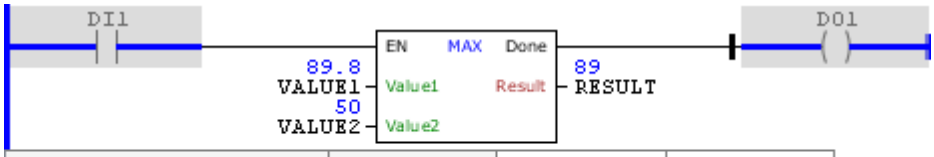


Example



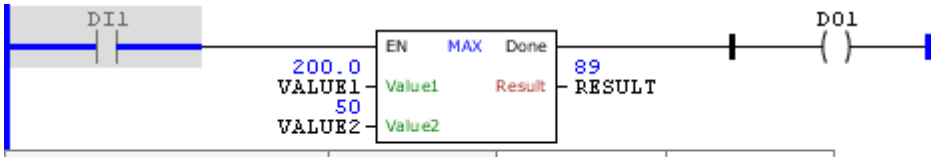
VALUE1	SINT	70	Decimal
VALUE2	SINT	-100	Decimal
RESULT	SINT	70	Decimal

The above example calculates the maximum value between VALUE 1 and VALUE2 variables, storing the final result in RESULT. The block ends with success and Done output is activated.



VALUE1	REAL	89.8	Float
VALUE2	SINT	50	Decimal
RESULT	SINT	89	Decimal

The above example calculates the maximum value between VALUE 1 and VALUE2 variables, storing the final result in RESULT. The block ends with success and Done output is activated. Notice that the types of the input variables can be different without causing execution problems.



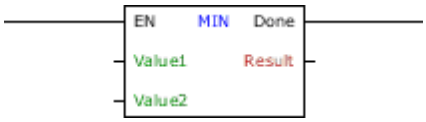
VALUE1	REAL	200.0	Float
VALUE2	SINT	50	Decimal
RESULT	SINT	89	Decimal

The above example calculates the maximum value between VALUE 1 and VALUE2 variables, storing the final result in RESULT. Since the result is higher than the maximum supported by SINT type, the block generates an error and Done output is disabled.

11.14.7.11.4.2 MIN

Block that compares the values of Value1 and Value2 and stores the lowest of them in Result.

Ladder Representation



Block Structure

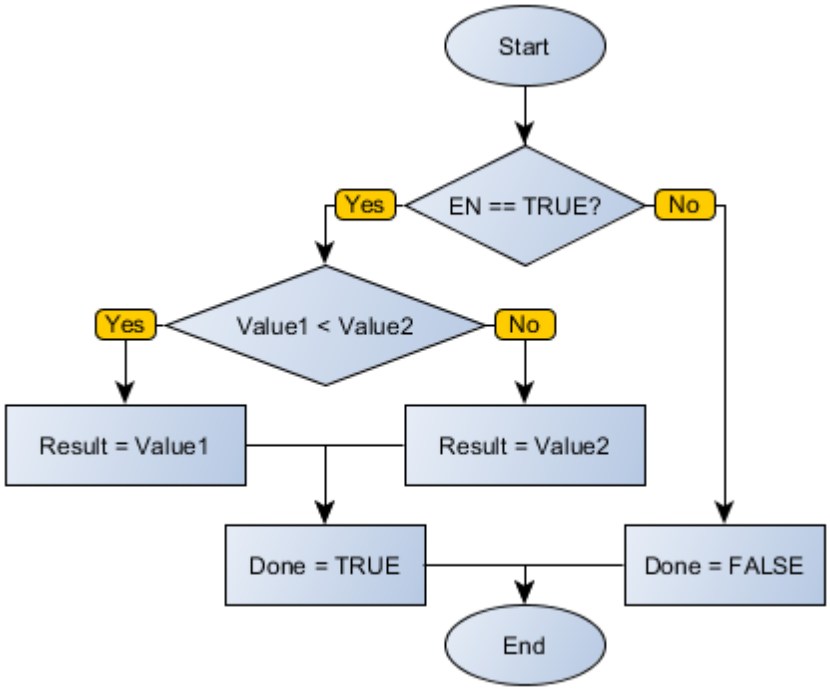
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value1	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	First value of comparison
	Value2	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Second value of comparison
VAR_OUTPUT	Done	BOOL	End of operation
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Low est of the values compared

Operation

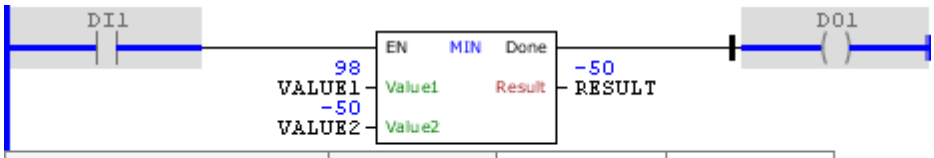
When this block has a TRUE value in EN, it sends to the Result output the lowest value in the comparison between Value1 and Value2. If no errors, the Done variable is set. If there is any error in the operation, Done is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Done remains in FALSE.

Block Flowchart

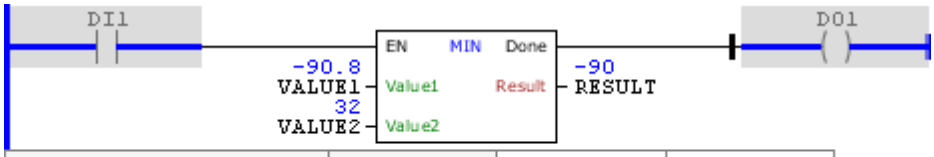


Example



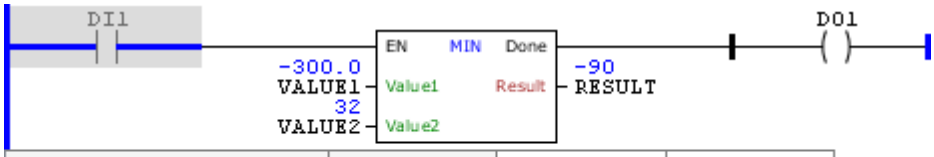
VALUE1	SINT	98	Decimal
VALUE2	SINT	-50	Decimal
RESULT	SINT	-50	Decimal

The above example calculates the minimum value between VALUE 1 and VALUE2 variables, storing the final result in RESULT. The block ends with success and Done output is activated.



VALUE1	REAL	-90.8	Float
VALUE2	SINT	32	Decimal
RESULT	SINT	-90	Decimal

The above example calculates the minimum value between VALUE 1 and VALUE2 variables, storing the final result in RESULT. The block ends with success and Done output is activated. Notice that the types of the input variables can be different without causing execution problems.



VALUE1	REAL	-300.0	Float
VALUE2	SINT	32	Decimal
RESULT	SINT	-90	Decimal

The above example calculates the minimum value between VALUE 1 and VALUE2 variables, storing the final result in RESULT. Since the result is lower than the minimum supported by SINT type, the block generates an error and Done output is disabled.

11.14.7.11.4.3 SAT

Block that performs a routine for saturation of the value found in Value in accordance with the limits for Minimum and Maximum, storing the result in Result.

Ladder Representation

Block Structure

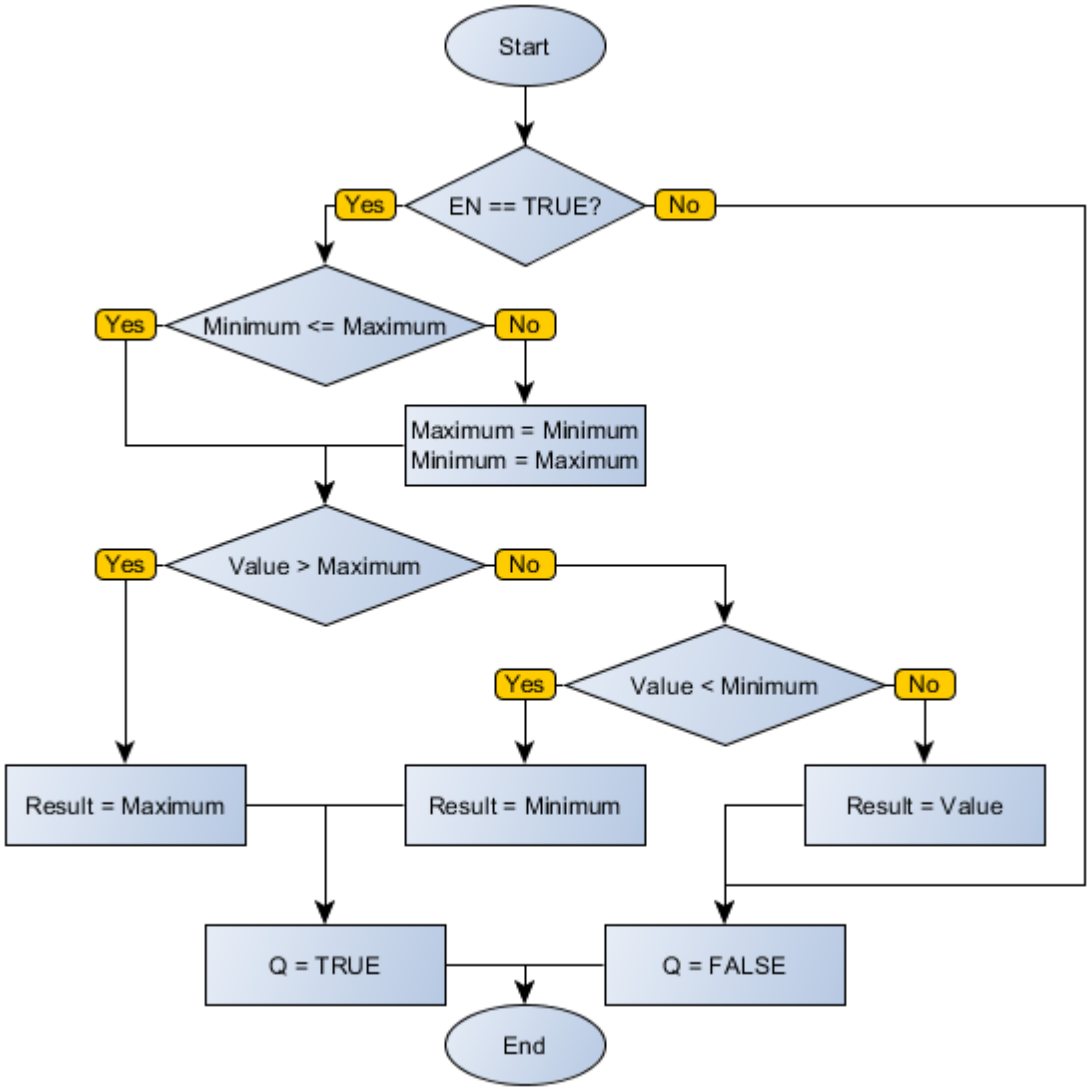
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	Value	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Reference value
	Minimum	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Inferior saturation value
	Maximum	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Superior saturation value
VAR_OUTPUT	Q	BOOL	Indicator that there w as saturation in the process
	Result	BYTE USINT SINT WORD UINT INT DWORD UDINT DINT REAL	Result of operation

Operation

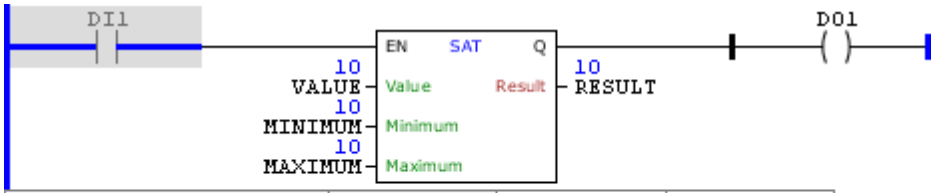
When this block has a TRUE value in EN, it performs a comparison between Value and Minimum and Maximum. If Value is in the range between Minimum and Maximum, Result receives the value of Value and Q remains FALSE. If Value is higher than Maximum, Result receives Maximum and Q receives TRUE. If Value is lower than Minimum, Result receives Minimum and Q receives TRUE. If there is any error in the operation, Q is not set, staying in FALSE status, while Result remains with its value unchanged.

When EN has FALSE value, Result remains unchanged and Q remains in FALSE.

Block Flowchart

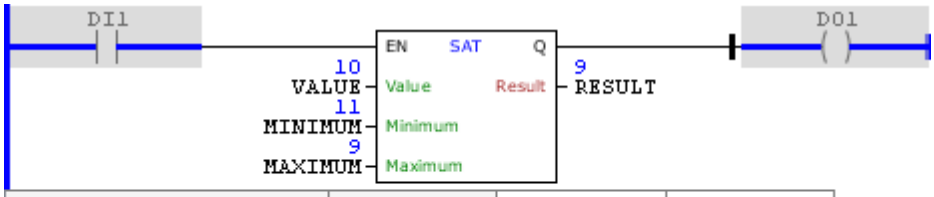


Example



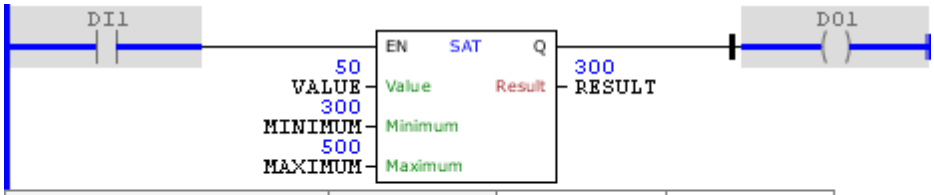
VALUE	INT	10	Decimal
MINIMUM	INT	10	Decimal
MAXIMUM	INT	10	Decimal
RESULT	INT	10	Decimal

The above example passes the VALUE value to RESULT, since it is not lower than MINIMUM or higher than MAXIMUM. The block ends successfully and the Q output is disabled, since there was no saturation.



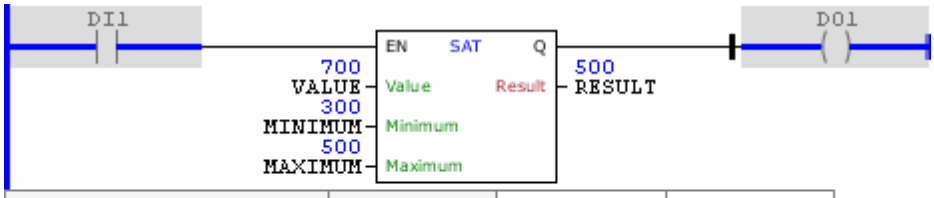
VALUE	INT	10	Decimal
MINIMUM	INT	11	Decimal
MAXIMUM	INT	9	Decimal
RESULT	INT	9	Decimal

The above example passes the MAXIMUM to RESULT, since VALUE is higher than MAXIMUM. The block ends successfully and the Q output is activated, since there was saturation.



VALUE	INT	50	Decimal
MINIMUM	INT	300	Decimal
MAXIMUM	INT	500	Decimal
RESULT	INT	300	Decimal

The above example passes the MINIMUM to RESULT, since VALUE is lower than MINIMUM. The block ends successfully and the Q output is activated, since there was saturation.



VALUE	INT	700	Decimal
MINIMUM	INT	300	Decimal
MAXIMUM	INT	500	Decimal
RESULT	INT	500	Decimal

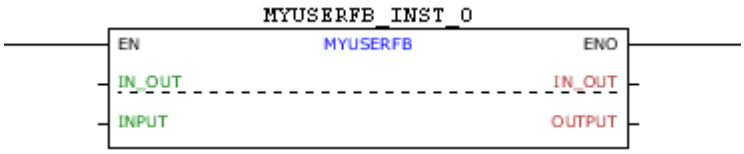
The above example passes the MAXIMUM value to RESULT, since VALUE is higher than MAXIMUM. The block ends successfully and the Q output is activated, since there was saturation.

