SIEMENS

SIMATIC

Ladder Logic (LAD) for S7-300 and S7-400 Programming

Reference Manual

This manual is part of the documentation package with the order number:
6ES7810-4CA08-8BW1

Edition 03/2006
A5E00706949-01

Safety Guidelines

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

Danger
indicates that death or severe personal injury **will** result if proper precautions are not taken.

Warning
indicates that death or severe personal injury **may** result if proper precautions are not taken.

Caution
with a safety alert symbol indicates that minor personal injury can result if proper precautions are not taken.

Caution
without a safety alert symbol indicates that property damage can result if proper precautions are not taken.

Notice
indicates that an unintended result or situation can occur if the corresponding notice is not taken into account.

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

Qualified Personnel

The device/system may only be set up and used in conjunction with this documentation. Commissioning and operation of a device/system may only be performed by **qualified personnel.** Within the context of the safety notices in this documentation qualified persons are defined as persons who are authorized to commission, ground and label devices, systems and circuits in accordance with established safety practices and standards.

Prescribed Usage

Note the following:

Warning
This device and its components may only be used for the applications described in the catalog or the technical description, and only in connection with devices or components from other manufacturers which have been approved or recommended by Siemens.
Correct, reliable operation of the product requires proper transport, storage, positioning and assembly as well as careful operation and maintenance.

Trademarks

Disclaimer of Liability

Siemens AG
Automation and Drives
Postfach 4848
90437 NÜRNBERG
GERMANY

A5E00706949-01
03/2006

Preface

Purpose

This manual is your guide to creating user programs in the Ladder Logic (LAD) programming language.

This manual also includes a reference section that describes the syntax and functions of the language elements of Ladder Logic.

Basic Knowledge Required

The manual is intended for S7 programmers, operators, and maintenance/service personnel.

In order to understand this manual, general knowledge of automation technology is required.

In addition to, computer literacy and the knowledge of other working equipment similar to the PC (e.g. programming devices) under the operating systems MS Windows 2000 Professional, MS Windows XP Professional or MS Windows Server 2003 are required.

Scope of the Manual

This manual is valid for release 5.4 of the STEP 7 programming software package.

Compliance with IEC 1131-3

LAD corresponds to the “Ladder Logic” language defined in the International Electrotechnical Commission's standard IEC 1131-3. For further details, refer to the table of standards in the STEP 7 file NORM_TBL.WRI.

Requirements

To use this Ladder Logic manual effectively, you should already be familiar with the theory behind S7 programs which is documented in the online help for STEP 7. The language packages also use the STEP 7 standard software, so you should be familiar with handling this software and have read the accompanying documentation.

This manual is part of the documentation package "STEP 7 Reference".

The following table displays an overview of the STEP 7 documentation:

Documentation	Purpose	Order Number
STEP 7 Basic Information with • Working with STEP 7, Getting Started Manual • Programming with STEP 7 • Configuring Hardware and Communication Connections, STEP 7 • From S5 to S7, Converter Manual	Basic information for technical personnel describing the methods of implementing control tasks with STEP 7 and the S7-300/400 programmable controllers.	6ES7810-4CA08-8BW0
STEP 7 Reference with • Ladder Logic (LAD) / Function Block Diagram (FDB) / Statement List (STL) for S7-300/400 manuals • Standard and System Function for S7-300/400 Volume 1 and Volume 2	Provides reference information and describes the programming languages LAD, FBD and STL, and standard and system function extending the scope of the STEP 7 basic information.	6ES7810-4CA08-8BW1

Online Helps	Purpose	Order Number
Help on STEP 7	Basic information on programming and configuring hardware with STEP 7 in the form of an online help.	Part of the STEP 7 Standard software.
Reference helps on AWL/KOP/FUP Reference help on SFBs/SFCs Reference help on Organization Blocks	Context-sensitive reference information.	Part of the STEP 7 Standard software.

Online Help

The manual is complemented by an online help which is integrated in the software. This online help is intended to provide you with detailed support when using the software.

The help system is integrated in the software via a number of interfaces:

- The context-sensitive help offers information on the current context, for example, an open dialog box or an active window. You can open the context-sensitive help via the menu command **Help > Context-Sensitive Help**, by pressing F1 or by using the question mark symbol in the toolbar.
- You can call the general Help on STEP 7 using the menu command **Help > Contents** or the "Help on STEP 7" button in the context-sensitive help window.
- You can call the glossary for all STEP 7 applications via the "Glossary" button.

This manual is an extract from the "Help on Ladder Logic". As the manual and the online help share an identical structure, it is easy to switch between the manual and the online help.

Further Support

If you have any technical questions, please get in touch with your Siemens representative or responsible agent.

You will find your contact person at:

http://www.siemens.com/automation/partner

You will find a guide to the technical documentation offered for the individual SIMATIC Products and Systems here at:

http://www.siemens.com/simatic-tech-doku-portal

The online catalog and order system is found under:

http://mall.automation.siemens.com/

Training Centers

Siemens offers a number of training courses to familiarize you with the SIMATIC S7 automation system. Please contact your regional training center or our central training center in D 90327 Nuremberg, Germany for details:

Telephone: +49 (911) 895-3200.

Internet: http://www.sitrain.com

Technical Support

You can reach the Technical Support for all A&D products

- Via the Web formula for the Support Request
  http://www.siemens.com/automation/support-request
- Phone: + 49 180 5050 222
- Fax: + 49 180 5050 223

Additional information about our Technical Support can be found on the Internet pages http://www.siemens.com/automation/service

Service & Support on the Internet

In addition to our documentation, we offer our Know-how online on the internet at:

http://www.siemens.com/automation/service&support

where you will find the following:

- The newsletter, which constantly provides you with up-to-date information on your products.
- The right documents via our Search function in Service & Support.
- A forum, where users and experts from all over the world exchange their experiences.
- Your local representative for Automation & Drives.
- Information on field service, repairs, spare parts and more under "Services".

Contents

1 Bit Logic Instructions

1.1 Overview of Bit Logic Instructions

Description

Bit logic instructions work with two digits, 1 and 0. These two digits form the base of a number system called the binary system. The two digits 1 and 0 are called binary digits or bits. In the world of contacts and coils, a 1 indicates activated or energized, and a 0 indicates not activated or not energized.

The bit logic instructions interpret signal states of 1 and 0 and combine them according to Boolean logic. These combinations produce a result of 1 or 0 that is called the ”result of logic operation” (RLO).

The logic operations that are triggered by the bit logic instructions perform a variety of functions.

There are bit logic instructions to perform the following functions:

• ---| |---	Normally Open Contact (Address)
• ---| / |---	Normally Closed Contact (Address)
• ---(SAVE)	Save RLO into BR Memory
• XOR	Bit Exclusive OR
• ---( )	Output Coil
• ---(#)---	Midline Output
• ---|NOT|---	Invert Power Flow

The following instructions react to an RLO of 1:

• ---(S)	Set Coil
• ---(R)	Reset Coil
• SR	Set-Reset Flip Flop
• RS	Reset-Set Flip Flop

Other instructions react to a positive or negative edge transition to perform the following functions:

• ---(N)---	Negative RLO Edge Detection
• ---(P)---	Positive RLO Edge Detection
• NEG	Address Negative Edge Detection
• POS	Address Positive Edge Detection

- Immediate Read
- Immediate Write

1.2 ---| |--- Normally Open Contact (Address)

Symbol

<address>

---| |---

Parameter	Data Type	Memory Area	Description
<address>	BOOL	I, Q, M, L, D, T, C	Checked bit

Description

---| |--- (Normally Open Contact) is closed when the bit value stored at the specified **<address>** is equal to "1". When the contact is closed, ladder rail power flows across the contact and the result of logic operation (RLO) = "1".

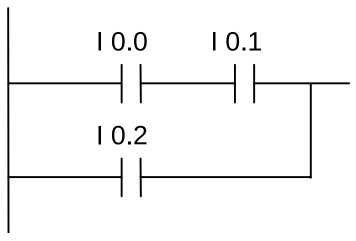
Otherwise, if the signal state at the specified **<address>** is "0", the contact is open. When the contact is open, power does not flow across the contact and the result of logic operation (RLO) = "0".

When used in series, **---| |---** is linked to the RLO bit by AND logic. When used in parallel, it is linked to the RLO by OR logic.

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	-	-	-	-	-	X	X	X	1

Example



Power flows if one of the following conditions exists:

The signal state is "1" at inputs I0.0 and I0.1

Or the signal state is "1" at input I0.2

1.3 ---| / |--- Normally Closed Contact (Address)

Symbol

<address>

---| / |---

Parameter	Data Type	Memory Area	Description
<address>	BOOL	I, Q, M, L, D, T, C	Checked bit

Description

---| / |--- (Normally Closed Contact) is closed when the bit value stored at the specified **<address>** is equal to "0". When the contact is closed, ladder rail power flows across the contact and the result of logic operation (RLO) = "1".

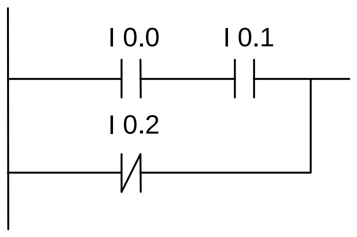
Otherwise, if the signal state at the specified **<address>** is "1", the contact is opened. When the contact is opened, power does not flow across the contact and the result of logic operation (RLO) = "0".

When used in series, ---| / |--- is linked to the RLO bit by AND logic. When used in parallel, it is linked to the RLO by OR logic.

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	-	-	-	-	-	X	X	X	1

Example



Power flows if one of the following conditions exists:

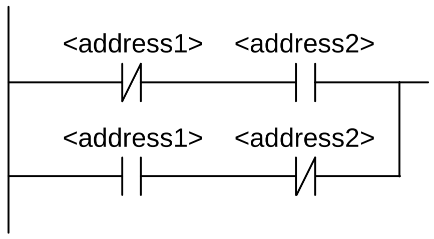
The signal state is "1" at inputs I0.0 and I0.1

Or the signal state is "1" at input I0.2

1.4 XOR Bit Exclusive OR

For the XOR function, a network of normally open and normally closed contacts must be created as shown below.

Symbols

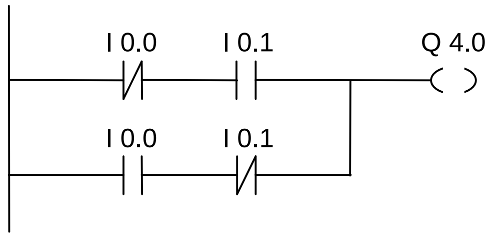


Parameter	Data Type	Memory Area	Description
<address1>	BOOL	I, Q, M, L, D, T, C	Scanned bit
<address2	BOOL	I, Q, M, L, D, T, C	Scanned bit

Description

XOR (Bit Exclusive OR) creates an RLO of "1" if the signal state of the two specified bits is different.

Example



The output Q4.0 is "1" if (I0.0 = "0" AND I0.1 = "1") OR (I0.0 = "1" AND I0.1 = "0").

1.5 --|NOT|-- Invert Power Flow

Symbol

---|NOT|---

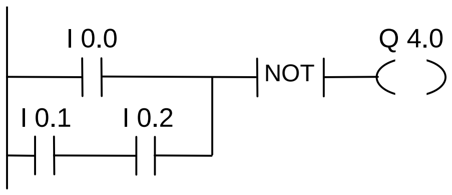
Description

---|NOT|--- (Invert Power Flow) negates the RLO bit.

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	-	-	-	-	-	-	1	X	-

Example



The signal state of output Q4.0 is "0" if one of the following conditions exists:

The signal state is "1" at input I0.0

Or the signal state is "1" at inputs I0.1 and I0.2.

1.6 ---( ) Output Coil

Symbol

<address>

---( )

Parameter	Data Type	Memory Area	Description
<address>	BOOL	I, Q, M, L, D	Assigned bit

Description

---( ) (Output Coil) works like a coil in a relay logic diagram. If there is power flow to the coil (RLO = 1), the bit at location **<address>** is set to "1". If there is no power flow to the coil (RLO = 0), the bit at location **<address>** is set to "0". An output coil can only be placed at the right end of a ladder rung. Multiple output elements (max. 16) are possible (see example). A negated output can be created by using the ---|NOT|--- (invert power flow) element.

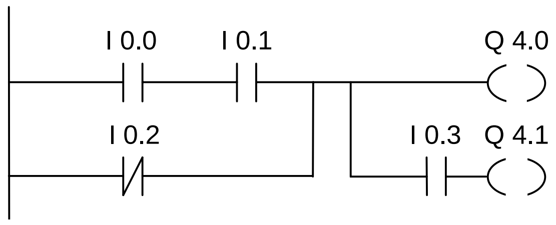
MCR (Master Control Relay) dependency

MCR dependency is activated only if an output coil is placed inside an active MCR zone. Within an activated MCR zone, if the MCR is on and there is power flow to an output coil; the addressed bit is set to the current status of power flow. If the MCR is off, a logic "0" is written to the specified address regardless of power flow status.

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	-	-	-	-	-	0	X	-	0

Example



The signal state of output Q4.0 is "1" if one of the following conditions exists:

The signal state is "1" at inputs I0.0 and I0.1

Or the signal state is "0" at input I0.2.

The signal state of output Q4.1 is "1" if one of the following conditions exists:

The signal state is "1" at inputs I0.0 and I0.1

Or the signal state is "0" at input I0.2 and "1" at input I0.3

If the example rungs are within an activated MCR zone:

When MCR is on, Q4.0 and Q4.1 are set according to power flow status as described above.

When MCR is off (=0), Q4.0 and Q4.1 are reset to 0 regardless of power flow.

1.7 ---(#)--- Midline Output

Symbol

<address>

---(#)---

Parameter	Data Type	Memory Area	Description
<address>	BOOL	I, Q, M, *L, D	Assigned bit

* An L area address can only be used if it is declared TEMP in the variable declaration table of a logic block (FC, FB, OB).

Description

---(#)--- (Midline Output) is an intermediate assigning element which saves the RLO bit (power flow status) to a specified **<address>**. The midline output element saves the logical result of the preceding branch elements. In series with other contacts, ---(#)--- is inserted like a contact. A ---(#)--- element may never be connected to the power rail or directly after a branch connection or at the end of a branch. A negated ---(#)--- can be created by using the ---|NOT|--- (invert power flow) element.

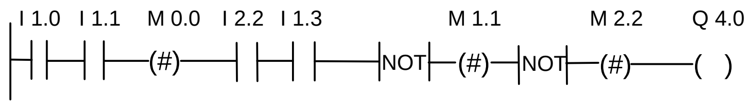
MCR (Master Control Relay) dependency

MCR dependency is activated only if a midline output coil is placed inside an active MCR zone. Within an activated MCR zone, if the MCR is on and there is power flow to a midline output coil; the addressed bit is set to the current status of power flow. If the MCR is off, a logic "0" is written to the specified address regardless of power flow status.

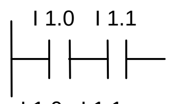
Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	-	-	-	-	-	0	X	-	1

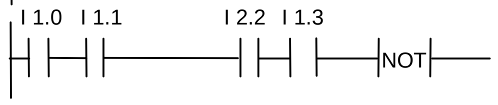
Example



M 0.0 has the RLO



M 1.1 has the RLO



M 2.2 has the RLO of the entire bit logic combination

1.8 ---(R) Reset Coil

Symbol

<address>

---(R)

Parameter	Data Type	Memory Area	Description
<address>	BOOL	I, Q, M, L, D, T, C	Reset bit

Description

---(R) (Reset Coil) is executed only if the RLO of the preceding instructions is "1" (power flows to the coil). If power flows to the coil (RLO is "1"), the specified **<address>** of the element is reset to "0". A RLO of "0" (no power flow to the coil) has no effect and the state of the element's specified address remains unchanged. The **<address>** may also be a timer (T no.) whose timer value is reset to "0" or a counter (C no.) whose counter value is reset to "0".

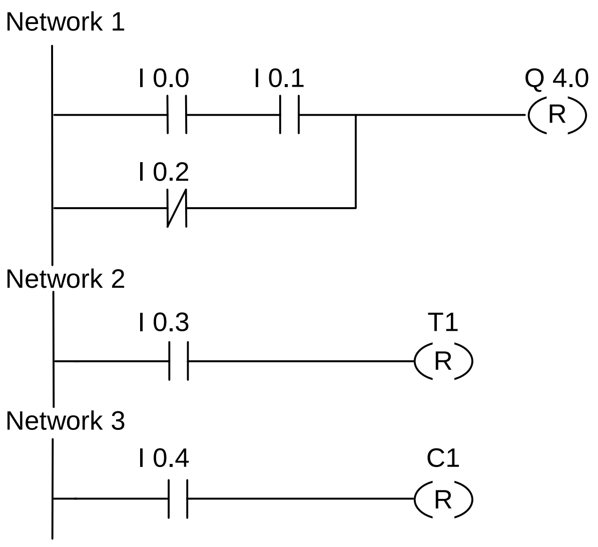
MCR (Master Control Relay) dependency

MCR dependency is activated only if a reset coil is placed inside an active MCR zone. Within an activated MCR zone, if the MCR is on and there is power flow to a reset coil; the addressed bit is reset to the "0" state. If the MCR is off, the current state of the element's specified address remains unchanged regardless of power flow status.

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	-	-	-	-	-	0	X	-	0

Example



The signal state of output Q4.0 is reset to "0" if one of the following conditions exists:

The signal state is "1" at inputs I0.0 and I0.1

Or the signal state is "0" at input I0.2.

If the RLO is "0", the signal state of output Q4.0 remains unchanged.

The signal state of timer T1 is only reset if:

the signal state is "1" at input I0.3.

The signal state of counter C1 is only reset if:

the signal state is "1" at input I0.4.

If the example rungs are within an activated MCR zone:

When MCR is on, Q4.0, T1, and C1 are reset as described above.

When MCR is off, Q4.0, T1, and C1 are left unchanged regardless of RLO state (power flow status).

1.9 ---(S) Set Coil

Symbol

<address>

---(S)

Parameter	**Data Type**	**Memory Area**	**Description**
<address>	BOOL	I, Q, M, L, D	Set bit

Description

---(S) (Set Coil) is executed only if the RLO of the preceding instructions is "1" (power flows to the coil). If the RLO is "1" the specified **<address>** of the element is set to "1".

An RLO = 0 has no effect and the current state of the element's specified address remains unchanged.

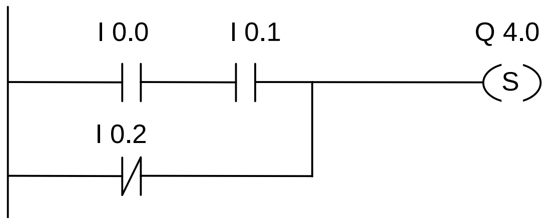
MCR (Master Control Relay) dependency

MCR dependency is activated only if a set coil is placed inside an active MCR zone. Within an activated MCR zone, if the MCR is on and there is power flow to a set coil; the addressed bit is set to the "1" state. If the MCR is off, the current state of the element's specified address remains unchanged regardless of power flow status.

Status word

	BR	**CC 1**	**CC 0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**/FC**
writes:	-	-	-	-	-	0	X	-	0

Example



The signal state of output Q4.0 is "1" if one of the following conditions exists:

The signal state is "1" at inputs I0.0 and I0.1

Or the signal state is "0" at input I0.2.

If the RLO is "0", the signal state of output Q4.0 remains unchanged.

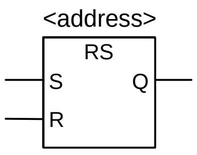
If the example rungs are within an activated MCR zone:

When MCR is on, Q4.0 is set as described above.

When MCR is off, Q4.0 is left unchanged regardless of RLO state (power flow status).

1.10 RS Reset-Set Flip Flop

Symbol



Parameter	Data Type	Memory Area	Description
<address>	BOOL	I, Q, M, L, D	Set or reset bit
S	BOOL	I, Q, M, L, D	Enabled reset instruction
R	BOOL	I, Q, M, L, D	Enabled reset instruction
Q	BOOL	I, Q, M, L, D	Signal state of <address>

Description

RS (Reset-Set Flip Flop) is reset if the signal state is "1" at the R input, and "0" at the S input. Otherwise, if the signal state is "0" at the R input and "1" at the S input, the flip flop is set. If the RLO is "1" at both inputs, the order is of primary importance. The RS flip flop executes first the reset instruction then the set instruction at the specified **<address>**, so that this address remains set for the remainder of program scanning.

The S (Set) and R (Reset) instructions are executed only when the RLO is "1". RLO "0" has no effect on these instructions and the address specified in the instruction remains unchanged.

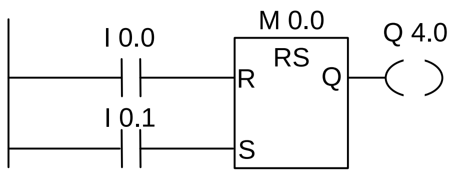
MCR (Master Control Relay) dependency

MCR dependency is activated only if a RS flip flop is placed inside an active MCR zone. Within an activated MCR zone, if the MCR is on, the addressed bit is reset to "0" or set to "1" as described above. If the MCR is off, the current state of the specified address remains unchanged regardless of input states.

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	-	-	-	-	-	x	x	x	1

Example



If the signal state is "1" at input I0.0 and "0" at I0.1, memory bit M0.0 is set and output Q4.0 is "0". Otherwise, if the signal state at input I0.0 is "0" and at I0.1 is "1", memory bit M0.0 is reset and output Q4.0 is "1". If both signal states are "0", nothing is changed. If both signal states are "1", the set instruction dominates because of the order; M0.0 is set and Q4.0 is "1".

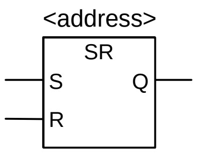
If the example is within an activated MCR zone:

When MCR is on, Q4.0 is reset or set as described above.

When MCR is off, Q4.0 is left unchanged regardless of input states.

1.11 SR Set-Reset Flip Flop

Symbol



Parameter	Data Type	Memory Area	Description
<address>	BOOL	I, Q, M, L, D	Set or reset bit
S	BOOL	I, Q, M, L, D	Enable set instruction
R	BOOL	I, Q, M, L, D	Enable reset instruction
Q	BOOL	I, Q, M, L, D	Signal state of <address>

Description

SR (Set-Reset Flip Flop) is set if the signal state is "1" at the S input, and "0" at the R input. Otherwise, if the signal state is "0" at the S input and "1" at the R input, the flip flop is reset. If the RLO is "1" at both inputs, the order is of primary importance. The SR flip flop executes first the set instruction then the reset instruction at the specified **<address>,** so that this address remains reset for the remainder of program scanning.

The S (Set) and R (Reset) instructions are executed only when the RLO is "1". RLO "0" has no effect on these instructions and the address specified in the instruction remains unchanged.

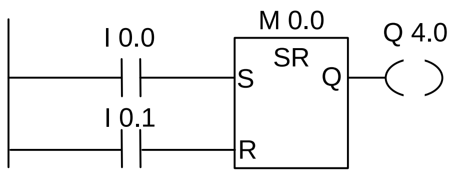
MCR (Master Control Relay) dependency

MCR dependency is activated only if a SR flip flop is placed inside an active MCR zone. Within an activated MCR zone, if the MCR is on ; the addressed bit is set to "1" or reset to "0" as described above. If the MCR is off, the current state of the specified address remains unchanged regardless of input states.

Status word

	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
writes:	-	-	-	-	-	x	x	x	1

Example



If the signal state is "1" at input I0.0 and "0" at I0.1, memory bit M0.0 is set and output Q4.0 is "1". Otherwise, if the signal state at input I0.0 is "0" and at I0.1 is "1", memory bit M0.0 is reset and output Q4.0 is "0". If both signal states are "0", nothing is changed. If both signal states are "1", the reset instruction dominates because of the order; M0.0 is reset and Q4.0 is "0".

If the example is within an activated MCR zone:

When MCR is on, Q4.0 is set or reset as described above.

When MCR is off, Q4.0 is left unchanged regardless of input states.

1.12 ---(N)--- Negative RLO Edge Detection

Symbol

<address>

---(N)

Parameter	Data Type	Memory Area	Description
<address>	BOOL	I, Q, M, L, D	Edge memory bit, storing the previous signal state of RLO

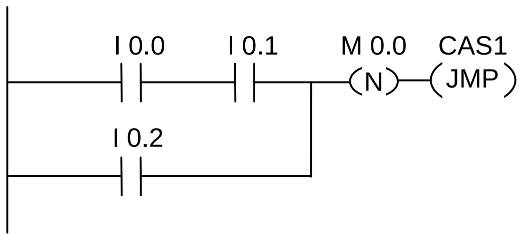
Description

---(N)--- (Negative RLO Edge Detection) detects a signal change in the address from "1" to "0" and displays it as RLO = "1" after the instruction. The current signal state in the RLO is compared with the signal state of the address, the edge memory bit. If the signal state of the address is "1" and the RLO was "0" before the instruction, the RLO will be "1" (pulse) after this instruction, and "0" in all other cases. The RLO prior to the instruction is stored in the address.

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	-	-	-	-	-	0	x	x	1

Example



The edge memory bit M0.0 saves the old RLO state. When there is a signal change at the RLO from "1" to "0", the program jumps to label CAS1.

1.13 ---(P)--- Positive RLO Edge Detection

Symbol

<address>

---(P)---

Parameter	Data Type	Memory Area	Description
<address>	BOOL	I, Q, M, L, D	Edge memory bit, storing the previous signal state of RLO

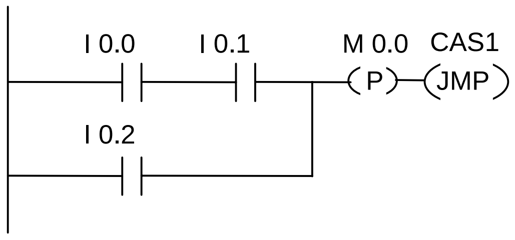
Description

---(P)--- (Positive RLO Edge Detection) detects a signal change in the address from "0" to "1" and displays it as RLO = "1" after the instruction. The current signal state in the RLO is compared with the signal state of the address, the edge memory bit. If the signal state of the address is "0" and the RLO was "1" before the instruction, the RLO will be "1" (pulse) after this instruction, and "0" in all other cases. The RLO prior to the instruction is stored in the address.

Status word

	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
writes:	-	-	-	-	-	0	X	X	1

Example



The edge memory bit M0.0 saves the old RLO state. When there is a signal change at the RLO from "0" to "1", the program jumps to label CAS1.

1.14 ---(SAVE) Save RLO into BR Memory

Symbol

---(SAVE)

Description

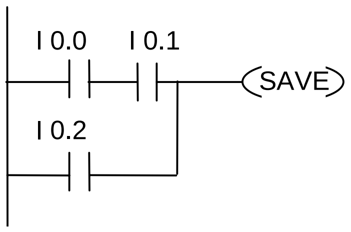
---(SAVE) (Save RLO into BR Memory) saves the RLO to the BR bit of the status word. The first check bit /FC is not reset. For this reason, the status of the BR bit is included in the AND logic operation in the next network.

For the instruction "SAVE" (LAD, FBD, STL), the following applies and not the recommended use specified in the manual and online help:
We do not recommend that you use SAVE and then check the BR bit in the same block or in subordinate blocks, because the BR bit can be modified by many instructions occurring inbetween. It is advisable to use the SAVE instruction before exiting a block, since the ENO output (= BR bit) is then set to the value of the RLO bit and you can then check for errors in the block.

Status word

	BR	**CC 1**	**CC 0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**/FC**
writes:	X	-	-	-	-	-	-	-	-

Example

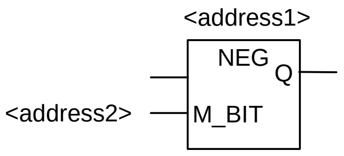


The status of the rung (=RLO) is saved to the BR bit.

BR Binary Result Bit (Status Word, Bit 8)

1.15 NEG Address Negative Edge Detection

Symbol



Parameter	Data Type	Memory Area	Description
<address1>	BOOL	I, Q, M, L, D	Scanned signal
<address2>	BOOL	I, Q, M, L, D	M_BIT edge memory bit, storing the previous signal state of <address1>
Q	BOOL	I, Q, M, L, D	One shot output

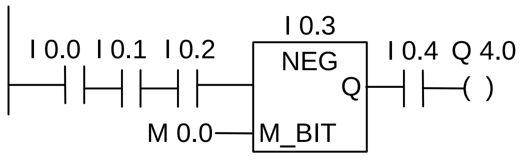
Description

NEG (Address Negative Edge Detection) compares the signal state of **<address1>** with the signal state from the previous scan, which is stored in **<address2>**. If the current RLO state is "1" and the previous state was "0" (detection of rising edge), the RLO bit will be "1" after this instruction.

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	-	-	-	-	-	x	1	x	1

Example

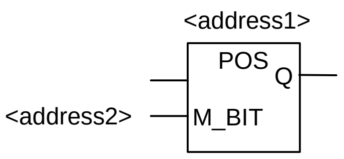


The signal state at output Q4.0 is "1" if the following conditions exist:

- The signal state is "1" at inputs I0.0 and I0.1 and I0.2
- And there is a negative edge at input I0.3
- And the signal state is "1" at input I0.4

1.16 POS Address Positive Edge Detection

Symbol



Parameter	Data Type	Memory Area	Description
<address1>	BOOL	I, Q, M, L, D	Scanned signal
<address2>	BOOL	I, Q, M, L, D	M_BIT edge memory bit, storing the previous signal state of <address1>
Q	BOOL	I, Q, M, L, D	One shot output

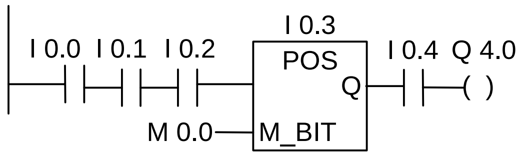
Description

POS (Address Positive Edge Detection) compares the signal state of **<address1>** with the signal state from the previous scan, which is stored in **<address2>**. If the current RLO state is "1" and the previous state was "0" (detection of rising edge), the RLO bit will be "1" after this instruction.

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	-	-	-	-	-	x	1	x	1

Example



The signal state at output Q4.0 is "1" if the following conditions exist:

- The signal state is "1" at inputs I0.0 and I0.1 and I0.2
- And there is a positive edge at input I0.3
- And the signal state is "1" at input I0.4

1.17 Immediate Read

Description

For the Immediate Read function, a network of symbols must be created as shown in the example below.

For time-critical applications, the current state of a digital input may be read faster than the normal case of once per OB1 scan cycle. An Immediate Read gets the state of a digital input from an input module at the time the Immediate Read rung is scanned. Otherwise, you must wait for the end of the next OB1 scan cycle when the I memory area is updated with the P memory state.

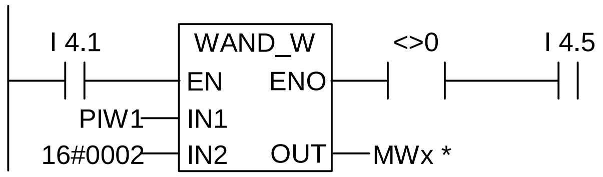
To perform an immediate read of an input (or inputs) from an input module, use the peripheral input (PI) memory area instead of the input (I) memory area. The peripheral input memory area can be read as a byte, a word, or a double word. Therefore, a single digital input cannot be read via a contact (bit) element.

To conditionally pass voltage depending on the status of an immediate input:

1. A word of PI memory that contains the input data of concern is read by the CPU.
2. The word of PI memory is then ANDed with a constant that yields a non-zero result if the input bit is on ("1").
3. The accumulator is tested for non-zero condition.

Example

Ladder Network with Immediate Read of Peripheral Input I1.1



***** MWx has to be specified in order to be able to store the network. x may be any permitted number.

Description of WAND_W instruction:

PIW1	0000000000101010
W#16#0002	0000000000000010
Result	0000000000000010

In this example immediate input I1.1 is in series with I4.1 and I4.5.

The word PIW1 contains the immediate status of I1.1. PIW1 is ANDed with W#16#0002. The result is not equal to zero if I1.1 (second bit) in PB1 is true ("1"). The contact A<>0 passes voltage if the result of the WAND_W instruction is not equal to zero.

1.18 Immediate Write

Description

For the Immediate Write function, a network of symbols must be created as shown in the example below.

For time-critical applications, the current state of a digital output may have to be sent to an output module faster than the normal case of once at the end of the OB1 scan cycle. An Immediate Write writes to a digital output to a input module at the time the Immediate Write rung is scanned. Otherwise, you must wait for the end of the next OB1 scan cycle when the Q memory area is updated with the P memory state.

To perform an immediate write of an output (or outputs) to an output module, use the peripheral output (PQ) memory area instead of the output (Q) memory area. The peripheral output memory area can be read as a byte, a word, or a double word. Therefore, a single digital output cannot be updated via a coil element. To write the state of a digital output to an output module immediately, a byte, word, or double word of Q memory that contains the relevant bit is conditionally copied to the corresponding PQ memory (direct output module addresses).