11.14.7.12 Module

11.14.7.12.1 USERFB

Block that performs a subroutine programmed by the user.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	INPUT	According to user programming	Block inputs
VAR_OUTPUT	ENO	BOOL	End of operation
	OUTPUT	According to user programming	Block outputs
VAR_IN_OUT	IN_OUT	According to user programming	Block inputs/outputs
VAR	MYUSERFB_INST_0	MYUSERFB	Instance of access to block structure

Operation

When this block has a TRUE value in EN, it updates the values of internal fields with the input variables, performs the Ladder routine programmed by the user and updates the values of the outputs after completing routine.

When EN has FALSE value, outputs remain unchanged.

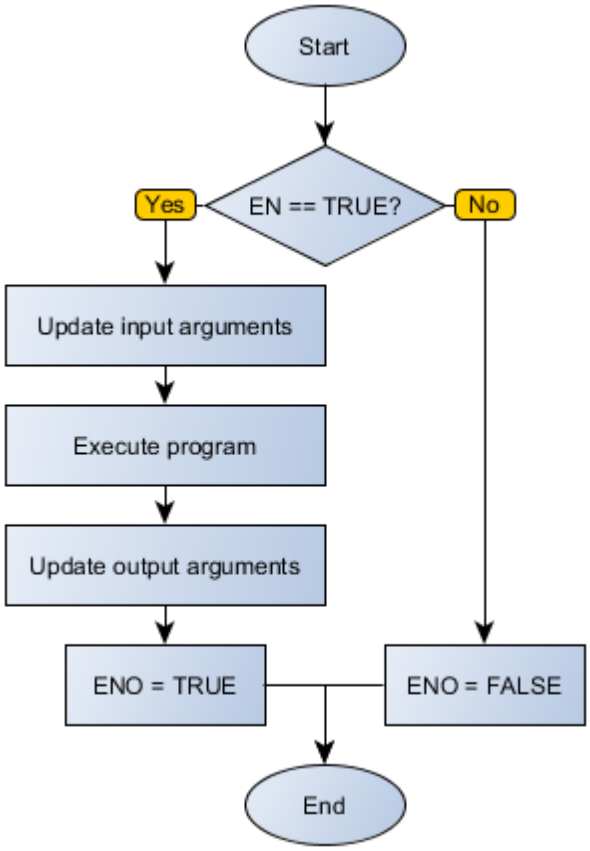
The ENO value forwards to the next Ladder block the EN value after the operation is completed.

NOTE!
Refer to section Working with USERFBs for further information.

Compatibility

Device	Version
PLC300	1.50 or higher
SCA06	2.00 or higher

Block Flowchart

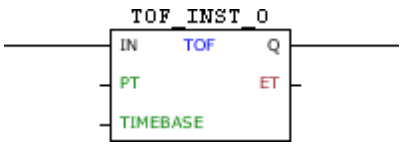


11.14.7.13 Timer

11.14.7.13.1 TOF

Timer block that, when energized, disables the output after a delay set by PT.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	IN	BOOL	Block enabling
	PT	WORD UINT DWORD UDINT	Delay of output deactivating
	TIMEBASE	WORD	Time base for PT and ET
VAR_OUTPUT	Q	BOOL	Block output
	ET	WORD UINT DWORD UDINT	Counter elapsed time
VAR	TOF_INST_0	TOF	Instance of access to block structure

NOTE!
In CFW300, the PT e ET fields can only be WORD ou UINT type.

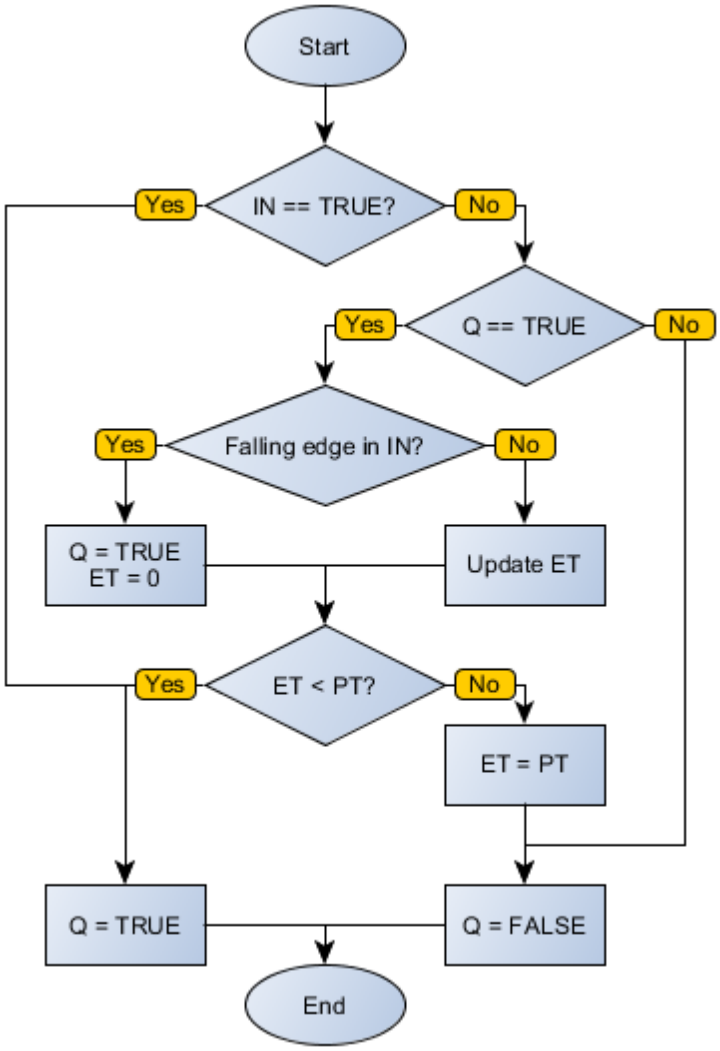
Operation

While the IN input is TRUE, the Q output is also TRUE and ET also receives the value zero.
On the negative transition edge in IN, counting is triggered and ET is incremented according to TIMEBASE. When ET equals PT, the Q output goes to state FALSE until IN revolutions to FALSE.

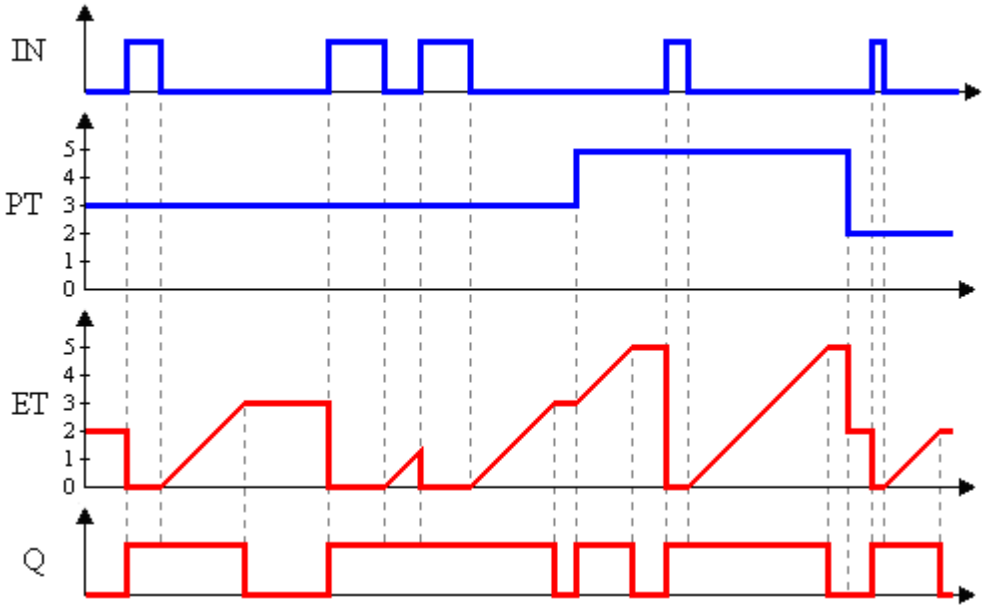
Compatibility

Device	Version
PLC300	1.50 or higher
SCA06	2.00 or higher

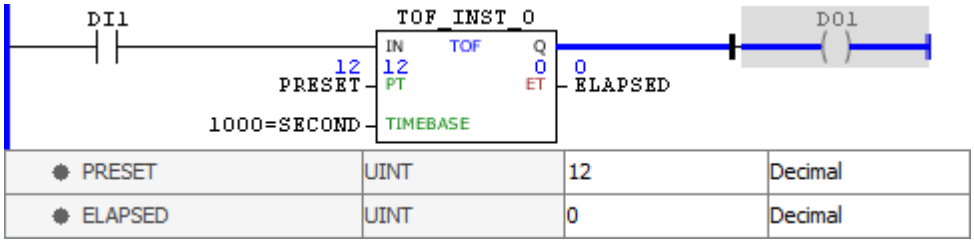
Block Flowchart



Operation Diagram



Example



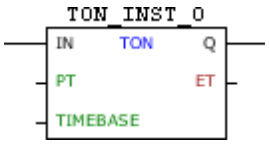
PRESET	UINT	12	Decimal
ELAPSED	UINT	0	Decimal

The above example disables the DO1 output to identify a low level in DI1 for 12 seconds, remaining disabled until DI1 again be TRUE.

11.14.7.13.2 TON

Timer block that, when energized, enables the output after a delay set by PT.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	IN	BOOL	Block enabling
	PT	WORD UINT DWORD UDINT	Delay of output drive
	TIMEBASE	WORD	Time base for PT and ET
VAR_OUTPUT	Q	BOOL	Block output
	ET	WORD UINT DWORD UDINT	Counter elapsed time
VAR	TON_INST_0	TON	Instance of access to block structure

NOTE!
In CFW300, the PT e ET fields can only be WORD ou UINT type.

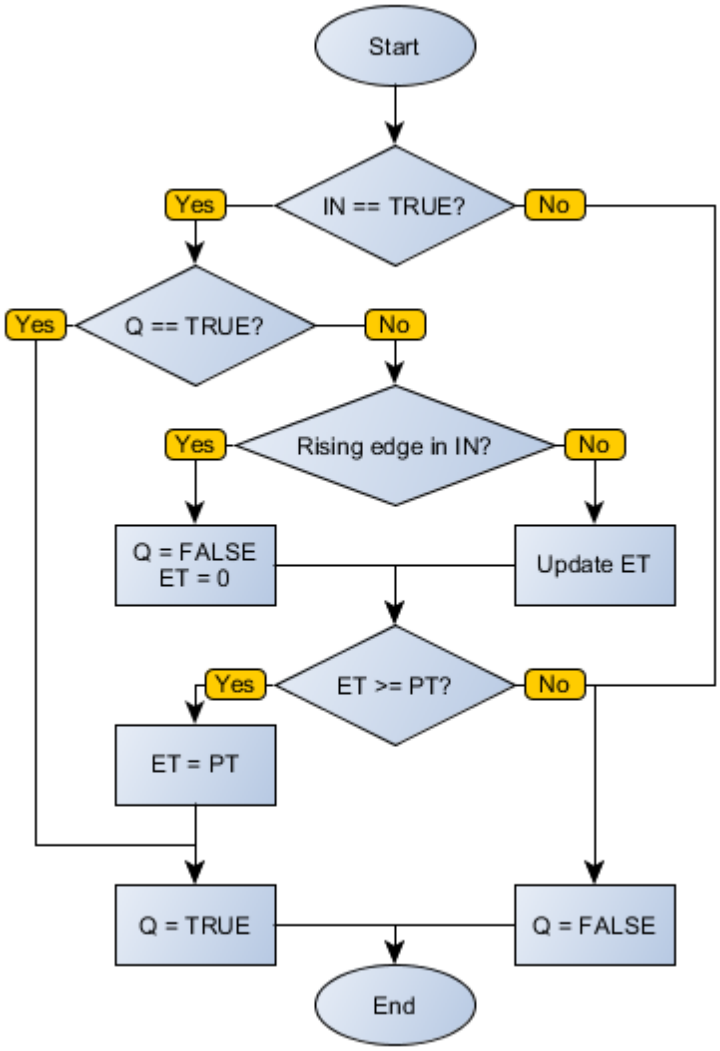
Operation

While the IN input is FALSE, the Q output is FALSE and ET also receives the value zero.
On the edge positive transition in IN, counting is triggered and ET is incremented according to TIMEBASE. When ET equals PT, the Q output goes to state TRUE until IN revolutions to FALSE.

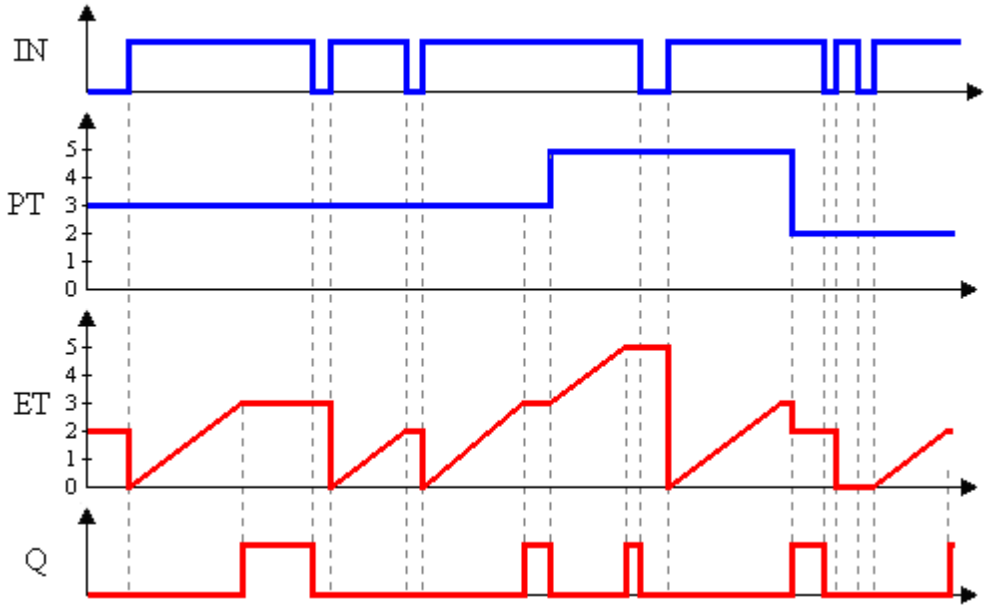
Compatibility

Device	Version
PLC300	1.50 or higher
SCA06	2.00 or higher

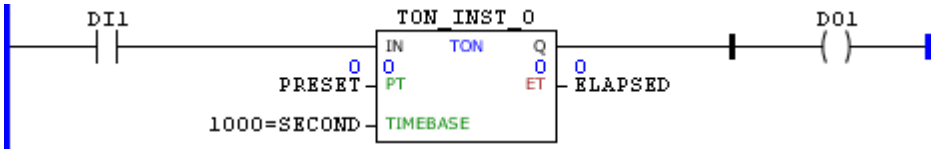
Block Flowchart



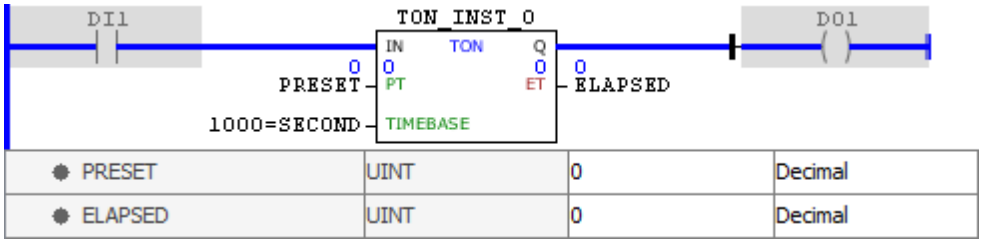
Operation Diagram



Example

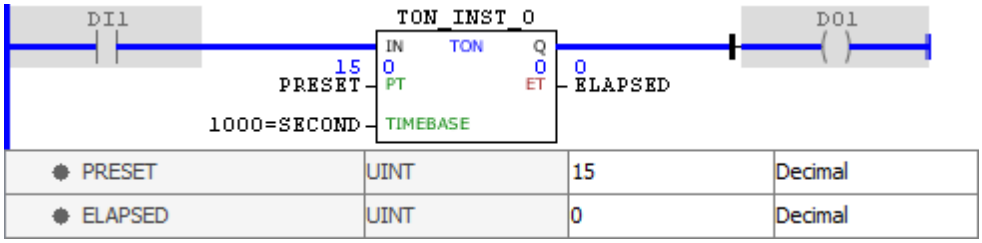


The above example shows the initial conditions of the block and of the routine variables.



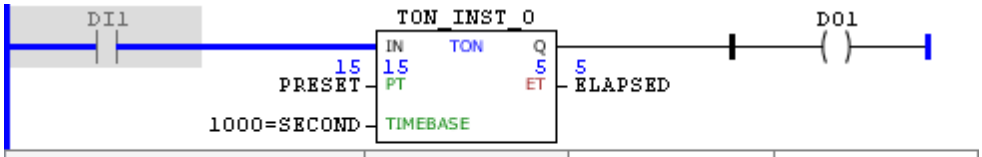
PRESET	UINT	0	Decimal
ELAPSED	UINT	0	Decimal

When activated the IN input, counting is triggered. Since ET equals PT, the Q output is enabled.



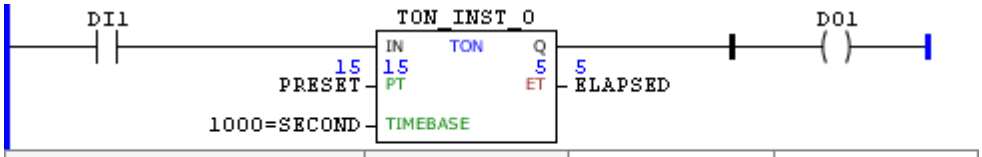
PRESET	UINT	15	Decimal
ELAPSED	UINT	0	Decimal

Note that a change in PRESET variable is not forwarded to the PT field while the IN entry remains enabled.



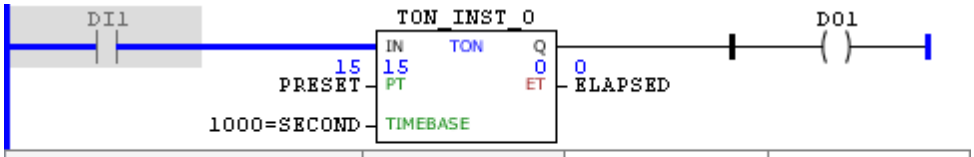
PRESET	UINT	15	Decimal
ELAPSED	UINT	5	Decimal

Disabling the IN input, the value of PT is updated and the Q output is disabled. When activating it again, counting is triggered.



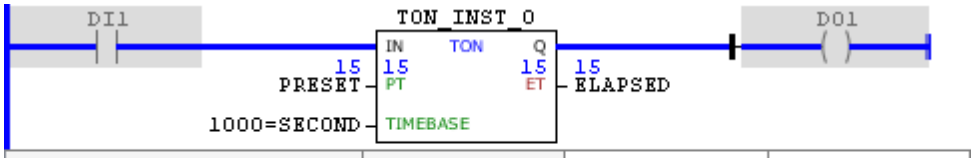
PRESET	UINT	15	Decimal
ELAPSED	UINT	5	Decimal

Disabling the IN input, the value of ET remains saved.



PRESET	UINT	15	Decimal
ELAPSED	UINT	0	Decimal

Enabling the IN input, the value of ET is reset and counting is triggered.



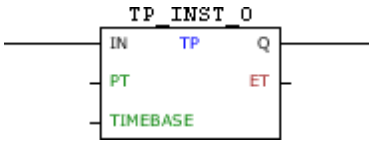
PRESET	UINT	15	Decimal
ELAPSED	UINT	15	Decimal

When ET reaches the value PT, the Q is output enabled and remains so while IN is at TRUE level.

11.14.7.13.3 TP

Timer block that, when identifies it is energized, enables the output after a delay set by PT.

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	IN	BOOL	Block enabling
	PT	WORD UINT DWORD UDINT	Time w hile the output is enabled
	TIMEBASE	WORD	Time base for PT and ET
VAR_OUTPUT	Q	BOOL	Block output
	ET	WORD UINT DWORD UDINT	Counter elapsed time
VAR	TP_INST_0	TP	Instance of access to block structure

NOTE!
In CFW300, the PT e ET fields can only be WORD ou UINT type.

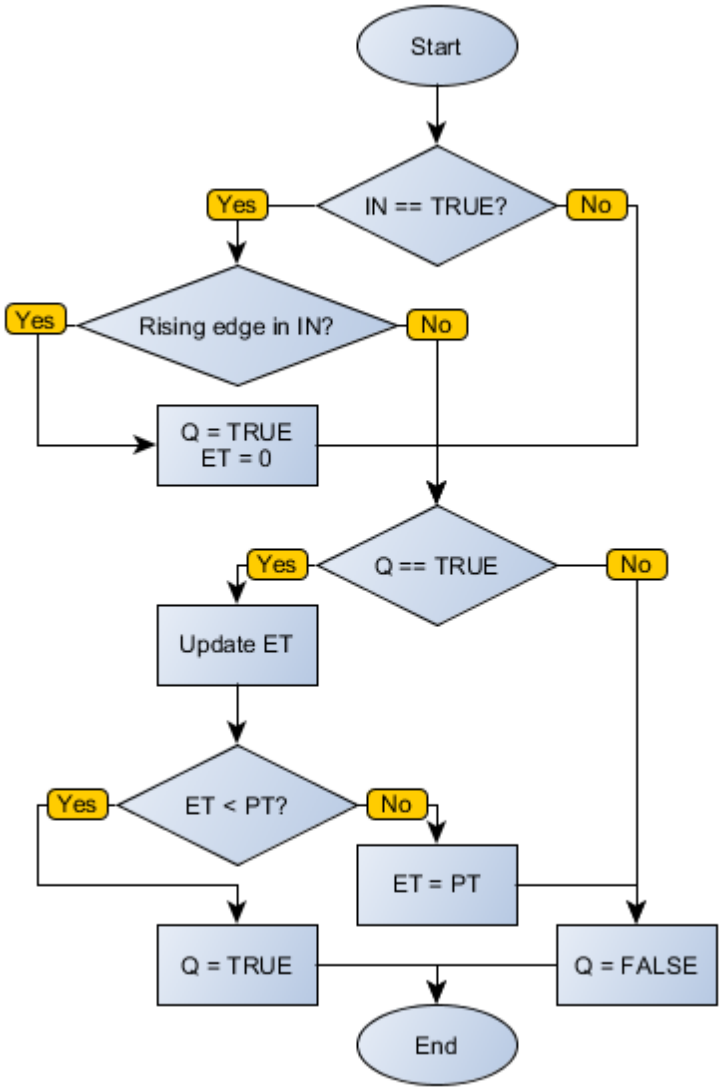
Operation

On the edge positive transition in IN, Q receives TRUE value, counting is triggered and ET is incremented according to TIMEBASE. When ET equals PT, the Q output goes to state FALSE until IN revolutions to FALSE. At that moment, if IN is at TRUE level, nothing happens. On the edge positive transition in IN, ET is automatically reset.

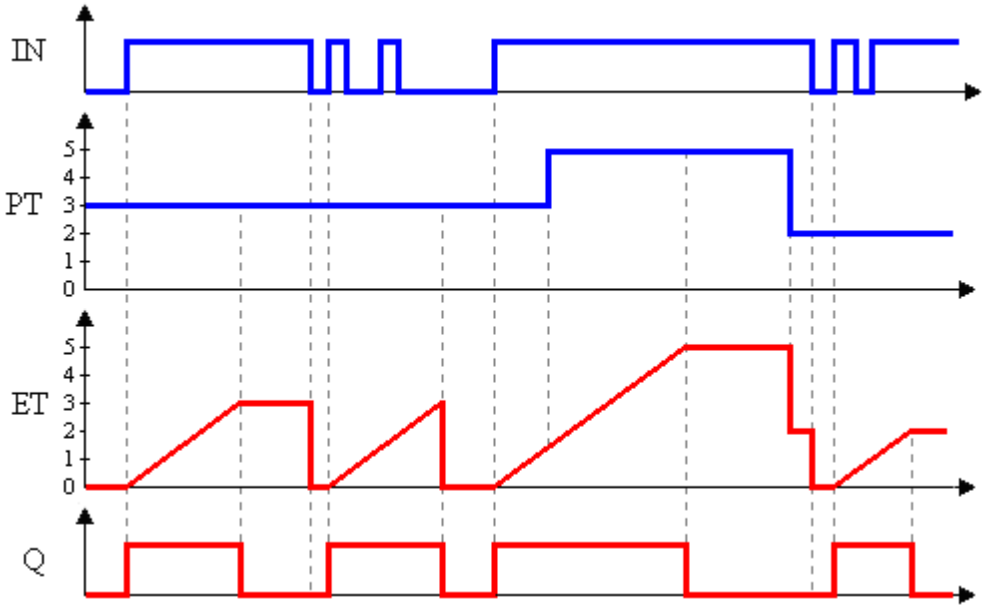
Compatibility

Device	Version
PLC300	1.50 or higher
SCA06	2.00 or higher

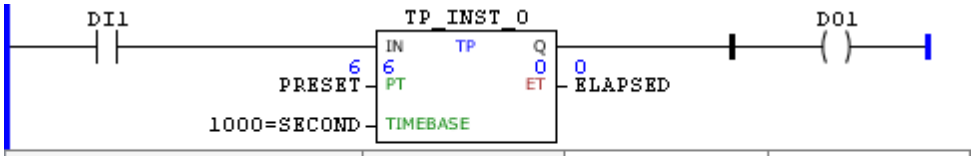
Block Flowchart



Operation Diagram



Example



PRESET	UINT	6	Decimal
ELAPSED	UINT	0	Decimal

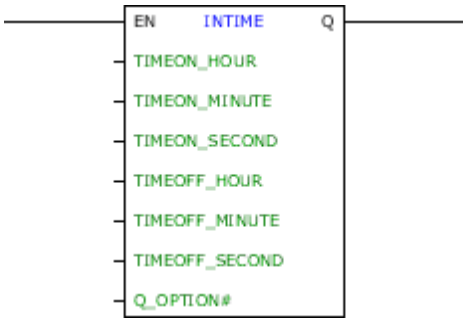
The above example enables the DO1 output for six seconds at each DI1 positive transition.

11.14.7.14 RTC

11.14.7.14.1 INTIME

Block that performs a programmed enabling for a time based on RTC (Real Time Clock).

Ladder Representation



Block Structure

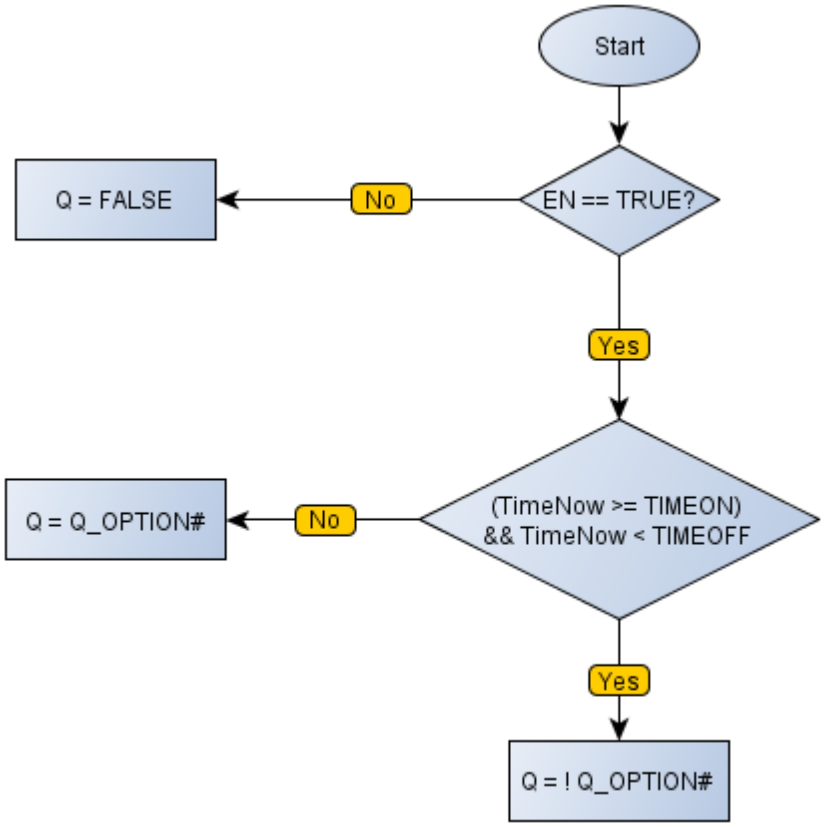
Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	TIMEON_HOUR	WORD UINT	Enabling hour
	TIMEON_MINUTE	WORD UINT	Enabling minute
	TIMEON_SECOND	WORD UINT	Enabling second
	TIMEOFF_HOUR	WORD UINT	Disabling hour
	TIMEOFF_MINUTE	WORD UINT	Disabling minute
	TIMEOFF_SECOND	WORD UINT	Disabling second
	Q_OPTION#	BYTE	Output operation
VAR_OUTPUT	Q	BOOL	Block output

Operation

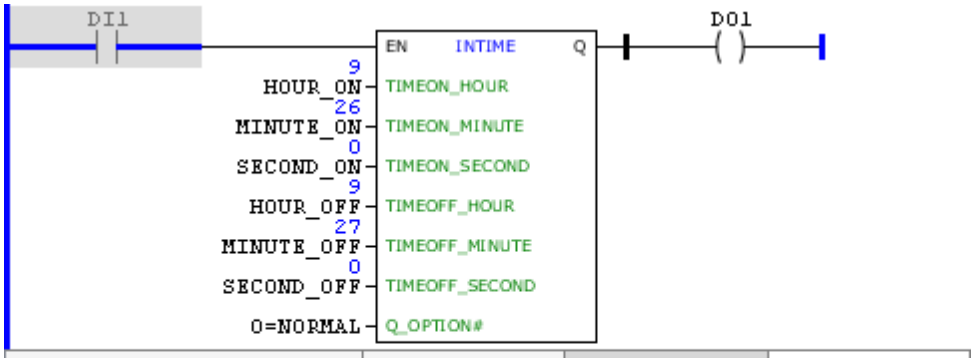
When this block has a TRUE value in EN, it has two modes of operation. If Q_OPTION# is Normal, Q is enabled when the internal clock's time is equal to that defined by the parameters TIMEON and disabled when the internal clock's time is equal to the parameters set by TIMEOFF. If Q_OPTION# is Inverted, Q is disabled when the internal clock's time is equal to that defined by the parameters TIMEON and enabled when the internal clock's time is equal to the parameters set by TIMEOFF.

When EN has FALSE value, Q remains FALSE.