Caution

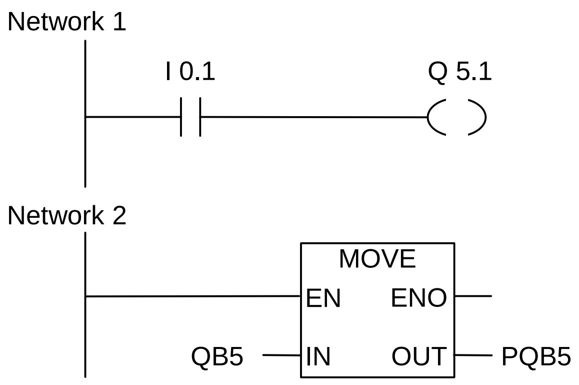
- Since the entire byte of Q memory is written to an output module, all outputs bits in that byte are updated when the immediate output is performed.
- If an output bit has intermediate states (1/0) occurring throughout the program that should not be sent to the output module, Immediate Writes could cause dangerous conditions (transient pulses at outputs) to occur.
- As a general design rule, an external output module should only be referenced once in a program as a coil. If you follow this design rule, most potential problems with immediate outputs can be avoided.

Example

Ladder network equivalent of Immediate Write to peripheral digital output module 5, channel 1.

The bit states of the addressed output Q byte (QB5) are either modified or left unchanged. Q5.1 is assigned the signal state of I0.1 in network 1. QB5 is copied to the corresponding direct peripheral output memory area (PQB5).

The word PIW1 contains the immediate status of I1.1. PIW1 is ANDed with W#16#0002. The result is not equal to zero if I1.1 (second bit) in PB1 is true ("1"). The contact A<>0 passes voltage if the result of the WAND_W instruction is not equal to zero.



In this example Q5.1 is the desired immediate output bit.

The byte PQB5 contains the immediate output status of the bit Q5.1.

The other 7 bits in PQB5 are also updated by the MOVE (copy) instruction.

2 Comparison Instructions

2.1 Overview of Comparison Instructions

Description

IN1 and IN2 are compared according to the type of comparison you choose:

==	IN1 is equal to IN2
<>	IN1 is not equal to IN2
>	IN1 is greater than IN2
<	IN1 is less than IN2
>=	IN1 is greater than or equal to IN2
<=	IN1 is less than or equal to IN2

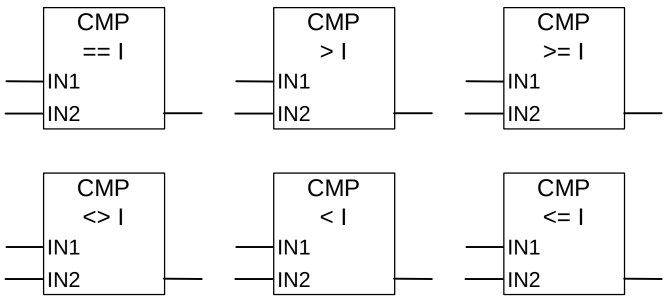
If the comparison is true, the RLO of the function is "1". It is linked to the RLO of a rung network by AND if the compare element is used in series, or by OR if the box is used in parallel.

The following comparison instructions are available:

- CMP ? I Compare Integer
- CMP ? D Compare Double Integer
- CMP ? R Compare Real

2.2 CMP ? I Compare Integer

Symbols



Parameter	Data Type	Memory Area	Description
box input	BOOL	I, Q, M, L, D	Result of the previous logic operation
box output	BOOL	I, Q, M, L, D	Result of the comparison, is only processed further if the RLO at the box input = 1
IN1	INT	I, Q, M, L, D or constant	First value to compare
IN2	INT	I, Q, M, L, D or constant	Second value to compare

Description

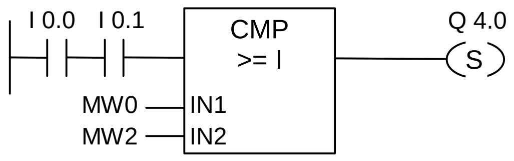
CMP ? I (Compare Integer) can be used like a normal contact. It can be located at any position where a normal contact could be placed. IN1 and IN2 are compared according to the type of comparison you choose.

If the comparison is true, the RLO of the function is "1". It is linked to the RLO of the whole rung by AND if the box is used in series, or by OR if the box is used in parallel.

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	x	x	x	0	-	0	x	x	1

Example

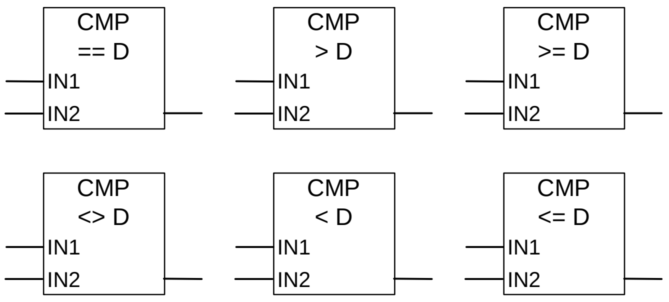


Output Q4.0 is set if the following conditions exist:

- There is a signal state of "1" at inputs I0.0 and at I0.1
- AND MW0 >= MW2

2.3 CMP ? D Compare Double Integer

Symbols



Parameter	Data Type	Memory Area	Description
box input	BOOL	I, Q, M, L, D	Result of the previous logic operation
box output	BOOL	I, Q, M, L, D	Result of the comparison, is only processed further if the RLO at the box input = 1
IN1	DINT	I, Q, M, L, D or constant	First value to compare
IN2	DINT	I, Q, M, L, D or constant	Second value to compare

Description

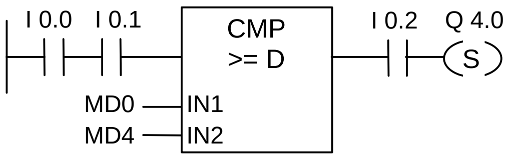
CMP ? D (Compare Double Integer) can be used like a normal contact. It can be located at any position where a normal contact could be placed. IN1 and IN2 are compared according to the type of comparison you choose.

If the comparison is true, the RLO of the function is "1". It is linked to the RLO of a rung network by AND if the compare element is used in series, or by OR if the box is used in parallel.

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	x	x	x	0	-	0	x	x	1

Example

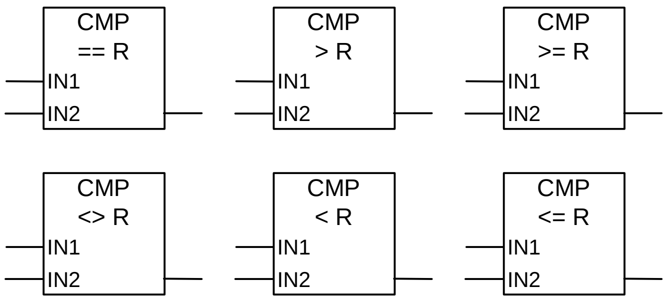


Output Q4.0 is set if the following conditions exist:

- There is a signal state of "1" at inputs I0.0 and at I0.1
- And MD0 >= MD4
- And there is a signal state of"1" at input I0.2

2.4 CMP ? R Compare Real

Symbols



Parameter	Data Type	Memory Area	Description
box input	BOOL	I, Q, M, L, D	Result of the previous logic operation
box output	BOOL	I, Q, M, L, D	Result of the comparison, is only processed further if the RLO at the box input = 1
IN1	REAL	I, Q, M, L, D or constant	First value to compare
IN2	REAL	I, Q, M, L, D or constant	Second value to compare

Description

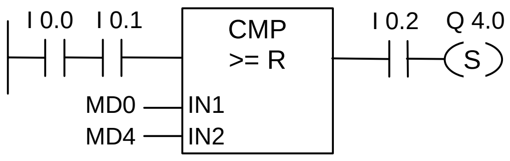
CMP ? R (Compare Real) can be used like a normal contact. It can be located at any position where a normal contact could be placed. IN1 and IN2 are compared according to the type of comparison you choose.

If the comparison is true, the RLO of the function is "1". It is linked to the RLO of the whole rung by AND if the box is used in series, or by OR if the box is used in parallel.

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	x	x	x	x	x	0	x	x	1

Example



Output Q4.0 is set if the following conditions exist:

- There is a signal state of "1" at inputs I0.0 and at I0.1
- And MD0 >= MD4
- And there is a signal state of"1" at input I0.2

3 Conversion Instructions

3.1 Overview of Conversion Instructions

Description

The conversion instructions read the contents of the parameters IN and convert these or change the sign. The result can be queried at the parameter OUT.

The following conversion instructions are available:

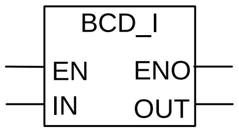
• BCD_I	BCD to Integer
• I_BCD	Integer to BCD
• BCD_DI	BCD to Double Integer
• I_DINT	Integer to Double Integer
• DI_BCD	Double Integer to BCD
• DI_REAL	Double Integer to Floating-Point

• INV_I	Ones Complement Integer
• INV_DI	Ones Complement Double Integer
• NEG_I	Twos Complement Integer
• NEG_DI	Twos Complement Double Integer

• NEG_R	Negate Floating-Point Number
• ROUND	Round to Double Integer
• TRUNC	Truncate Double Integer Part
• CEIL	Ceiling
• FLOOR	Floor

3.2 BCD_I BCD to Integer

Symbol



Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, L, D	Enable input
ENO	BOOL	I, Q, M, L, D	Enable output
IN	WORD	I, Q, M, L, D	BCD number
OUT	INT	I, Q, M, L, D	Integer value of BCD number

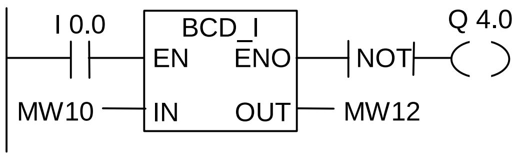
Description

BCD_I (Convert BCD to Integer) reads the contents of the IN parameter as a three-digit, BCD coded number (+/- 999) and converts it to an integer value (16-bit). The integer result is output by the parameter OUT. ENO always has the same signal state as EN.

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	1	-	-	-	-	0	1	1	1

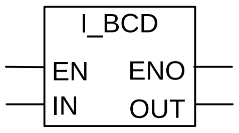
Example



If input I0.0 is "1" , then the content of MW10 is read as a three-digit BCD coded number and converted to an integer. The result is stored in MW12. The output Q4.0 is "1" if the conversion is not executed (ENO = EN = 0).

3.3 I_BCD Integer to BCD

Symbol



Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, L, D	Enable input
ENO	BOOL	I, Q, M, L, D	Enable output
IN	INT	I, Q, M, L, D	Integer number
OUT	WORD	I, Q, M, L, D	BCD value of integer number

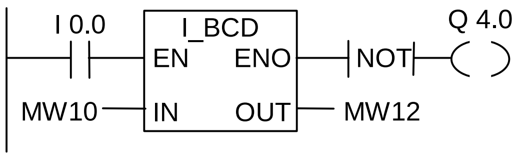
Description

I_BCD (Convert Integer to BCD) reads the content of the IN parameter as an integer value (16-bit) and converts it to a three-digit BCD coded number (+/- 999). The result is output by the parameter OUT. If an overflow occurred, ENO will be "0".

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	x	-	-	x	x	0	x	x	1

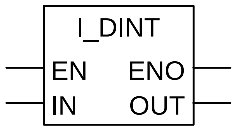
Example



If I0.0 is "1", then the content of MW10 is read as an integer and converted to a three-digit BCD coded number. The result is stored in MW12. The output Q4.0 is "1" if there was an overflow, or the instruction was not executed (I0.0 = 0).

3.4 I_DINT Integer to Double Integer

Symbol



Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, L, D	Enable input
ENO	BOOL	I, Q, M, L, D	Enable output
IN	INT	I, Q, M, L, D	Integer value to convert
OUT	DINT	I, Q, M, L, D	Double integer result

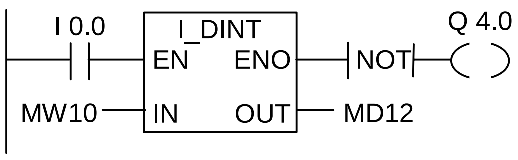
Description

I_DINT (Convert Integer to Double Integer) reads the content of the IN parameter as an integer (16-bit) and converts it to a double integer (32-bit). The result is output by the parameter OUT. ENO always has the same signal state as EN.

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	1	-	-	-	-	0	1	1	1

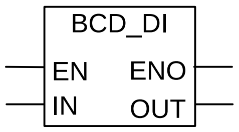
Example



If I0.0 is "1", then the content of MW10 is read as an integer and converted to a double integer. The result is stored in MD12. The output Q4.0 is "1" if the conversion is not executed (ENO = EN = 0).

3.5 BCD_DI BCD to Double Integer

Symbol



Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, L, D	Enable input
ENO	BOOL	I, Q, M, L, D	Enable output
IN	DWORD	I, Q, M, L, D	BCD number
OUT	DINT	I, Q, M, L, D	Double integer value of BCD number

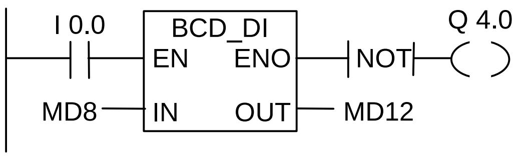
Description

BCD_DI (Convert BCD to Double Integer) reads the content of the IN parameter as a seven-digit, BCD coded number (+/- 9999999) and converts it to a double integer value (32-bit). The double integer result is output by the parameter OUT. ENO always has the same signal state as EN.

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	1	-	-	-	-	0	1	1	1

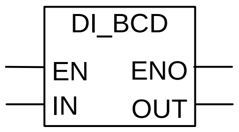
Example



If I0.0 is "1" , then the content of MD8 is read as a seven-digit BCD coded number and converted to a double integer. The result is stored in MD12. The output Q4.0 is "1" if the conversion is not executed (ENO = EN = 0).

3.6 DI_BCD Double Integer to BCD

Symbol



Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, L, D	Enable input
ENO	BOOL	I, Q, M, L, D	Enable output
IN	DINT	I, Q, M, L, D	Double integer number
OUT	DWORD	I, Q, M, L, D	BCD value of a double integer number

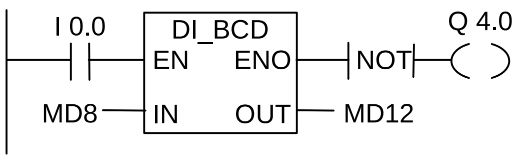
Description

DI_BCD (Convert Double Integer to BCD) reads the content of the IN parameter as a double integer (32-bit) and converts it to a seven-digit BCD coded number (+/- 9999999). The result is output by the parameter OUT. If an overflow occurred, ENO will be "0".

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	x	-	-	x	x	0	x	x	1

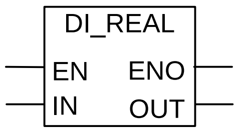
Example



If I0.0 is "1", then the content of MD8 is read as a double integer and converted to a seven-digit BCD number. The result is stored in MD12. The output Q4.0 is "1" if an overflow occurred, or the instruction was not executed (I0.0 = 0).

3.7 DI_REAL Double Integer to Floating-Point

Symbol



Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, L, D	Enable input
ENO	BOOL	I, Q, M, L, D	Enable output
IN	DINT	I, Q, M, L, D	Double integer value to convert
OUT	REAL	I, Q, M, L, D	Floating-point number result

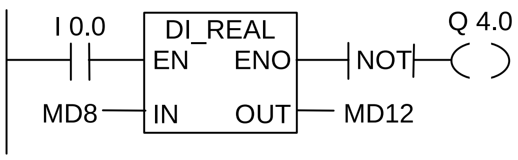
Description

DI_REAL (Convert Double Integer to Floating-Point) reads the content of the IN parameter as a double integer and converts it to a floating-point number. The result is output by the parameter OUT. ENO always has the same signal state as EN.

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	1	-	-	-	-	0	1	1	1

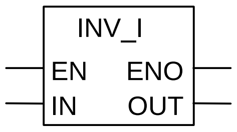
Example



If I0.0 is "1", then the content of MD8 is read as an double integer and converted to a floating-point number. The result is stored in MD12. The output Q4.0 is "1" if the conversion is not executed (ENO = EN = 0).

3.8 INV_I Ones Complement Integer

Symbol



Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, L, D	Enable input
ENO	BOOL	I, Q, M, L, D	Enable output
IN	INT	I, Q, M, L, D	Integer input value
OUT	INT	I, Q, M, L, D	Ones complement of the integer IN

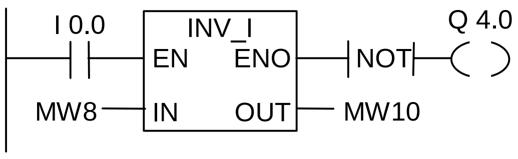
Description

INV_I (Ones Complement Integer) reads the content of the IN parameter and performs a Boolean XOR function with the hexadecimal mask W#16#FFFF. This instruction changes every bit to its opposite state. ENO always has the same signal state as EN.

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	1	-	-	-	-	0	1	1	1

Example



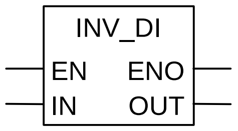
If I0.0 is "1", then every bit of MW8 is reversed, for example:

MW8 = 01000001 10000001 results in MW10 = 10111110 01111110.

The output Q4.0 is "1" if the conversion is not executed (ENO = EN = 0).

3.9 INV_DI Ones Complement Double Integer

Symbol



Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, L, D	Enable input
ENO	BOOL	I, Q, M, L, D	Enable output
IN	DINT	I, Q, M, L, D	Double integer input value
OUT	DINT	I, Q, M, L, D	Ones complement of the double integer IN

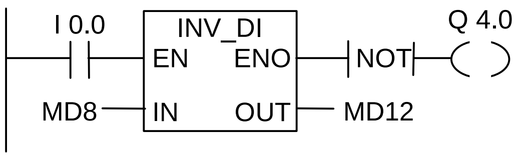
Description

INV_DI (Ones Complement Double Integer) reads the content of the IN parameter and performs a Boolean XOR function with the hexadecimal mask W#16#FFFF FFFF .This instruction changes every bit to its opposite state. ENO always has the same signal state as EN.

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	1	-	-	-	-	0	1	1	1

Example



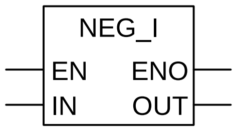
If I0.0 is "1", then every bit of MD8 is reversed, for example:

MD8 = F0FF FFF0 results in MD12 = 0F00 000F.

The output Q4.0 is "1" if the conversion is not executed (ENO = EN = 0).

3.10 NEG_I Twos Complement Integer

Symbol



Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, L, D	Enable input
ENO	BOOL	I, Q, M, L, D	Enable output
IN	INT	I, Q, M, L, D	Integer input value
OUT	INT	I, Q, M, L, D	Twos complement of integer IN

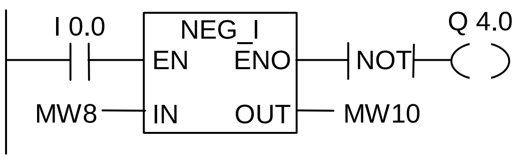
Description

NEG_I (Twos Complement Integer) reads the content of the IN parameter and performs a twos complement instruction. The twos complement instruction is equivalent to multiplication by (-1) and changes the sign (for example: from a positive to a negative value). ENO always has the same signal state as EN with the following exception: if the signal state of EN = 1 and an overflow occurs, the signal state of ENO = 0.

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	x	x	x	x	x	0	x	x	1

Example



If I0.0 is "1", then the value of MW8 with the opposite sign is output by the OUT parameter to MW10.

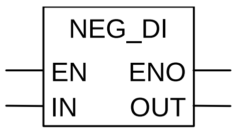
MW8 = + 10 results in MW10 = - 10.

The output Q4.0 is "1" if the conversion is not executed (ENO = EN = 0).

If the signal state of EN = 1 and an overflow occurs, the signal state of ENO = 0.

3.11 NEG_DI Twos Complement Double Integer

Symbol



Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, L, D	Enable input
ENO	BOOL	I, Q, M, L, D	Enable output
IN	DINT	I, Q, M, L, D	Double integer input value
OUT	DINT	I, Q, M, L, D	Twos complement of IN value

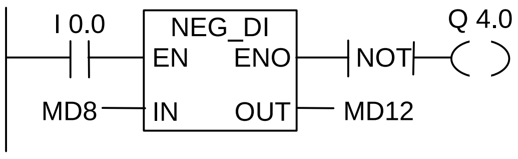
Description

NEG_DI (Twos Complement Double Integer) reads the content of the IN parameter and performs a twos complement instruction. The twos complement instruction is equivalent to multiplication by (-1) and changes the sign (for example: from a positive to a negative value). ENO always has the same signal state as EN with the following exception: if the signal state of EN = 1 and an overflow occurs, the signal state of ENO = 0.

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	x	x	x	x	x	0	x	x	1

Example



If I0.0 is "1", then the value of MD8 with the opposite sign is output by the OUT parameter to MD12.

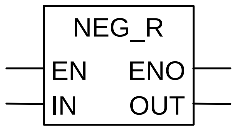
MD8 = + 1000 results in MD12 = - 1000.

The output Q4.0 is "1" if the conversion is not executed (ENO = EN = 0).

If the signal state of EN = 1 and an overflow occurs, the signal state of ENO = 0.

3.12 NEG_R Negate Floating-Point Number

Symbol



Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, L, D	Enable input
ENO	BOOL	I, Q, M, L, D	Enable output
IN	REAL	I, Q, M, L, D	Floating-point number input value
OUT	REAL	I, Q, M, L, D	Floating-point number IN with negative sign

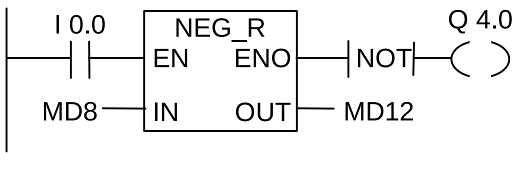
Description

NEG_R (Negate Floating-Point) reads the contents of the IN parameter and changes the sign. The instruction is equivalent to multiplication by (-1) and changes the sign (for example: from a positive to a negative value). ENO always has the same signal state as EN.

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	x	-	-	-	-	0	x	x	1

Example



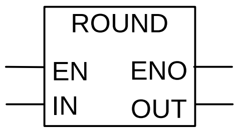
If I0.0 is "1", then the value of MD8 with the opposite sign is output by the OUT parameter to MD12.

MD8 = + 6.234 results in MD12 = - 6.234.

The output Q4.0 is "1" if the conversion is not executed (ENO = EN = 0).

3.13 ROUND Round to Double Integer

Symbol



Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, L, D	Enable input
ENO	BOOL	I, Q, M, L, D	Enable output
IN	REAL	I, Q, M, L, D	Value to round
OUT	DINT	I, Q, M, L, D	IN rounded to nearest whole number

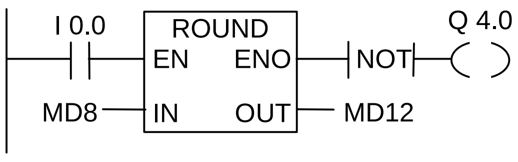
Description

ROUND (Round Double Integer) reads the content of the IN parameter as a floating-point number and converts it to a double integer (32-bit). The result is the closest integer number ("Round to nearest"). If the floating-point number lies between two integers, the even number is returned. The result is output by the parameter OUT. If an overflow occurred ENO will be "0".

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	x	-	-	x	x	0	x	x	1

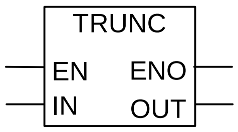
Example



If I0.0 is "1", then the content of MD8 is read as a floating-point number and converted to the closest double integer. The result of this "Round to nearest" function is stored in MD12. The output Q4.0 is "1" if an overflow occurred or the instruction was not executed (I0.0 = 0).

3.14 TRUNC Truncate Double Integer Part

Symbol



Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, L, D	Enable input
ENO	BOOL	I, Q, M, L, D	Enable output
IN	REAL	I, Q, M, L, D	Floating-point value to convert
OUT	DINT	I, Q, M, L, D	Whole number part of IN value

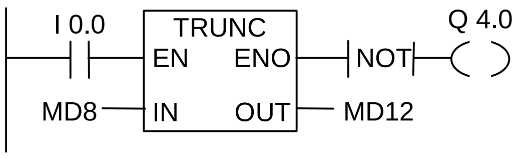
Description

TRUNC (Truncate Double Integer) reads the content of the IN parameter as a floating-point number and converts it to a double integer (32-bit). The double integer result of the ("Round to zero mode") is output by the parameter OUT. If an overflow occurred, ENO will be "0".

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	x	-	-	x	x	0	x	x	1

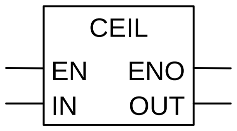
Example



If I0.0 is "1", then the content of MD8 is read as a real number and converted to a double integer. The integer part of the floating-point number is the result and is stored in MD12. The output Q4.0 is "1" if an overflow occurred, or the instruction was not executed (I0.0 = 0).

3.15 CEIL Ceiling

Symbol



Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, L, D	Enable input
ENO	BOOL	I, Q, M, L, D	Enable output
IN	REAL	I, Q, M, L, D	Floating-point value to convert
OUT	DINT	I, Q, M, L, D	Lowest greater double integer

Description

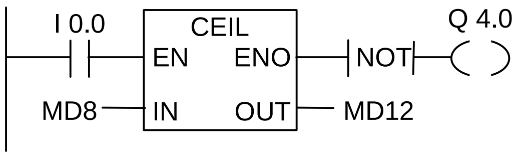
CEIL (Ceiling) reads the contents of the IN parameter as a floating-point number and converts it to a double integer (32-bit). The result is the lowest integer which is greater than the floating-point number ("Round to + infinity"). If an overflow occurs, ENO will be "0".

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes*:	X	-	-	X	X	0	X	X	1
writes**:	0	-	-	-	-	0	0	0	1

* Function is executed (EN = 1)
** Function is not executed (EN = 0)

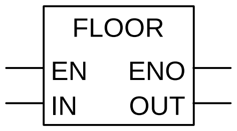
Example



If I0.0 is 1, the contents of MD8 are read as a floating-point number which is converted into a double integer using the function **Round**. The result is stored in MD12. The output Q4.0 is "1" if an overflow occured or the instruction was not processed (I0.0 = 0).

3.16 FLOOR Floor

Symbol



Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, L, D	Enable input
ENO	BOOL	I, Q, M, L, D	Enable output
IN	REAL	I, Q, M, L, D	Floating-point value to convert
OUT	DINT	I, Q, M, L, D	Greatest lower double integer

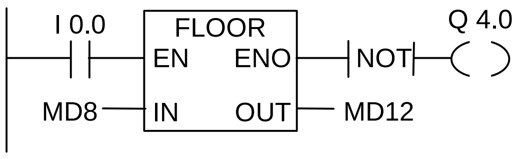
Description

FLOOR (Floor) reads the content of the IN parameter as a floating-point number and converts it to a double integer (32-bit). The result is the greatest integer component which is lower than the floating-point number ("Round to - infinity"). If an overflow occurred ENO will be "0".

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	x	-	-	x	x	0	x	x	1

Example



If I0.0 is "1", then the content of MD8 is read as a floating-point number and converted to a double integer by the round to - infinity mode. The result is stored in MD12. The output Q4.0 is "1" if an overflow occurred, or the instruction was not executed (I0.0 = 0).

4 Counter Instructions

4.1 Overview of Counter Instructions

Area in Memory

Counters have an area reserved for them in the memory of your CPU. This memory area reserves one 16-bit word for each counter address. The ladder logic instruction set supports 256 counters.

The counter instructions are the only functions that have access to the counter memory area.

Count Value

Bits 0 through 9 of the counter word contain the count value in binary code. The count value is moved to the counter word when a counter is set. The range of the count value is 0 to 999.

You can vary the count value within this range by using the following counter instructions:

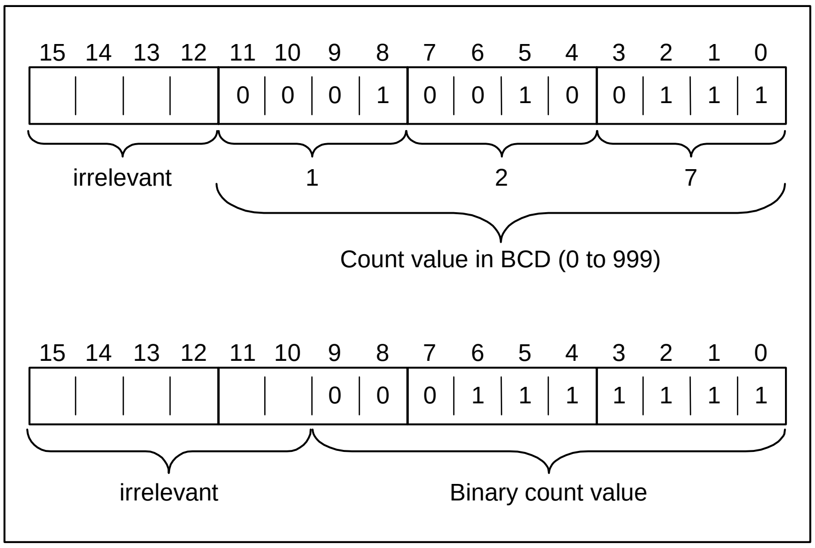
• S_CUD	Up-Down Counter
• S_CD	Down Counter
• S_CU	Up Counter
• ---(SC)	Set Counter Coil
• ---(CU)	Up Counter Coil
• ---(CD)	Down Counter Coil

Bit Configuration in the Counter

You provide a counter with a preset value by entering a number from 0 to 999, for example 127, in the following format: C#127. The C# stands for binary coded decimal format (BCD format: each set of four bits contains the binary code for one decimal value).

Bits 0 through 11 of the counter contain the count value in binary coded decimal format.

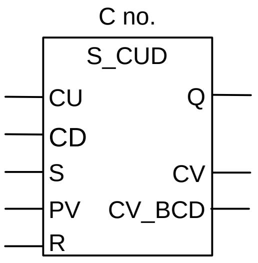
The following figure shows the contents of the counter after you have loaded the count value 127, and the contents of the counter cell after the counter has been set.



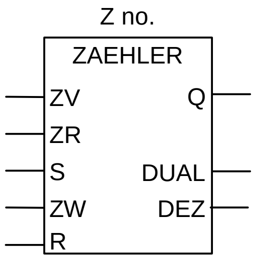
4.2 S_CUD Up-Down Counter

Symbol

English



German



Parameter English	Parameter German	Data Type	Memory Area	Description
C no.	Z no.	COUNTER	C	Counter identification number; range depends on CPU
CU	ZV	BOOL	I, Q, M, L, D	Count up input
CD	ZR	BOOL	I, Q, M, L, D	Count down input
S	S	BOOL	I, Q, M, L, D	Set input for presetting counter
PV	ZW	WORD	I, Q, M, L, D or constant	Enter counter value as C#<value> in the range from 0 to 999
PV	ZW	WORD	I, Q, M, L, D	Value for presetting counter
R	R	BOOL	I, Q, M, L, D	Reset input
CV	DUAL	WORD	I, Q, M, L, D	Current counter value, hexadecimal number
CV_BCD	DEZ	WORD	I, Q, M, L, D	Current counter value, BCD coded
Q	Q	BOOL	I, Q, M, L, D	Status of the counter

Description

S_CUD (Up-Down Counter) is preset with the value at input PV if there is a positive edge at input S. If there is a 1 at input R, the counter is reset and the count is set to zero. The counter is incremented by one if the signal state at input CU changes from "0" to "1" and the value of the counter is less than "999". The counter is decremented by one if there is a positive edge at input CD and the value of the counter is greater than "0".

If there is a positive edge at both count inputs, both instructions are executed and the count value remains unchanged.

If the counter is set and if RLO = 1 at the inputs CU/CD, the counter will count accordingly in the next scan cycle, even if there was no change from a positive to a negative edge or viceversa.

The signal state at output Q is "1" if the count is greater than zero and "0" if the count is equal to zero.

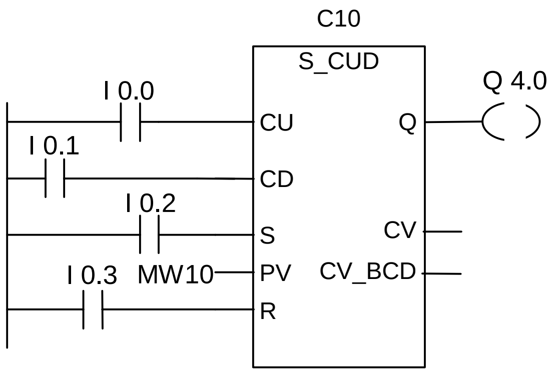
Status word

	BR	**CC 1**	**CC 0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**/FC**
writes:	-	-	-	-	-	x	x	x	1

Note

Avoid to use a counter at several program points (risk of counting errors).

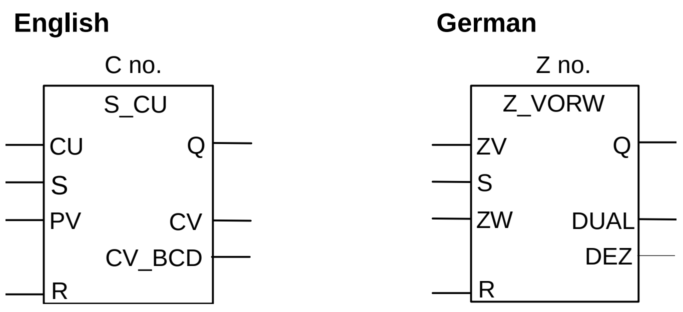
Example



If I0.2 changes from "0" to "1", the counter is preset with the value of MW10. If the signal state of I0.0 changes from "0" to "1", the value of counter C10 will be incremented by one - except when the value of C10 is equal than "999". If I0.1 changes from "0" to "1", C10 is decremented by one - except when the value of C10 is equal to "0". Q4.0 is "1" if C10 is not equal to zero.