Block Flowchart

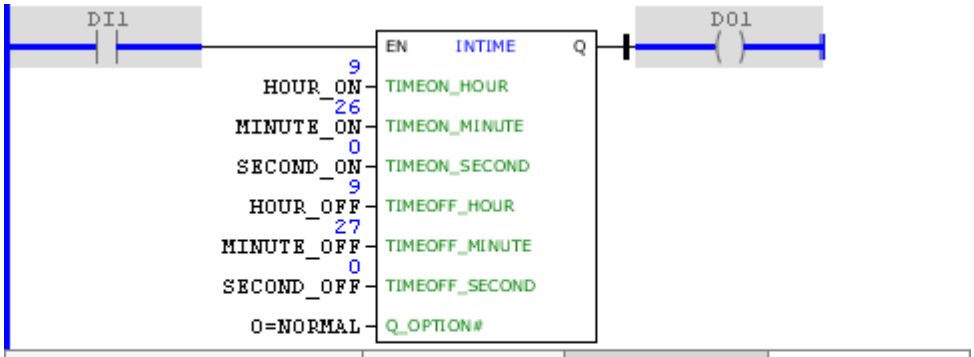


Example



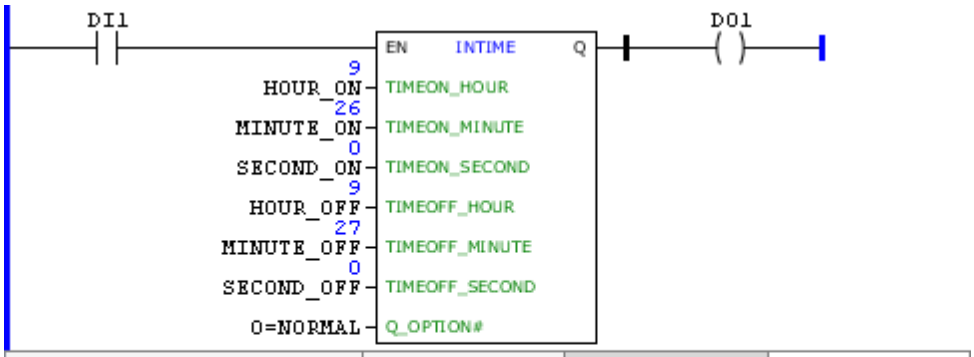
HOUR	UINT	9	Decimal
MINUTE	UINT	25	Decimal
SECOND	UINT	56	Decimal

In the example above, the INTIME block is enabled, the Q_OPTION# input is enabled for NORMAL operation and the current time of the internal clock of the device is lower than the registered enabling inputs of the block (HOUR_ON, MINUTE_ON and SECOND_ON). This way, the Q output is disabled.



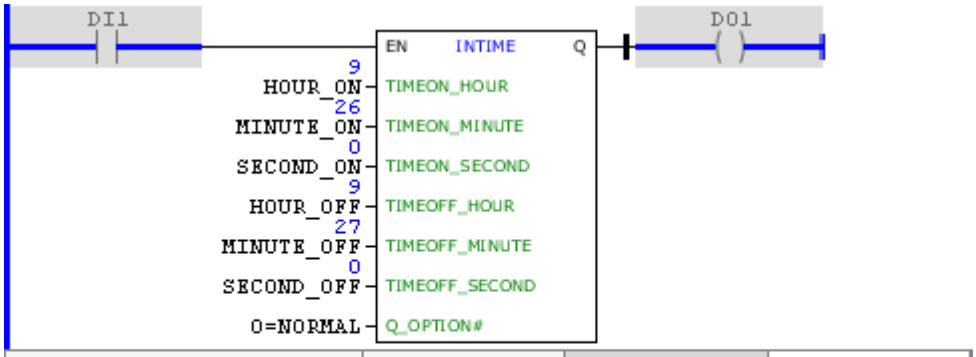
HOUR	UINT	9	Decimal
MINUTE	UINT	26	Decimal
SECOND	UINT	0	Decimal

In the example above, the INTIME block is enabled, the Q_OPTION# input is enabled for NORMAL operation and the current time of the internal clock of the device is equal to the registered in the enabling inputs of the block (HOUR_ON, MINUTE_ON and SECOND_ON). This way, the Q output is disabled.



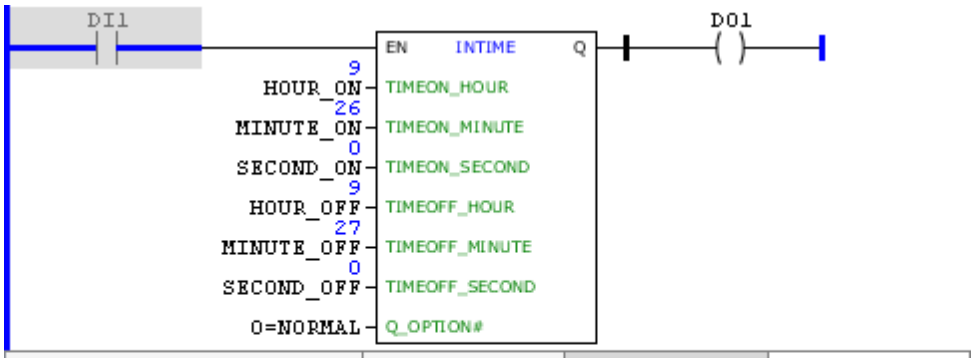
HOUR	UINT	9	Decimal
MINUTE	UINT	26	Decimal
SECOND	UINT	28	Decimal

In the above example, the INTIME block is disabled. This way, regardless of the input, the Q output is disabled.



HOUR	UINT	9	Decimal
MINUTE	UINT	27	Decimal
SECOND	UINT	0	Decimal

In the example above, the INTIME block is enabled, the Q_OPTION# input is enabled for NORMAL operation and the current time of the internal clock of the device is equal to the registered in the disabling inputs of the block (HOUR_OFF, MINUTE_OFF and SECOND_OFF). This way, the Q output is enabled.



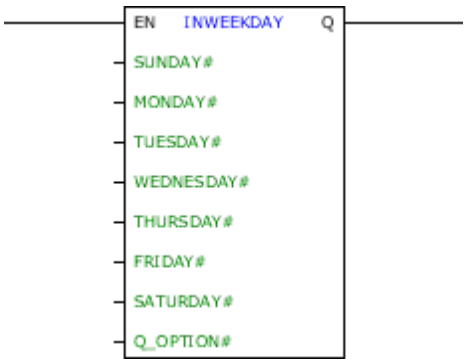
HOUR	UINT	9	Decimal
MINUTE	UINT	27	Decimal
SECOND	UINT	1	Decimal

In the example above, the INTIME block is enabled, the Q_OPTION# input is enabled for NORMAL operation and the current time of the internal clock of the device is superior to the registered in the disabling inputs of the block (HOUR_OFF, MINUTE_OFF and SECOND_OFF). Thus, the Q output is disabled.

11.14.7.14.2 INWEEKDAY

Block that performs a programmed enabling for weekdays based on RTC (Real Time Clock).

Ladder Representation



Block Structure

Variable Type	Name	Data Type	Description
VAR_INPUT	EN	BOOL	Block enabling
	SUNDAY#	BOOL	Enabled on Sundays
	MONDAY#	BOOL	Enabled on Mondays
	TUESDAY#	BOOL	Enabled on Tuesdays
	WEDNESDAY#	BOOL	Enabled on Wednesdays
	THURSDAY#	BOOL	Enabled on Thursdays
	FRIDAY#	BOOL	Enabled on Fridays
	SATURDAY#	BOOL	Enabled on Saturdays
	Q_OPTION#	BYTE	Output operation
VAR_OUTPUT	Q	BOOL	Block output

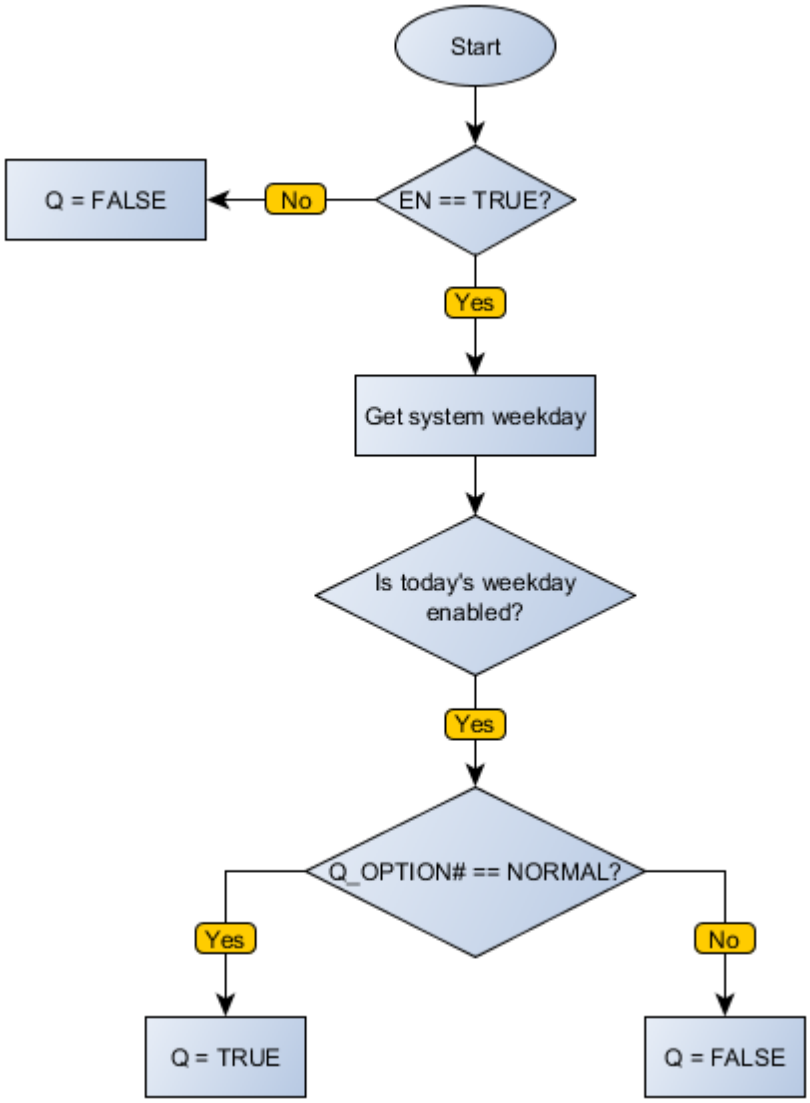
Operation

When this block has a TRUE value in EN, it has two modes of operation. If Q_OPTION# is Normal, Q is enabled if the day of week of the internal clock has Enabled parameter in the block. If Q_OPTION# is Inverted, Q is disabled if the day of week of the internal clock has Enabled parameter in the block.

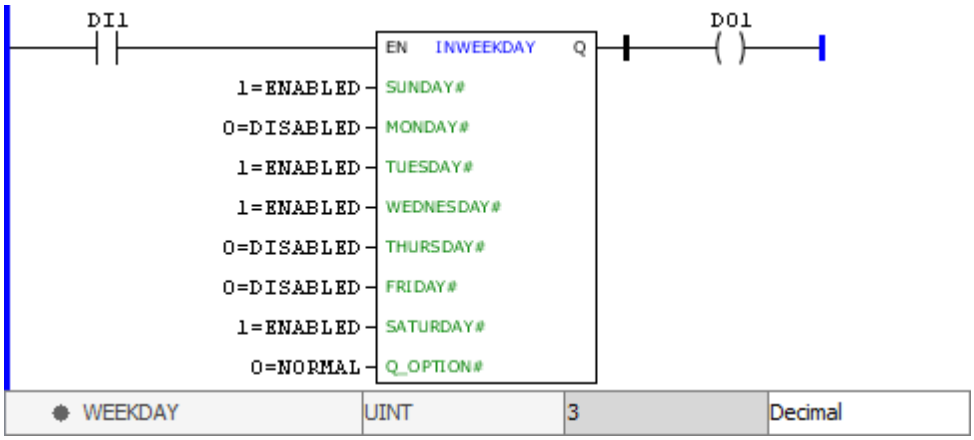
When EN has FALSE value, Q remains FALSE.

NOTE!
The weekdays are identified by numbers, with Sunday being day 0 and Saturday day 6.

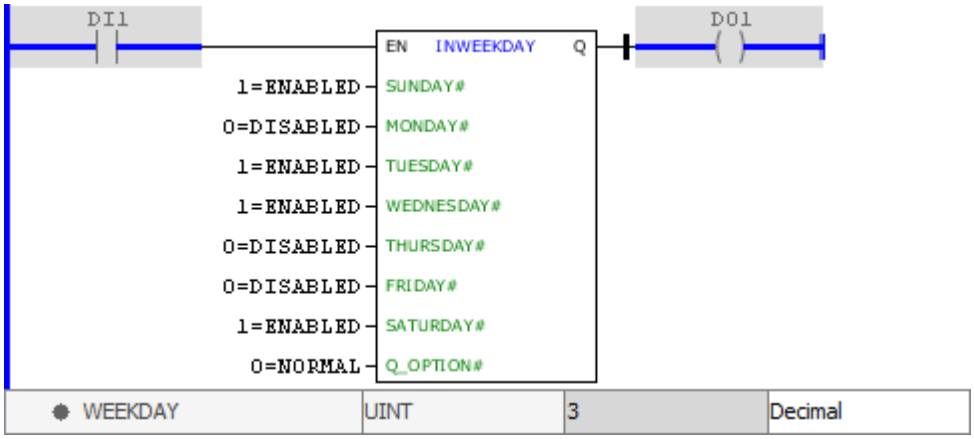
Block Flowchart


Start
EN == TRUE?
No
Q = FALSE
Yes
Get system weekday
Is today's weekday enabled?
Yes
Q_OPTION# == NORMAL?
Yes
No
Q = TRUE
Q = FALSE

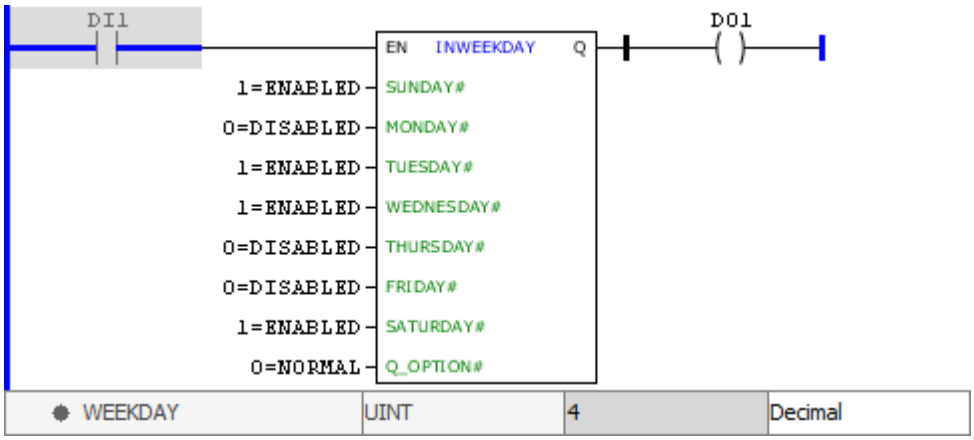
Example


DI1
EN
INWEEKDAY
Q
DO1
1=ENABLED
SUNDAY#
0=DISABLED
MONDAY#
1=ENABLED
TUESDAY#
1=ENABLED
WEDNESDAY#
0=DISABLED
THURSDAY#
0=DISABLED
FRIDAY#
1=ENABLED
SATURDAY#
0=NORMAL
Q_OPTION#
WEEKDAY
UINT
3
Decimal

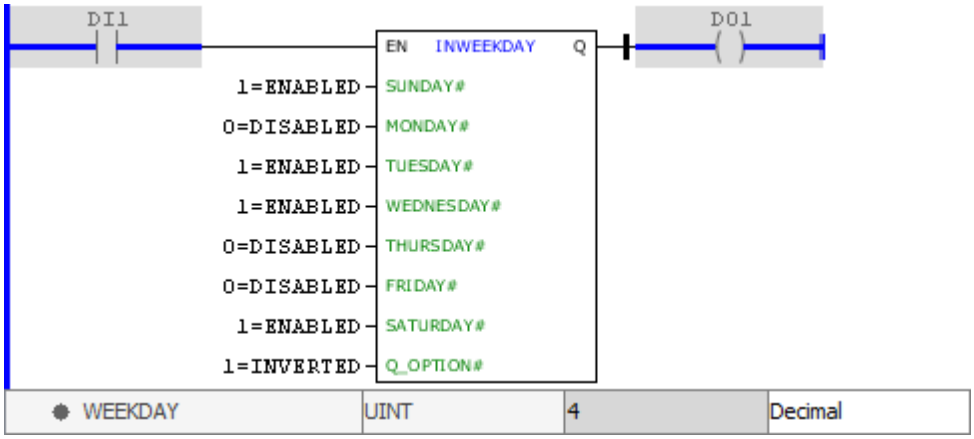
In the above example, the INWEEKDAY block is disabled. This way, regardless of the input, the Q output is disabled.