4.3 S_CU Up Counter

Symbol



Parameter English	Parameter German	Data Type	Memory Area	Description
C no.	Z no.	COUNTER	C	Counter identification number; range depends of CPU
CU	ZV	BOOL	I, Q, M, L, D	Count up input
S	S	BOOL	I, Q, M, L, D	Set input for presetting counter
PV	ZW	WORD	I, Q, M, L, D or constant	Enter counter value as C#<value> in the range from 0 to 999
PV	ZW	WORD	I, Q, M, L, D	Value for presetting counter
R	R	BOOL	I, Q, M, L, D	Reset input
CV	DUAL	WORD	I, Q, M, L, D	Current counter value, hexadecimal number
CV_BCD	DEZ	WORD	I, Q, M, L, D	Current counter value, BCD coded
Q	Q	BOOL	I, Q, M, L, D	Status of the counter

Description

S_CU (Up Counter) is preset with the value at input PV if there is a positive edge at input S.

The counter is reset if there is a "1" at input R and the count value is then set to zero.

The counter is incremented by one if the signal state at input CU changes from "0" to "1" and the value of the counter is less than "999".

If the counter is set and if RLO = 1 at the inputs CU, the counter will count accordingly in the next scan cycle, even if there was no change from a positive to a negative edge or viceversa.

The signal state at output Q is "1" if the count is greater than zero and "0" if the count is equal to zero.

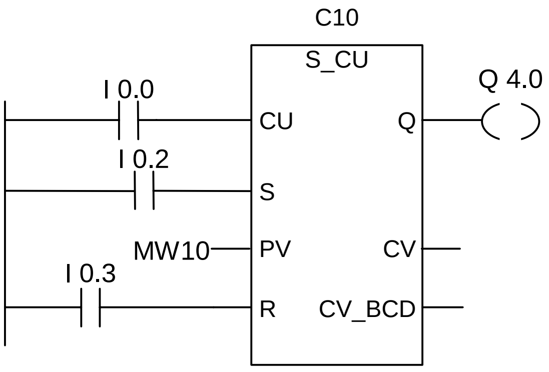
Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	-	-	-	-	-	x	x	x	1

Note

Avoid to use a counter at several program points (risk of counting errors).

Example

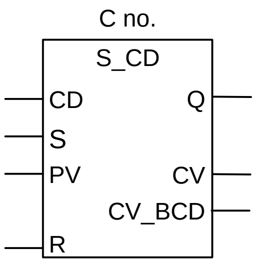


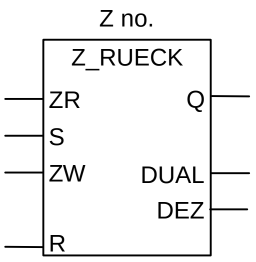
If I0.2 changes from "0" to "1", the counter is preset with the value of MW10. If the signal state of I0.0 changes from "0" to "1", the value of counter C10 will be incremented by one - unless the value of C10 is equal to "999". Q4.0 is "1" if C10 is not equal to zero.

4.4 S_CD Down Counter

Symbol

English **German**





Parameter English	Parameter German	Data Type	Memory Area	Description
C no.	Z no.	COUNTER	C	Counter identification num-ber; range depends of CPU
CD	ZR	BOOL	I, Q, M, L, D	Count down input
S	S	BOOL	I, Q, M, L, D	Set input for presetting counter
PV	ZW	WORD	I, Q, M, L, D or constant	Enter counter value as C#<value> in the range from 0 to 999
PV	ZW	WORD	I, Q, M, L, D	Value for presetting counter
R	R	BOOL	I, Q, M, L, D	Reset input
CV	DUAL	WORD	I, Q, M, L, D	Current counter value, hexadecimal number
CV_BCD	DEZ	WORD	I, Q, M, L, D	Current counter value, BCD coded
Q	Q	BOOL	I, Q, M, L, D	Status counter

Description

S_CD (Down Counter) is set with the value at input PV if there is a positive edge at input S.

The counter is reset if there is a 1 at input R and the count value is then set to zero.

The counter is decremented by one if the signal state at input CD changes from "0" to "1" and the value of the counter is greater than zero.

If the counter is set and if RLO = 1 at the inputs CD, the counter will count accordingly in the next scan cycle, even if there was no change from a positive to a negative edge or viceversa.

The signal state at output Q is "1" if the count is greater than zero and "0" if the count is equal to zero.

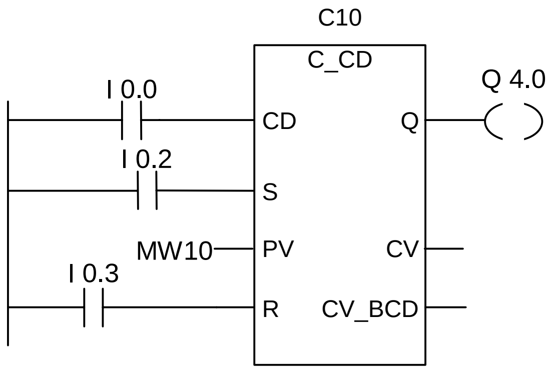
Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	-	-	-	-	-	x	x	x	1

Note

Avoid to use a counter at several program points (risk of counting errors).

Example



If I0.2 changes from "0" to "1", the counter is preset with the value of MW10. If the signal state of I0.0 changes from "0" to "1", the value of counter C10 will be decremented by one - unless the value of C10 is equal to "0". Q4.0 is "1" if C10 is not equal to zero.

4.5 ---(SC) Set Counter Value

Symbol

English	**German**
<C no.>	<Z no.>
---(SC)	**---(SZ)**
<preset value>	<preset value>

Parameter English	Parameter German	Data Type	Memory Area	Description
<C no.>	<Z no.>	COUNTER	C	Counter number to be preset
<preset value>	<preset value>	WORD	I, Q, M, L, D or constant	Value for presetting BCD (0 to 999)

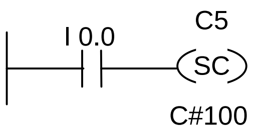
Description

---(SC) (Set Counter Value) executes only if there is a positive edge in RLO. At that time, the preset value transferred into the specified counter.

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	x	-	-	-	-	0	x	-	0

Example



The counter C5 is preset with the value of 100 if there is a positive edge at input I0.0 (change from "0" to "1"). If there is no positive edge, the value of counter C5 remains unchanged.

4.6 ---(CU) Up Counter Coil

Symbol

English	**German**
<C no.>	<Z no.>
---(CU)	**---(ZV)**

Parameter English	**Parameter German**	**Data Type**	**Memory Area**	**Description**
<C no.>	<Z no.>	COUNTER	C	Counter identification number; range depends on CPU

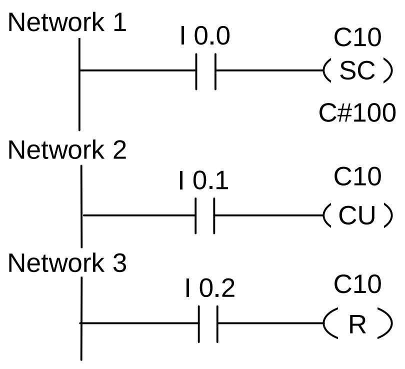
Description

---(CU) (Up Counter Coil) increments the value of the specified counter by one if there is a positive edge in the RLO and the value of the counter is less than "999". If there is no positive edge in the RLO or the counter already has the value "999", the value of the counter will be unchanged.

Status word

	BR	**CC 1**	**CC 0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**/FC**
writes:	-	-	-	-	-	0	-	-	0

Example



If the signal state of input I0.0 changes from "0" to "1" (positive edge in RLO), the preset value of 100 is loaded to counter C10.

If the signal state of input I0.1 changes from "0" to "1" (positive edge in RLO), counter C10 count value will be incremented by one unless the value of C10 is equal to "999". If there is no positive edge in RLO, the value of C10 will be unchanged.

If the signal state of I0.2 is "1", the counter C10 is reset to "0".

4.7 ---(CD) Down Counter Coil

Symbol

English	**German**
<C no.>	<Z no.>
---(CD)	**---(ZD)**

Parameter English	Parameter German	Data Type	Memory Area	Description
<C no.>	<Z no.>	COUNTER	C	Counter identification number; range depends on CPU

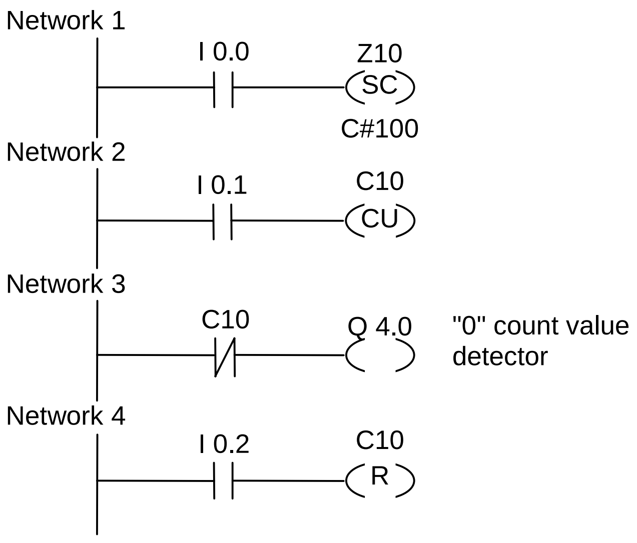
Description

---(CD) (Down Counter Coil) decrements the value of the specified counter by one, if there is a positive edge in the RLO state and the value of the counter is more than "0". If there is no positive edge in the RLO or the counter has already the value "0", the value of the counter will be unchanged.

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	-	-	-	-	-	0	-	-	0

Example



If the signal state of input I0.0 changes from "0" to "1" (positive edge in RLO), the preset value of 100 is loaded to counter C10.

If the signal state of input I0.1 changes from "0" to "1" (positive edge in RLO), counter C10 count value will be decremented by one unless the value of C10 is equal to "0". If there is no positive edge in RLO, the value of C10 will be unchanged.

If the count value = 0, then Q4.0 is turned on.

If the signal state of input I0.2 is "1", the counter C10 is reset to "0".

5 Data Block Instructions

5.1 ---(OPN) Open Data Block: DB or DI

Symbol

<DB no.> or <DI no.>

---(OPN)

Parameter	Data Type	Memory Area	Description
<DB no.> <DI no.>	BLOCK_DB	DB, DI	Number of DB/DI; range depends on CPU

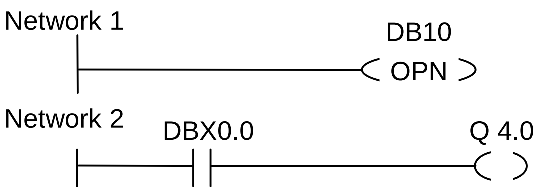
Description

---(OPN) (Open a Data Block) opens a shared data block (DB) or an instance data block (DI). The **---(OPN)** function is an unconditional call of a data block. The number of the data block is transferred into the DB or DI register. The subsequent DB and DI commands access the corresponding blocks, depending on the register contents.

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	-	-	-	-	-	-	-	-	-

Example



Data block 10 (DB10) is opened. The contact address (DBX0.0) refers to bit zero of data byte zero of the current data record contained in DB10. The signal state of this bit is assigned to the output Q4.0.

6 Logic Control Instructions

6.1 Overview of Logic Control Instructions

Description

You can use logic control instructions in all logic blocks: organization blocks (OBs), function blocks (FBs), and functions (FCs).

There are logic control instructions to perform the following functions:

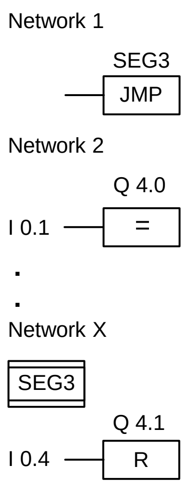
- ---(JMP)--- Unconditional Jump
- ---(JMP)--- Conditional Jump
- ---(JMPN)--- Jump-If-Not

Label as Address

The address of a Jump instruction is a label. A label consists of a maximum of four characters. The first character must be a letter of the alphabet; the other characters can be letters or numbers (for example, SEG3). The jump label indicates the destination to which you want the program to jump.

Label as Destination

The destination label must be at the beginning of a network. You enter the destination label at the beginning of the network by selecting LABEL from the ladder logic browser. An empty box appears. In the box, you type the name of the label.



6.2 ---(JMP)--- Unconditional Jump

Symbol

<label name>

---(JMP)

Description

---(JMP) (jump within the block when 1) functions as an absolute jump when there is no other Ladder element between the left-hand power rail and the instruction (see example).

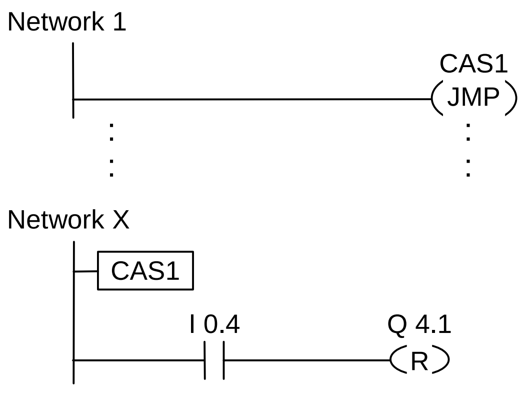
A destination (LABEL) must also exist for every ---(JMP).

All instructions between the jump instruction and the label are not executed.

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	-	-	-	-	-	-	-	-	-

Example



The jump is always executed and the instructions between the jump instruction and the jump label are missed out.

6.3 ---(JMP)--- Conditional Jump

Symbol

<label name>

---(JMP)

Description

---(JMP) (jump within the block when 1) functions as a conditional jump when the RLO of the previous logic operation is "1".

A destination (LABEL) must also exist for every ---(JMP).

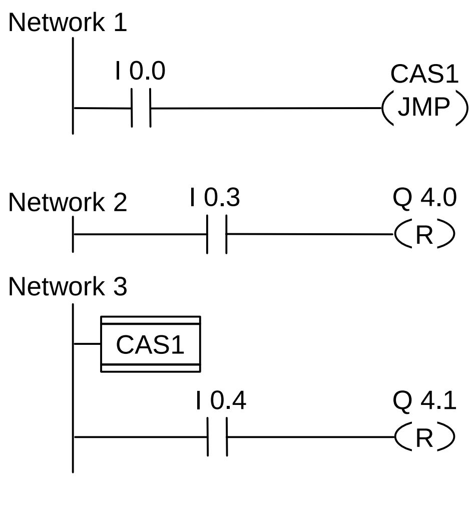
All instructions between the jump instruction and the label are not executed.

If a conditional jump is not executed, the RLO changes to "1" after the jump instruction.

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	-	-	-	-	-	0	1	1	0

Example



If I0.0 = "1", the jump to label CAS1 is executed. Because of the jump, the instruction to reset output Q4.0 is not executed even if there is a logic "1" at I0.3.

6.4 ---(JMPN) Jump-If-Not

Symbol

<label name>

---(JMPN)

Description

---(JMPN) (Jump-If-not) corresponds to a "goto label" function which is executed if the RLO is "0".

A destination (LABEL) must also exist for every ---(JMPN).

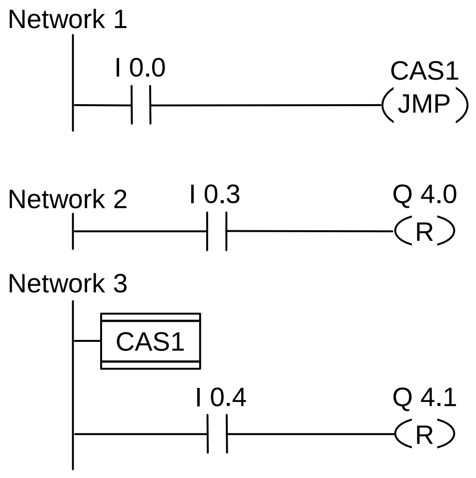
All instructions between the jump instruction and the label are not executed.

If a conditional jump is not executed, the RLO changes to "1" after the jump instruction.

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	-	-	-	-	-	0	1	1	0

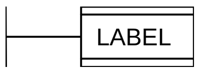
Example



If I0.0 = "0", the jump to label CAS1 is executed. Because of the jump, the instruction to reset output Q4.0 is not executed even if there is a logic "1" at I0.3.

6.5 LABEL Label

Symbol



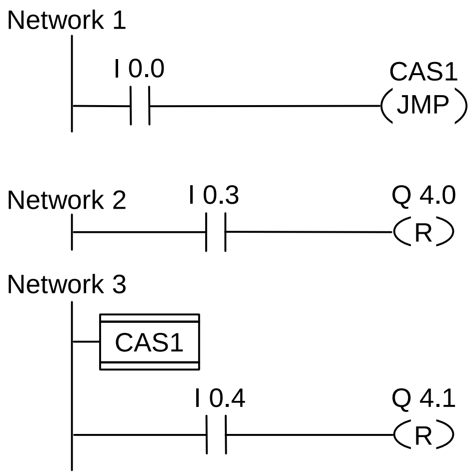
Description

LABEL is the identifier for the destination of a jump instruction.

The first character must be a letter of the alphabet; the other characters can be letters or numbers (for example, CAS1).

A jump label (**LABEL**) must exist for every ---(JMP) or ---(JMPN).

Example



If I0.0 = "1", the jump to label CAS1 is executed. Because of the jump, the instruction to reset output Q4.0 is not executed even if there is a logic "1" at I0.3.

7 Integer Math Instructions

7.1 Overview of Integer Math Instructions

Description

Using integer math, you can carry out the following operations with **two integer numbers** (16 and 32 bits):

• ADD_I	Add Integer
• SUB_I	Subtract Integer
• MUL_I	Multiply Integer
• DIV_I	Divide Integer

• ADD_DI	Add Double Integer
• SUB_DI	Subtract Double Integer
• MUL_DI	Multiply Double Integer
• DIV_DI	Divide Double Integer
• MOD_DI	Return Fraction Double Integer

7.2 Evaluating the Bits of the Status Word with Integer Math Instructions

Description

The integer math instructions affect the following bits in the Status word: CC1 and CC0, OV and OS.

The following tables show the signal state of the bits in the status word for the results of instructions with Integers (16 and 32 bits):

Valid Range for the Result	**CC 1**	**CC 0**	**OV**	**OS**
0 (zero)	0	0	0	*
16 bits: -32 768 <= result < 0 (negative number) 32 bits: -2 147 483 648 <=result < 0 (negative number)	0	1	0	*
16 bits: 32 767 >= result > 0 (positive number) 32 bits: 2 147 483 647 >= result > 0 (positive number)	1	0	0	*

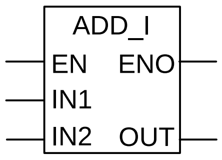
* The OS bit is not affected by the result of the instruction.

Invalid Range for the Result	**A1**	**A0**	**OV**	**OS**
Underflow (addition) 16 bits: result = -65536 32 bits: result = -4 294 967 296	0	0	1	1
Underflow (multiplication) 16 bits: result < -32 768 (negative number) 32 bits: result < -2 147 483 648 (negative number)	0	1	1	1
Overflow (addition, subtraction) 16 bits: result > 32 767 (positive number) 32 bits: result > 2 147 483 647 (positive number)	0	1	1	1
Overflow (multiplication, division) 16 bits: result > 32 767 (positive number) 32 bits: result > 2 147 483 647 (positive number)	1	0	1	1
Underflow (addition, subtraction) 16 bits: result < -32. 768 (negative number) 32 bits: result < -2 147 483 648 (negative number)	1	0	1	1
Division by 0	1	1	1	1

Operation	**A1**	**A0**	**OV**	**OS**
+D: result = -4 294 967 296	0	0	1	1
/D or MOD: division by 0	1	1	1	1

7.3 ADD_I Add Integer

Symbol



Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, L, D	Enable input
ENO	BOOL	I, Q, M, L, D	Enable output
IN1	INT	I, Q, M, L, D or constant	First value for addition
IN2	INT	I, Q, M, L, D or constant	Second value for addition
OUT	INT	I, Q, M, L, D	Result of addition

Description

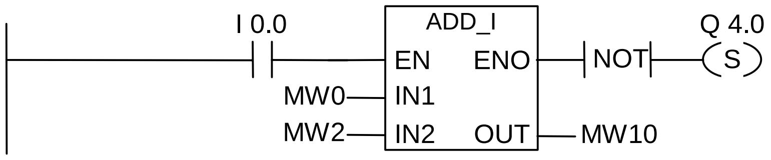
ADD_I (Add Integer) is activated by a logic "1" at the Enable (EN) Input. IN1 and IN2 are added and the result can be scanned at OUT. If the result is outside the permissible range for an integer (16-bit), the OV bit and OS bit will be "1" and ENO is logic "0", so that other functions after this math box which are connected by the ENO (cascade arrangement) are not executed.

See also Evaluating the Bits of the Status Word with Integer Math Instructions.

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	x	x	x	x	x	0	x	x	1

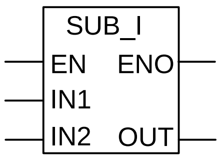
Example



The ADD_I box is activated if I0.0 = "1". The result of the addition MW0 + MW2 is output to MW10. If the result was outside the permissible range for an integer, the output Q4.0 is set.

7.4 SUB_I Subtract Integer

Symbol



Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, L, D	Enable input
ENO	BOOL	I, Q, M, L, D	Enable output
IN1	INT	I, Q, M, L, D or constant	First value for subtraction
IN2	INT	I, Q, M, L, D or constant	Value to subtract
OUT	INT	I, Q, M, L, D	Result of subtraction

Description

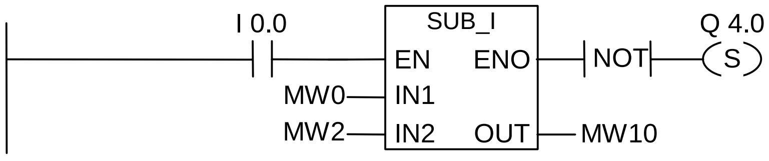
SUB_I (Subtract Integer) is activated by a logic "1" at the Enable (EN) Input. IN2 is subtracted from IN1 and the result can be scanned at OUT. If the result is outside the permissible range for an integer (16-bit), the OV bit and OS bit will be "1" and ENO is logic "0", so that other functions after this math box which are connected by the ENO (cascade arrangement) are not executed.

See also Evaluating the Bits of the Status Word with Integer Math Instructions.

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	x	x	x	x	x	0	x	x	1

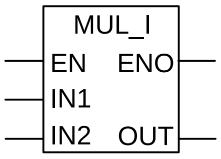
Example



The SUB_I box is activated if I0.0 = "1". The result of the subtraction MW0 - MW2 is output to MW10. If the result was outside the permissible range for an integer or the signal state of I0.0 = 0, the output Q4.0 is set.

7.5 MUL_I Multiply Integer

Symbol



Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, L, D	Enable input
ENO	BOOL	I, Q, M, L, D	Enable output
IN1	INT	I, Q, M, L, D or constant	First value for multiplication
IN2	INT	I, Q, M, L, D or constant	Second value for multiplication
OUT	INT	I, Q, M, L, D	Result of multiplication

Description

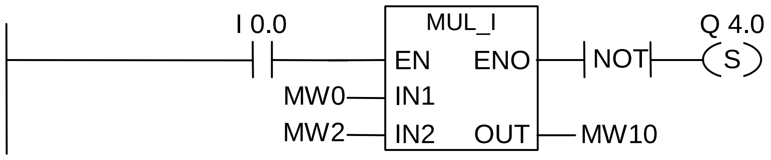
MUL_I (Multiply Integer) is activated by a logic "1" at the Enable (EN) Input. IN1 and IN2 are multiplied and the result can be scanned at OUT. If the result is outside the permissible range for an integer (16-bit), the OV bit and OS bit will be "1" and ENO is logic "0", so that other functions after this math box which are connected by the ENO (cascade arrangement) are not executed.

See also Evaluating the Bits of the Status Word with Integer Math Instructions.

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	x	x	x	x	x	0	x	x	1

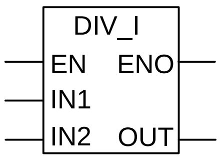
Example



The MUL_I box is activated if I0.0 = "1". The result of the multiplication MW0 x MW2 is output to MD10. If the result was outside the permissible range for an integer, the output Q4.0 is set.

7.6 DIV_I Divide Integer

Symbol



Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, L, D	Enable input
ENO	BOOL	I, Q, M, L, D	Enable output
IN1	INT	I, Q, M, L, D or constant	Dividend
IN2	INT	I, Q, M, L, D or constant	Divisor
OUT	INT	I, Q, M, L, D	Result of division

Description

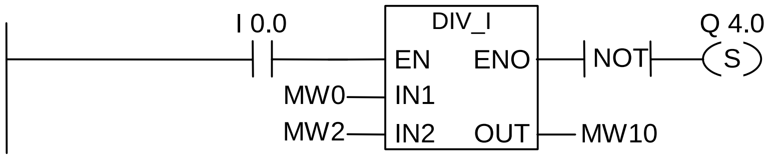
DIV_I (Divide Integer) is activated by a logic "1" at the Enable (EN) Input. IN1 is divided by IN2 and the result can be scanned at OUT. If the result is outside the permissible range for an integer (16-bit), the OV bit and OS bit is "1" and ENO is logic "0", so that other functions after this math box which are connected by ENO (cascade arrangement) are not executed.

See also Evaluating the Bits of the Status Word with Integer Math Instructions.

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	x	x	x	x	x	0	x	x	1

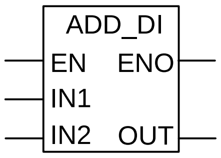
Example



The DIV_I box is activated if I0.0 = "1". The result of the division MW0 by MW2 is output to MW10. If the result was outside the permissible range for an integer, the output Q4.0 is set.

7.7 ADD_DI Add Double Integer

Symbol



Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, L, D	Enable input
ENO	BOOL	I, Q, M, L, D	Enable output
IN1	DINT	I, Q, M, L, D or constant	First value for addition
IN2	DINT	I, Q, M, L, D or constant	Second value for addition
OUT	DINT	I, Q, M, L, D	Result of addition

Description

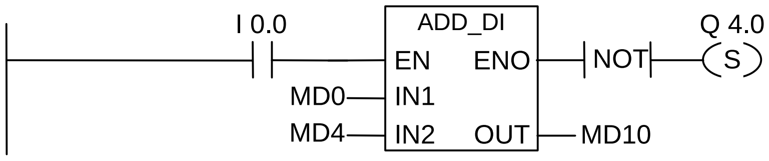
ADD_DI (Add Double Integer) is activated by a logic "1" at the Enable (EN) Input. IN1 and IN2 are added and the result can be scanned at OUT. If the result is outside the permissible range for a double integer (32-bit), the OV bit and OS bit will be "1" and ENO is logic "0", so that other functions after this math box which are connected by the ENO (cascade arrangement) are not executed.

See also Evaluating the Bits of the Status Word with Integer Math Instructions.

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	x	x	x	x	x	0	x	x	1

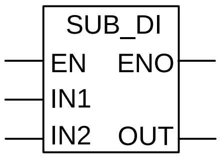
Example



The ADD_DI box is activated if I0.0 = "1". The result of the addition MD0 + MD4 is output to MD10. If the result was outside the permissible range for a double integer, the output Q4.0 is set.

7.8 SUB_DI Subtract Double Integer

Symbol



Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, L, D	Enable input
ENO	BOOL	I, Q, M, L, D	Enable output
IN1	DINT	I, Q, M, L, D or constant	First value for subtraction
IN2	DINT	I, Q, M, L, D or constant	Value to subtract
OUT	DINT	I, Q, M, L, D	Result of subtraction

Description

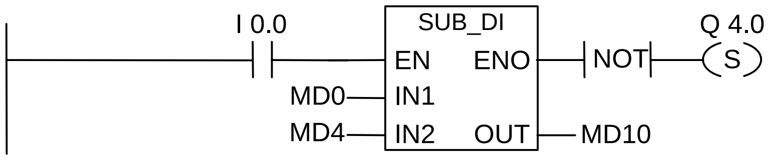
SUB_DI (Subtract Double Integer) is activated by a logic "1" at the Enable (EN) Input. IN2 is subtracted from IN1 and the result can be scanned at OUT. If the result is outside the permissible range for a double integer (32-bit), the OV bit and OS bit will be "1" and ENO is logic "0", so that other functions after this math box which are connected by the ENO (cascade arrangement) are not executed.

See also Evaluating the Bits of the Status Word with Integer Math Instructions.

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	x	x	x	x	x	0	x	x	1

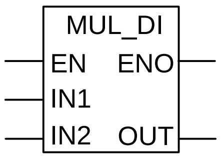
Example



The SUB_DI box is activated if I0.0 = "1". The result of the subtraction MD0 - MD4 is output to MD10. If the result was outside the permissible range for a double integer, the output Q4.0 is set.

7.9 MUL_DI Multiply Double Integer

Symbol



Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, L, D	Enable input
ENO	BOOL	I, Q, M, L, D	Enable output
IN1	DINT	I, Q, M, L, D or constant	First value for multiplication
IN2	DINT	I, Q, M, L, D or constant	Second value for multiplication
OUT	DINT	I, Q, M, L, D	Result of multiplication

Description

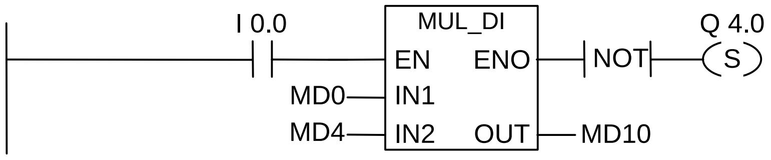
MUL_DI (Multiply Double Integer) is activated by a logic "1" at the Enable (EN) Input. IN1 and IN2 are multiplied and the result can be scanned at OUT. If the result is outside the permissible range for a double integer (32-bit), the OV bit and OS bit will be "1" and ENO is logic "0", so that other functions after this math box which are connected by the ENO (cascade arrangement) are not executed.

See also Evaluating the Bits of the Status Word with Integer Math Instructions.

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	x	x	x	x	x	0	x	x	1

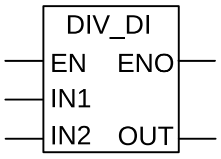
Example



The MUL_DI box is activated if I0.0 = "1". The result of the multiplication MD0 x MD4 is output to MD10. If the result was outside the permissible range for a double integer, the output Q4.0 is set.

7.10 DIV_DI Divide Double Integer

Symbol



Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, L, D	Enable input
ENO	BOOL	I, Q, M, L, D	Enable output
IN1	DINT	I, Q, M, L, D or constant	Dividend
IN2	DINT	I, Q, M, L, D or constant	Divisor
OUT	DINT	I, Q, M, L, D	Whole-number result of division

Description

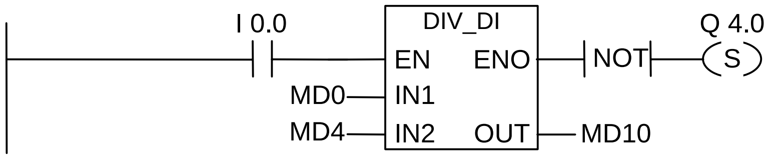
DIV_DI (Divide Double Integer) is activated by a logic "1" at the Enable (EN) Input. IN1 is divided by IN2 and the result can be scanned at OUT. The Divide Double Integer element does not produce a remainder. If the result is outside the permissible range for a double integer (32-bit), the OV bit and OS bit is "1" and ENO is logic "0", so that other functions after this math box which are connected by the ENO (cascade arrangement) are not executed.

See also Evaluating the Bits of the Status Word with Integer Math Instructions.

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	x	x	x	x	x	0	x	x	1

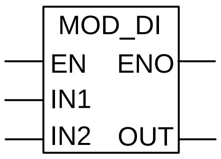
Example



The DIV_DI box is activated if I0.0 = "1". The result of the division MD0 : MD4 is output to MD10. If the result was outside the permissible range for a double integer, the output Q4.0 is set.

7.11 MOD_DI Return Fraction Double Integer

Symbol



Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, L, D	Enable input
ENO	BOOL	I, Q, M, L, D	Enable output
IN1	DINT	I, Q, M, L, D or constant	Dividend
IN2	DINT	I, Q, M, L, D or constant	Divisor
OUT	DINT	I, Q, M, L, D	Remainder of division

Description

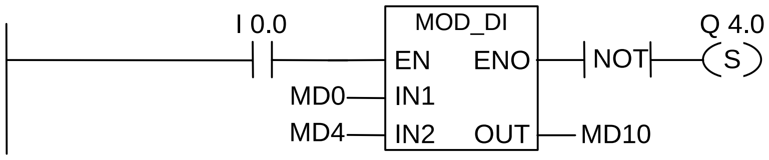
MOD_DI (Return Fraction Double Integer) is activated by a logic "1" at the Enable (EN) Input. IN1 is divided by IN2 and the fraction can be scanned at OUT. If the result is outside the permissible range for a double integer (32-bit), the OV bit and OS bit is "1" and ENO is logic "0", so that other functions after this math box which are connected by the ENO (cascade arrangement) are not executed.

See also Evaluating the Bits of the Status Word with Integer Math Instructions.

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	x	x	x	x	x	0	x	x	1

Example



The DIV_DI box is activated if I0.0 = "1". The remainder of the division MD0:MD4 is output to MD10. If the remainder was outside the permissible range for a double integer, the output Q4.0 is set.

8 Floating Point Math Instructions

8.1 Overview of Floating-Point Math Instruction

Description

The IEEE 32-bit floating-point numbers belong to the data type called REAL. You can use the floating-point math instructions to perform the following math instructions using **two 32-bit IEEE floating-point numbers**:

- ADD_R Add Real
- SUB_R Subtract Real
- MUL_R Multiply Real
- DIV_R Divide Real

Using floating-point math, you can carry out the following operations with **one 32-bit IEEE floating-point number**:

- Establish the Absolute Value (ABS)
- Establish the Square (SQR) and the Square Root (SQRT)
- Establish the Natural Logarithm (LN)
- Establish the Exponential Value (EXP) to base e (= 2,71828)
- Establish the following trigonometrical functions of an angle represented as a 32-bit IEEE floating-point number
  - Sine (SIN) and Arc Sine (ASIN)
  - Cosine (COS) and Arc Cosine (ACOS)
  - Tangent (TAN) and Arc Tangent (ATAN)

See also Evaluating the Bits of the Status Word.

8.2 Evaluating the Bits of the Status Word with Floating-Point Math Instructions

Description

Floating–point instructions affect the following bits in the status word: CC 1 and CC 0, OV and OS.

The following tables show the signal state of the bits in the status word for the results of instructions with floating-point numbers (32 bits):

Valid Area for a Result	CC 1	CC 0	OV	OS
+0, -0 (zero)	0	0	0	*
-3.402823E+38 < result < -1.175494E-38 (negative number)	0	1	0	*
+1.175494E-38 < result < 3.402824E+38 (positive number)	1	0	0	*

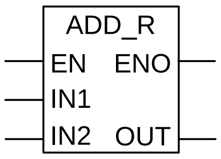
* The OS bit is not affected by the result of the instruction.

Invalid Area for a Result	CC 1	CC 0	OV	OS
Underflow -1.175494E-38 < result < - 1.401298E-45 (negative number)	0	0	1	1
Underflow +1.401298E-45 < result < +1.175494E-38 (positive number)	0	0	1	1
Overflow Result < -3.402823E+38 (negative number)	0	1	1	1
Overflow Result > 3.402823E+38 (positive number)	1	0	1	1
Not a valid floating-point number or illegal instruction (input value outside the valid range)	1	1	1	1

8.3 Basic Instructions

8.3.1 ADD_R Add Real

Symbol



Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, L, D	Enable input
ENO	BOOL	I, Q, M, L, D	Enable output
IN1	REAL	I, Q, M, L, D or constant	First value for addition
IN2	REAL	I, Q, M, L, D or constant	Second value for addition
OUT	REAL	I, Q, M, L, D	Result of addition

Description

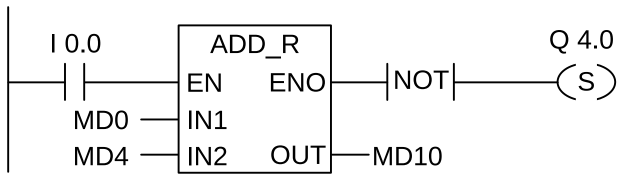
ADD_R (Add Real) is activated by a logic "1" at the Enable (EN) Input. IN1 and IN2 are added and the result can be scanned at OUT. If the result is outside the permissible range for a floating-point number (overflow or underflow), the OV bit and OS bit will be "1" and ENO is "0", so that other functions after this math box which are connected by the ENO (cascade arrangement) are not executed.

See also Evaluating the Bits of the Status Word.

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	x	x	x	x	x	0	x	x	1

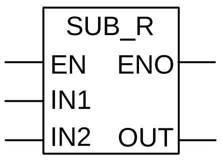
Example



The ADD_R box is activated by logic "1" at I0.0. The result of the addition MD0 + MD4 is output to MD10. If the result was outside the permissible range for a floating-point number or if the program statement was not processed (I0.0 = 0), the output Q4.0 is set.

8.3.2 SUB_R Subtract Real

Symbol



Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, L, D	Enable input
ENO	BOOL	I, Q, M, L, D	Enable output
IN1	REAL	I, Q, M, L, D or constant	First value for subtraction
IN2	REAL	I, Q, M, L, D or constant	Value to subtract
OUT	REAL	I, Q, M, L, D	Result of subtraction

Description

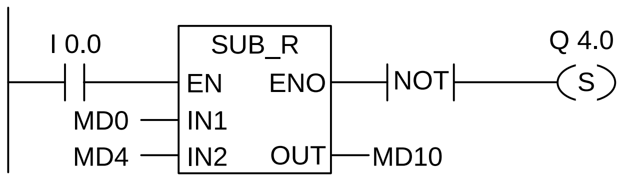
SUB_R (Subtract Real) is activated by a logic "1" at the Enable (EN) Input. IN2 is subtracted from IN1 and the result can be scanned at OUT. If the result is outside the permissible range for a floating-point number (overflow or underflow), the OV bit and OS bit will be "1" and ENO is logic "0", so that other functions after this math box which are connected by the ENO (cascade arrangement) are not executed.

See also Evaluating the Bits of the Status Word.

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	x	x	x	x	x	0	x	x	1

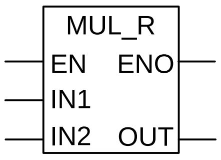
Example



The SUB_R box is activated by logic "1" at I0.0. The result of the subtraction MD0 - MD4 is output to MD10. If the result was outside the permissible range for a floating-point number or if the program statement was not processed, the output Q4.0 is set.

8.3.3 MUL_R Multiply Real

Symbol



Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, L, D	Enable input
ENO	BOOL	I, Q, M, L, D	Enable output
IN1	REAL	I, Q, M, L, D or constant	First value for multiplication
IN2	REAL	I, Q, M, L, D or constant	Second value for multiplication
OUT	REAL	I, Q, M, L, D	Result of multiplication

Description

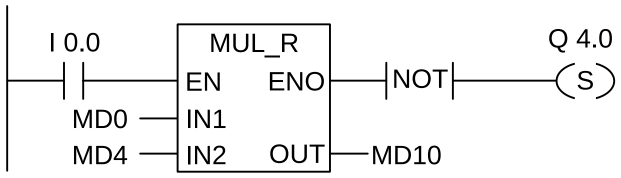
MUL_R (Multiply Real) is activated by a logic "1" at the Enable (EN) Input. IN1 and IN2 are multiplied and the result can be scanned at OUT. If the result is outside the permissible range for a floating-point number (overflow or underflow), the OV bit and OS bit will be "1" and ENO is logic "0", so that other functions after this math box which are connected by the ENO (cascade arrangement) are not executed.

See also Evaluating the Bits of the Status Word.

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	x	x	x	x	x	0	x	x	1

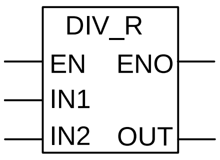
Example



The MUL_R box is activated by logic "1" at I0.0. The result of the multiplication MD0 x MD4 is output to MD0. If the result was outside the permissible range for a floating-point number or if the program statement was not processed, the output Q4.0 is set.

8.3.4 DIV_R Divide Real

Symbol



Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, L, D	Enable input
ENO	BOOL	I, Q, M, L, D	Enable output
IN1	REAL	I, Q, M, L, D or constant	Dividend
IN2	REAL	I, Q, M, L, D or constant	Divisor
OUT	REAL	I, Q, M, L, D	Result of division

Description

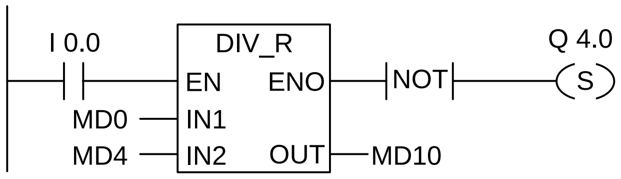
DIV_R (Divide Real) is activated by a logic "1" at the Enable (EN) Input. IN1 is divided by IN2 and the result can be scanned at OUT. If the result is outside the permissible range for a floating-point number (overflow or underflow), the OV bit and OS bit is "1" and ENO is logic "0", so that other functions after this math box which are connected by the ENO (cascade arrangement) are not executed.

See also Evaluating the Bits of the Status Word.

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	x	x	x	x	x	0	x	x	1

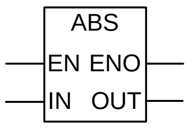
Example



The DIV_R box is activated by logic "1" at I0.0. The result of the division MD0 by MD4 is output to MD10. If the result was outside the permissible range for a floating-point number or if the program statement was not processed, the output Q4.0 is set.

8.3.5 ABS Establish the Absolute Value of a Floating-Point Number

Symbol



Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, L, D	Enable input
ENO	BOOL	I, Q, M, L, D	Enable output
IN	REAL	I, Q, M, L, D or constant	Input value: floating-point
OUT	REAL	I, Q, M, L, D	Output value: absolute value of the floating-point number

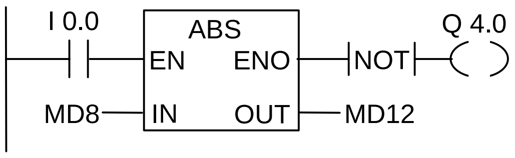
Description

ABS establishes the absolute value of a floating-point number.

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	1	-	-	-	-	0	1	1	1

Example



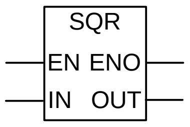
If I0.0 = "1", the absolute value of MD8 is output at MD12.

MD8 = + 6.234 gives MD12 = 6.234. Output Q4.0 is "1" when the conversion is not executed (ENO = EN = 0).

8.4 Extended Instructions

8.4.1 SQR Establish the Square

Symbol



Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, L, D	Enable input
ENO	BOOL	I, Q, M, L, D	Enable output
IN	REAL	I, Q, M, L, D or constant	Input value: floating-point
OUT	REAL	I, Q, M, L, D	Output value: square of floating-point number

Description

SQR establishes the square of a floating-point number.

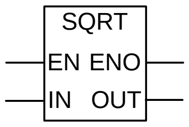
See also Evaluating the Bits of the Status Word.

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	x	x	x	x	x	0	x	x	1

8.4.2 SQRT Establish the Square Root

Symbol



Parameter	**Data Type**	**Memory Area**	**Description**
EN	BOOL	I, Q, M, L, D	Enable input
ENO	BOOL	I, Q, M, L, D	Enable output
IN	REAL	I, Q, M, L, D or constant	Input value: floating-point
OUT	REAL	I, Q, M, L, D	Output value: square root of floating-point number

Description

SQRT establishes the square root of a floating-point number. This instruction issues a positive result when the address is greater than "0". Sole exception: the square root of -0 is -0.

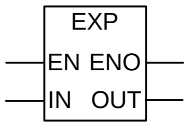
See also Evaluating the Bits of the Status Word.

Status word

	BR	**CC 1**	**CC 0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**/FC**
writes:	x	x	x	x	x	0	x	x	1

8.4.3 EXP Establish the Exponential Value

Symbol



Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, L, D	Enable input
ENO	BOOL	I, Q, M, L, D	Enable output
IN	REAL	I, Q, M, L, D or constant	Input value: floating-point
OUT	REAL	I, Q, M, L, D	Output value: exponential value of the floating-point number

Description

EXP establishes the exponential value of a floating-point number on the basis e (=2,71828...).

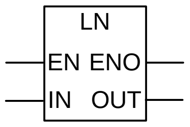
See also Evaluating the Bits of the Status Word.

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	x	x	x	x	x	0	x	x	1

8.4.4 LN Establish the Natural Logarithm

Symbol



Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, L, D	Enable input
ENO	BOOL	I, Q, M, L, D	Enable output
IN	REAL	I, Q, M, L, D or constant	Input value: floating-point
OUT	REAL	I, Q, M, L, D	Output value: natural logarithm of the floating-point number

Description

LN establishes the natural logarithm of a floating-point number.

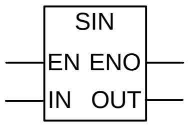
See also Evaluating the Bits of the Status Word.

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	x	x	x	x	x	0	x	x	1

8.4.5 SIN Establish the Sine Value

Symbol



Parameter	**Data Type**	**Memory Area**	**Description**
EN	BOOL	I, Q, M, L, D	Enable input
ENO	BOOL	I, Q, M, L, D	Enable output
IN	REAL	I, Q, M, L, D or constant	Input value: floating-point
OUT	REAL	I, Q, M, L, D	Output value: sine of the floating-point number

Description

SIN establishes the sine value of a floating-point number. The floating-point number represents an angle in a radian measure here.

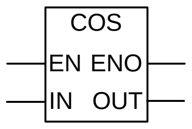
See also Evaluating the Bits of the Status Word.

Status word

	BR	**CC 1**	**CC 0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**/FC**
writes:	x	x	x	x	x	0	x	x	1

8.4.6 COS Establish the Cosine Value

Symbol



Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, L, D	Enable input
ENO	BOOL	I, Q, M, L, D	Enable output
IN	REAL	I, Q, M, L, D or constant	Input value: floating-point
OUT	REAL	I, Q, M, L, D	Output value: cosine of the floating-point number

Description

COS establishes the cosine value of a floating-point number. The floating-point number represents an angle in a radian measure here.

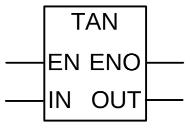
See also Evaluating the Bits of the Status Word.

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	x	x	x	x	x	0	x	x	1

8.4.7 TAN Establish the Tangent Value

Symbol



Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, L, D	Enable input
ENO	BOOL	I, Q, M, L, D	Enable output
IN	REAL	I, Q, M, L, D or constant	Input value: floating-point
OUT	REAL	I, Q, M, L, D	Output value: tangent of the floating-point number

Description

TAN establishes the tangent value of a floating-point number. The floating-point number represents an angle in a radian measure here.

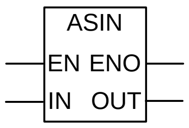
See also Evaluating the Bits of the Status Word.

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	x	x	x	x	x	0	x	x	1

8.4.8 ASIN Establish the Arc Sine Value

Symbol



Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, L, D	Enable input
ENO	BOOL	I, Q, M, L, D	Enable output
IN	REAL	I, Q, M, L, D or constant	Input value: floating-point
OUT	REAL	I, Q, M, L, D	Output value: arc sine of the floating-point number

Description

ASIN establishes the arc sine value of a floating-point number with a definition range -1 <= input value <= 1. The result represents an angle in a radian measure within the range

$$-\pi/2 \leq \text{output value} \leq +\pi/2$$

where $\pi = 3.1415....$

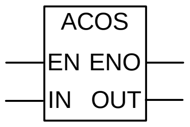
See also Evaluating the Bits of the Status Word.

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	x	x	x	x	x	0	x	x	1

8.4.9 ACOS Establish the Arc Cosine Value

Symbol



Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, L, D	Enable input
ENO	BOOL	I, Q, M, L, D	Enable output
IN	REAL	I, Q, M, L, D or constant	Input value: floating-point
OUT	REAL	I, Q, M, L, D	Output value: arc cosine of the floating-point number

Description

ACOS establishes the arc cosine value of a floating-point number with a definition range -1 <= input value <= 1. The result represents an angle in a radian measure within the range

$$0 \leq \text{output value} \leq +\pi$$

where $\pi = 3.1415....$

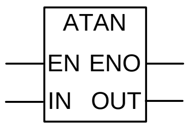
See also Evaluating the Bits of the Status Word.

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	x	x	x	x	x	0	x	x	1

8.4.10 ATAN Establish the Arc Tangent Value

Symbol



Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, L, D	Enable input
ENO	BOOL	I, Q, M, L, D	Enable output
IN	REAL	I, Q, M, L, D or constant	Input value: floating-point
OUT	REAL	I, Q, M, L, D	Output value: arc tangent of the floating-point number

Description

ATAN establishes the arc tangent value of a floating-point number. The result represents an angle in a radian measure within the range

$-\pi/2 \leq \text{output value} \leq +\pi/2$

where π = 3.1415....

See also Evaluating the Bits of the Status Word.

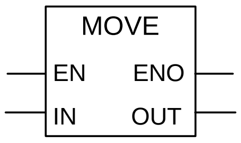
Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	x	x	x	x	x	0	x	x	1

9 Move Instructions

9.1 MOVE Assign a Value

Symbol



Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, L, D	Enable input
ENO	BOOL	I, Q, M, L, D	Enable output
IN	All elementary data types with a length of 8, 16, or 32 bits	I, Q, M, L, D or constant	Source value
OUT	All elementary data types with a length of 8, 16, or 32 bits	I, Q, M, L, D	Destination address

Description

MOVE (Assign a Value) is activated by the Enable EN Input. The value specified at the IN input is copied to the address specified at the OUT output. ENO has the same logic state as EN. **MOVE** can copy only BYTE, WORD, or DWORD data objects. User-defined data types like arrays or structures have to be copied with the system function "BLKMOVE" (SFC 20).

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	1	-	-	-	-	0	1	1	1

MCR (Master Control Relay) dependency

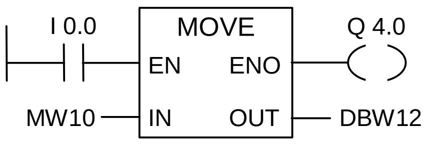
MCR dependency is activated only if a Move box is placed inside an active MCR zone. Within an activated MCR zone, if the MCR is on and there is power flow to the enable input; the addressed data is copied as described above. If the MCR is off, and a MOVE is executed, a logic "0" is written to the specified OUT address regardless of current IN states.

Note

When moving a value to a data type of a different length, higher-value bytes are truncated as necessary or filled up with zeros:

Example: Double Word	**1111 1111**	**0000 1111**	**1111 0000**	**0101 0101**
Move	**Result**			
to a double word:	1111 1111	0000 1111	1111 0000	0101 0101
to a byte:				0101 0101
to a word:			1111 0000	0101 0101
Example: Byte				**1111 0000**
Move	**Result**			
to a byte:				1111 0000
to a word:			0000 0000	1111 0000
to a double word:	0000 0000	0000 0000	0000 0000	1111 0000

Example



The instruction is executed if I0.0 is "1". The content of MW10 is copied to data word 12 of the currently open DB.

Q4.0 is "1" if the instruction is executed.

If the example rungs are within an activated MCR zone:

- When MCR is on, MW10 data is copied to DBW12 as described above.
- When MCR is off, "0" is written to DBW12.

10 Program Control Instructions

10.1 Overview of Program Control Instructions

Description

The following program control instructions are available:

• ---(CALL)	Call FC SFC from Coil (without Parameters)
• CALL_FB	Call FB from Box
• CALL_FC	Call FC from Box
• CALL_SFB	Call System FB from Box
• CALL_SFC	Call System FC from Box
• Call Multiple Instance	
• Call Block from a Library	
• Important Notes on Using MCR Functions	
• ---(MCR<)	Master Control Relay On
• ---(MCR>)	Master Control Relay Off
• ---(MCRA)	Master Control Relay Activate
• ---(MCRD)	Master Control Relay Deactivate
• RET	Return

10.2 ---(Call) Call FC SFC from Coil (without Parameters)

Symbol

<FC/SFC no.>

---(CALL)

Parameter	Data Type	Memory Area	Description
<FC/SFC no.>	BLOCK_FC BLOCK_SFC	-	Number of FC/SFC; range depends on CPU

Description

---(Call) (Call FC or SFC without Parameters) is used to call a function (FC) or system function (SFC) that has no passed parameters. A call is only executed if RLO is "1" at the CALL coil. If **---(Call)** is executed,

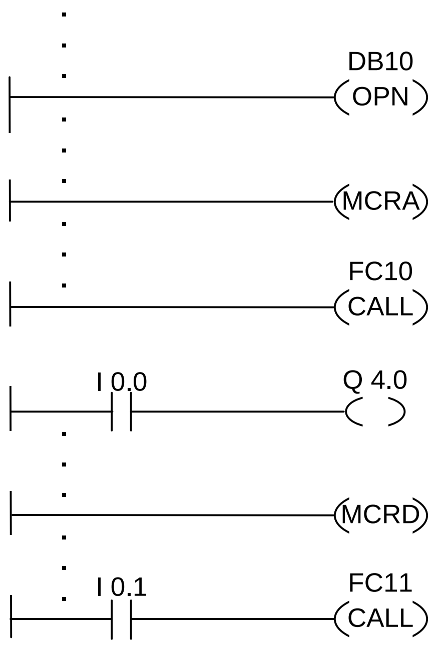
- The return address of the calling block is stored,
- The previous local data area is replaced by the current local data area,
- The MA bit (active MCR bit) is shifted to the B stack,
- A new local data area for the called function is created.

After this, program processing continues in the called FC or SFC.

Status word

		BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Unconditional:	writes:	-	-	-	-	0	0	1	-	0
Conditional:	writes:	-	-	-	-	0	0	1	1	0

Example



The Ladder rungs shown above are program sections from a function block written by a user. In this FB, DB10 is opened and MCR functionality is activated. If the unconditional call of FC10 is executed, the following occurs:

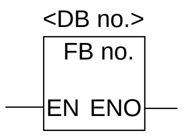
The return address of the calling FB plus selection data for DB10 and for the instance data block for the calling FB are saved. The MA bit, set to "1" in the MCRA instruction, is pushed to the B stack and then set to "0" for the called block (FC10). Program processing continues in FC10. If MCR functionality is required by FC10, it must be re-activated within FC10. When FC10 is finished, program processing returns to the calling FB. The MA bit is restored, DB10 and the instance data block for the user-written FB become the current DBs again, regardless of which DBs FC10 has used. The program continues with the next rung by assigning the logic state of I0.0 to output Q4.0. The call of FC11 is a conditional call. It is only executed if I0.1 is "1". If it is executed, the process of passing program control to and returning from FC11 is the same as was described for FC10.

Note

After returning to the calling block, the previously open DB is not always open again. Please make sure you read the note in the README file.

10.3 CALL_FB Call FB from Box

Symbol



The symbol depends on the FB (whether it has parameters and how many of them). It must have the EN, ENO, and the name or number of the FB.

Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, L, D	Enable input
ENO	BOOL	I, Q, M, L, D	Enable output
FB no. DB no.	BLOCK_FB BLOCK_DB	- -	Number of FB/DB; range depends on CPU

Description

CALL_FB (Call a Function Block from a Box) executed if EN is "1". If CALL_FB is executed,

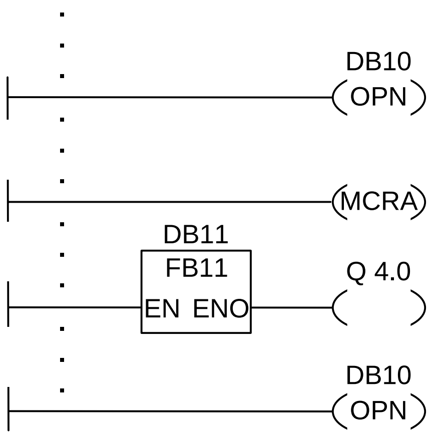
- The return address of the calling block is stored,
- The selection data for the two current data blocks (DB and instance DB) are stored,
- The previous local data area is replaced by the current local data area,
- The MA bit (active MCR bit) is shifted to the B stack,
- A new local data area for the called function block is created.

After this, program processing continues within the called function block. The BR bit is scanned in order to find out the ENO. The user has to assign the required state (error evaluation) to the BR bit in the called block using ---(SAVE).

Status word

		BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Unconditional:	writes:	x	-	-	-	0	0	x	x	x
Conditional:	writes:	-	-	-	-	0	0	x	x	x

Example



The Ladder rungs shown above are program sections from a function block written by a user. In this FB, DB10 is opened and MCR functionality is activated. If the unconditional call of FB11 is executed, the following occurs:

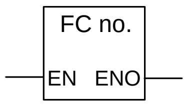
The return address of the calling FB plus selection data for DB10 and for the instance data block for the calling FB are saved. The MA bit, set to "1" in the MCRA instruction, is pushed to the B stack and then set to "0" for the called block (FB11). Program processing continues in FB11. If MCR functionality is required by FB11, it must be re-activated within FB11. The state of the RLO must be saved in the BR bit by the instruction ---(SAVE) in order to be able to evaluate errors in the calling FB. When FB11 is finished, program processing returns to the calling FB. The MA bit is restored and the instance data block of the user-written FB is opened again. If the FB11 is processed correctly, ENO = "1" and therefore Q4.0 = "1".

Note

When opening an FB or SFB, the number of the previously opened DB is lost. The required DB has to be reopened.

10.4 CALL_FC Call FC from Box

Symbol



The symbol depends on the FC (whether it has parameters and how many of them). It must have EN, ENO, and the name or number of the FC.

Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, L, D	Enable input
ENO	BOOL	I, Q, M, L, D	Enable output
FC no.	BLOCK_FC	-	Number of FC; range depends on CPU

Description

CALL_FC (Call a Function from a Box) is used to call a function (FC). The call is executed if EN is "1". If CALL_FC is executed,

- The return address of the calling block is stored,
- The previous local data area is replaced by the current local data area,
- The MA bit (active MCR bit) is shifted to the B stack,
- A new local data area for the called function is created.

After this, program processing continues in the called function.

The BR bit is scanned in order to find out the ENO. The user has to assign the required state (error evaluation) to the BR bit in the called block using **---(SAVE)**.

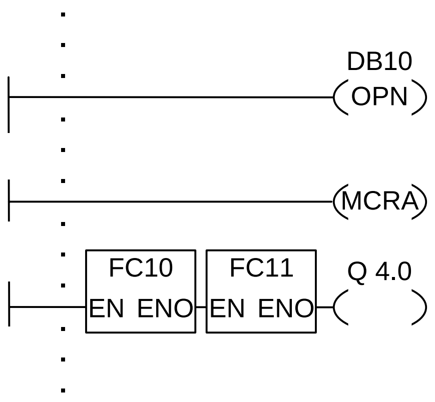
If you call a function and the variable declaration table of the called block has IN, OUT, and IN_OUT declarations, these variables are added in the program for the calling block as a list of formal parameters.

When calling the function, you **must** assign actual parameters to the formal parameters at the call location. Any initial values in the function declaration have no significance.

Status word

		BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Unconditional:	writes:	x	-	-	-	0	0	x	x	x
Conditional:	writes:	-	-	-	-	0	0	x	x	x

Example



The Ladder rungs shown above are program sections from a function block written by a user. In this FB, DB10 is opened and MCR functionality is activated. If the unconditional call of FC10 is executed, the following occurs:

The return address of the calling FB plus selection data for DB10 and for the instance data block for the calling FB are saved. The MA bit, set to "1" in the MCRA instruction, is pushed to the B stack and then set to "0" for the called block (FC10). Program processing continues in FC10. If MCR functionality is required by FC10, it must be re-activated within FC10. The state of the RLO must be saved in the BR bit by the instruction ---(SAVE) in order to be able to evaluate errors in the calling FB. When FC10 is finished, program processing returns to the calling FB. The MA bit is restored. After execution of FC10, program processing is continued in the calling FB depending on the ENO:

ENO = "1" FC11 is processed

ENO = "0" processing starts in the next network

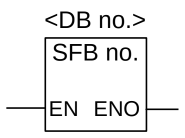
If FC11 is also processed correctly, ENO = "1" and therefore Q**4.0 = "1"**.

Note

After returning to the calling block, the previously open DB is not always open again. Please make sure you read the note in the README file.

10.5 CALL_SFB Call System FB from Box

Symbol



The symbol depends on the SFB (whether it has parameters and how many of them). It must have the EN, ENO, and the name or number of the SFB.

Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, L, D	Enable input
ENO	BOOL	I, Q, M, L, D	Enable output
SFB no. DB no.	BLOCK_SFB BLOCK_DB	- -	Number of SFB; range depends on CPU

Description

CALL_SFB (Call a System Function Block from a Box) is executed if EN is "1". If CALL_SFB is executed,

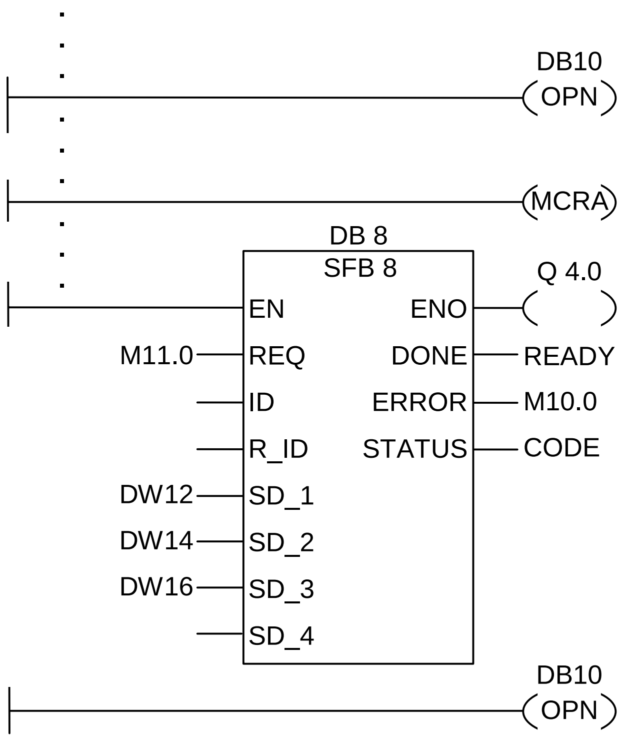
- The return address of the calling block is stored,
- The selection data for the two current data blocks (DB and instance DB) are stored,
- The previous local data area is replaced by the current local data area,
- The MA bit (active MCR bit) is shifted to the B stack,
- A new local data area for the called system function block is created.

Program processing then continues in the called SFB. ENO is "1" if the SFB was called (EN = "1") and no error occurs.

Status word

		BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Unconditional:	writes:	x	-	-	-	0	0	x	x	x
Conditional:	writes:	-	-	-	-	0	0	x	x	x

Example



The Ladder rungs shown above are program sections from a function block written by a user. In this FB, DB10 is opened and MCR functionality is activated. If the unconditional call of SFB8 is executed, the following occurs:

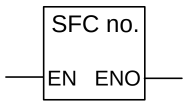
The return address of the calling FB plus selection data for DB10 and for the instance data block for the calling FB are saved. The MA bit, set to "1" in the MCRA instruction, is pushed to the B stack and then set to "0" for the called block (SFB8). Program processing continues in SFB8. When SFB8 is finished, program processing returns to the calling FB. The MA bit is restored and the instance data block of the user-written FB becomes the current instance DB. If the SFB8 is processed correctly, ENO = "1" and therefore Q4.0 = "1".

Note

When opening an FB or SFB, the number of the previously opened DB is lost. The required DB has to be reopened.

10.6 CALL_SFC Call System FC from Box

Symbol



The symbol depends on the SFC (whether it has parameters and how many of them). It must have EN, ENO, and the name or number of the SFC.

Parameter	Data Type	Memory Area	Description
EN	BOOL	-	Enable input
ENO	BOOL	-	Enable output
SFC no.	BLOCK_SFC	-	Number of SFC; range depends on CPU

Description

CALL_SFC (Call a System Function from a Box) is used to call an SFC. The call is executed if EN is "1". If CALL_SFC is executed,

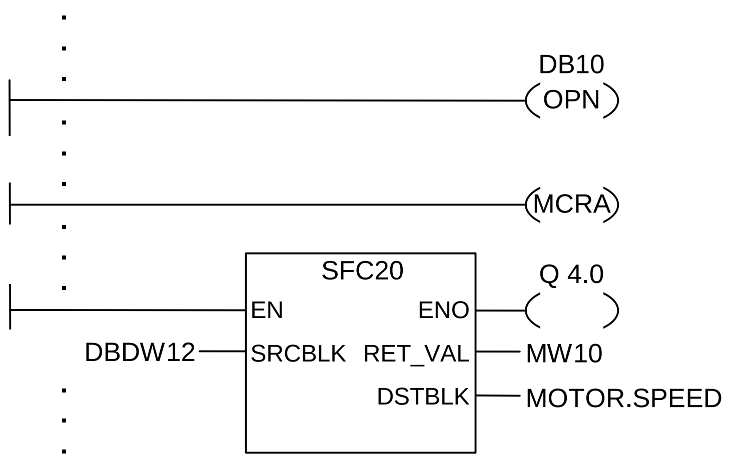
- The return address of the calling block is stored,
- The previous local data area is replaced by the current local data area,
- The MA bit (active MCR bit) is shifted to the B stack,
- A new local data area for the called system function is created.

After this, program processing continues in the called SFC. ENO is "1" if the SFC was called (EN = "1") and no error occurs.

Status word

		BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
Unconditional:	writes:	x	-	-	-	0	0	x	x	x
Conditional:	writes:	-	-	-	-	0	0	x	x	x

Example



The Ladder rungs shown above are program sections from a function block written by a user. In this FB, DB10 is opened and MCR functionality is activated. If the unconditional call of SFC20 is executed, the following occurs:

The return address of the calling FB plus selection data for DB10 and for the instance data block for the calling FB are saved. The MA bit, set to "1" in the MCRA instruction, is pushed to the B stack and then set to "0" for the called block (SFC20). Program processing continues in SFC20. When SFC20 is finished, program processing returns to the calling FB. The MA bit is restored.

After processing the SFC20, the program is continued in the calling FB depending on the ENO:

ENO = "1" Q4.0 = "1"

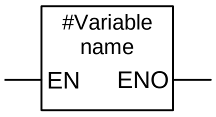
ENO = "0" Q4.0 = "0"

Note

After returning to the calling block, the previously open DB is not always open again. Please make sure you read the note in the README file.