In the example above, the INWEEKDAY block is enabled and Q_OPTION# input is enabled for NORMAL operation. The current day of the week of the device's internal clock is Wednesday (value 3), which has ENABLED status in the programming. This way, the Q output is enabled.



In the example above, the INWEEKDAY block is enabled and Q_OPTION# input is enabled for NORMAL operation. The current day of the week of the device's internal clock is Thursday (value 4), which has DISABLED status in the programming. Thus, the Q output is disabled.



In the example above, the INWEEKDAY block is enabled and Q_OPTION# input is enabled for INVERTED operation. The current day of the week of the device's internal clock is Thursday (value 4), which has DISABLED status in the programming. This way, the Q output is enabled.

11.14.7.15 Structures

Structure is a data grouping used to define a recipe or an object.

In the Ladder program, it is possible to create variables of the structure type and use them in the blocks. To access the internal members of the structure, the '.' is used followed by its respective member.

Creating a structure

1. With the right button of the mouse on the folder **Structure**, click on **New file**.

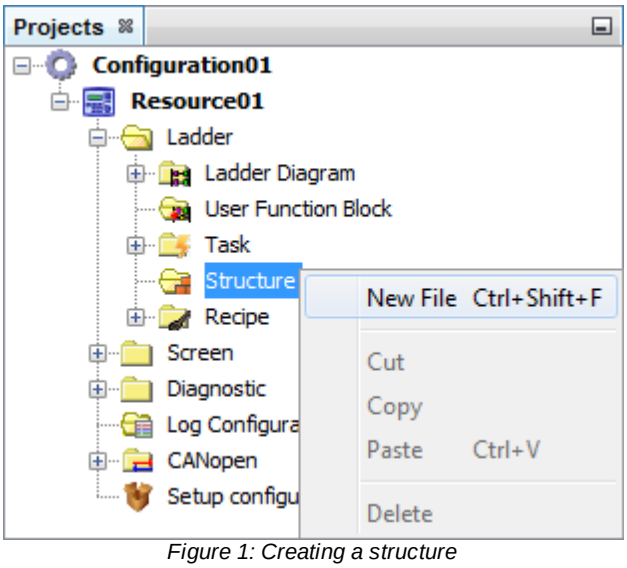


Figure 1: Creating a structure

2. Define the file name and press the **Next** button.

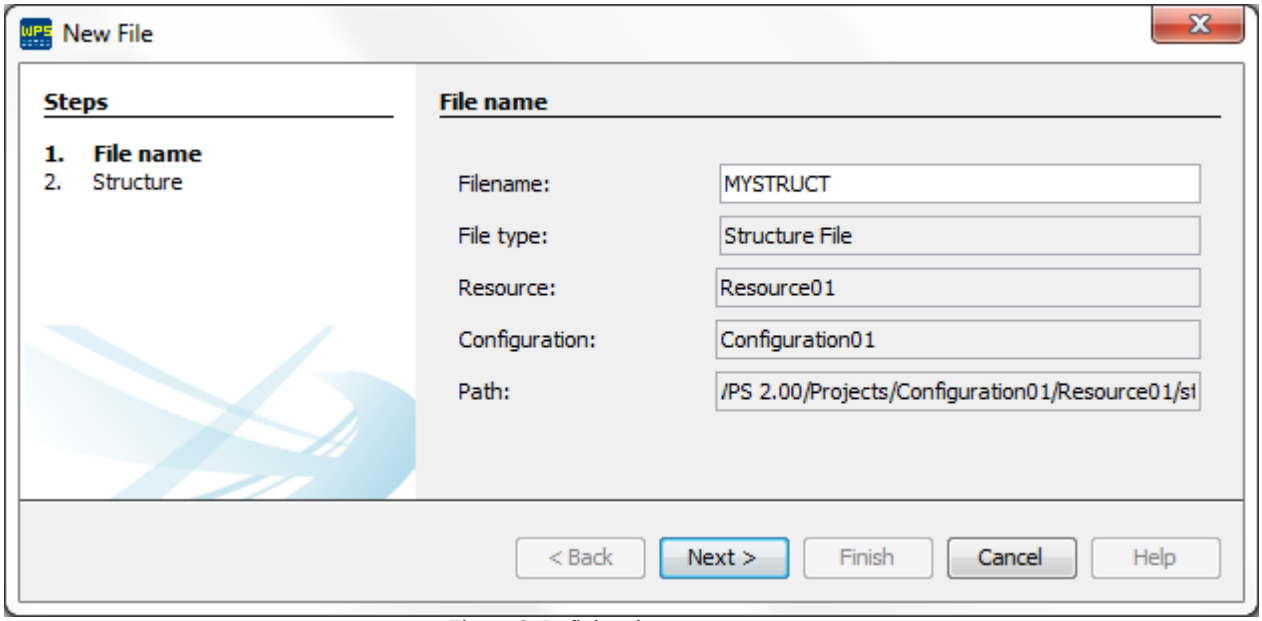


Figure 2: Defining the structure name

3. Configure the structure using the buttons presented in the figure below.

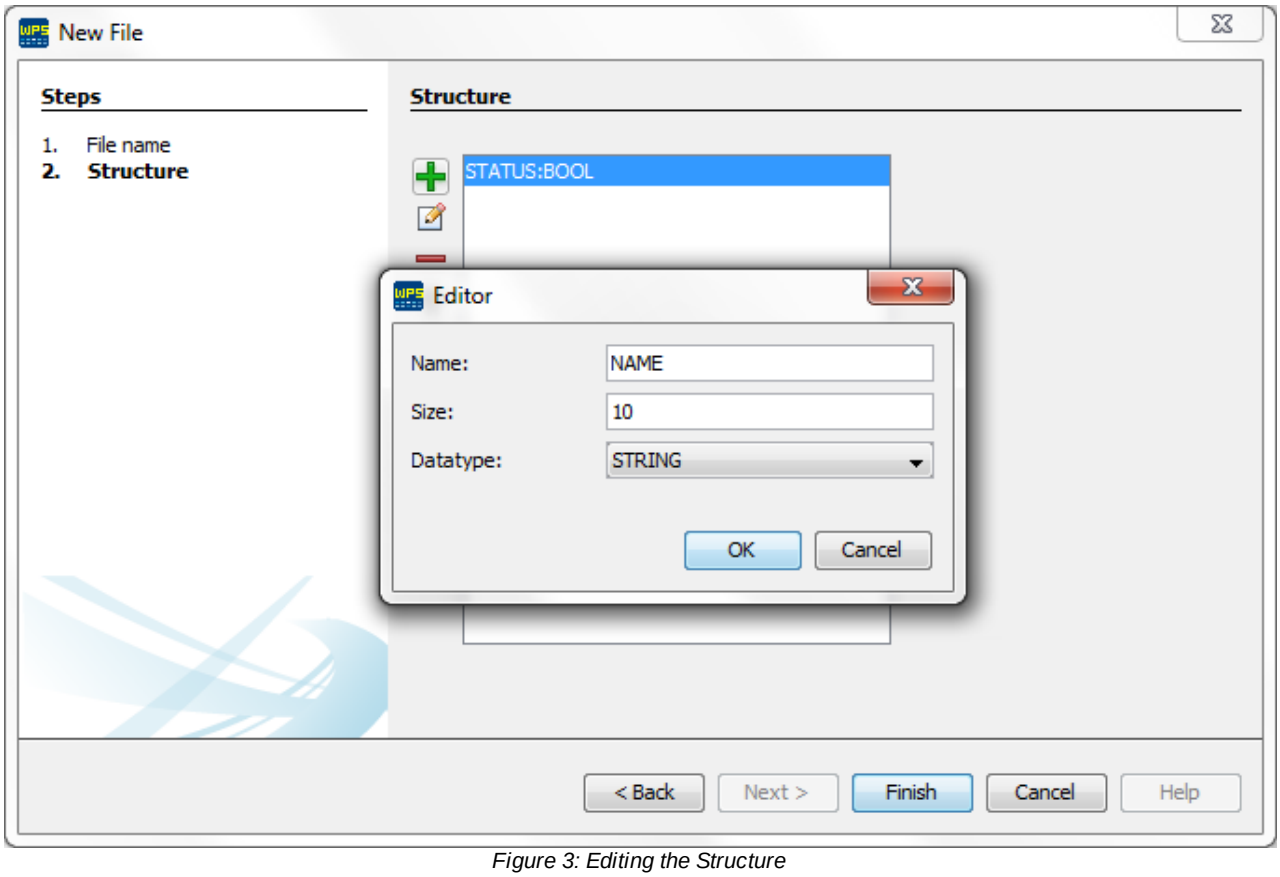


Figure 3: Editing the Structure

4. After finishing the edition of the structure, click on the button **Finish**.

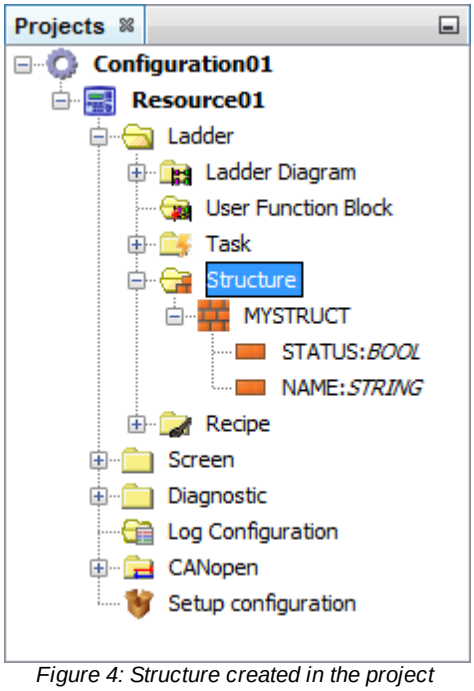


Figure 4: Structure created in the project

Editing a structure

Just double click on the desired structure, as shown in figure 4, and a window will open as shown in figure 3, allowing to insert new data, erase or move the position of the data.

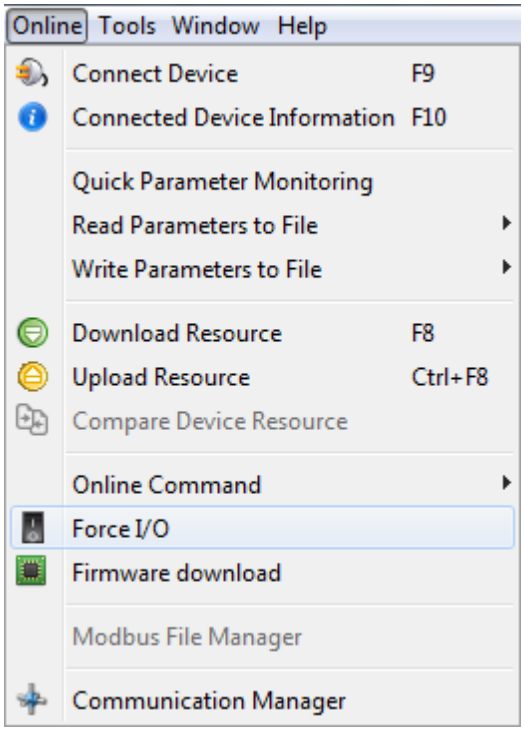
11.14.8Communication

11.14.8.1 Force I/O

Overview

The force inputs and outputs window is used for the values of the digital and analog inputs to be read by the program, by values manipulated by the user, regardless their physical state. It also allows the manipulation of the physical states of the digital and analog outputs by the user independently of the values calculated by the program.
In order to force the device inputs and outputs, it is necessary that the online monitoring be active and the option **Run cyclically** be enabled. The data are sent to the device every 2 seconds.
The values can be edited with the device disconnected. The configurations are stored in the resources and recorded whenever the main resource selection is changed.
The data displayed on the force I/O window contain the values belonging to the resource (and configuration) selected as main.

The force I/O window is open trough the menu **Online** > **Force I/O**:



Toolbar

The toolbar of the force window has the options to run cyclically, upload the device force configuration, enable all and disable all:

Run cyclically: Sends the user's configurations to the device and updates the state of the inputs and outputs in a cyclic way.

Upload configuration: Allows the current configuration of the device to be read. For this option to be enabled, it is necessary that the online monitoring be active and the option run cyclically be disabled.

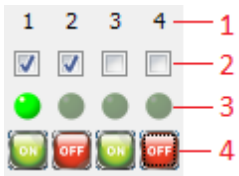
Enable all: Enables the force I/O of all of the inputs and outputs of the device.

Disable all: Disables the force I/O of all of the inputs and outputs of the device.

Input and Output commands

For each digital and analog input and output there is a selection box linked to enable the force, a status field and an edition field.

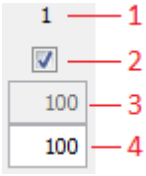
Digital:



1. Number of the digital inputs/output
2. Enable/disable Force I/O

3. Current status of the I/O: It has three statuses: 1. light green LED: activated; 2. dark green LED: deactivated; 3. gray LED: the value is not being read.
4. Enable/disable the input/output

Analog:



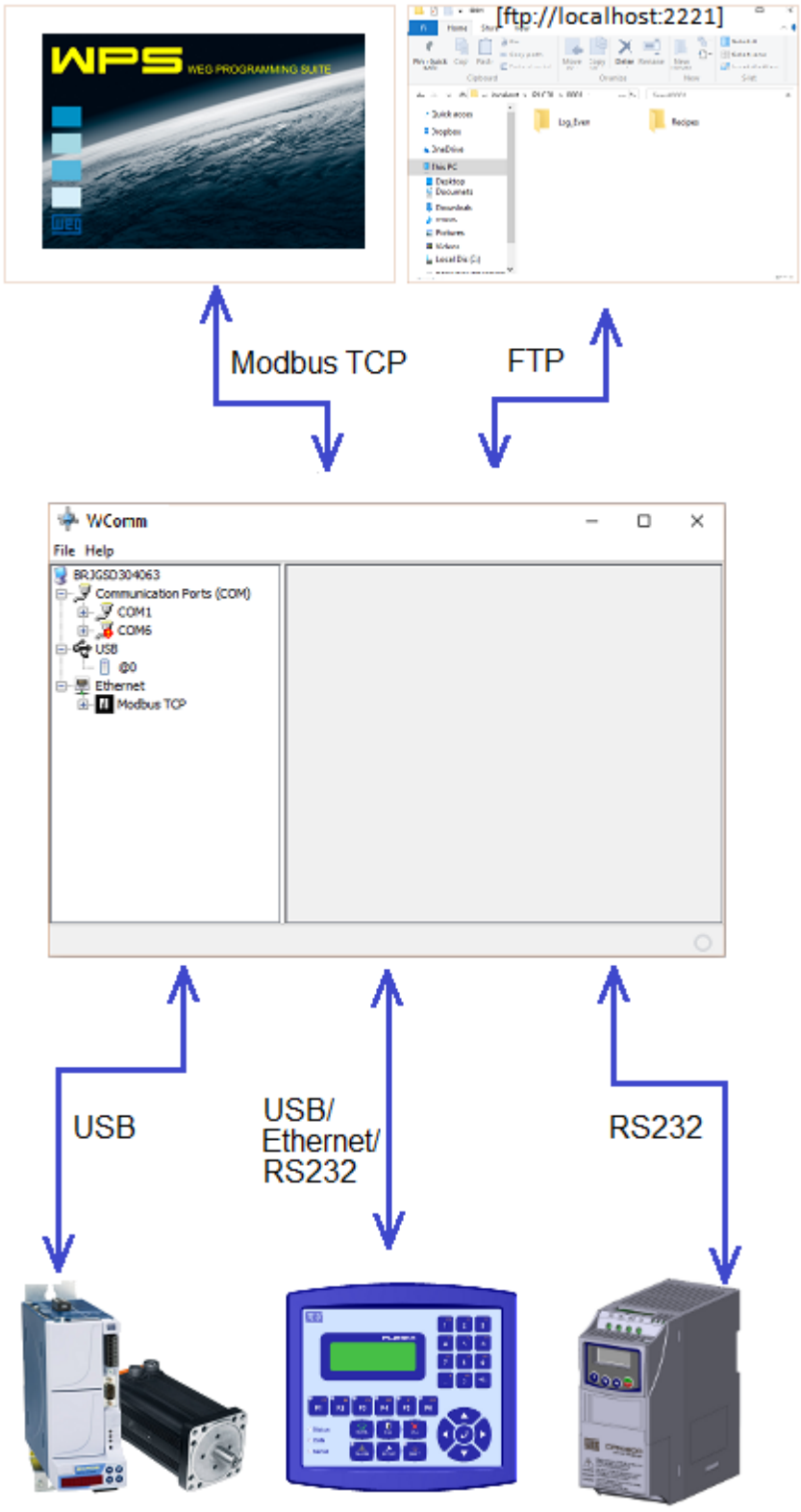
1. Number of the analog input/output
2. Enable/disable Force I/O
3. Current value of the input/output
4. Value of the input/output configured by the user

NOTE!
The analog signal scale has 15 bits plus 1 bit for signal, except for SSW900 which it has only 10 unsigned bits.