10.7 Call Multiple Instance

Symbol



Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, L, D	Enable input
ENO	BOOL	I, Q, M, L, D	Enable output
#Variable name	FB, SFB	-	Name of multiple instance

Description

A multiple instance is created by declaring a static variable with the data type of a function block. Only multiple instances that have already been declared are included in the program element catalog. The symbol for a multiple instance varies depending on whether and how many parameters are present. EN, ENO and the variable name are always present.

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	-	-	-	-	0	0	x	x	x

10.8 Call Block from a Library

The libraries available in the SIMATIC Manager can be used here to select a block that

- Is integrated in your CPU operating system ("Standard Library" library for STEP 7 projects in version 3 and "stdlibs (V2)" for STEP 7 projects in version 2)
- You saved yourself in a library because you wanted to use it a number of times.

10.9 Important Notes on Using MCR Functions

Take care with blocks in which the Master Control Relay was activated with MCRA:

- If the MCR is deactivated, the value 0 is written by all assignments in program segments between ---(MCR<) and ---(MCR>). This is valid for **all** boxes which contain an assignment, including the parameter transfer to blocks.
- The MCR is deactivated if the RLO was = 0 before an **MCR<** instruction.

Danger: PLC in STOP or undefined runtime characteristics!

The compiler also uses write access to local data behind the temporary variables defined in VAR_TEMP for calculating addresses. This means the following command sequences will set the PLC to STOP or lead to undefined runtime characteristics:

Formal parameter access

- Access to components of complex FC parameters of the type STRUCT, UDT, ARRAY, STRING
- Access to components of complex FB parameters of the type STRUCT, UDT, ARRAY, STRING from the IN_OUT area in a block with multiple instance capability (version 2 block).
- Access to parameters of a function block with multiple instance capability (version 2 block) if its address is greater than 8180.0.
- Access in a function block with multiple instance capability (version 2 block) to a parameter of the type BLOCK_DB opens DB0. Any subsequent data access sets the CPU to STOP. T 0, C 0, FC0, or FB0 are also always used for TIMER, COUNTER, BLOCK_FC, and BLOCK_FB.

Parameter passing

- Calls in which parameters are transferred.

LAD/FBD

- T branches and midline outputs in Ladder or FBD starting with RLO = 0.

Remedy

Free the above commands from their dependence on the MCR:

- Deactivate the Master Control Relay using the **Master Control Relay Deactivate** instruction before the statement or network in question.
- Activate the Master Control Relay again using the **Master Control Relay Activate** instruction after the statement or network in question.

10.10 ---(MCR<) Master Control Relay On

Important Notes on Using MCR Functions

Symbol

---(MCR<)

Description

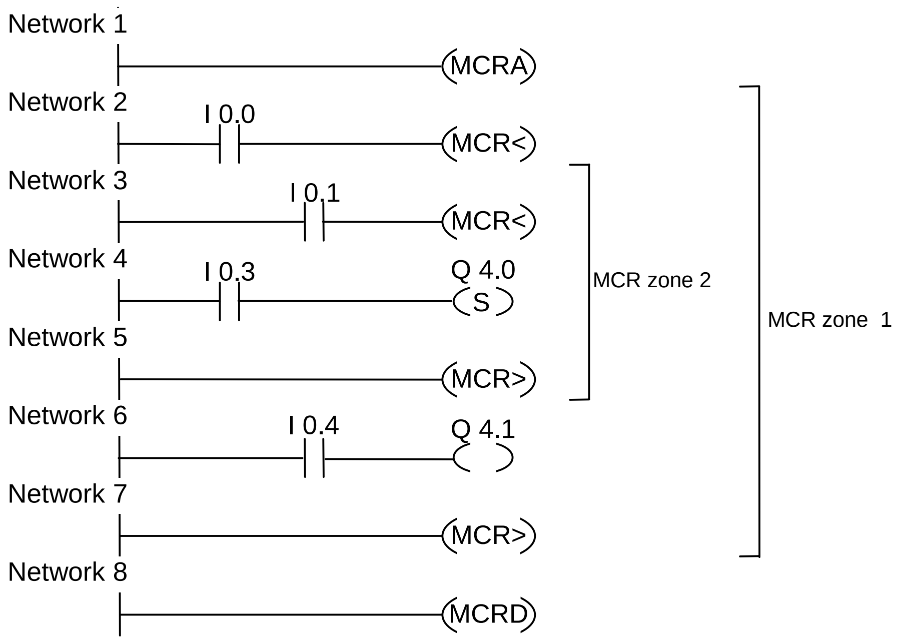
---(MCR<) (Open a Master Control Relay zone) saves the RLO in the MCR stack. The MCR nesting stack is a LIFO stack (last in, first out) and only 8 stack entries (nesting levels) are possible. If the stack is already full, the ---(MCR<) function produces an MCR stack fault (MCRF). The following elements are MCR-dependent and influenced by the RLO state that is saved to the MCR stack while opening an MCR zone:

• --(#)	Midline Output
• --( )	Output
• --(S)	Set Output
• --(R)	Reset Output
• RS	Reset Flip Flop
• SR	Set Flip Flop
• MOVE	Assign a Value

Status word

	BR	**CC 1**	**CC 0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**/FC**
writes:	-	-	-	-	-	0	1	-	0

Example



MCR functionality is activated by the MCRA rung. It is then possible to create up to eight nested MCR zones. In the example there are two MCR zones. The functions are executed as follows:

I0.0 = "1" (MCR is ON for zone 1): the logic state of I0.4 is assigned to Q4.1

I0.0 = "0" (MCR is OFF for zone 1): Q4.1 is "0" regardless of the logic state of I0.4

I0.1 = "1" (MCR is ON for zone 2): Q4.0 is set to "1" if I0.3 is "1"

I0.1 = "0" (MCR is OFF for zone 2): Q4.0 remains unchanged regardless the logic state of I0.3

10.11 ---(MCR>) Master Control Relay Off

Important Notes on Using MCR Functions

Symbol

---(MCR>)

Description

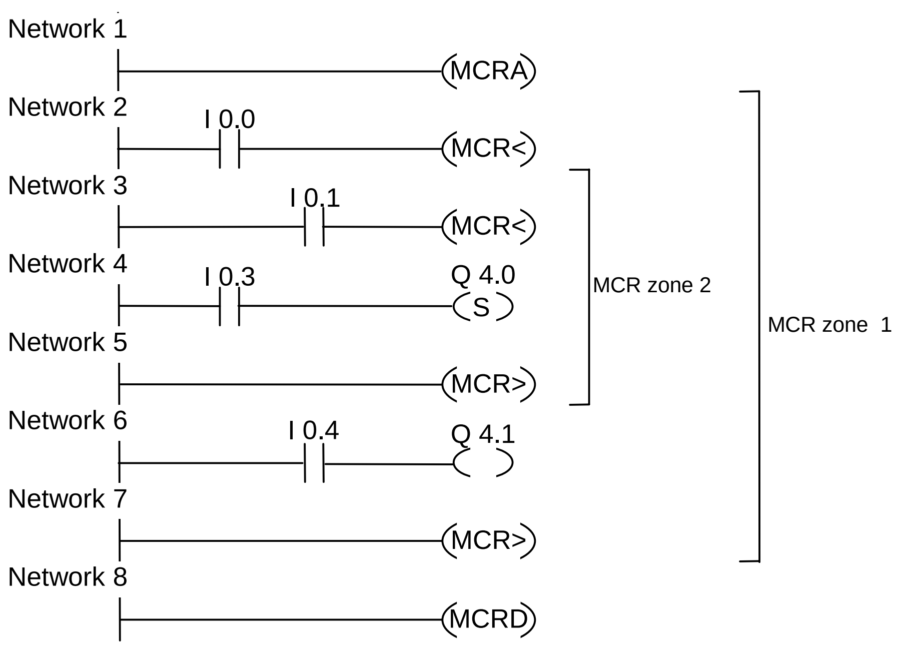
---(MCR>) (close the last opened MCR zone) removes an RLO entry from the MCR stack. The MCR nesting stack is a LIFO stack (last in, first out) and only 8 stack entries (nesting levels) are possible. If the stack is already empty, ---(MCR>) produces an MCR stack fault (MCRF). The following elements are MCR-dependent and influenced by the RLO state that is saved to the MCR stack while opening the MCR zone:

• --(#)	Midline Output
• --( )	Output
• --(S)	Set Output
• --(R)	Reset Output
• RS	Reset Flip Flop
• SR	Set Flip Flop
• MOVE	Assign a Value

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	-	-	-	-	-	0	1	-	0

Example



MCR functionality is activated by the ---(MCRA) rung. It is then possible to create up to eight nested MCR zones. In the example there are two MCR zones. The first ---(MCR>) (MCR OFF) rung belongs to the second ---(MCR<) (MCR ON) rung. All rungs between belong to the MCR zone 2. The functions are executed as follows:

I0.0 = "1": the logic state of I0.4 is assigned to Q4.1

I0.0 = "0": Q4.1 is "0" regardless of the logic state of I0.4

I0.1 = "1": Q4.0 is set to "1" if I0.3 is "1"

I0.1 = "0": Q4.0 remains unchanged regardless of the logic state of I0.3

10.12 ---(MCRA) Master Control Relay Activate

Important Notes on Using MCR Functions

Symbol

---(MCRA)

Description

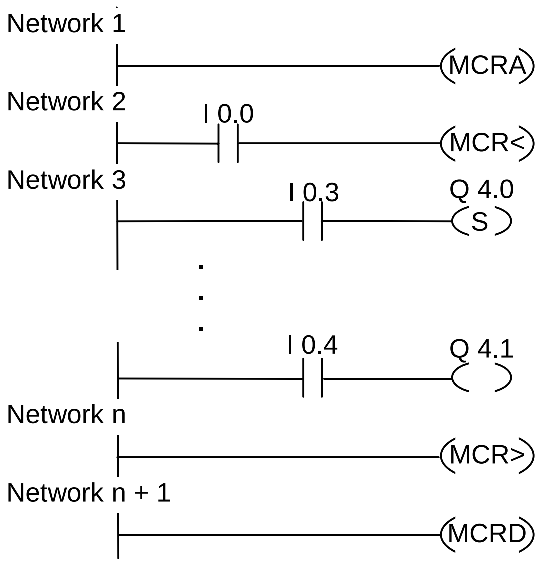
---(MCRA) (Activate Master Control Relay) activates master control relay function. After this command, it is possible to program MCR zones with the commands:

- ---(MCR<)
- ---(MCR>)

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	-	-	-	-	-	-	-	-	-

Example



MCR functionality is activated by the MCRA rung. The rungs between the MCR< and the MCR> (outputs Q4.0, Q4.1) are executed as follows:

I0.0 = "1" (MCR is ON): Q4.0 is set to "1" if I0.3 is logic "1", or will remain unchanged if I0.3 is "0" and the logic state of I0.4 is assigned to Q4.1

I0.0 = "0" (MCR is OFF): Q4.0 remains unchanged regardless of the logic state of I0.3 and Q4.1 is "0" regardless of the logic state of I0.4

In the next rung, the instruction ---(MCRD) deactivates the MCR. This means that you cannot program any more MCR zones using the instruction pair ---(MCR<) and ---(MCR>).

10.13 ---(MCRD) Master Control Relay Deactivate

Important Notes on Using MCR Functions

Symbol

---(MCRD)

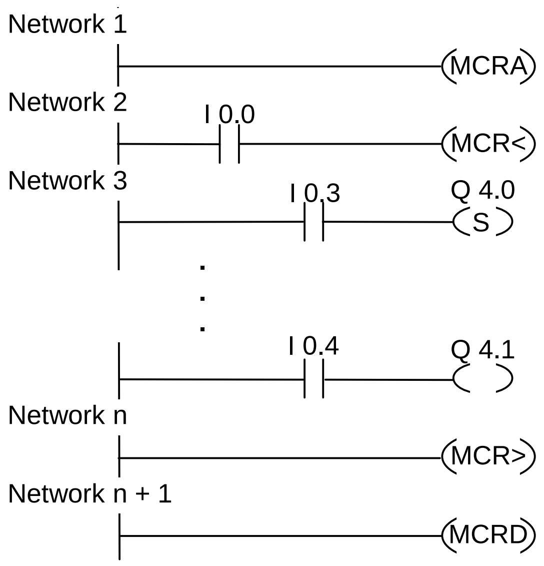
Description

---(MCRD) (Deactivate Master Control Relay) deactivates MCR functionality. After this command, you cannot program MCR zones.

Status word

	BR	**CC 1**	**CC 0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**/FC**
writes:	-	-	-	-	-	-	-	-	-

Example



MCR functionality is activated by the MCRA rung. The rungs between the MCR< and the MCR> (outputs Q4.0, Q4.1) are executed as follows:

I0.0 = "1" (MCR is ON): Q4.0 is set to "1" if I0.3 is logic "1" and the logic state of I0.4 is assigned to Q4.1.

I0.0 = "0" (MCR is OFF): Q4.0 remains unchanged regardless of the logic state of I0.3 and Q4.1 is "0" regardless of the logic state of I0.4.

In the next rung, the instruction ---(MCRD) deactivates the MCR. This means that you cannot program any more MCR zones using the instruction pair ---(MCR<) and ---(MCR>).

10.14 ---(RET) Return

Symbol

---(RET)

Description

RET (Return) is used to conditionally exit blocks. For this output, a preceding logic operation is required.

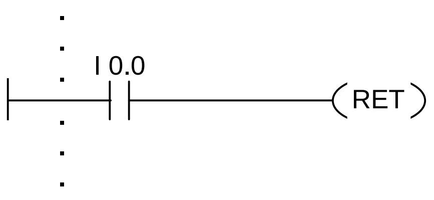
Status word

Conditional Return (Return if RLO = "1"):

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	*	-	-	-	0	0	1	1	0

* The operation **RET** is shown internally in the sequence "SAVE; BEC, ". This also affects the BR bit.

Example



The block is exited if I0.0 is "1".

11 Shift and Rotate Instructions

11.1 Shift Instructions

11.1.1 Overview of Shift Instructions

Description

You can use the Shift instructions to move the contents of input IN bit by bit to the left or the right (see also CPU Registers). Shifting to the left multiplies the contents of input IN by 2 to the power n (2^n); shifting to the right divides the contents of input IN by 2 to the power n (2^n). For example, if you shift the binary equivalent of the decimal value 3 to the left by 3 bits, you obtain the binary equivalent of the decimal value 24 in the accumulator. If you shift the binary equivalent of the decimal value 16 to the right by 2 bits, you obtain the binary equivalent of the decimal value 4 in the accumulator.

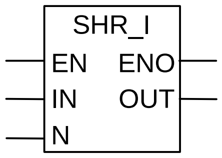
The number that you supply for input parameter N indicates the number of bits by which to shift. The bit places that are vacated by the Shift instruction are either filled with zeros or with the signal state of the sign bit (a 0 stands for positive and a 1 stands for negative). The signal state of the bit that is shifted last is loaded into the CC 1 bit of the status word. The CC 0 and OV bits of the status word are reset to 0. You can use jump instructions to evaluate the CC 1 bit.

The following shift instructions are available:

• SHR_I	Shift Right Integer
• SHR_DI	Shift Right Double Integer
• SHL_W	Shift Left Word
• SHR_W	Shift Right Word
• SHL_DW	Shift Left Double Word
• SHR_DW	Shift Right Double Word

11.1.2 SHR_I Shift Right Integer

Symbol

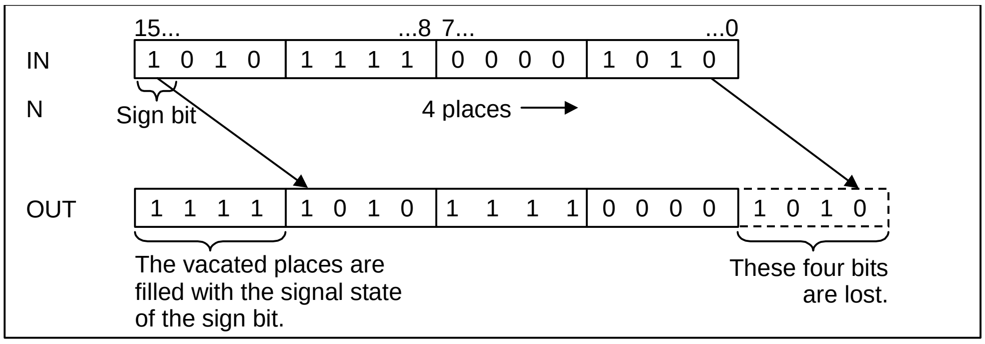


Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, L, D	Enable input
ENO	BOOL	I, Q, M, L, D	Enable output
IN	INT	I, Q, M, L, D	Value to shift
N	WORD	I, Q, M, L, D	Number of bit positions to shift
OUT	INT	I, Q, M, L, D	Result of shift instruction

Description

SHR_I (Shift Right Integer) is activated by a logic "1" at the Enable (EN) Input. The SHR_I instruction is used to shift bits 0 to 15 of input IN bit by bit to the right. Bits 16 to 31 are not affected. The input N specifies the number of bits by which to shift. If N is larger than 16, the command acts as if N were equal to 16. The bit positions shifted in from the left to fill vacated bit positions are assigned the logic state of bit 15 (sign bit for the integer). This means these bit positions are assigned "0" if the integer is positive and "1" if the integer is negative. The result of the shift instruction can be scanned at output OUT. The CC 0 bit and the OV bit are set to "0" by SHR_I if N is not equal to 0.

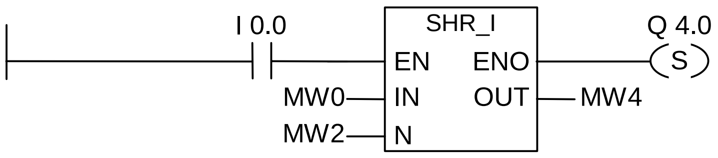
ENO has the same signal state as EN.



Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	x	x	x	x	-	x	x	x	1

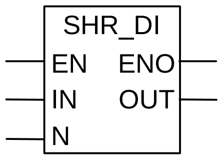
Example



The SHR_I box is activated by logic "1" at I0.0. MW0 is loaded and shifted right by the number of bits specified with MW2. The result is written to MW4. Q4.0 is set.

11.1.3 SHR_DI Shift Right Double Integer

Symbol



Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, L, D	Enable input
ENO	BOOL	I, Q, M, L, D	Enable output
IN	DINT	I, Q, M, L, D	Value to shift
N	WORD	I, Q, M, L, D	Number of bit positions to shift
OUT	DINT	I, Q, M, L, D	Result of shift instruction

Description

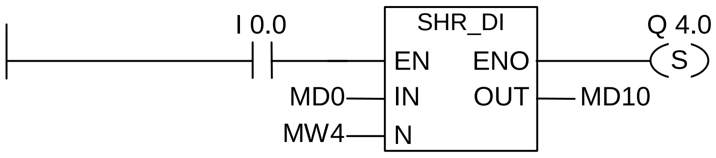
SHR_DI (Shift Right Double Integer) is activated by a logic "1" at the Enable (EN) Input. The SHR_DI instruction is used to shift bits 0 to 31 of input IN bit by bit to the right. The input N specifies the number of bits by which to shift. If N is larger than 32, the command acts as if N were equal to 32. The bit positions shifted in from the left to fill vacated bit positions are assigned the logic state of bit 31 (sign bit for the double integer). This means these bit positions are assigned "0" if the integer is positive and "1" if the integer is negative. The result of the shift instruction can be scanned at output OUT. The CC 0 bit and the OV bit are set to "0" by SHR_DI if N is not equal to 0.

ENO has the same signal state as EN.

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	x	x	x	x	-	x	x	x	1

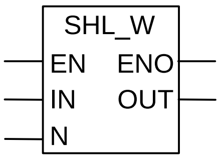
Example



The SHR_DI box is activated by logic "1" at I0.0. MD0 is loaded and shifted right by the number of bits specified with MW4. The result is written to MD10. Q4.0 is set.

11.1.4 SHL_W Shift Left Word

Symbol

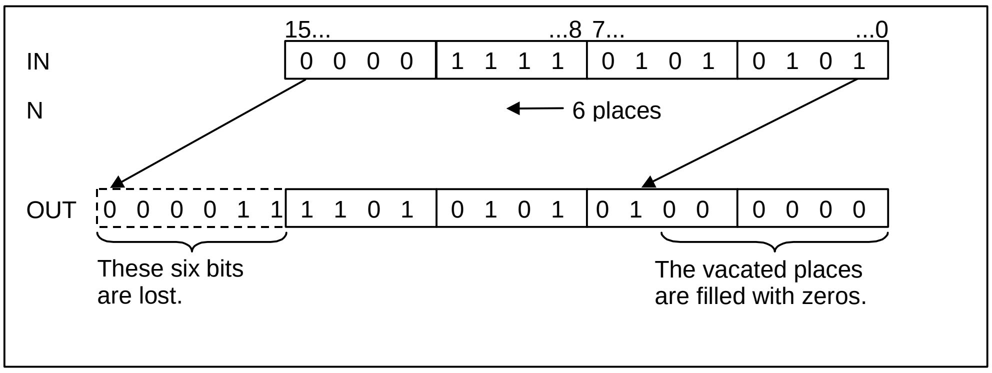


Parameter	**Data Type**	**Memory Area**	**Description**
EN	BOOL	I, Q, M, L, D	Enable input
ENO	BOOL	I, Q, M, L, D	Enable output
IN	WORD	I, Q, M, L, D	Value to shift
N	WORD	I, Q, M, L, D	Number of bit positions to shift
OUT	WORD	I, Q, M, L, D	Result of shift instruction

Description

SHL_W (Shift Left Word) is activated by a logic "1" at the Enable (EN) Input. The SHL_W instruction is used to shift bits 0 to 15 of input IN bit by bit to the left. Bits 16 to 31 are not affected. The input N specifies the number of bits by which to shift. If N is larger than 16, the command writes a "0" at output OUT and sets the bits CC 0 and OV in the status word to "0". N zeros are also shifted in from the right to fill vacated bit positions. The result of the shift instruction can be scanned at output OUT. The CC 0 bit and the OV bit are set to "0" by SHL_W if N is not equal to 0.

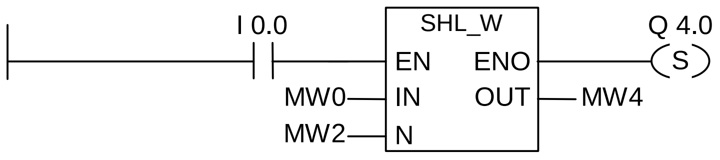
ENO has the same signal state as EN.



Status word

	BR	**CC 1**	**CC 0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**/FC**
writes:	x	x	x	x	-	x	x	x	1

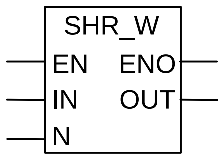
Example



The SHL_W box is activated by logic "1" at I0.0. MW0 is loaded and shifted left by the number of bits specified with MW2. The result is written to MW4. Q4.0 is set.

11.1.5 SHR_W Shift Right Word

Symbol



Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, L, D	Enable input
ENO	BOOL	I, Q, M, L, D	Enable output
IN	WORD	I, Q, M, L, D	Value to shift
N	WORD	I, Q, M, L, D	Number of bit positions to shift
OUT	WORD	I, Q, M, L, D	Result word of shift instruction

Description

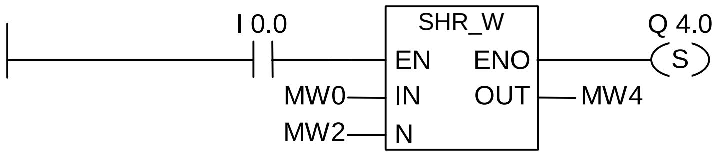
SHR_W (Shift Right Word) is activated by a logic "1" at the Enable (EN) Input. The SHR_W instruction is used to shift bits 0 to 15 of input IN bit by bit to the right. Bits 16 to 31 are not affected. The input N specifies the number of bits by which to shift. If N is larger than 16, the command writes a "0" at output OUT and sets the bits CC 0 and OV in the status word to "0". N zeros are also shifted in from the left to fill vacated bit positions. The result of the shift instruction can be scanned at output OUT. The CC 0 bit and the OV bit are set to "0" by SHR_W if N is not equal to 0.

ENO has the same signal state as EN.

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	x	x	x	x	-	x	x	x	1

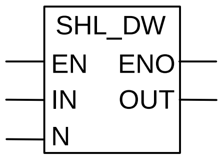
Example



The SHR_W box is activated by logic "1" at I0.0. MW0 is loaded and shifted right by the number of bits specified with MW2. The result is written to MW4. Q4.0 is set.

11.1.6 SHL_DW Shift Left Double Word

Symbol



Parameter	**Data Type**	**Memory Area**	**Description**
EN	BOOL	I, Q, M, L, D	Enable input
ENO	BOOL	I, Q, M, L, D	Enable output
IN	DWORD	I, Q, M, L, D	Value to shift
N	WORD	I, Q, M, L, D	Number of bit positions to shift
OUT	DWORD	I, Q, M, L, D	Result double word of shift instruction

Description

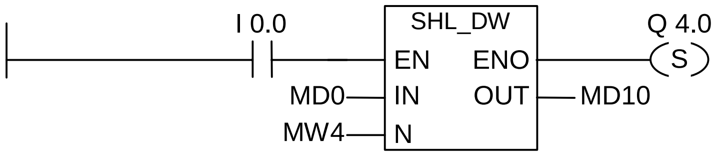
SHL_DW (Shift Left Double Word) is activated by a logic "1" at the Enable (EN) Input. The SHL_DW instruction is used to shift bits 0 to 31 of input IN bit by bit to the left. The input N specifies the number of bits by which to shift. If N is larger than 32, the command writes a "0" at output OUT and sets the bits CC 0 and OV in the status word to "0". N zeros are also shifted in from the right to fill vacated bit positions. The result double word of the shift instruction can be scanned at output OUT. The CC 0 bit and the OV bit are set to "0" by SHL_DW if N is not equal to 0.

ENO has the same signal state as EN.

Status word

	BR	**CC 1**	**CC 0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**/FC**
writes:	x	x	x	x	-	x	x	x	1

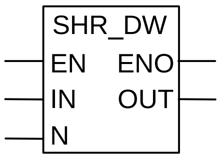
Example



The SHL_DW box is activated by logic "1" at I0.0. MD0 is loaded and shifted left by the number of bits specified with MW4. The result is written to MD10. Q4.0 is set.

11.1.7 SHR_DW Shift Right Double Word

Symbol

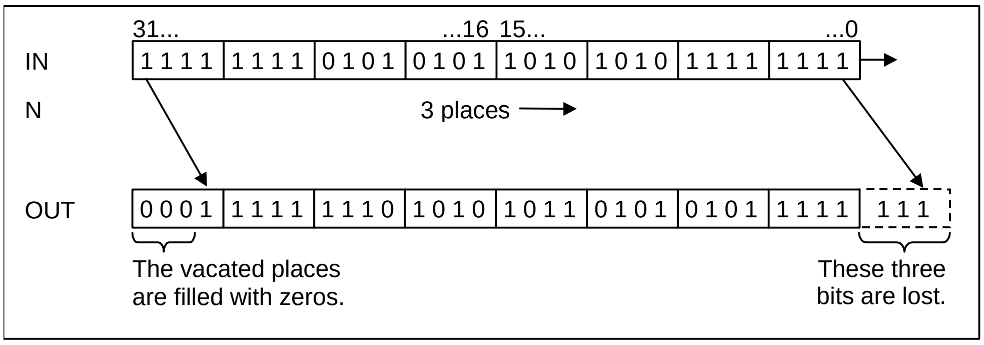


Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, L, D	Enable input
ENO	BOOL	I, Q, M, L, D	Enable output
IN	DWORD	I, Q, M, L, D	Value to shift
N	WORD	I, Q, M, L, D	Number of bit positions to shift
OUT	DWORD	I, Q, M, L, D	Result double word of shift instruction

Description

SHR_DW (Shift Right Double Word) is activated by a logic "1" at the Enable (EN) Input. The SHR_DW instruction is used to shift bits 0 to 31 of input IN bit by bit to the right. The input N specifies the number of bits by which to shift. If N is larger than 32, the command writes a "0" at output OUT and sets the bits CC 0 and OV in the status word to "0". N zeros are also shifted in from the left to fill vacated bit positions. The result double word of the shift instruction can be scanned at output OUT. The CC 0 bit and the OV bit are set to "0" by SHR_DW if N is not equal to 0.

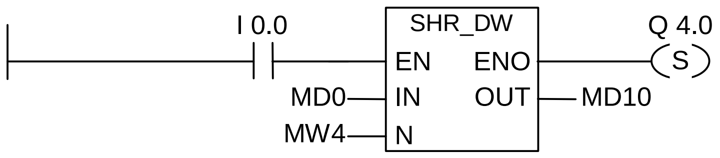
ENO has the same signal state as EN.



Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	x	x	x	x	-	x	x	x	1

Example



The SHR_DW box is activated by logic "1" at I0.0. MD0 is loaded and shifted right by the number of bits specified with MW4. The result is written to MD10. Q4.0 is set.

11.2 Rotate Instructions

11.2.1 Overview of Rotate Instructions

Description

You can use the Rotate instructions to rotate the entire contents of input IN bit by bit to the left or to the right. The vacated bit places are filled with the signal states of the bits that are shifted out of input IN.

The number that you supply for input parameter N specifies the number of bits by which to rotate.

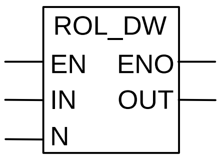
Depending on the instruction, rotation takes place via the CC 1 bit of the status word. The CC 0 bit of the status word is reset to 0.

The following rotate instructions are available:

- ROL_DW Rotate Left Double Word
- ROR_DW Rotate Right Double Word

11.2.2 ROL_DW Rotate Left Double Word

Symbol

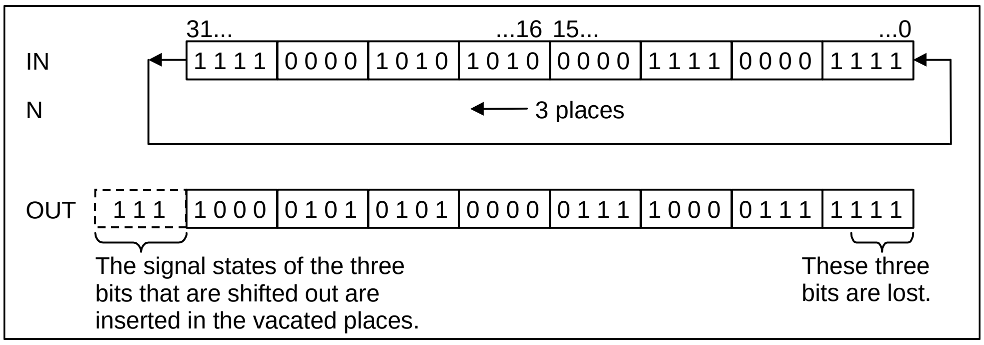


Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, L, D	Enable input
ENO	BOOL	I, Q, M, L, D	Enable output
IN	DWORD	I, Q, M, L, D	Value to rotate
N	WORD	I, Q, M, L, D	Number of bit positions to rotate
OUT	DWORD	I, Q, M, L, D	Result double word of rotate instruction

Description

ROL_DW (Rotate Left Double Word) is activated by a logic "1" at the Enable (EN) Input. The ROL_DW instruction is used to rotate the entire contents of input IN bit by bit to the left. The input N specifies the number of bits by which to rotate. If N is larger than 32, the double word IN is rotated by ((N-1) modulo 32)+1 positions. The bit positions shifted in from the right are assigned the logic states of the bits which were rotated out to the left. The result double word of the rotate instruction can be scanned at output OUT. The CC 0 bit and the OV bit are set to "0" by ROL_DW if N is not equal to 0.

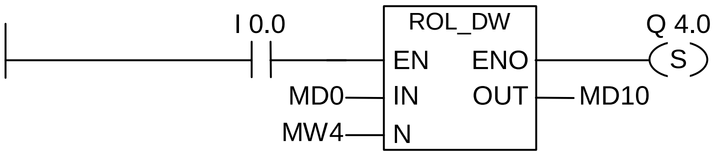
ENO has the same signal state as EN.



Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	x	x	x	x	-	x	x	x	1

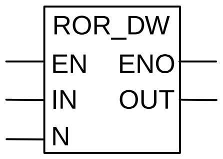
Example



The ROL_DW box is activated by logic "1" at I0.0. MD0 is loaded and rotated to the left by the number of bits specified with MW4. The result is written to MD10. Q4.0 is set.

11.2.3 ROR_DW Rotate Right Double Word

Symbol

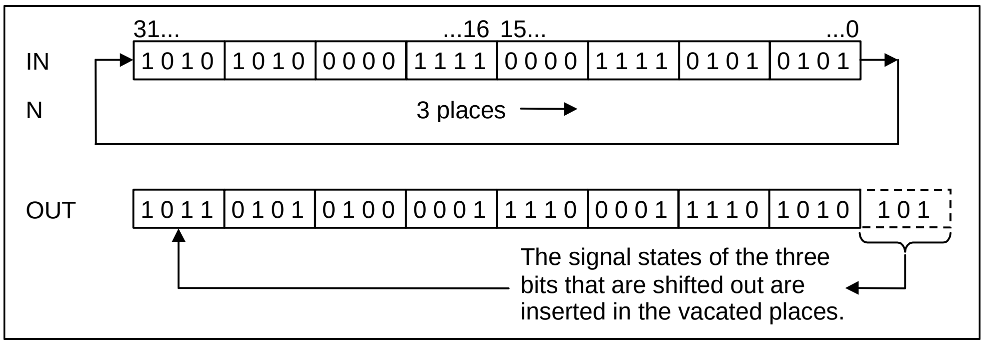


Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, L, D	Enable input
ENO	BOOL	I, Q, M, L, D	Enable output
IN	DWORD	I, Q, M, L, D	Value to rotate
N	WORD	I, Q, M, L, D	Number of bit positions to rotate
OUT	DWORD	I, Q, M, L, D	Result double word of rotate instruction

Description

ROR_DW (Rotate Right Double Word) is activated by a logic "1" at the Enable (EN) Input. The ROR_DW instruction is used to rotate the entire contents of input IN bit by bit to the right. The input N specifies the number of bits by which to rotate. If N is larger than 32, the double word IN is rotated by ((N-1) modulo 32)+1 positions. The bit positions shifted in from the left are assigned the logic states of the bits which were rotated out to the right. The result double word of the rotate instruction can be scanned at output OUT. The CC 0 bit and the OV bit are set to "0" by ROR_DW if N is not equal to 0.

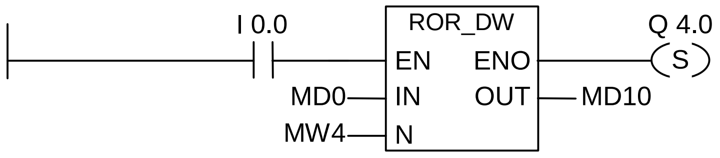
ENO has the same signal state as EN.



Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	x	x	x	x	-	x	x	x	1

Example



The ROR_DW box is activated by logic "1" at I0.0. MD0 is loaded and rotated to the right by the number of bits specified with MW4. The result is written to MD10. Q4.0 is set.

12 Status Bit Instructions

12.1 Overview of Statusbit Instructions

Description

The status bit instructions are bit logic instructions that work with the bits of the status word. Each of these instructions reacts to one of the following conditions that is indicated by one or more bits of the status word:

- The Binary Result bit (BR ---I I---) is set (that is, has a signal state of 1).
- A math function had an Overflow (OV ---I I---) or a Stored Overflow (OS ---I I---).
- The result of a math function is unordered (UO ---I I---).
- The result of a math function is related to 0 in one of the following ways: == 0, <> 0, > 0, < 0, >= 0, <= 0.

When a status bit instruction is connected in series, it combines the result of its signal state check with the previous result of logic operation according to the And truth table. When a status bit instruction is connected in parallel, it combines its result with the previous RLO according to the Or truth table.

Status word

The status word is a register in the memory of your CPU that contains bits that you can reference in the address of bit and word logic instructions. Structure of the status word:

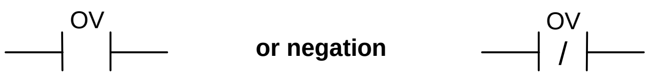
2^{15}... ...2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC

You can evaluate the bits in the status word

- by Integer Math Functions,
- by Floating-point Functions.

12.2 OV ---| |--- Exception Bit Overflow

Symbol



Description

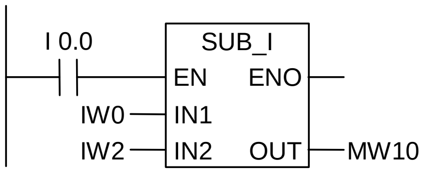
OV ---| |--- (Exception Bit Overflow) or **OV ---| / |---** (Negated Exception Bit Overflow) contact symbols are used to recognize an overflow in the last math function executed. This means that after the function executes, the result of the instruction is outside the permissible negative or positive range. Used in series, the result of the scan is linked to the RLO by AND, used in parallel, it is linked to the RLO by OR.

Status word

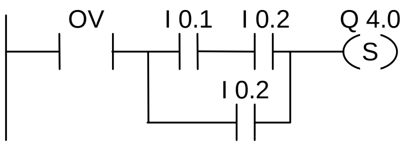
	BR	**CC 1**	**CC 0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**/FC**
writes:	-	-	-	-	-	x	x	x	1

Example

Network 1



Network 2



The box is activated by signal state "1" at I0.0. If the result of the math function "IW0 - IW2" is outside the permissible range for an integer, the OV bit is set.

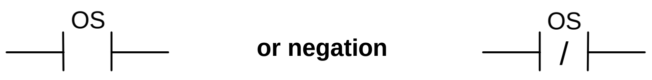
The signal state scan at OV is "1". Q4.0 is set if the scan of OV is signal state "1" and the RLO of network 2 is "1".

Note

The scan with OV is only necessary because of the two separate networks. Otherwise it is possible to take the ENO output of the math function that is "0" if the result is outside the permissible range.

12.3 OS ---| |--- Exception Bit Overflow Stored

Symbol



Description

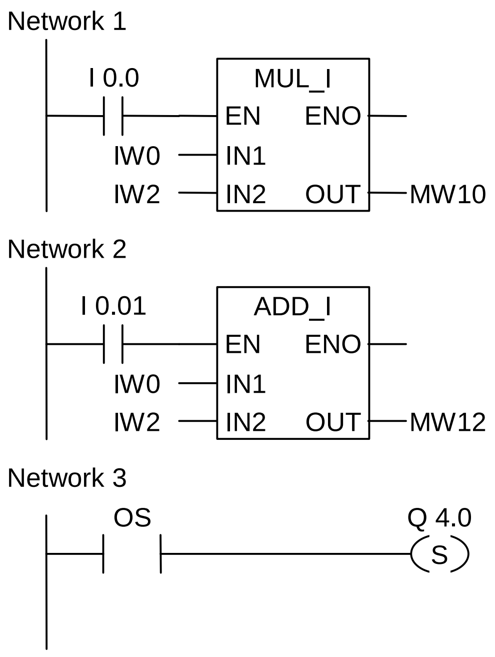
OS ---| |--- (Exception Bit Overflow Stored) or **OS ---| / |---** (Negated Exception Bit Overflow Stored) contact symbols are used to recognize and store a latching overflow in a math function. If the result of the instruction lies outside the permissible negative or positive range, the OS bit in the status word is set. Unlike the OV bit, which is rewritten for subsequent math functions, the OS bit stores an overflow when it occurs. The OS bit remains set until the block is left.

Used in series, the result of the scan is linked to the RLO by AND, used in parallel, it is linked to the RLO by OR.

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	-	-	-	-	-	x	x	x	1

Example



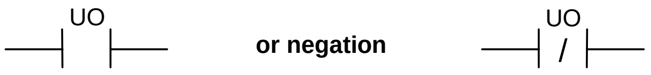
The MUL_I box is activated by signal state "1" at I0.0. The ADD_I box is activated by logic "1" at I0.1. If the result of one of the math functions was outside the permissible range for an integer, the OS bit in the status word is set to "1". Q4.0 is set if the scan of OS is logic "1".

Note

The scan with OS is only necessary because of the two separate networks. Otherwise it is possible to take the ENO output of the first math function and connect it with the EN input of the second (cascade arrangement).

12.4 UO ---| |--- Exception Bit Unordered

Symbol



Description

UO ---| |--- (Exception Bit Unordered) or **UO ---| / |---** (Negated Exception Bit Unordered) contact symbols are used to recognize if the math function with floating-point numbers is unordered (meaning, whether one of the values in the math function is an invalid floating-point number).

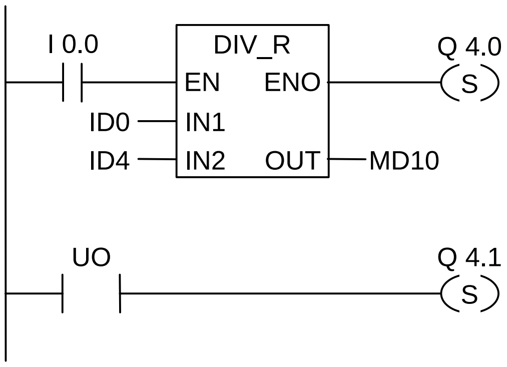
If the result of a math function with floating-point numbers (UO) is invalid, the signal state scan is "1". If the logic operation in CC 1 and CC 0 shows "not invalid", the result of the signal state scan is "0".

Used in series, the result of the scan is linked to the RLO by AND, used in parallel it is linked to the RLO by OR.

Status word

	BR	**CC 1**	**CC 0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**/FC**
writes:	-	-	-	-	-	x	x	x	1

Example

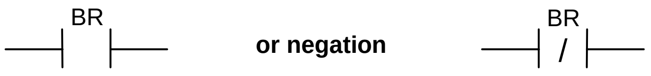


The box is activated by signal state "1" at I0.0. If the value of ID0 or ID4 is an invalid floating-point number, the math function is invalid. If the signal state of EN = 1 (activated) and if an error occurs during the processing of the function DIV_R, the signal state of ENO = 0.

Output Q4.1 is set when the function DIV_R is executed but one of the values is not a valid **floating-point number**.

12.5 BR ---| |--- Exception Bit Binary Result

Symbol



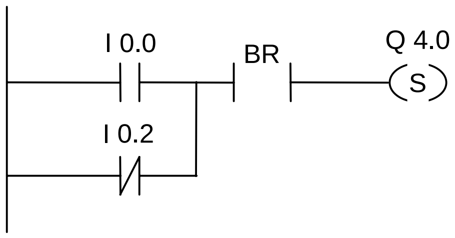
Description

BR ---| |--- (Exception Bit BR Memory) or **BR ---| / |---** (Negated Exception Bit BR Memory) contact symbols are used to test the logic state of the BR bit in the status word. Used in series, the result of the scan is linked to the RLO by AND, used in parallel, it is linked to the RLO by OR. The BR bit is used in the transition from word to bit processing.

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	-	-	-	-	-	x	x	x	1

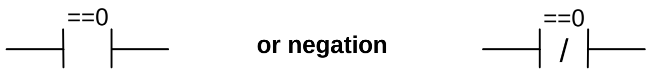
Example



Q4.0 is set if I0.0 is "1" or I0.2 is "0" and in addition to this RLO the logic state of the BR bit is "1".

12.6 ==0 ---| |--- Result Bit Equal 0

Symbol



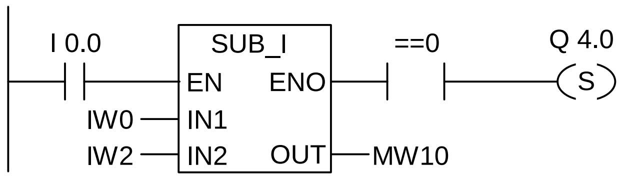
Description

==0 ---| |--- (Result Bit Equal 0) or **==0 ---| / |---** (Negated Result Bit Equal 0) contact symbols are used to recognize if the result of a math function is equal to "0". The instructions scan the condition code bits CC 1 and CC 0 in the status word in order to determine the relation of the result to "0". Used in series, the result of the scan is linked to the RLO by AND, used in parallel, it is linked to the RLO by OR.

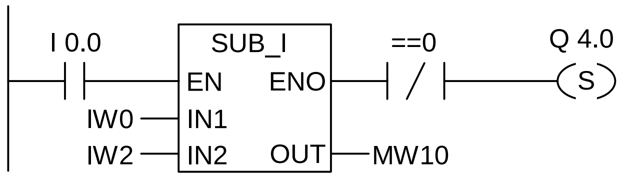
Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	-	-	-	-	-	x	x	x	1

Examples



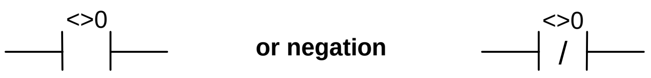
The box is activated by signal state "1" at I0.0. If the value of IW0 is equal to the value of IW2, the result of the math function "IW0 - IW2" is "0". Q4.0 is set if the function is properly executed and the result is **equal** to "0".



Q4.0 is set if the function is properly executed and the result is **not equal** to "0".

12.7 <>0 ---| |--- Result Bit Not Equal 0

Symbol



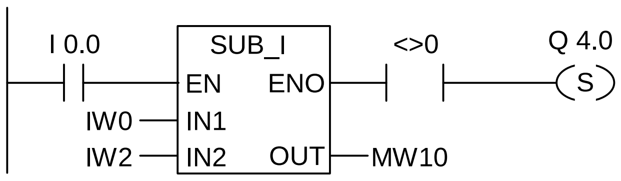
Description

<>0 ---| |--- (Result Bit Not Equal 0) or **<>0 ---| / |---** (Negated Result Bit Not Equal 0) contact symbols are used to recognize if the result of a math function is not equal to "0". The instructions scan the condition code bits CC 1 and CC 0 in the status word in order to determine the relation of the result to "0". Used in series, the result of the scan is linked to the RLO by AND, used in parallel, it is linked to the RLO by OR.

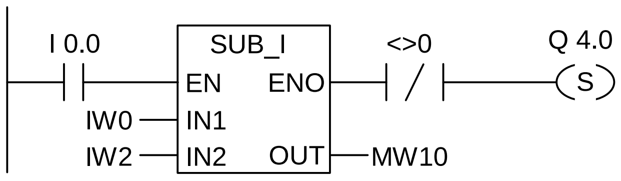
Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	-	-	-	-	-	x	x	x	1

Examples



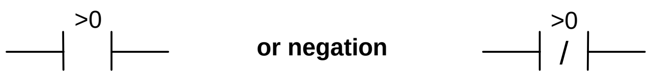
The box is activated by signal state "1" at I0.0. If the value of IW0 is different to the value of IW2, the result of the math function "IW0 - IW2" is not equal to "0". Q4.0 is set if the function is properly executed and the result is **not equal** to "0".



Q4.0 is set if the function is properly executed and the result is **equal** to "0".

12.8 >0 ---| |--- Result Bit Greater Than 0

Symbol



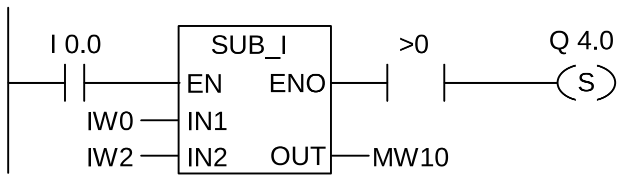
Description

>0 ---| |--- (Result Bit Greater Than 0) or **>0 ---| / |---** (Negated Result Bit Greater Than Zero) contact symbols are used to recognize if the result of a math function is greater than "0". The instructions scan the condition code bits CC 1 and CC 0 in the status word in order to determine the relation to "0". Used in series, the result of the scan is linked to the RLO by AND, used in parallel, it is linked to the RLO by OR.

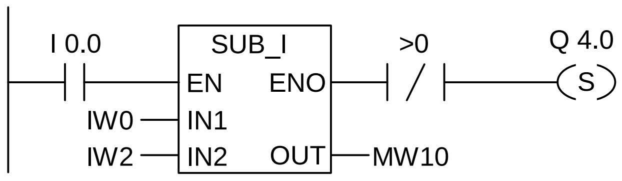
Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	-	-	-	-	-	x	x	x	1

Example



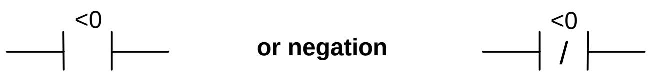
The box is activated by signal state "1" at I0.0. If the value of IW0 is higher than the value of IW2, the result of the math function "IW0 - IW2" is greater than "0". Q4.0 is set if the function is properly executed and the result is **greater** than "0".



Q4.0 is set if the function is properly executed and the result is **not** greater than "0".

12.9 <0 ---| |--- Result Bit Less Than 0

Symbol



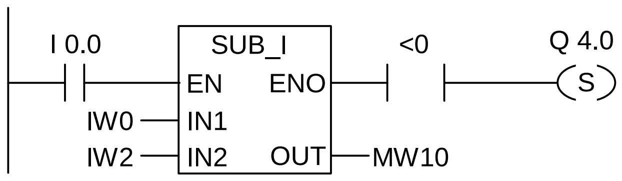
Description

<0 ---| |--- (Result Bit Less Than 0) or **<0 ---| / |---** (Negated Result Bit Less Than 0) contact symbols are used to recognize if the result of a math function is less than "0". The instructions scan the condition code bits CC 1 and CC 0 in the status word in order to determine the relation of the result to "0". Used in series, the result of the scan is linked to the RLO by AND, used in parallel, it is linked to the RLO by OR.

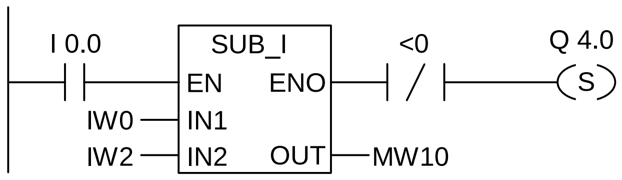
Status word

	BR	**CC 1**	**CC 0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**/FC**
writes:	-	-	-	-	-	x	x	x	1

Example



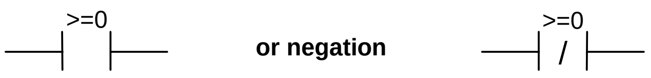
The box is activated by signal state "1" at I0.0. If the value of IW0 is lower than the value of IW2, the result of the math function "IW0 - IW2" is less than "0". Q4.0 is set if the function is properly executed and the result is **less** than "0".



Q4.0 is set if the function is properly executed and the result is **not** less than "0".

12.10 >=0 ---| |--- Result Bit Greater Equal 0

Symbol



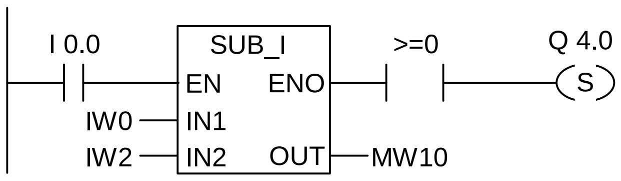
Description

>=0 ---| |--- (Result Bit Greater Equal 0) or **>=0 ---| / |---** (Negated Result Bit Greater Equal 0) contact symbols are used to recognize if the result of a math function is greater than or equal to "0". The instructions scan the condition code bits CC 1 and CC 0 in the status word in order to determine the relation to "0". Used in series, the result of the scan is linked to the RLO by AND, used in parallel, it is linked to the RLO by OR.

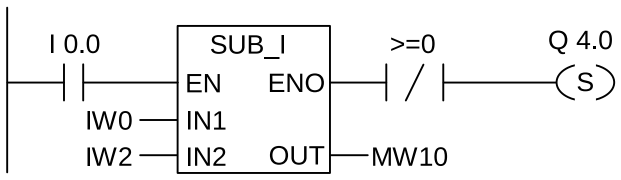
Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	-	-	-	-	-	x	x	x	1

Example



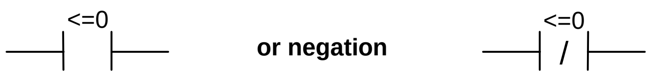
The box is activated by signal state "1" at I0.0. If the value of IW0 is higher or equal to the value of IW2, the result of the math function "IW0 - IW2" is greater than or equal to "0". Q4.0 is set if the function is properly executed and the result is **greater** than or equal to "0".



Q4.0 is set if the function is properly executed and the result is **not** greater than or equal to "0".

12.11 <=0 ---| |--- Result Bit Less Equal 0

Symbol



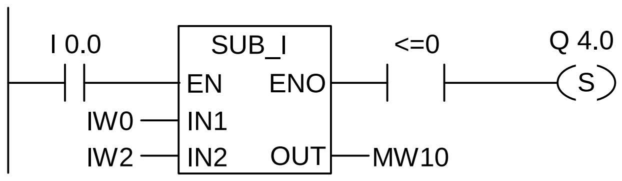
Description

<=0 ---| |--- (Result Bit Less Equal 0) or **<=0 ---| / |---** (Negated Result Bit Less Equal 0) contact symbols are used to recognize if the result of a math function is less than or equal to "0". The instructions scan the condition code bits CC 1 and CC 0 in the status word in order to determine the relation of the result to "0". Used in series, the result of the scan is linked to the RLO by AND, used in parallel, it is linked to the RLO by OR.

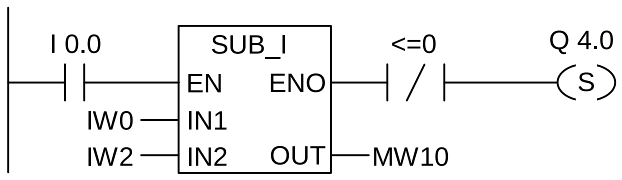
Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	-	-	-	-	-	x	x	x	1

Examples



The box is activated by signal state "1" at I0.0. If the value of IW0 is less than or equal to the value of IW2 the result of the math function "IW0 - IW2" is less than or equal to "0". Q4.0 is set if the function is well properly executed and the result is **less** than or equal to "0".



Q4.0 is set if the function is properly executed and the result is **not less** than or equal to "0".

13 Timer Instructions

13.1 Overview of Timer Instructions

Description

You can find information for setting and selecting the correct time under "Location of a Timer in Memory and Components of a Timer".

The following timer instructions are available:

• S_PULSE	Pulse S5 Timer
• S_PEXT	Extended Pulse S5 Timer
• S_ODT	On-Delay S5 Timer
• S_ODTS	Retentive On-Delay S5 Timer
• S_OFFDT	Off-Delay S5 Timer
• ---(SP)	Pulse Timer Coil
• ---(SE)	Extended Pulse Timer Coil
• ---(SD)	On-Delay Timer Coil
• ---(SS)	Retentive On-Delay Timer Coil
• ---(SA)	Off-Delay Timer Coil

13.2 Location of a Timer in Memory and Components of a Timer

Area in Memory

Timers have an area reserved for them in the memory of your CPU. This memory area reserves one 16-bit word for each timer address. The ladderlogic instruction set supports 256 timers. Please refer to your CPU's technical information to establish the number of timer words available.

The following functions have access to the timer memory area:

- Timer instructions
- Updating of timer words by means of clock timing. This function of your CPU in the RUN mode decrements a given time value by one unit at the interval designated by the time base until the time value is equal to zero.

Time Value

Bits 0 through 9 of the timer word contain the time value in binary code. The time value specifies a number of units. Time updating decrements the time value by one unit at an interval designated by the time base. Decrementing continues until the time value is equal to zero. You can load a time value into the low word of accumulator 1 in binary, hexadecimal, or binary coded decimal (BCD) format.

You can pre-load a time value using either of the following formats:

- W#16#wxyz
  - Where w = the time base (that is, the time interval or resolution)
  - Where xyz = the time value in binary coded decimal format
- S5T#a**H**_b**M**_c**S**_d**MS**
  - Where H = hours, M = minutes, S = seconds, and MS = milliseconds; a, b, c, d are defined by the user.
  - The time base is selected automatically, and the value is rounded to the next lower number with that time base.

The maximum time value that you can enter is 9,990 seconds, or 2H_46M_30S.

S5TIME#4S = 4 seconds
s5t#2h_15m = 2 hours and 15 minutes
S5T#1H_12M_18S = 1 hour, 12 minutes, and 18 seconds

Time Base

Bits 12 and 13 of the timer word contain the time base in binary code. The time base defines the interval at which the time value is decremented by one unit. The smallest time base is 10 ms; the largest is 10 s.

Time Base	Binary Code for the Time Base
10 ms	00
100 ms	01
1 s	10
10 s	11

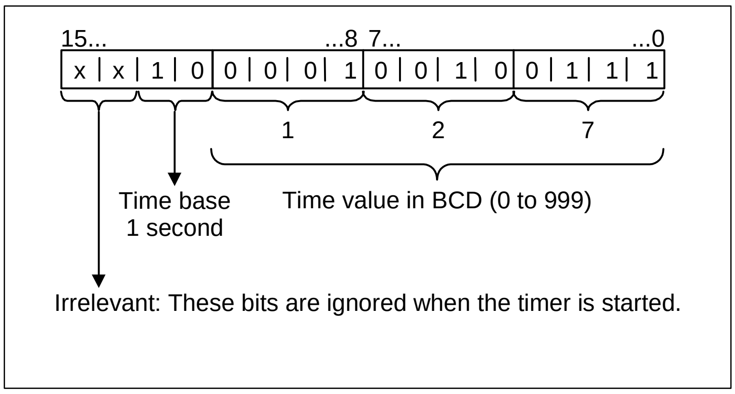
Values that exceed 2h46m30s are not accepted. A value whose resolution is too high for the range limits (for example, 2h10ms) is truncated down to a valid resolution. The general format for S5TIME has limits to range and resolution as shown below:

Resolution	Range
0.01 second	10MS to 9S_990MS
0.1 second	100MS to 1M_39S_900MS
1 second	1S to 16M_39S
10 seconds	10S to 2H_46M_30S

Bit Configuration in the Time Cell

When a timer is started, the contents of the timer cell are used as the time value. Bits 0 through 11 of the timer cell hold the time value in binary coded decimal format (BCD format: each set of four bits contains the binary code for one decimal value). Bits 12 and 13 hold the time base in binary code.

The following figure shows the contents of the timer cell loaded with timer value 127 and a time base of 1 second:

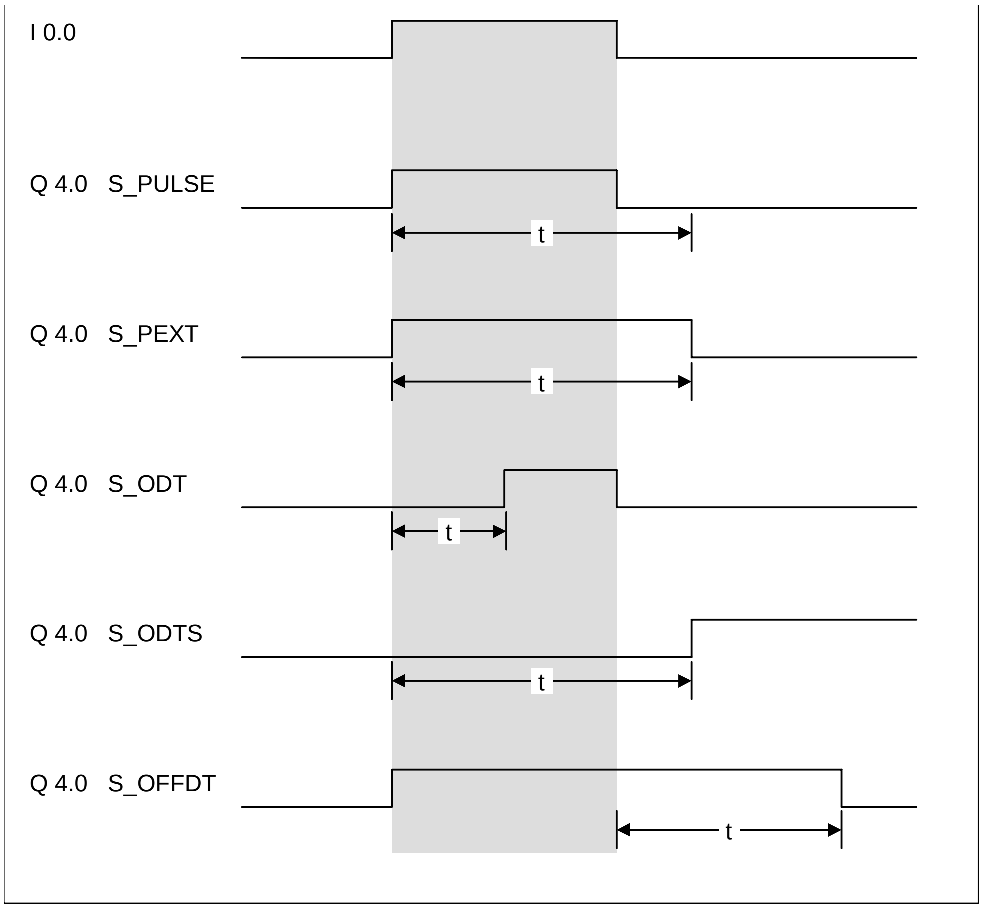

15...
...8
7...
...0
x x 1 0 0 0 0 1 0 0 1 0 0 1 1 1
1
2
7
Time base
1 second
Time value in BCD (0 to 999)
Irrelevant: These bits are ignored when the timer is started.

Reading the Time and the Time Base

Each timer box provides two outputs, BI and BCD, for which you can indicate a word location. The BI output provides the time value in binary format. The BCD output provides the time base and the time value in binary coded decimal (BCD) format.

Choosing the right Timer

This overview is intended to help you choose the right timer for your timing job.

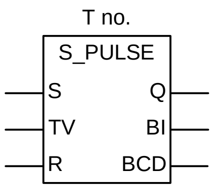


Timer	Description
S_PULSE Pulse timer	The maximum time that the output signal remains at 1 is the same as the programmed time value t. The output signal stays at 1 for a shorter period if the input signal changes to 0.
S_PEXT Extended pulse timer	The output signal remains at 1 for the programmed length of time, regardless of how long the input signal stays at 1.
S_ODT On-delay timer	The output signal changes to 1 only when the programmed time has elapsed and the input signal is still 1.
S_ODTS Retentive on-delay timer	The output signal changes from 0 to 1 only when the programmed time has elapsed, regardless of how long the input signal stays at 1.
S_OFFDT Off-delay timer	The output signal changes to 1 when the input signal changes to 1 or while the timer is running. The time is started when the input signal changes from 1 to 0.

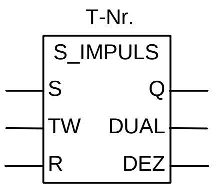
13.3 S_PULSE Pulse S5 Timer

Symbol

English



German



Parameter English	Parameter German	Data Type	Memory Area	Description
T no.	T-Nr.	TIMER	T	Timer identification number; range depends on CPU
S	S	BOOL	I, Q, M, L, D	Start input
TV	TW	S5TIME	I, Q, M, L, D	Preset time value
R	R	BOOL	I, Q, M, L, D	Reset input
BI	DUAL	WORD	I, Q, M, L, D	Remaining time value, integer format
BCD	DEZ	WORD	I, Q, M, L, D	Remaining time value, BCD format
Q	Q	BOOL	I, Q, M, L, D	Status of the timer

Description

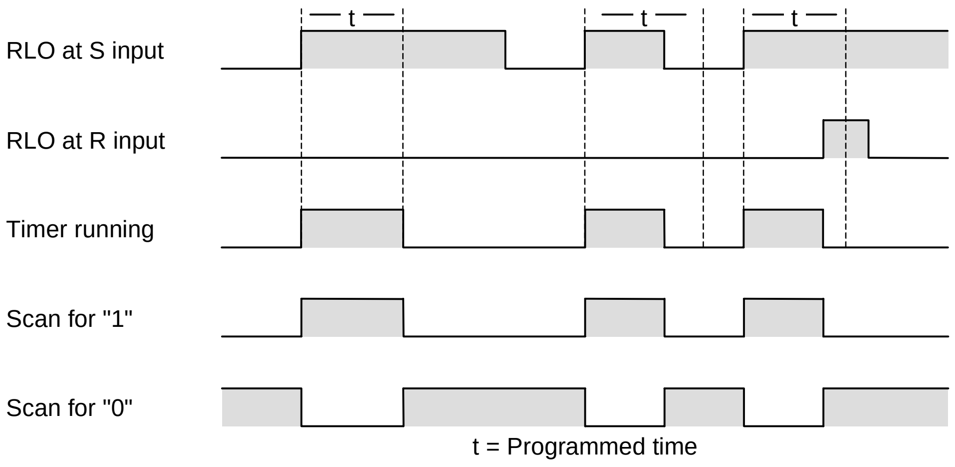
S_PULSE (Pulse S5 Timer) starts the specified timer if there is a positive edge at the start (S) input. A signal change is always necessary in order to enable a timer. The timer runs as long as the signal state at input S is "1", the longest period, however, is the time value specified by input TV. The signal state at output Q is "1" as long as the timer is running. If there is a change from "1" to "0" at the S input before the time interval has elapsed the timer will be stopped. In this case the signal state at output Q is "0".

The timer is reset when the timer reset (R) input changes from "0" to "1" while the timer is running. The current time and the time base are also set to zero. Logic "1" at the timer's R input has no effect if the timer is not running.

The current time value can be scanned at the outputs BI and BCD. The time value at BI is binary coded, at BCD it is BCD coded. The current time value is the initial TV value minus the time elapsed since the timer was started.

Timing Diagram

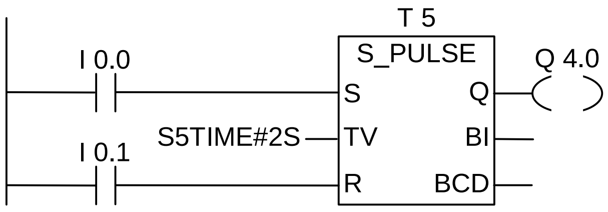
Pulse timer characteristics:



Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	-	-	-	-	-	x	x	x	1

Example

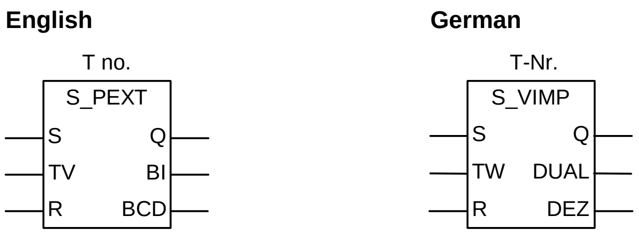


If the signal state of input I0.0 changes from "0" to "1" (positive edge in RLO), the timer T5 will be started. The timer will continue to run for the specified time of two seconds (2 s) as long as I0.0 is "1". If the signal state of I0.0 changes from "1" to "0" before the timer has expired the timer will be stopped. If the signal state of input I0.1 changes from "0" to "1" while the timer is running, the time is reset.

The output Q4.0 is logic "1" as long as the timer is running and "0" if the time has elapsed or was reset.