12 WComm

12.1 Introduction

The WComm (Communication Manager) configures the communication channels of the USB interfaces, communication ports (COM) and Ethernet, providing such connections via Modbus TCP protocol and data access via FTP for access through other applications.

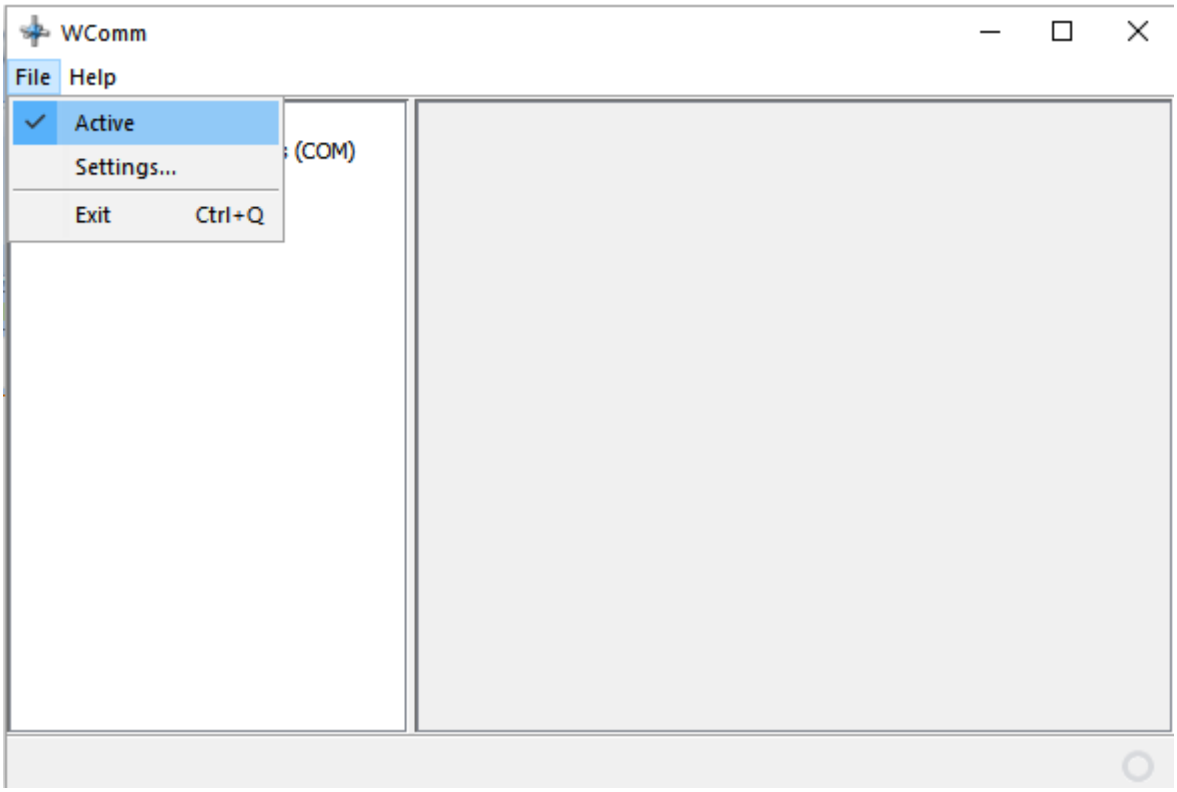

WPS
WEG PROGRAMMING SUITE
[ftp://localhost:2221]
Modbus TCP
FTP
WComm
File Help
BRJGSD304063
Communication Ports (COM)
COM1
COM6
USB
@0
Ethernet
Modbus TCP
USB
USB/
Ethernet/
RS232
RS232

12.2 Configuration

12.2.1 Menus

The WComm offers the configuration options available in this menu:

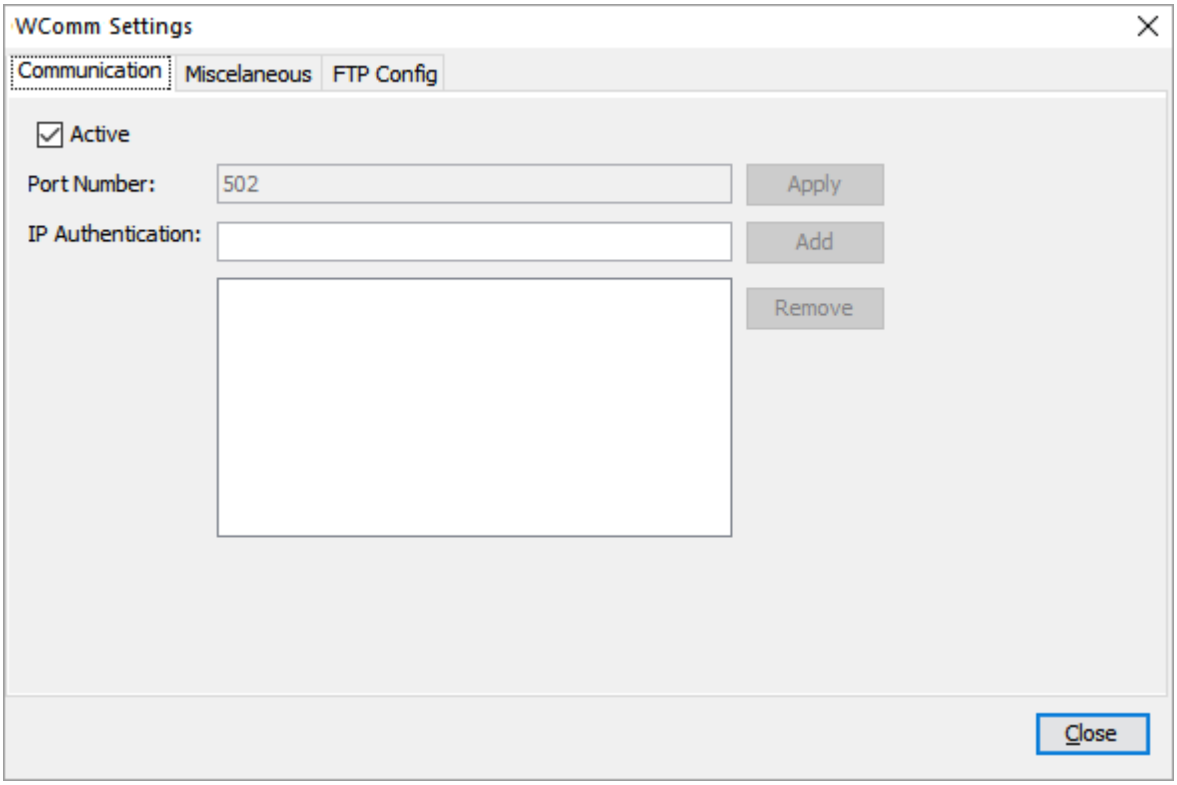
- Menu **File:**
- Option to activate or deactivate the communication with the applications (standard: active);
- **Configuration...** presents a dialogue box with options.



The dialogue box has the options: **Communication**, **Miscellaneous** and **FTP Config**.

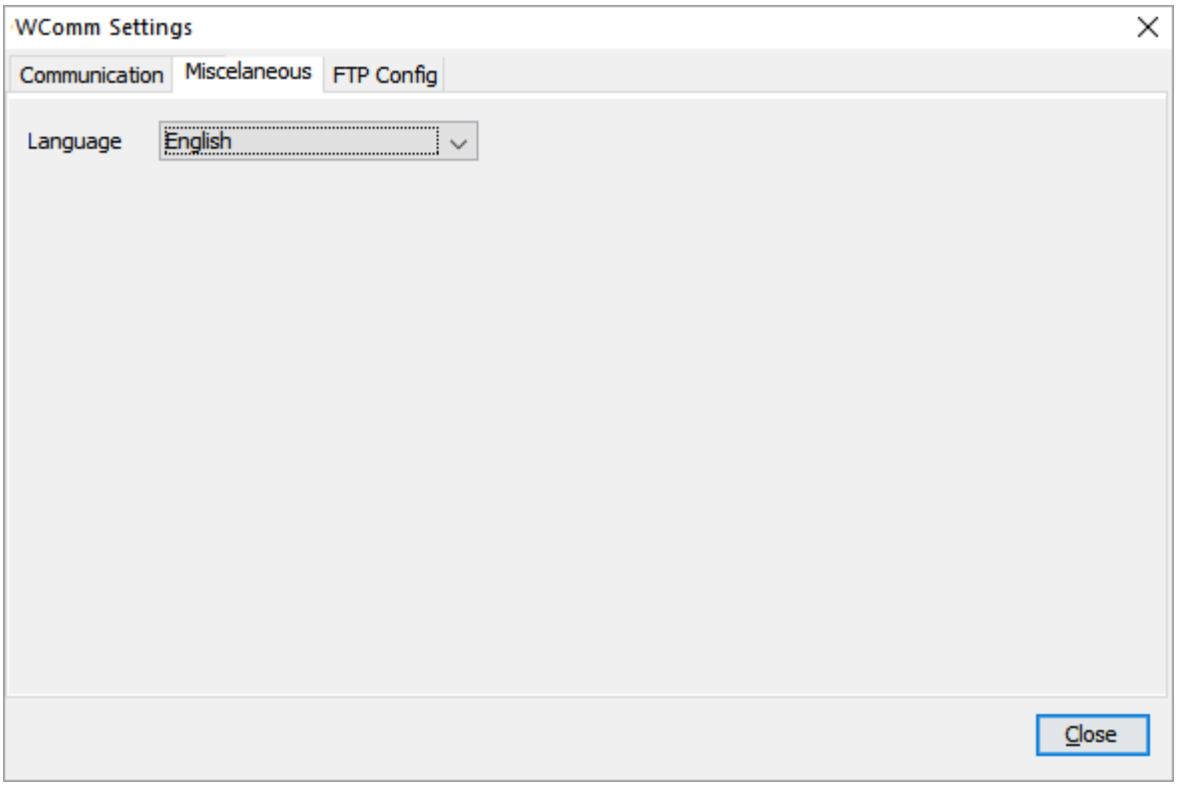
- **Communication**

- Checkbox: when checked, it enables the WComm for remote connection via TCP/IP;
- Port Number: number of the TCP/IP communication port the WComm uses for communication with applications that communicate with the devices through it;
- IP Authentication: list of the IP addresses that can access the WComm. An empty list indicates that any IP address can access the WComm.



- **Miscellaneous**

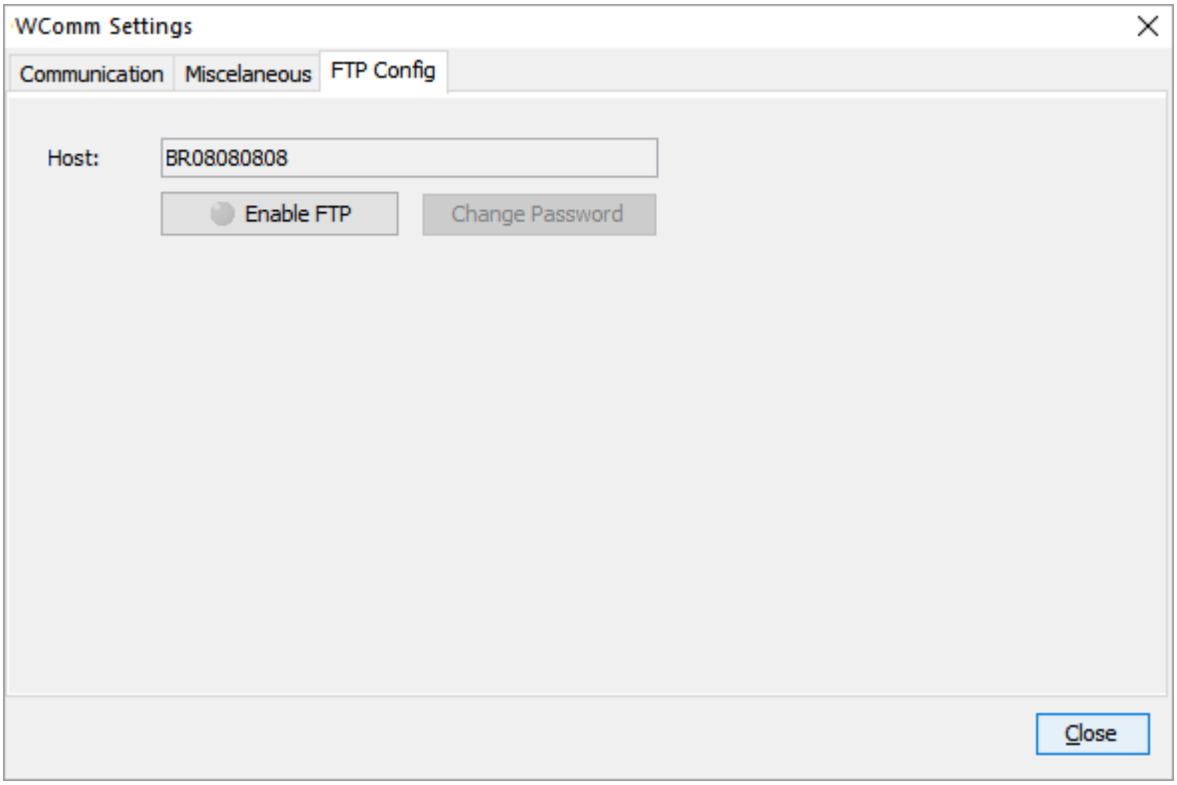
- Language: allows selecting the language of the WComm from the options available in the menu.