13.4 S_PEXT Extended Pulse S5 Timer

Symbol



Parameter English	Parameter German	Data Type	Memory Area	Description
T no.	T-Nr.	TIMER	T	Timer identification number; range depends on CPU
S	S	BOOL	I, Q, M, L, D	Start input
TV	TW	S5TIME	I, Q, M, L, D	Preset time value
R	R	BOOL	I, Q, M, L, D	Reset input
BI	DUAL	WORD	I, Q, M, L, D	Remaining time value, integer format
BCD	DEZ	WORD	I, Q, M, L, D	Remaining time value, BCD format
Q	Q	BOOL	I, Q, M, L, D	Status of the timer

Description

S_PEXT (Extended Pulse S5 Timer) starts the specified timer if there is a positive edge at the start (S) input. A signal change is always necessary in order to enable a timer. The timer runs for the preset time interval specified at input TV even if the signal state at the S input changes to "0" before the time interval has elapsed. The signal state at output Q is "1" as long as the timer is running. The timer will be restarted ("re-triggered") with the preset time value if the signal state at input S changes from "0" to "1" while the timer is running.

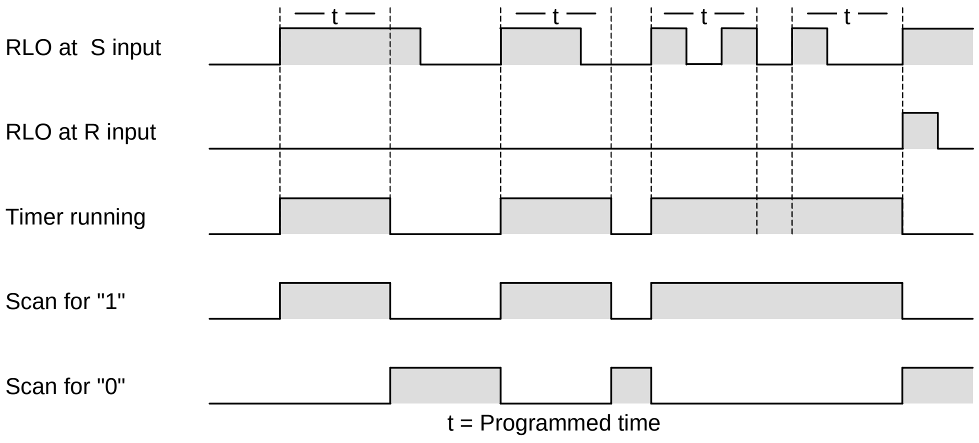
The timer is reset if the reset (R) input changes from "0" to "1" while the timer is running. The current time and the time base are set to zero.

The current time value can be scanned at the outputs BI and BCD. The time value at BI is binary coded, at BCD is BCD coded. The current time value is the initial TV value minus the time elapsed since the timer was started.

See also "Location of a Timer in Memory and Components of a Timer".

Timing Diagram

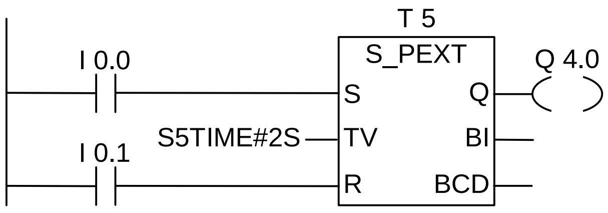
Extended pulse timer characteristics:



Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	-	-	-	-	-	x	x	x	1

Example

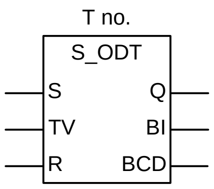


If the signal state of input I0.0 changes from "0" to "1" (positive edge in RLO), the timer T5 will be started. The timer will continue to run for the specified time of two seconds (2 s) without being affected by a negative edge at input S. If the signal state of I0.0 changes from "0" to "1" before the timer has expired the timer will be re-triggered. The output Q4.0 is logic "1" as long as the timer is running.

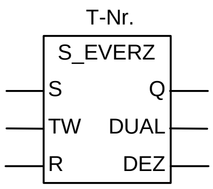
13.5 S_ODT On-Delay S5 Timer

Symbol

English



German



Parameter English	Parameter German	Data Type	Memory Area	Description
T no.	T-Nr.	TIMER	T	Timer identification number; range depends on CPU
S	S	BOOL	I, Q, M, L, D	Start input
TV	TW	S5TIME	I, Q, M, L, D	Preset time value
R	R	BOOL	I, Q, M, L, D	Reset input
BI	DUAL	WORD	I, Q, M, L, D	Remaining time value, integer format
BCD	DEZ	WORD	I, Q, M, L, D	Remaining time value, BCD format
Q	Q	BOOL	I, Q, M, L, D	Status of the timer

Description

S_ODT (On-Delay S5 Timer) starts the specified timer if there is a positive edge at the start (S) input. A signal change is always necessary in order to enable a timer. The timer runs for the time interval specified at input TV as long as the signal state at input S is positive. The signal state at output Q is "1" when the timer has elapsed without error and the signal state at the S input is still "1". When the signal state at input S changes from "1" to "0" while the timer is running, the timer is stopped. In this case the signal state of output Q is "0".

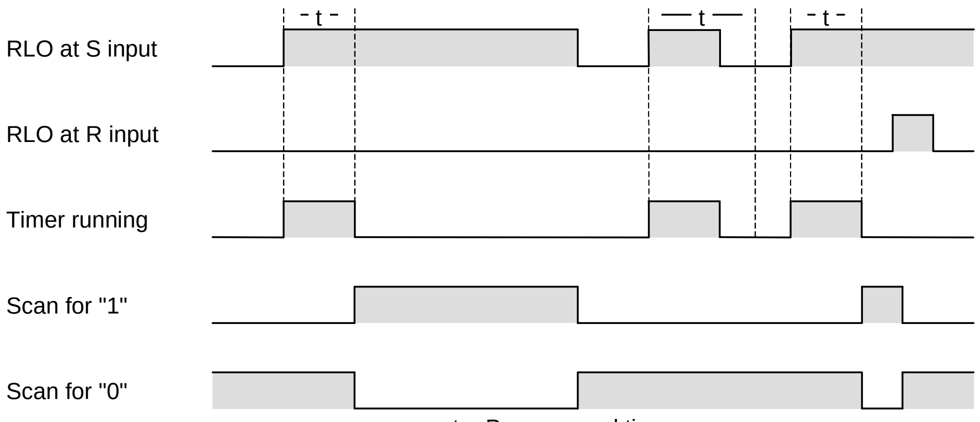
The timer is reset if the reset (R) input changes from "0" to "1" while the timer is running. The current time and the time base are set to zero. The signal state at output Q is then "0". The timer is also reset if there is a logic "1" at the R input while the timer is not running and the RLO at input S is "1".

The current time value can be scanned at the outputs BI and BCD. The time value at BI is binary coded, at BCD is BCD coded. The current time value is the initial TV value minus the time elapsed since the timer was started.

See also "Location of a Timer in Memory and Components of a Timer".

Timing Diagram

On-Delay timer characteristics:

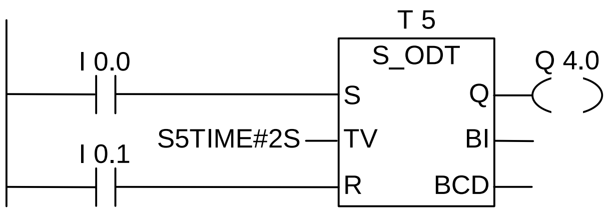


t = Programmed time

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	-	-	-	-	-	x	x	x	1

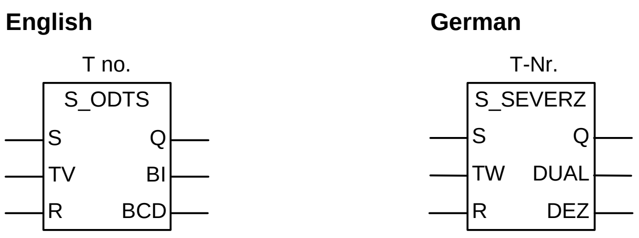
Example



If the signal state of I0.0 changes from "0" to "1" (positive edge in RLO), the timer T5 will be started. If the time of two seconds elapses and the signal state at input I0.0 is still "1", the output Q4.0 will be "1". If the signal state of I0.0 changes from "1" to "0", the timer is stopped and Q4.0 will be "0" (if the signal state of I0.1 changes from "0" to "1", the time is reset regardless of whether the timer is running or not).

13.6 S_ODTS Retentive On-Delay S5 Timer

Symbol



Parameter English	Parameter German	Data Type	Memory Area	Description
T no.	T-Nr.	TIMER	T	Timer identification number; range depends on CPU
S	S	BOOL	I, Q, M, L, D	Start input
TV	TW	S5TIME	I, Q, M, L, D	Preset time value
R	R	BOOL	I, Q, M, L, D	Reset input
BI	DUAL	WORD	I, Q, M, L, D	Remaining time value, integer format
BCD	DEZ	WORD	I, Q, M, L, D	Remaining time value, BCD format
Q	Q	BOOL	I, Q, M, L, D	Status of the timer

Description

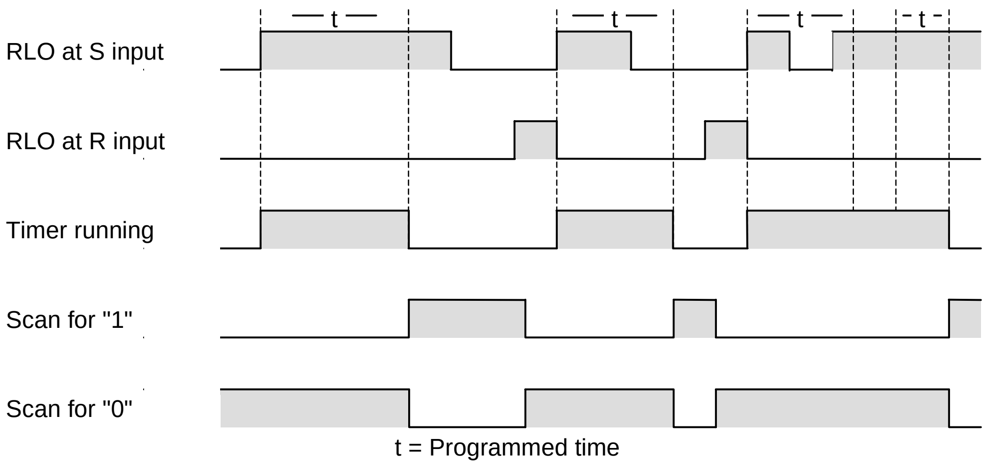
S_ODTS (Retentive On-Delay S5 Timer) starts the specified timer if there is a positive edge at the start (S) input. A signal change is always necessary in order to enable a timer. The timer runs for the time interval specified at input TV even if the signal state at input S changes to "0" before the time interval has elapsed. The signal state at output Q is "1" when the timer has elapsed without regard to the signal state at input S. The timer will be restarted (re-triggered) with the specified time if the signal state at input S changes from "0" to "1" while the timer is running.

The timer is reset if the reset (R) input changes from "0" to "1" without regard to the RLO at the S input. The signal state at output Q is then "0".

The current time value can be scanned at the outputs BI and BCD. The time value at BI is binary coded, at BCD it is BCD coded. The current time value is the initial TV value minus the time elapsed since the timer was started.

Timing Diagram

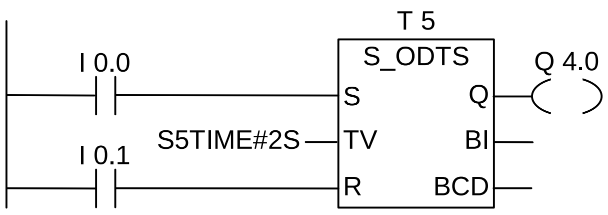
Retentive On-Delay timer characteristics:



Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	-	-	-	-	-	x	x	x	1

Example

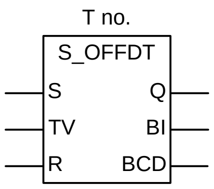


If the signal state of I0.0 changes from "0" to "1" (positive edge in RLO), the timer T5 will be started. The timer runs without regard to a signal change at I0.0 from "1" to "0". If the signal state at I0.0 changes from "0" to "1" before the timer has expired, the timer will be re-triggered. The output Q4.0 will be "1" if the timer elapsed. (If the signal state of input I0.1 changes from "0" to "1", the time will be reset irrespective of the RLO at S.)

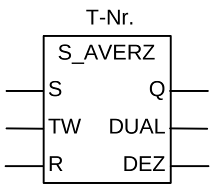
13.7 S_OFFDT Off-Delay S5 Timer

Symbol

English



German



Parameter English	Parameter German	Data Type	Memory Area	Description
T no.	T-Nr.	TIMER	T	Timer identification number; range depends on CPU
S	S	BOOL	I, Q, M, L, D	Start input
TV	TW	S5TIME	I, Q, M, L, D	Preset time value
R	R	BOOL	I, Q, M, L, D	Reset input
BI	DUAL	WORD	I, Q, M, L, D	Remaining time value, integer format
BCD	DEZ	WORD	I, Q, M, L, D	Remaining time value, BCD format
Q	Q	BOOL	I, Q, M, L, D	Status of the timer

Description

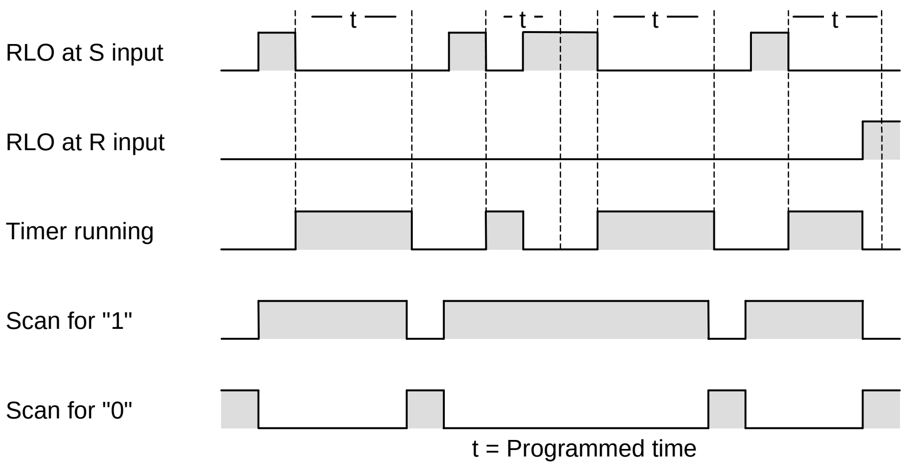
S_OFFDT (Off-Delay S5 Timer) starts the specified timer if there is a negative edge at the start (S) input. A signal change is always necessary in order to enable a timer. The signal state at output Q is "1" if the signal state at the S input is "1" or while the timer is running. The timer is reset when the signal state at input S goes from "0" to "1" while the timer is running. The timer is not restarted until the signal state at input S changes again from "1" to "0".

The timer is reset when the reset (R) input changes from "0" to "1" while the timer is running.

The current time value can be scanned at the outputs BI and BCD. The time value at BI is binary coded, at BCD it is BCD coded. The current time value is the initial TV value minus the time elapsed since the timer was started.

Timing Diagram

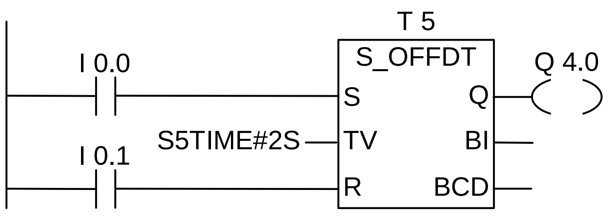
Off-Delay timer characteristics:



Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	-	-	-	-	-	x	x	x	1

Example



If the signal state of I0.0 changes from "1" to "0", the timer is started.

Q4.0 is "1" when I0.0 is "1" or the timer is running. (if the signal state at I0.1 changes from "0" to "1" while the time is running, the timer is reset).

13.8 ---(SP) Pulse Timer Coil

Symbol

English	**German**
<T no..>	<T no.>
---(SP)	**---(SI)**
<time value>	<time value>

Parameter	Data Type	Memory Area	Description
<T no.>	TIMER	T	Timer identification number; range depends on CPU
<time value>	S5TIME	I, Q, M, L, D	Preset time value

Description

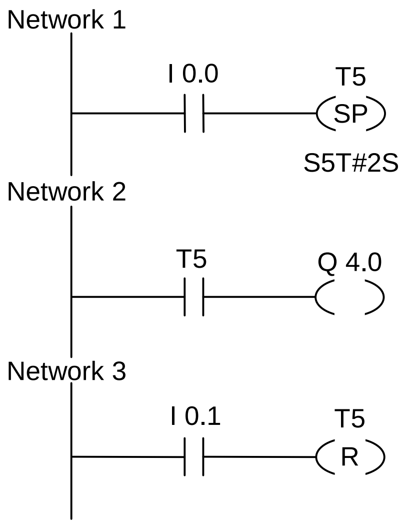
---(SP) (Pulse Timer Coil) starts the specified timer with the **<time value>** when there is a positive edge on the RLO state. The timer continues to run for the specified time interval as long as the RLO remains positive ("1"). The signal state of the counter is "1" as long as the timer is running. If there is a change from "1" to "0" in the RLO before the time value has elapsed, the timer will stop. In this case, a scan for "1" always produces the result "0".

See also "Location of a Timer in Memory and Components of a Timer" and S_PULSE (Pulse S5 Timer).

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	-	-	-	-	-	0	-	-	0

Example



If the signal state of input I0.0 changes from "0" to "1" (positive edge in RLO), the timer T5 is started. The timer continues to run with the specified time of two seconds as long as the signal state of input I0.0 is "1". If the signal state of input I0.0 changes from "1" to "0" before the specified time has elapsed, the timer stops.

The signal state of output Q4.0 is "1" as long as the timer is running. A signal state change from "0" to "1" at input I0.1 will reset timer T5 which stops the timer and clears the remaining portion of the time value to "0".

13.9 ---(SE) Extended Pulse Timer Coil

Symbol

English	**German**
<T no.>	<T no>
---(SE)	**---(SV)**
<time value>	<time value>

Parameter	Data Type	Memory Area	Description
<T no.>	TIMER	T	Timer identification number; range depends on CPU
<time value>	S5TIME	I, Q, M, L, D	Preset time value

Description

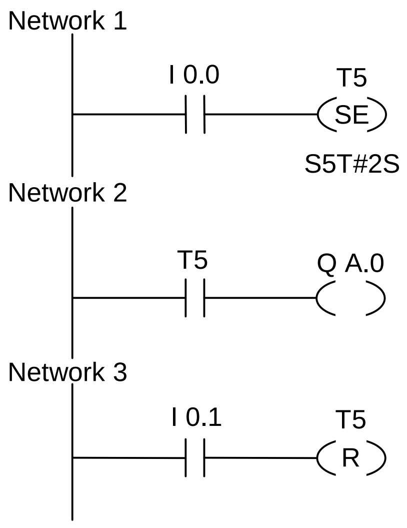
---(SE) (Extended Pulse Timer Coil) starts the specified timer with the specified **<time value>** when there is a positive edge on the RLO state. The timer continues to run for the specified time interval even if the RLO changes to "0" before the timer has expired. The signal state of the counter is ”1” as long as the timer is running. The timer will be restarted (re-triggered) with the specified time value if the RLO changes from "0" to "1" while the timer is running.

See also "Location of a Timer in Memory and Components of a Timer" and S_PEXT (Extended Pulse S5 Timer).

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	-	-	-	-	-	0	-	-	0

Example



If the signal state of input I0.0 changes from "0" to "1" (positive edge in RLO), the timer T5 is started. The timer continues to run without regard to a negative edge of the RLO. If the signal state of I0.0 changes from "0" to "1" before the timer has expired, the timer is re-triggered.

The signal state of output Q4.0 is "1" as long as the timer is running. A signal state change from "0" to "1" at input I0.1 will reset timer T5 which stops the timer and clears the remaining portion of the time value to "0".

13.10 ---(SD) On-Delay Timer Coil

Symbol

English	**German**
<T no.>	<T no.>
---(SD)	**---(SE)**
<time value>	<time value>

Parameter	Data Type	Memory Area	Description
<T no.>	TIMER	T	Timer identification number; range depends on CPU
<time value>	S5TIME	I, Q, M, L, D	Preset time value

Description

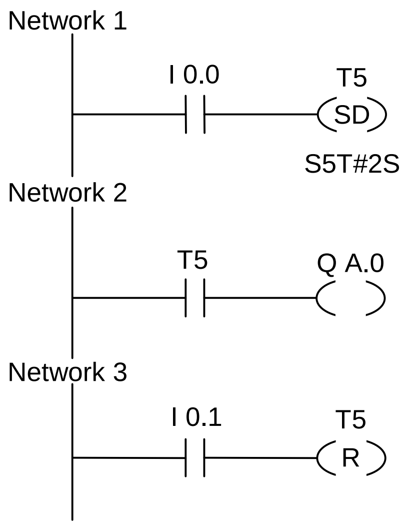
---(SD) (On Delay Timer Coil) starts the specified timer with the **<time value>** if there is a positive edge on the RLO state. The signal state of the timer is "1" when the **<time value>** has elapsed without error and the RLO is still "1". When the RLO changes from "1" to "0" while the timer is running, the timer is reset. In this case, a scan for "1" always produces the result "0".

See also "Location of a Timer in Memory and Components of a Timer" and S_ODT (On-Delay S5 Timer).

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	-	-	-	-	-	0	-	-	0

Example



If the signal state of input I0.0 changes from "0" to "1" (positive edge in RLO), the timer T5 is started. If the time elapses and the signal state of input I0.0 is still "1", the signal state of output Q4.0 will be "1".

If the signal state of input I0.0 changes from "1" to "0", the timer remains idle and the signal state of output Q4.0 will be "0". A signal state change from "0" to "1" at input I0.1 will reset timer T5 which stops the timer and clears the remaining portion of the time value to "0".

13.11 ---(SS) Retentive On-Delay Timer Coil

Symbol

English	**German**
<T no.>	<T no.>
---(SS)	**---(SS)**
<time value>	<time value>

Parameter	Data Type	Memory Area	Description
<T no.>	TIMER	T	Timer identification number; range depends on CPU
<time value>	S5TIME	I, Q, M, L, D	Preset time value

Description

---(SS) (Retentive On-Delay Timer Coil) starts the specified timer if there is a positive edge on the RLO state. The signal state of the timer is "1" if the time value has elapsed. A restart of the timer is only possible if it is reset explicitly. Only a reset causes the signal state of the timer to be set to "0".

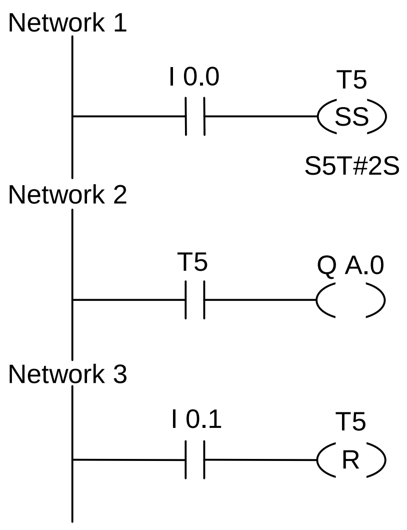
The timer restarts with the specified time value if the RLO changes from "0" to "1" while the timer is running.

See also "Location of a Timer in Memory and Components of a Timer" and S_ODTS (Retentive On-Delay S5 Timer).

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	-	-	-	-	-	0	-	-	0

Example



If the signal state of input I0.0 changes from "0" to "1" (positive edge in RLO), the timer T5 is started. If the signal state of input I0.0 changes from "0" to "1" before the timer has expired, the timer is re-triggered. The output Q4.0 will be "1" if the timer elapsed. A signal state "1" at input I0.1 will reset timer T5, which stops the timer and clears the remaining portion of the time value to "0".

13.12 ---(SF) Off-Delay Timer Coil

Symbol

English	**German**
<T no.>	<T no.>
---(SF)	**---(SA)**
<time value>	<time value>

Parameter	Data Type	Memory Area	Description
<T no.>	TIMER	T	Timer identification number; range depends on CPU
<time value>	S5TIME	I, Q, M, L, D	Preset time value

Description

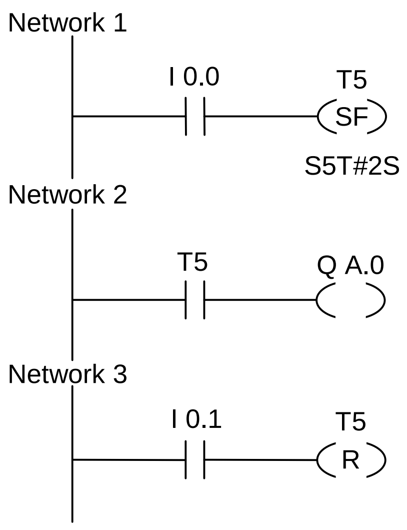
---(SF) (Off-Delay Timer Coil) starts the specified timer if there is a negative edge on the RLO state. The timer is "1" when the RLO is "1" or as long as the timer is running during the **<time value>** interval. The timer is reset when the RLO goes from "0" to "1" while the timer is running. The timer is always restarted when the RLO changes from "1" to "0".

See also "Location of a Timer in Memory and Components of a Timer" and S_OFFDT (Off-Delay S5 Timer).

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	-	-	-	-	-	0	-	-	0

Example



If the signal state of input I0.0 changes from "1" to "0" the timer is started.

The signal state of output Q4.0 is "1" when input I0.0 is "1" or the timer is running. A signal state change from "0" to "1" at input I0.1 will reset timer T5 which stops the timer and clears the remaining portion of the time value to "0".

14 Word Logic Instructions

14.1 Overview of Word logic instructions

Description

Word logic instructions compare pairs of words (16 bits) and double words (32 bits) bit by bit, according to Boolean logic.

If the result at output OUT does not equal 0, bit CC 1 of the status word is set to "1".

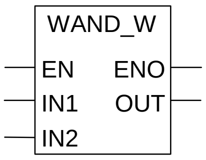
If the result at output OUT does equal 0, bit CC 1 of the status word is set to "0".

The following word logic instructions are available:

• WAND_W	(Word) AND Word
• WOR_W	(Word) OR Word
• WXOR_W	(Word) Exclusive OR Word
• WAND_DW	(Word) AND Double Word
• WOR_DW	(Word) OR Double Word
• WXOR_DW	(Word) Exclusive OR Double Word

14.2 WAND_W (Word) AND Word

Symbol



Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, L, D	Enable input
ENO	BOOL	I, Q, M, L, D	Enable output
IN1	WORD	I, Q, M, L, D	First value for logic operation
IN2	WORD	I, Q, M, L, D	Second value for logic operation
OUT	WORD	I, Q, M, L, D	Result word of logic operation

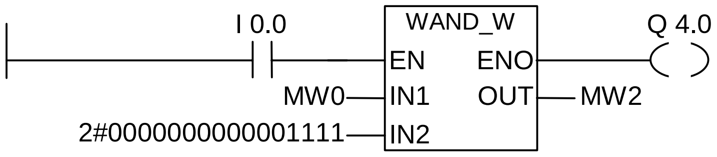
Description

WAND_W (AND Words) is activated by signal state "1" at the enable (EN) input and ANDs the two word values present at IN1 and IN2 bit by bit. The values are interpreted as pure bit patterns. The result can be scanned at the output OUT. ENO has the same logic state as EN.

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	1	x	0	0	-	x	1	1	1

Example



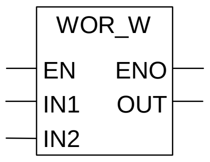
The instruction is executed if I0.0 is "1". Only bits 0 to 3 of MW0 are relevant, the rest of MW0 is masked by the IN2 word bit pattern:

MW0	=	01010101 01010101
IN2	=	00000000 00001111
MW0 AND IN2 = MW2	=	00000000 00000101

Q4.0 is "1" if the instruction is executed.

14.3 WOR_W (Word) OR Word

Symbol



Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, L, D	Enable input
ENO	BOOL	I, Q, M, L, D	Enable output
IN1	WORD	I, Q, M, L, D	First value for logic operation
IN2	WORD	I, Q, M, L, D	Second value for logic operation
OUT	WORD	I, Q, M, L, D	Result word of logic operation

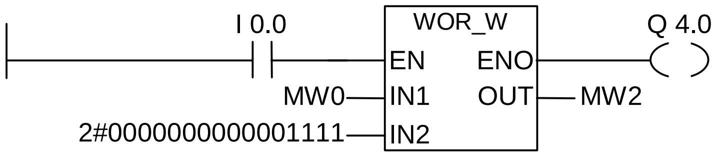
Description

WOR_W (OR Words) is activated by signal state "1" at the enable (EN) input and ORs the two word values present at IN1 and IN2 bit by bit. The values are interpreted as pure bit patterns. The result can be scanned at the output OUT. ENO has the same logic state as EN.

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	1	x	0	0	-	x	1	1	1

Example



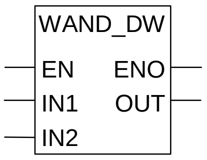
The instruction is executed if I0.0 is "1". Bits 0 to 3 are set to "1", all other MW0 bits are not changed.

MW0	=	01010101 01010101
IN2	=	00000000 00001111
MW0 OR IN2=MW2	=	01010101 01011111

Q4.0 is "1" if the instruction is executed.

14.4 WAND_DW (Word) AND Double Word

Symbol



Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, L, D	Enable input
ENO	BOOL	I, Q, M, L, D	Enable output
IN1	DWORD	I, Q, M, L, D	First value for logic operation
IN2	DWORD	I, Q, M, L, D	Second value for logic operation
OUT	DWORD	I, Q, M, L, D	Result double word of logic operation

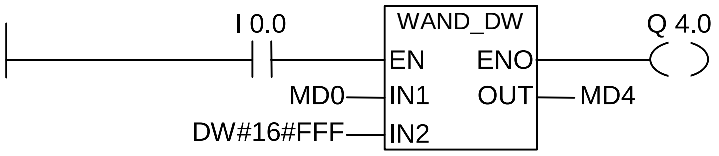
Description

WAND_DW (AND Double Words) is activated by signal state "1" at the enable (EN) input and ANDs the two word values present at IN1 and IN2 bit by bit. The values are interpreted as pure bit patterns. The result can be scanned at the output OUT. ENO has the same logic state as EN.

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	1	x	0	0	-	x	1	1	1

Example



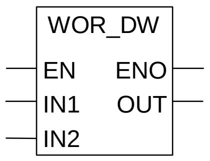
The instruction is executed if I0.0 is "1". Only bits 0 and 11 of MD0 are relevant, the rest of MD0 is masked by the IN2 bit pattern:

MD0	=	01010101 01010101 01010101 01010101
IN2	=	00000000 00000000 00001111 11111111
MD0 AND IN2 = MD4	=	00000000 00000000 00000101 01010101

Q4.0 is "1" if the instruction is executed.

14.5 WOR_DW (Word) OR Double Word

Symbol



Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, L, D	Enable input
ENO	BOOL	I, Q, M, L, D	Enable output
IN1	DWORD	I, Q, M, L, D	First value for logic operation
IN2	DWORD	I, Q, M, L, D	Second value for logic operation
OUT	DWORD	I, Q, M, L, D	Result double word of logic operation

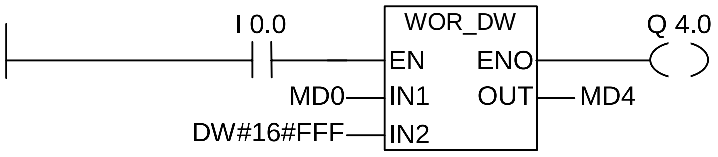
Description

WOR_DW (OR Double Words) is activated by signal state "1" at the enable (EN) input and ORs the two word values present at IN1 and IN2 bit by bit. The values are interpreted as pure bit patterns. The result can be scanned at the output OUT. ENO has the same logic state as EN.

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	1	x	0	0	-	x	1	1	1

Example



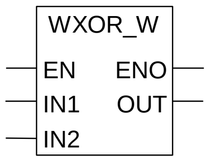
The instruction is executed if I0.0 is "1". Bits 0 to 11 are set to "1", the remaining MD0 bits are not changed:

MD0	=	01010101 01010101 01010101 01010101
IN2	=	00000000 00000000 00001111 11111111
MD0 OR IN2 = MD4	=	01010101 01010101 01011111 11111111

Q4.0 is "1" if the instruction is executed.

14.6 WXOR_W (Word) Exclusive OR Word

Symbol



Parameter	**Data Type**	**Memory Area**	**Description**
EN	BOOL	I, Q, M, L, D	Enable input
ENO	BOOL	I, Q, M, L, D	Enable output
IN1	WORD	I, Q, M, L, D	First value for logic operation
IN2	WORD	I, Q, M, L, D	Second value for logic operation
OUT	WORD	I, Q, M, L, D	Result word of logic operation

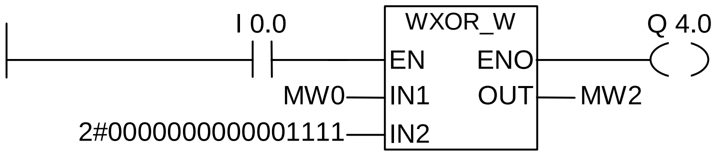
Description

WXOR_W (Exclusive OR Word) is activated by signal state "1" at the enable (EN) input and XORs the two word values present at IN1 and IN2 bit by bit. The values are interpreted as pure bit patterns. The result can be scanned at the output OUT. ENO has the same logic state as EN.