- **FTP Config**

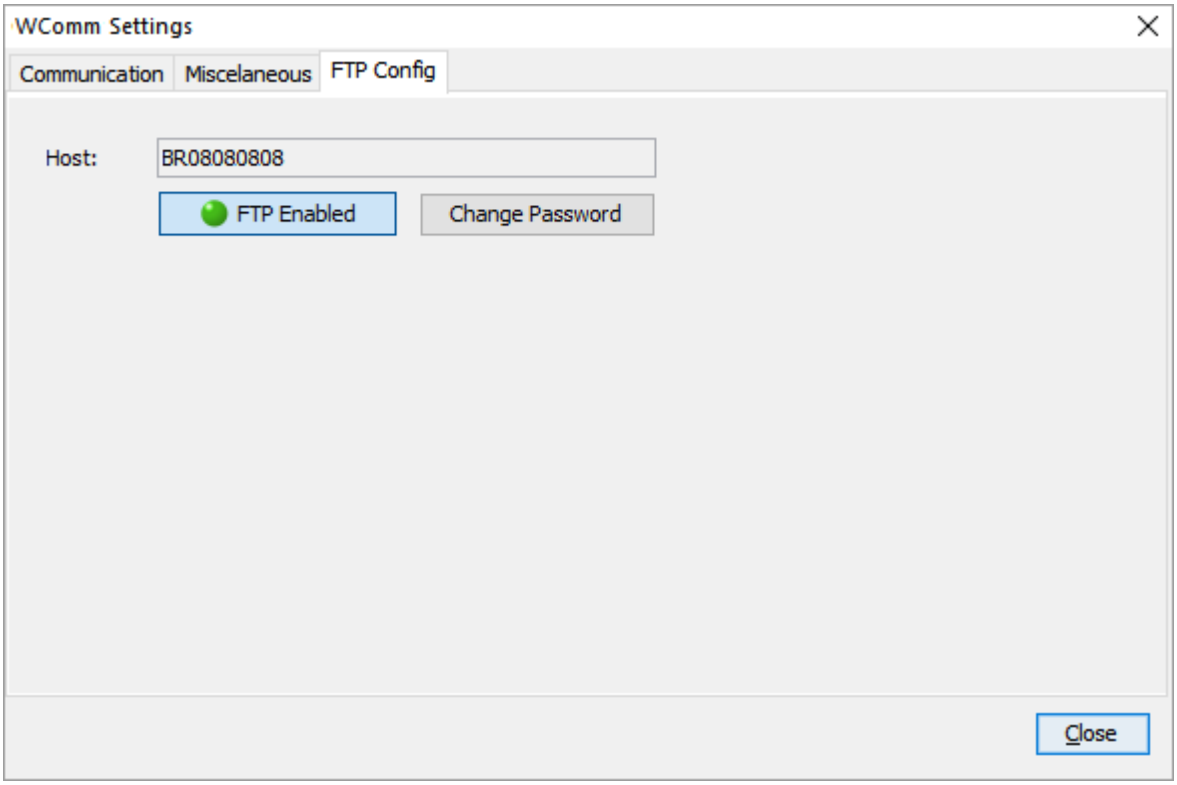
The FTP server (File Transfer Protocol) allows the files contained in the PLC300 SD card to be handled by an FTP client. In order to start the FTP server, see the information below:

- Host: presents the hostname to access the FTP service;
- Button to enable or disable the FTP service (default: disabled);
- Button to modify the administrator's password.

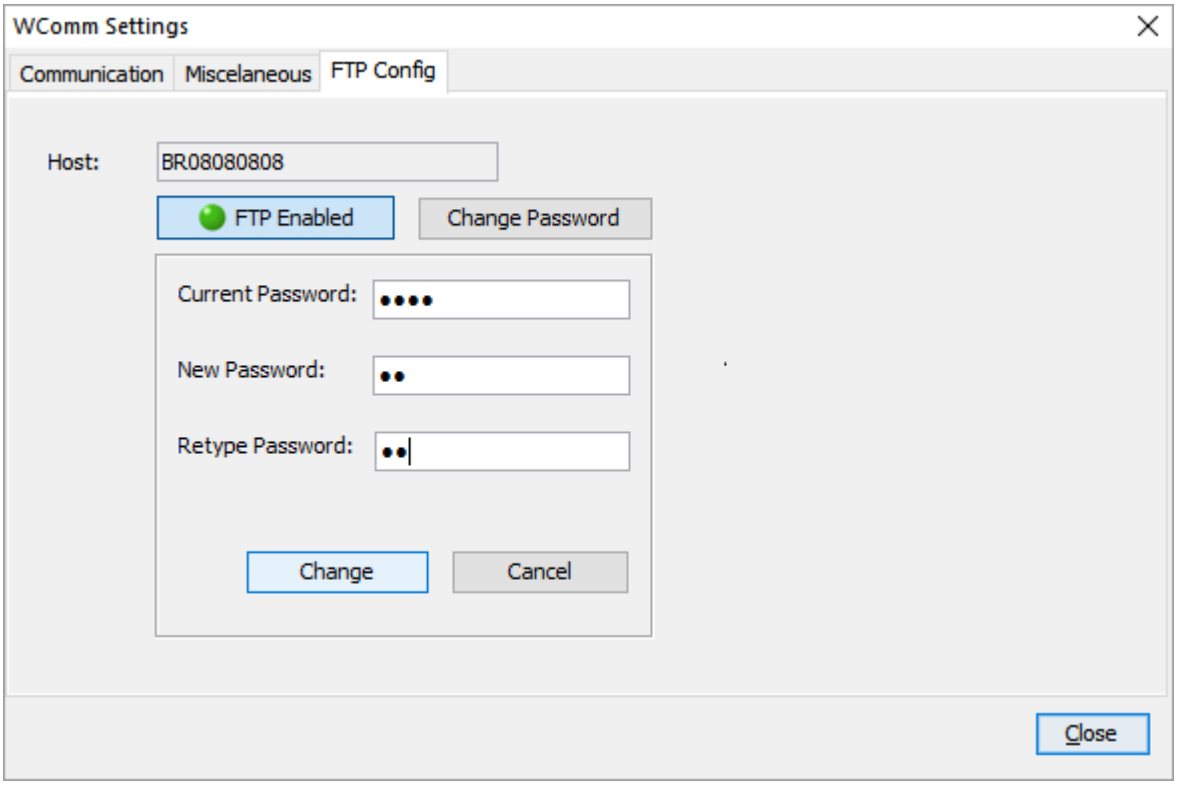


- FTP Status:

The image below presents the FTP service enabled (**Enable** button pressed):



- **Modify the administrator's password:**
This function will be only available when the FTP service is enabled, after clicking on the button to change the password.
Enter: current password, new password, retype the new password.



NOTE: The FTP service will be available using any FTP client.
- Hostname: indicated in the window above as "BR08080808".
- Port: 2221.
- Username: admin.
- Password: default: weg, or defined by the user.

Other configurations of the FTP service will be available in the Configuration File.

12.2.2 Quickstart Guide for FTP

The first step is to install the WPS, using the installer file available at http://www.weg.net.
Copy the folder name where WPS was installed (usually **C:\WEG\WPS 2.50**)

WComm is a WPS's module and its standard configuration allows to access PLC300 device through USB port, giving access since root folder **/PLC300/** on SD Card (connected to PLC300).

The FTP service starts **enabled**, because of that is necessary to activate it at first use: the instructions are available in session Configuration.

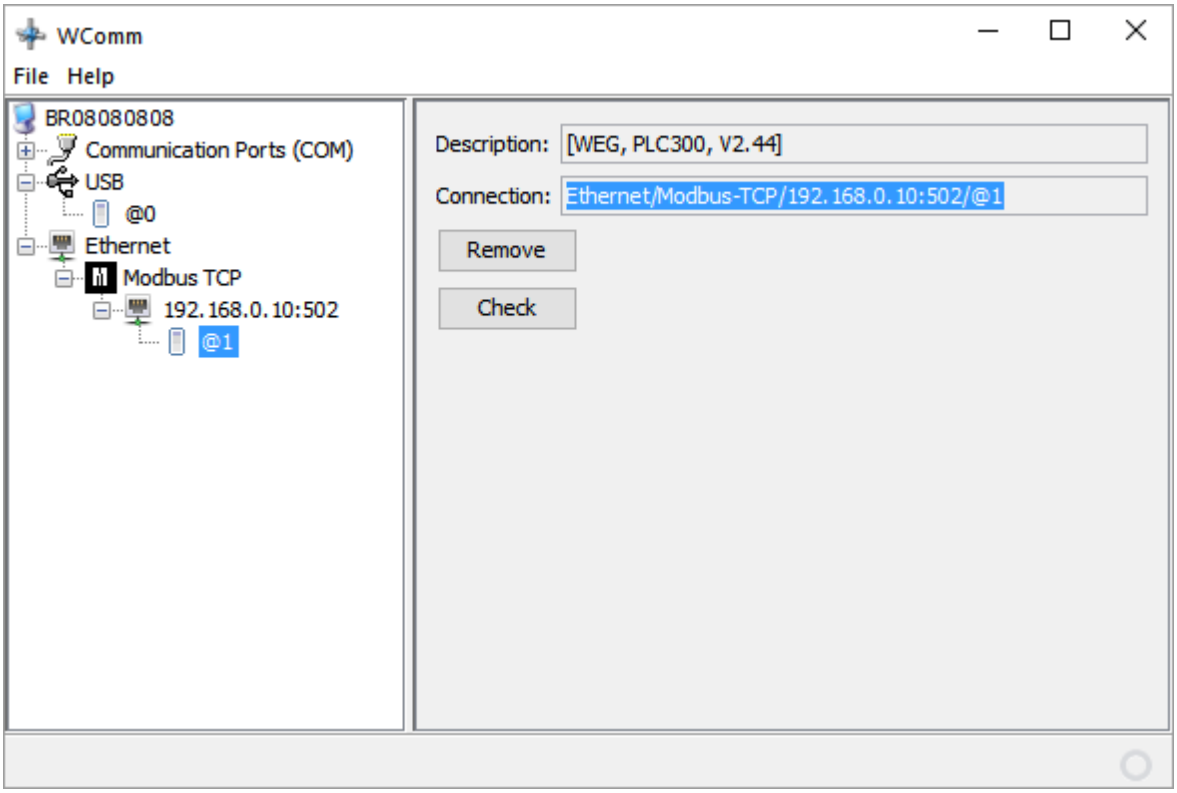
Once enabled, the WComm's FTP service uses the standard configuration.
To change the configuration, please follow the steps below.

If the WComm's configuration file was damaged/corrupted during edition, it may be necessary to uninstall and reinstall WPS.
Thus, if necessary to change configuration file, it is recommended to only change the connection type line.
Leave another changes to a next file edition.

- **Changing the connection type.**

- Using an regular text editor (Notepad), edit the configuration file located in folder where WPS is installed, usually **C:\WEG\WPS 2.50\wps\modules\gateway\conf\users.properties**.

- In WComm, click the button to check communication, making sure that PLC300 is accessible:



- Select and copy the connection type line as shown in image above (text in blue highlight).

- In **users.properties** file, change the connection´s variable, changing by text copied in previous step:
  From: **ftpserver.user.admin.connection=USB/@0**
  To: **ftpserver.user.admin.connection=Ethernet/Modbus-TCP/192.168.0.10\:502/@1**
  This instruction sets the communication type will be used to access the FTP service when logged as administrator (**admin**).
  There are samples in configuration file for all possibilities of connections: **USB**, **Serial** and **Ethernet**.
  These samples starts with character #, therefore changing it does not affects the software operation (since the # character being kept at line start).

- Save changes in **users.properties** file.

- Close * the WPS.

- After that, open * WPS. It will load the WComm new configuration.

- **Accessing data.**

- WComm module has two user profiles:
  admin - permission to read and write on SD Card. Can create, read and delete files and folders.
  anonymous - read only permission on files at SD Card.
- Restrictions:
  It is not possible to execute (open) files directly from SD Card: it is necessary to copy file to a local folder an then is possible to execute it from local folder.
  It is not possible to multi select files and folders: to transfer files and folders, please copy one at a time.

- Be sure that PLC300 is communicating with WComm (check corresponding connection).

- Access the FTP service as **admin** user:
  Access file explorer ("My Computer") from Operational System.
  Click on address bar, changing the selection "**This Computer**" for **ftp://admin@localhost:2221.** Press **Enter**.
  Type user: **admin** and password: **weg** when requested in a login window.
  The SD Card content will be shown. If there is no SD Card in PLC300 the system shows an error.

Also is possible to login as **anonymous** user. To do that, please follow steps above, using the folder **ftp://anonymous@localhost:2221**.

- **Disabling and Enabling users.**

- Using an regular text editor (Notepad), edit the configuration file located in folder where WPS is installed, usually **C:\WEG\WPS 2.50\wps\modules\gateway\conf\users.properties**.

- In **users.properties** file, change the variable **ftpserver.user.anonymous.enableflag**, changing the value as shown:
  From: **ftpserver.user.anonymous.enableflag=true**
  To: **ftpserver.user.anonymous.enableflag=false**
This instruction enables (**true**) or disables (**false**) the user **anonymous**.
Not only anonymous, but also **admin** can be changed, though it is not recommended.

- Save changes in **users.properties** file.

- Close * the WPS.

- After that, open * WPS. It will load the WComm new configuration.

- **Changing root folder.**

- Using an regular text editor (Notepad), edit the configuration file located in folder where WPS is installed, usually **C:\WEG\WPS 2.50\wps\modules\gateway\conf\users.properties**.

- In **users.properties** file, change the variable **ftpserver.user.admin.homedirectory**, changing the value by the new path, as follows:
  From: **ftpserver.user.admin.homedirectory=/PLC300/**
  To: **ftpserver.user.admin.homedirectory=/PLC300/0001/**
  This instruction **ftpserver.user.admin.homedirectory** changes the root folder available by FTP service when logged as **admin** user.

- It is also possible to change this configuration for **anonymous** user. To use the same path as shown in

sample above, the configuration should be:
From: **ftpserver.user.anonymous.homedirectory=/PLC300/**
To: **ftpserver.user.anonymous.homedirectory=/PLC300/0001/**

- Save changes in **users.properties** file.

- Close * the WPS.

- After that, open * WPS. It will load the WComm new configuration.

- **Changing the password.**

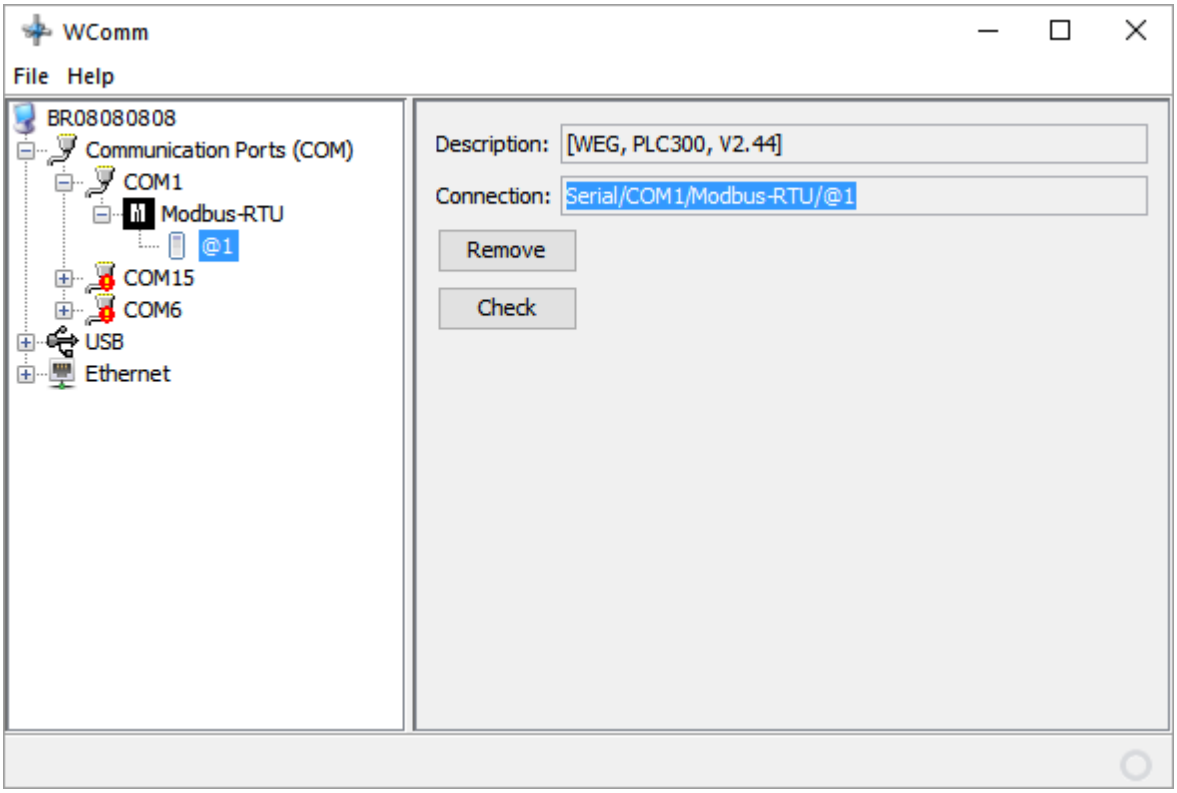
- Informations about the procedure to change password (and another configurations) are available in chapter Configuration.

NOTE!
* It is possible to execute just the WComm module, which is available at the WPS's install folder, usually at **C:\WEG\WPS 2.50\wps\modules\gateway\CommunicationGateway.exe**.
At this condition, when necessary to close the WComm, click the right mouse button over icon  in systray and select the option to close.

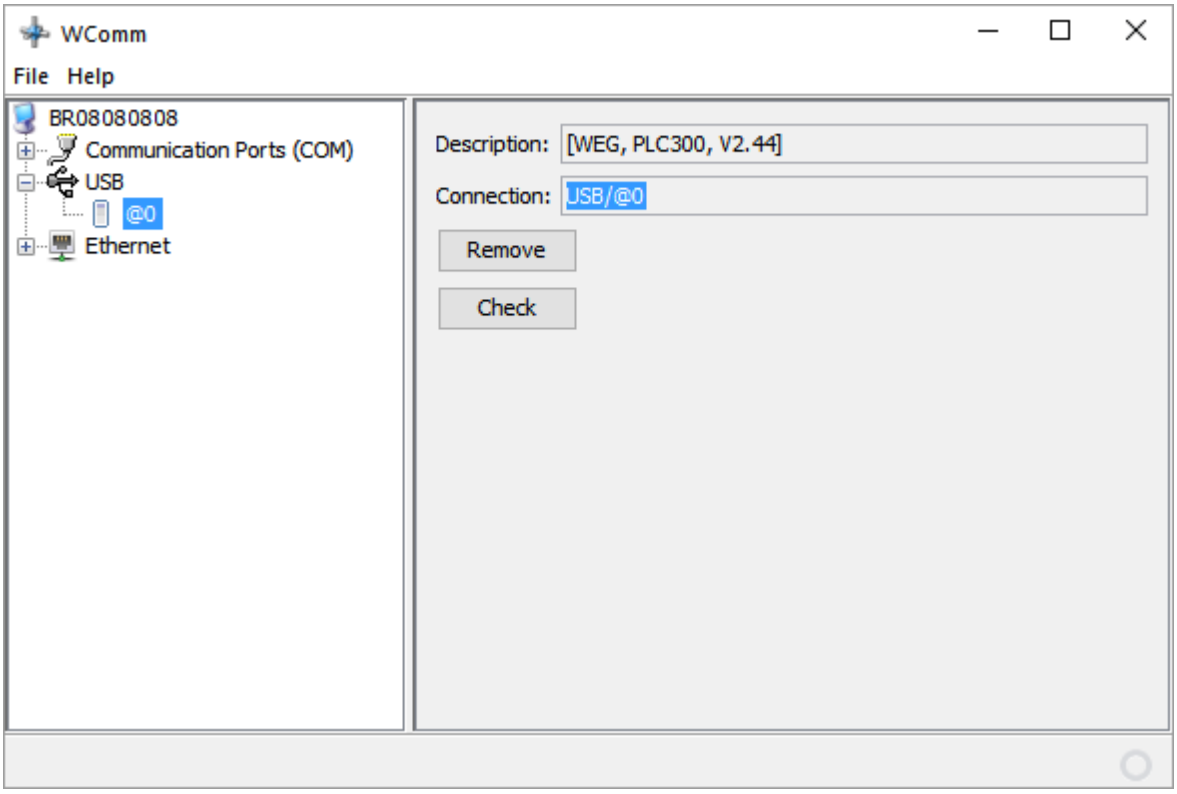
12.2.3 Configuration File

The FTP service configuration file allows customizing the following data:

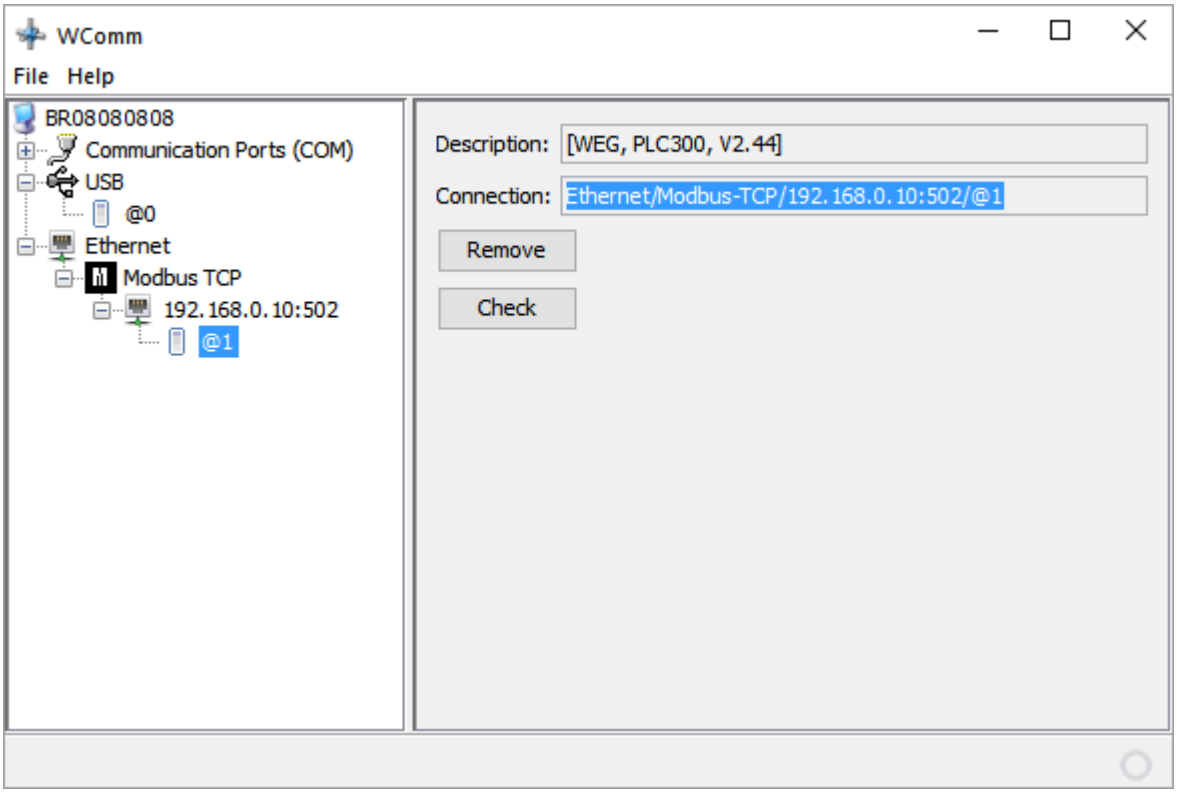
- Read-only or read and write permission for administrator user;
- FTP default path: the initial path that will be displayed on the FTP client after the access;
- Type of communication with the resource: indicate the communication form between the WComm and the resource. Such information is available in the communication port selection, which must be selected and copied, as shown in the following images:


WComm
File Help
BR08080808
Communication Ports (COM)
COM1
Modbus-RTU
@1
COM15
COM6
USB
Ethernet
Description: [WEG, PLC300, V2.44]
Connection: Serial/COM1/Modbus-RTU/@1
Remove
Check

Serial


WComm
File Help
BR08080808
Communication Ports (COM)
USB
@0
Ethernet
Description: [WEG, PLC300, V2.44]
Connection: USB/@0
Remove
Check

USB



Ethernet

The configuration file name is **users.properties**, available in folder **(...)/wps/modules/gateway/conf/** of the WPS.

Configuration file structure with example information (setting up anonymous and admin profiles):

#Generated file - don't edit (please) #Fri May 12 15:35:59 BRT 2017	
ftpserver.user.anonymous.idletime=0	***Time (in s) for automatic disconnection of the FTP.*** *Default: 0 (without automatic disconnection - recommended)*
ftpserver.user.anonymous.enableflag=true	***Enable anonymous user?*** *Default: true (enabled), false (disabled).*
ftpserver.user.anonymous.homedirectory=/PLC300/0001/	***Path to the SD Card files that will be the root folder in the FTP server.*** *Default:* ***/***
ftpserver.user.anonymous.connection=USB/@0	***Connection to the resource when accessing an FTP with an anonymous user.***
ftpserver.user.admin.writepermission=true	*Read/write permission for admin user, where:* ***true:*** *read and write to the SD Card;* ***false****: read-only.*
ftpserver.user.admin.idletime=0	***Time (in s) for automatic disconnection of the FTP.*** *Default: 0 (without automatic disconnection - recommended)*
ftpserver.user.admin.enableflag=true	***Enable admin user?*** *Default: true (enabled), false (disabled).*
ftpserver.user.admin. userpassword=XXXXXXXXXXXXXXXXXXXXXXXXXXXX	***Password in encrypted format.*** ***Do not change ****
ftpserver.user.admin.homedirectory=/PLC300/	***Path to the SD Card files that will be the root folder in the FTP server.*** *Default:* ***/***
ftpserver.user.admin.connection=USB/@0	***Connection to the resource when accessing an FTP with an admin user.***
#ftpserver.user.admin.connection=USB/@0	*Example of USB connection to the resource through the admin FTP user.*
#ftpserver.user.admin.connection=Serial/COM1/Modbus-RTU/@1	*Example of RS232 Serial connection to the resource, via COM1, (admin FTP user).*
#ftpserver.user.admin.connection=Ethernet/Modbus-TCP/192.168.0.10\:502/@1	*Example of Ethernet connection to the resource, via IP 192.168.0.10, port 502 and Unit Id 1, (admin FTP user).*

NOTE: Additional information to configure WComm, in a first utilization to read SD Card data, is available in Quickstart Guide for FTP.

NOTE: Any change in this file requires the reset of the WPS in order to implement the new configuration of WComm module.
* **Changing the password in the users.properties file compromises the access of the user to the application FTP service.**

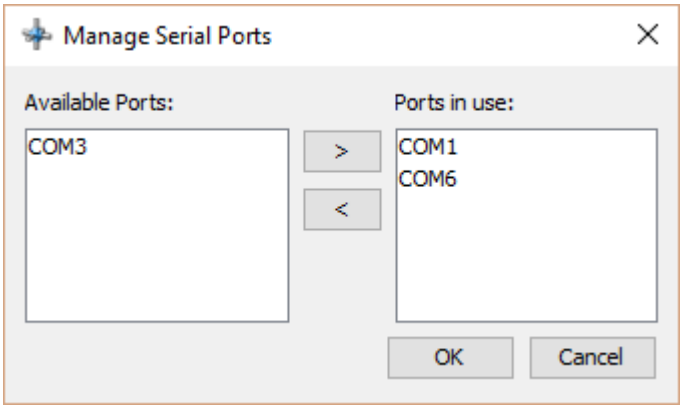
12.3 Add/Remove Connections

In order to add a new connection, select one of the available connections: **Communication Ports (COM)**, **USB** and **Ethernet.**

Communication Port (COM):

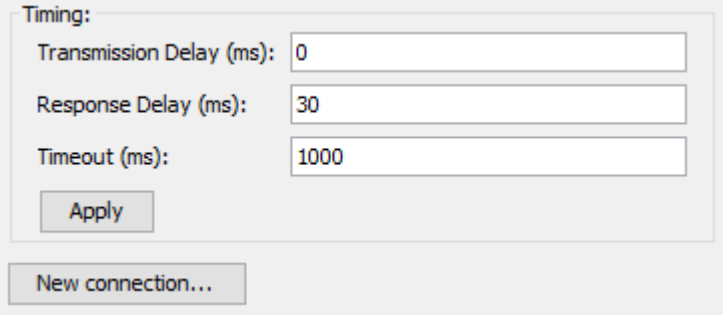
- Expand the “Communication Ports (COM)” item.

- In case the COM port is not listed, select “Manage Serial Ports” in the “Communication Port” item, and choose the used port.



- Change the baud rate, data bits, stop bits and parity settings according to the equipment configuration.

- Select the communication protocol (Modbus-RTU);
- Configure the Modbus-RTU protocol:



- Transmission delay(ms): waits for this time before sending the next telegram.
  - Adjustable range: 0 ... 20000;
  - Default: 0.

- Response delay (ms): waits for this time before trying to receive the response from the sent telegram.
  - Adjustable range: 0 ... 20000;
  - Default: 0.

- Timeout (ms): waits for this time before indicating that the response was not received.
  - Adjustable range: 100 ... 20000;
  - Default: 5000.

- Click on the **New connection...** button:
  - Select the slave address for the new connection according to the equipment configuration.

USB Interface

- Expand the USB item.
- Configure the USB interface:
  - Transmission delay(ms): waits for this time before sending the next telegram.
    - Adjustable range: 0 ... 20000;
    - Default: 0.

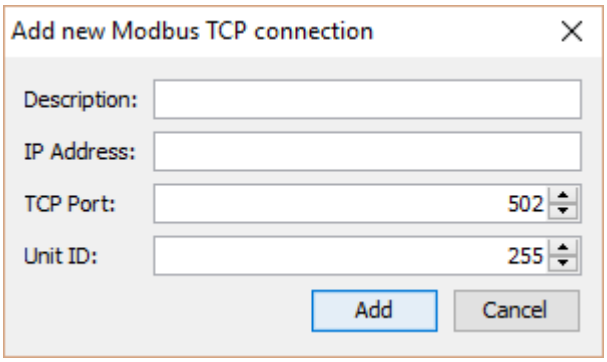
  - Response delay (ms): waits for this time before trying to receive the response from the sent telegram.
    - Adjustable range: 0 ... 20000;
    - Default: 0.

  - Timeout (ms): waits for this time before indicating that the response was not received.
    - Adjustable range: 100 ... 20000;
    - Default: 1000.

- Click on the **New connection...** button.

Ethernet

- Expand the Ethernet item.
- Select the communication protocol (Modbus-TCP).
- In case the IP address is not listed, select the port with the **New Connection** option in the “Modbus-TCP” item, and configure the connection.



  - Description: Connection description to be displayed.
  - IP Address: Address configured in the Modbus-TCP slave device.
  - TCP Port: Number of the port used for the communication with the Modbus-TCP slave (Default 502).
  - Unit ID: Modbus-TCP slave address.

- Configure the connection times:
  - Transmission delay(ms): waits for this time before sending the next telegram.

  - Adjustable range: 0 ... 20000;
  - Default: 0.

  - Response delay (ms): waits for this time before trying to receive the response from the sent telegram.

  - Adjustable range: 0 ... 20000;
  - Default: 0.

  - Timeout (ms): waits for this time before indicating that the response was not received.

  - Adjustable range: 100 ... 20000;
  - Default: 5000.

- Click on the **New connection...** button.
  - Select the slave address for the new connection according to the equipment configuration.

Index