Status word

	BR	**CC 1**	**CC 0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**/FC**
writes:	1	x	0	0	-	x	1	1	1

Example



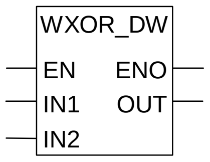
The instruction is executed if I0.0 is "1":

MW0	=	01010101 01010101
IN2	=	00000000 00001111
MW0 XOR IN2 = MW2	=	01010101 01011010

Q4.0 is "1" if the instruction is executed.

14.7 WXOR_DW (Word) Exclusive OR Double Word

Symbol



Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, L, D	Enable input
ENO	BOOL	I, Q, M, L, D	Enable output
IN1	DWORD	I, Q, M, L, D	First value for logic operation
IN2	DWORD	I, Q, M, L, D	Second value for logic operation
OUT	DWORD	I, Q, M, L, D	Result double word of logic operation

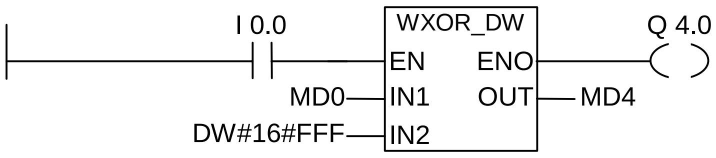
Description

WXOR_DW (Exclusive OR Double Word) is activated by signal state "1" at the enable (EN) input and XORs the two word values present at IN1 and IN2 bit by bit. The values are interpreted as pure bit patterns. The result can be scanned at the output OUT. ENO has the same logic state as EN.

Status word

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
writes:	1	x	0	0	-	x	1	1	1

Example



The instruction is executed if I0.0 is "1":

MD0	=	01010101 01010101 01010101 01010101
IN2	=	00000000 00000000 00001111 11111111
MW2 = MD0 XOR IN2	=	01010101 01010101 01011010 10101010

Q4.0 is "1" if the instruction is executed.

A Overview of All LAD Instructions

A.1 LAD Instructions Sorted According to English Mnemonics (International)

English Mnemonics	German Mnemonics	Program Elements Catalog	Description
---| |---	---| |---	Bit logic Instruction	Normally Open Contact (Address)
---|/|---	---|/|---	Bit logic Instruction	Normally Closed Contact (Address)
---( )	---( )	Bit logic Instruction	Output Coil
---(#)---	---(#)---	Bit logic Instruction	Midline Output
==0 ---| |---	==0 ---| |---	Status bits	Result Bit Equal 0
>0 ---| |---	>0 ---| |---	Status bits	Result Bit Greater Than 0
>=0 ---| |---	>=0 ---| |---	Status bits	Result Bit Greater Equal 0
<=0 ---| |---	<=0 ---| |---	Status bits	Result Bit Less Equal 0
<0 ---| |---	<0 ---| |---	Status bits	Result Bit Less Than 0
<>0 ---| |---	<>0 ---| |---	Status bits	Result Bit Not Equal 0
ABS	ABS	Floating point Instruction	Establish the Absolute Value of a Floating-Point Number
ACOS	ACOS	Floating point Instruction	Establish the Arc Cosine Value
ADD_DI	ADD_DI	Integer Math Instruction	Add Double Integer
ADD_I	ADD_I	Integer Math Instruction	Add Integer
ADD_R	ADD_R	Floating point Instruction	Add Real
ASIN	ASIN	Floating point Instruction	Establish the Arc Sine Value
ATAN	ATAN	Floating point Instruction	Establish the Arc Tangent Value
BCD_DI	BCD_DI	Convert	BCD to Double Integer
BCD_I	BCD_I	Convert	BCD to Integer
BR ---| |---	BIE ---| |---	Status bits	Exception Bit Binary Result
----(CALL)	----(CALL)	Program control	Call FC SFC from Coil (without Parameters)
CALL_FB	CALL_FB	Program control	Call FB from Box
CALL_FC	CALL_FC	Program control	Call FC from Box
CALL_SFB	CALL_SFB	Program control	Call System FB from Box
CALL_SFC	CALL_SFC	Program control	Call System FC from Box
----(CD)	----(ZR)	Counters	Down Counter Coil
CEIL	CEIL	Convert	Ceiling
CMP >=D	CMP >=D	Compare	Compare Double Integer (==, <>, >, <, >=, <=)
CMP >=I	CMP >=I	Compare	Compare Integer (==, <>, >, <, >=, <=)
CMP >=R	CMP >=R	Compare	Compare Real (==, <>, >, <, >=, <=)
COS	COS	Floating point Instruction	Establish the Cosine Value

English Mnemonics	German Mnemonics	Program Elements Catalog	Description
----(CU)	---(ZV)	Counters	Up Counter Coil
DI_BCD	DI_BCD	Convert	Double Integer to BCD
DI_R	DI_R	Convert	Double Integer to Floating-Point
DIV_DI	DIV_DI	Integer Math Instruction	Divide Double Integer
DIV_I	DIV_I	Integer Math Instruction	Divide Integer
DIV_R	DIV_R	Floating point Instruction	Divide Real
EXP	EXP	Floating point Instruction	Establish the Exponential Value
FLOOR	FLOOR	Convert	Floor
I_BCD	I_BCD	Convert	Integer to BCD
I_DI	I_DI	Convert	Integer to Double Integer
INV_I	INV_I	Convert	Ones Complement Integer
INV_DI	INV_DI	Convert	Ones Complement Double Integer
---(JMP)	---(JMP)	Jumps	Unconditional Jump
---(JMP)	---(JMP)	Jumps	Conditional Jump
---(JMPN)	---(JMPN)	Jumps	Jump-If-Not
LABEL	LABEL	Jumps	Label
LN	LN	Floating point Instruction	Establish the Natural Logarithm
---(MCR>)	---(MCR>)	Program control	Master Control Relay Off
---(MCR<)	---(MCR<)	Program control	Master Control Relay On
---(MCRA)	---(MCRA)	Program control	Master Control Relay Activate
---(MCRD)	---(MCRD)	Program control	Master Control Relay Deactivate
MOD_DI	MOD_DI	Integer Math Instruction	Return Fraction Double Integer
MOVE	MOVE	Move	Assign a Value
MUL_DI	MUL_DI	Integer Math Instruction	Multiply Double Integer
MUL_I	MUL_I	Integer Math Instruction	Multiply Integer
MUL_R	MUL_R	Floating point Instruction	Multiply Real
---(N)---	---(N)---	Bit logic Instruction	Negative RLO Edge Detection
NEG	NEG	Bit logic Instruction	Address Negative Edge Detection
NEG_DI	NEG_DI	Convert	Twos Complement Double Integer
NEG_I	NEG_I	Convert	Twos Complement Integer
NEG_R	NEG_R	Convert	Negate Floating-Point Number
---| NOT |---	---| NOT |---	Bit logic Instruction	Invert Power Flow
---(OPN)	---(OPN)	DB call	Open Data Block: DB or DI
OS ---| |---	OS ---| |---	Status bits	Exception Bit Overflow Stored
OV ---| |---	OV ---| |---	Status bits	Exception Bit Overflow
---(P)---	---(P)---	Bit logic Instruction	Positive RLO Edge Detection
POS	POS	Bit logic Instruction	Address Positive Edge Detection
---(R)	---(R)	Bit logic Instruction	Reset Coil
---(RET)	---(RET)	Program control	Return
ROL_DW	ROL_DW	Shift/Rotate	Rotate Left Double Word
ROR_DW	ROR_DW	Shift/Rotate	Rotate Right Double Word
ROUND	ROUND	Convert	Round to Double Integer
RS	RS	Bit logic Instruction	Reset-Set Flip Flop
---(S)	---(S)	Bit logic Instruction	Set Coil
---(SAVE)	---(SAVE)	Bit logic Instruction	Save RLO into BR Memory
---(SC)	---(SZ)	Counters	Set Counter Value

English Mnemonics	German Mnemonics	Program Elements Catalog	Description
S_CD	Z_RUECK	Counters	Down Counter
S_CU	Z_VORW	Counters	Up Counter
S_CUD	ZAEHLER	Counters	Up-Down Counter
---(SD)	---(SE)	Timers	On-Delay Timer Coil
---(SE)	---(SV)	Timers	Extended Pulse Timer Coil
---(SF)	---(SA)	Timers	Off-Delay Timer Coil
SHL_DW	SHL_DW	Shift/Rotate	Shift Left Double Word
SHL_W	SHL_W	Shift/Rotate	Shift Left Word
SHR_DI	SHR_DI	Shift/Rotate	Shift Right Double Integer
SHR_DW	SHR_DW	Shift/Rotate	Shift Right Double Word
SHR_I	SHR_I	Shift/Rotate	Shift Right Integer
SHR_W	SHR_W	Shift/Rotate	Shift Right Word
SIN	SIN	Floating point Instruction	Establish the Sine Value
S_ODT	S_EVERZ	Timers	On-Delay S5 Timer
S_ODTS	S_SEVERZ	Timers	Retentive On-Delay S5 Timer
S_OFFDT	S_AVERZ	Timers	Off-Delay S5 Timer
---(SP)	---(SI)	Timers	Pulse Timer Coil
S_PEXT	S_VIMP	Timers	Extended Pulse S5 Timer
S_PULSE	S_IMPULS	Timers	Pulse S5 Timer
SQR	SQR	Floating point Instruction	Establish the Square
SQRT	SQRT	Floating point Instruction	Establish the Square Root
SR	SR	Bit logic Instruction	Set-Reset Flip Flop
---(SS)	---(SS)	Timers	Retentive On-Delay Timer Coil
SUB_DI	SUB_DI	Integer Math Instruction	Subtract Double Integer
SUB_I	SUB_I	Integer Math Instruction	Subtract Integer
SUB_R	SUB_R	Floating point Instruction	Subtract Real
TAN	TAN	Floating point Instruction	Establish the Tangent Value
TRUNC	TRUNC	Convert	Truncate Double Integer Part
UO ---| |---	UO ---| |---	Status bits	Exception Bit Unordered
WAND_DW	WAND_DW	Word logic Instruction	AND Double Word
WAND_W	WAND_W	Word logic Instruction	AND Word
WOR_DW	WOR_DW	Word logic Instruction	OR Double Word
WOR_W	WOR_W	Word logic Instruction	OR Word
WXOR_DW	WXOR_DW	Word logic Instruction	Exclusive OR Double Word
WXOR_W	WXOR_W	Word logic Instruction	Exclusive OR Word

A.2 LAD Instructions Sorted According to German Mnemonics (SIMATIC)

German Mnemonics	English Mnemonics	Program Elements Catalog	Description
---| |---	---| |---	Bit logic Instruction	Normally Open Contact (Address)
---|/|---	---|/|---	Bit logic Instruction	Normally Closed Contact (Address)
---( )	---( )	Bit logic Instruction	Output Coil
---(#)---	---(#)---	Bit logic Instruction	Midline Output
==0 ---| |---	==0 ---| |---	Status bits	Result Bit Equal 0
>0 ---| |---	>0 ---| |---	Status bits	Result Bit Greater Than 0
>=0 ---| |---	>=0 ---| |---	Status bits	Result Bit Greater Equal 0
<=0 ---| |---	<=0 ---| |---	Status bits	Result Bit Less Equal 0
<0 ---| |---	<0 ---| |---	Status bits	Result Bit Less Than 0
<>0 ---| |---	<>0 ---| |---	Status bits	Result Bit Not Equal 0
ABS	ABS	Floating point Instruction	Establish the Absolute Value of a Floating-Point Number
ACOS	ACOS	Floating point Instruction	Establish the Arc Cosine Value
ADD_DI	ADD_DI	Integer Math Instruction	Add Double Integer
ADD_I	ADD_I	Integer Math Instruction	Add Integer
ADD_R	ADD_R	Floating point Instruction	Add Real
ASIN	ASIN	Floating point Instruction	Establish the Arc Sine Value
ATAN	ATAN	Floating point Instruction	Establish the Arc Tangent Value
BCD_DI	BCD_DI	Convert	BCD to Double Integer
BCD_I	BCD_I	Convert	BCD to Integer
BIE ---| |---	BR ---| |---	Status bits	Exception Bit Binary Result
----(CALL)	----(CALL)	Program control	Call FC SFC from Coil (without Parameters)
CALL_FB	CALL_FB	Program control	Call FB from Box
CALL_FC	CALL_FC	Program control	Call FC from Box
CALL_SFB	CALL_SFB	Program control	Call System FB from Box
CALL_SFC	CALL_SFC	Program control	Call System FC from Box
CEIL	CEIL	Convert	Ceiling
CMP >=D	CMP >=D	Compare	Compare Double Integer (==, <>, >, <, >=, <=)
CMP >=I	CMP >=I	Compare	Compare Integer (==, <>, >, <, >=, <=)
CMP >=R	CMP >=R	Compare	Compare Real (==, <>, >, <, >=, <=)
COS	COS	Floating point Instruction	Establish the Cosine Value
DI_BCD	DI_BCD	Convert	Double Integer to BCD

German Mnemonics	**English Mnemonics**	**Program Elements Catalog**	**Description**
DI_R	DI_R	Convert	Double Integer to Floating-Point
DIV_DI	DIV_DI	Integer Math Instruction	Divide Double Integer
DIV_I	DIV_I	Integer Math Instruction	Divide Integer
DIV_R	DIV_R	Floating point Instruction	Divide Real
EXP	EXP	Floating point Instruction	Establish the Exponential Value
FLOOR	FLOOR	Convert	Floor
I_BCD	I_BCD	Convert	Integer to BCD
I_DI	I_DI	Convert	Integer to Double Integer
INV_I	INV_I	Convert	Ones Complement Integer
INV_DI	INV_DI	Convert	Ones Complement Double Integer
---(JMP)	---(JMP)	Jumps	Conditional Jump
---(JMP)	---(JMP)	Jumps	Unconditional Jump
---(JMPN)	---(JMPN)	Jumps	Jump-If-Not
LABEL	LABEL	Jumps	Label
LN	LN	Floating point Instruction	Establish the Natural Logarithm
---(MCR>)	---(MCR>)	Program control	Master Control Relay Off
---(MCR<)	---(MCR<)	Program control	Master Control Relay On
---(MCRA)	---(MCRA)	Program control	Master Control Relay Activate
---(MCRD)	---(MCRD)	Program control	Master Control Relay Deactivate
MOD_DI	MOD_DI	Integer Math Instruction	Return Fraction Double Integer
MOVE	MOVE	Move	Assign a Value
MUL_DI	MUL_DI	Integer Math Instruction	Multiply Double Integer
MUL_I	MUL_I	Integer Math Instruction	Multiply Integer
MUL_R	MUL_R	Floating point Instruction	Multiply Real
---(N)---	---(N)---	Bit logic Instruction	Negative RLO Edge Detection
NEG	NEG	Bit logic Instruction	Address Negative Edge Detection
NEG_DI	NEG_DI	Convert	Twos Complement Double Integer
NEG_I	NEG_I	Convert	Twos Complement Integer
NEG_R	NEG_R	Convert	Negate Floating-Point Number
---| NOT |---	---| NOT |---	Bit logic Instruction	Invert Power Flow
---(OPN)	---(OPN)	DB call	Open Data Block: DB or DI
OS ---| |---	OS ---| |---	Status bits	Exception Bit Overflow Stored
OV ---| |---	OV ---| |---	Status bits	Exception Bit Overflow
---(P)---	---(P)---	Bit logic Instruction	Positive RLO Edge Detection

German Mnemonics	**English Mnemonics**	**Program Elements Catalog**	**Description**
POS	POS	Bit logic Instruction	Address Positive Edge Detection
---(R)	---(R)	Bit logic Instruction	Reset Coil
---(RET)	---(RET)	Program control	Return
ROL_DW	ROL_DW	Shift/Rotate	Rotate Left Double Word
ROR_DW	ROR_DW	Shift/Rotate	Rotate Right Double Word
ROUND	ROUND	Convert	Round to Double Integer
RS	RS	Bit logic Instruction	Reset-Set Flip Flop
---(S)	---(S)	Bit logic Instruction	Set Coil
---(SA)	---(SF)	Timers	Off-Delay Timer Coil
---(SAVE)	---(SAVE)	Bit logic Instruction	Save RLO into BR Memory
S_AVERZ	S_OFFDT	Timers	Off-Delay S5 Timer
---(SE)	---(SD)	Timers	On-Delay Timer Coil
S_EVERZ	S_ODT	Timers	On-Delay S5 Timer
SHL_DW	SHL_DW	Shift/Rotate	Shift Left Double Word
SHL_W	SHL_W	Shift/Rotate	Shift Left Word
SHR_DI	SHR_DI	Shift/Rotate	Shift Right Double Integer
SHR_DW	SHR_DW	Shift/Rotate	Shift Right Double Word
SHR_I	SHR_I	Shift/Rotate	Shift Right Integer
SHR_W	SHR_W	Shift/Rotate	Shift Right Word
---(SI)	---(SP)	Timers	Pulse Timer Coil
S_IMPULS	S_PULSE	Timers	Pulse S5 Timer
SIN	SIN	Floating point Instruction	Establish the Sine Value
SQR	SQR	Floating point Instruction	Establish the Square
SQRT	SQRT	Floating point Instruction	Establish the Square Root
SR	SR	Bit logic Instruction	Set-Reset Flip Flop
---(SS)	---(SS)	Timers	Retentive On-Delay Timer Coil
S_SEVERZ	S_ODTS	Timers	Retentive On-Delay S5 Timer
SUB_DI	SUB_DI	Integer Math Instruction	Subtract Double Integer
SUB_I	SUB_I	Integer Math Instruction	Subtract Integer
SUB_R	SUB_R	Floating point Instruction	Subtract Real
---(SV)	---(SE)	Timers	Extended Pulse Timer Coil
S_VIMP	S_PEXT	Timers	Extended Pulse S5 Timer
---(SZ)	---(SC)	Counters	Set Counter Value
TAN	TAN	Floating point Instruction	Establish the Tangent Value
TRUNC	TRUNC	Convert	Truncate Double Integer Part
UO ---| |---	UO ---| |---	Status bits	Exception Bit Unordered

German Mnemonics	English Mnemonics	Program Elements Catalog	Description
WAND_DW	WAND_DW	Word logic Instruction	AND Double Word
WAND_W	WAND_W	Word logic Instruction	AND Word
WOR_DW	WOR_DW	Word logic Instruction	OR Double Word
WOR_W	WOR_W	Word logic Instruction	OR Word
WXOR_DW	WXOR_DW	Word logic Instruction	Exclusive OR Double Word
WXOR_W	WXOR_W	Word logic Instruction	Exclusive OR Word
ZAEHLER	S_CUD	Counters	Up-Down Counter
----(ZR)	----(CD)	Counters	Down Counter Coil
Z_RUECK	S_CD	Counters	Down Counter
---(ZV)	----(CU)	Counters	Up Counter Coil
Z_VORW	S_CU	Counters	Up Counter

B Programming Examples

B.1 Overview of Programming Examples

Practical Applications

Each ladder logic instruction described in this manual triggers a specific operation. When you combine these instructions into a program, you can accomplish a wide variety of automation tasks. This chapter provides the following examples of practical applications of the ladder logic instructions:

- Controlling a conveyor belt using bit logic instructions
- Detecting direction of movement on a conveyor belt using bit logic instructions
- Generating a clock pulse using timer instructions
- Keeping track of storage space using counter and comparison instructions
- Solving a problem using integer math instructions
- Setting the length of time for heating an oven

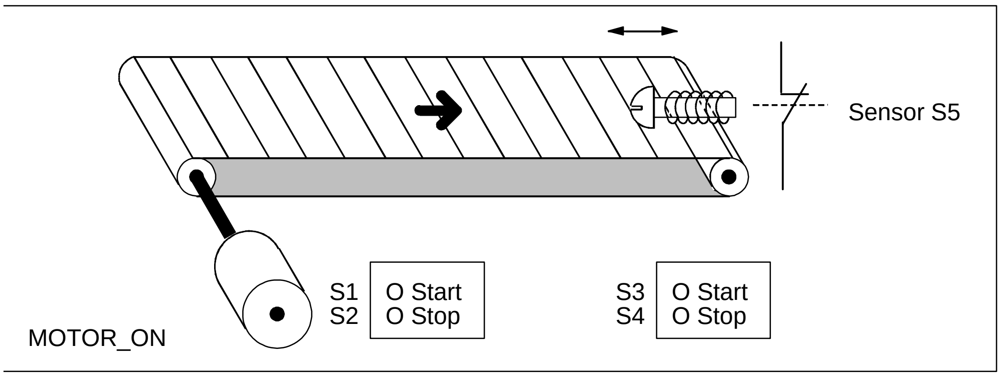
Instructions Used

Mnemonik	**Program Elements Catalog**	**Description**
WAND_W	Word logic instruction	(Word) And Word
WOR_W	Word logic instruction	(Word) Or Word
--- (CD)	Counters	Down Counter Coil
--- (CU)	Counters	Up Counter Coil
---(R)	Bit logic instruction	Reset Coil
---(S)	Bit logic instruction	Set Coil
---(P)	Bit logic instruction	Positive RLO Edge Detection
ADD_I	Floating-Point instruction	Add Integer
DIV_I	Floating-Point instruction	Divide Integer
MUL_I	Floating-Point instruction	Multiply Integer
CMP <=I, CMP >=I	Compare	Compare Integer
—| |—	Bit logic instruction	Normally Open Contact
—| / |—	Bit logic instruction	Normally Closed Contact
—( )	Bit logic instruction	Output Coil
---(JMPN)	Jumps	Jump-If-Not
---(RET)	Program control	Return
MOVE	Move	Assign a Value
--- (SE)	Timers	Extended Pulse Timer Coil

B.2 Example: Bit Logic Instructions

Example 1: Controlling a Conveyor Belt

The following figure shows a conveyor belt that can be activated electrically. There are two push button switches at the beginning of the belt: S1 for START and S2 for STOP. There are also two push button switches at the end of the belt: S3 for START and S4 for STOP. It is possible to start or stop the belt from either end. Also, sensor S5 stops the belt when an item on the belt reaches the end.



Absolute and symbolic Programming

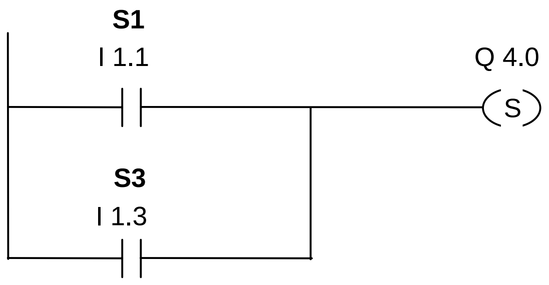
You can write a program to control the conveyor belt using **absolute values** or **symbols** that represent the various components of the conveyor system.

You need to make a symbol table to correlate the symbols you choose with absolute values (see the STEP 7 Online Help).

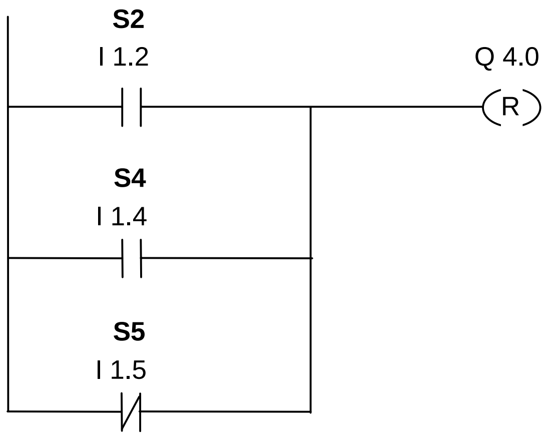
System Component	Absolute Address	Symbol	Symbol Table
Push Button Start Switch	I 1.1	S1	I 1.1 S1
Push Button Stop Switch	I 1.2	S2	I 1.2 S2
Push Button Start Switch	I 1.3	S3	I 1.3 S3
Push Button Stop Switch	I 1.4	S4	I 1.4 S4
Sensor	I 1.5	S5	I 1.5 S5
Motor	Q 4.0	MOTOR_ON	Q 4.0 MOTOR_ON

Ladder Logic Program to control the conveyor belt

Network 1: Pressing either start switch turns the motor on.

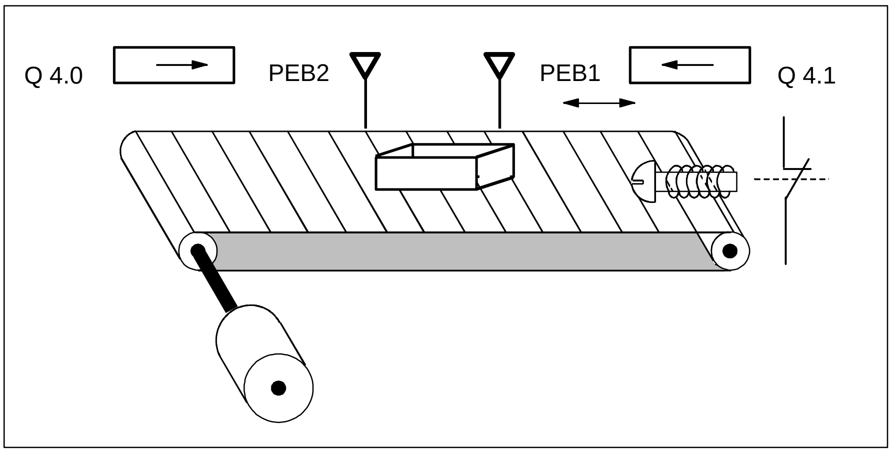


Network 2: Pressing either stop switch or opening the normally closed contact at the end of the belt turns the motor off.



Example 2: Detecting the Direction of a Conveyor Belt

The following figure shows a conveyor belt that is equipped with two photoelectric barriers (PEB1 and PEB2) that are designed to detect the direction in which a package is moving on the belt. Each photoelectric light barrier functions like a normally open contact.



Absolute and symbolic Programming

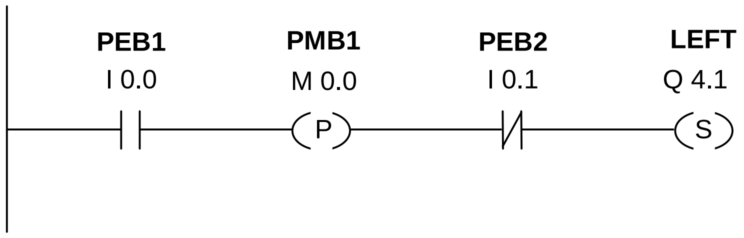
You can write a program to activate a direction display for the conveyor belt system using **absolute values** or **symbols** that represent the various components of the conveyor system.

You need to make a symbol table to correlate the symbols you choose with absolute values (see the STEP 7 Online Help).

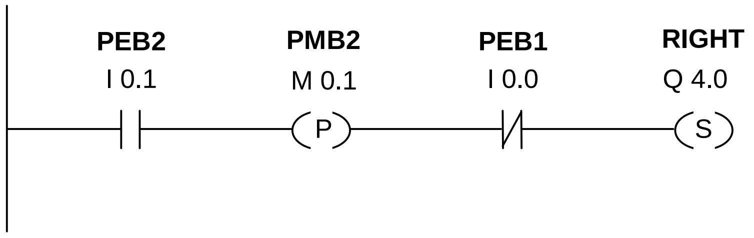
System Component	**Absolute Address**	**Symbol**	**Symbol Table**
Photoelectric barrier 1	I 0.0	PEB1	I 0.0 PEB1
Photoelectric barrier 2	I 0.1	PEB2	I 0.1 PEB2
Display for movement to right	Q 4.0	RIGHT	Q 4.0 RIGHT
Display for movement to left	Q 4.1	LEFT	Q 4.1 LEFT
Pulse memory bit 1	M 0.0	PMB1	M 0.0 PMB1
Pulse memory bit 2	M 0.1	PMB2	M 0.1 PMB2

Ladder Logic Program for Detecting the Direction of a Conveyor Belt

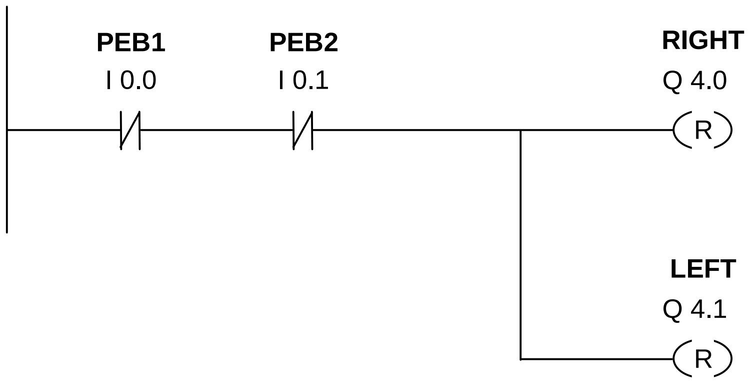
Network 1: If there is a transition in signal state from 0 to 1 (positive edge) at input I 0.0 and, at the same time, the signal state at input I 0.1 is 0, then the package on the belt is moving to the left.



Network 2: If there is a transition in signal state from 0 to 1 (positive edge) at input I 0.1 and, at the same time, the signal state at input I 0.0 is 0, then the package on the belt is moving to the right. If one of the photoelectric light barriers is broken, this means that there is a package between the barriers.



Network 3: If neither photoelectric barrier is broken, then there is no package between the barriers. The direction pointer shuts off.



B.3 Example: Timer Instructions

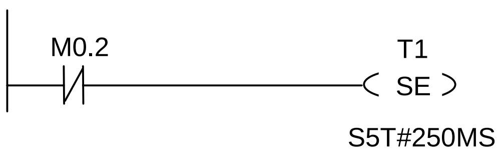
Clock Pulse Generator

You can use a clock pulse generator or flasher relay when you need to produce a signal that repeats periodically. A clock pulse generator is common in a signalling system that controls the flashing of indicator lamps.

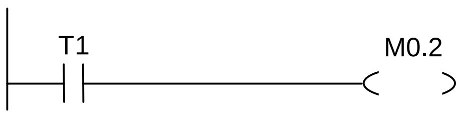
When you use the S7-300, you can implement the clock pulse generator function by using time-driven processing in special organization blocks. The example shown in the following ladder logic program, however, illustrates the use of timer functions to generate a clock pulse. The sample program shows how to implement a freewheeling clock pulse generator by using a timer.

Ladder Logic Program to Generate a Clock Pulse (pulse duty factor 1:1)

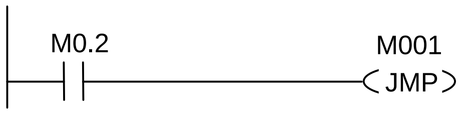
Network 1: If the signal state of timer T1 is 0, load the time value 250 ms into T1 and start T1 as an extended-pulse timer.



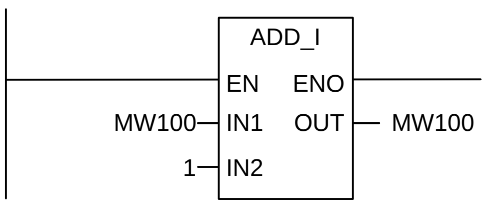
Network 2: The state of the timer is saved temporarily in an auxiliary memory marker.



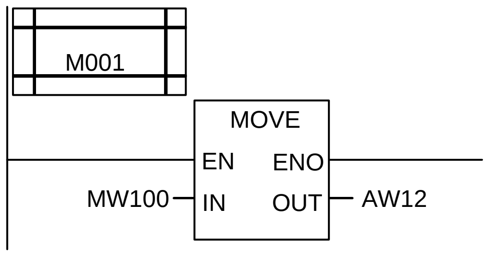
Network 3: If the signal state of timer T1 is 1, jump to jump label M001.



Network 4: When the timer T1 expires, the memory word 100 is incremented by 1.

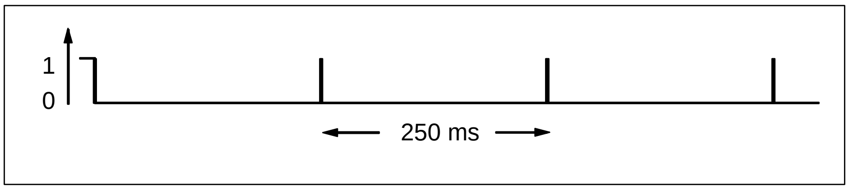


Network 5: The **MOVE** instruction allows you to output the different clock frequencies at outputs Q12.0 through Q13.7.



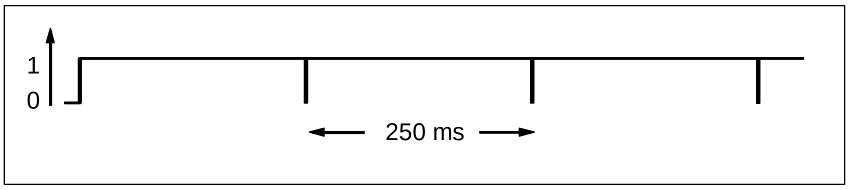
Signal Check

A signal check of timer T1 produces the following result of logic operation (RLO) for opener M0.2.



As soon as the time runs out, the timer is restarted. Because of this, the signal check made by —| / |— M0.2 produces a signal state of 1 only briefly.

The negated (inverted) RLO:



Every 250 ms the RLO bit is 0. The jump is ignored and the contents of memory word MW100 is incremented by 1.

Achieving a Specific Frequency

From the individual bits of memory bytes MB101 and MB100 you can achieve the following frequencies:

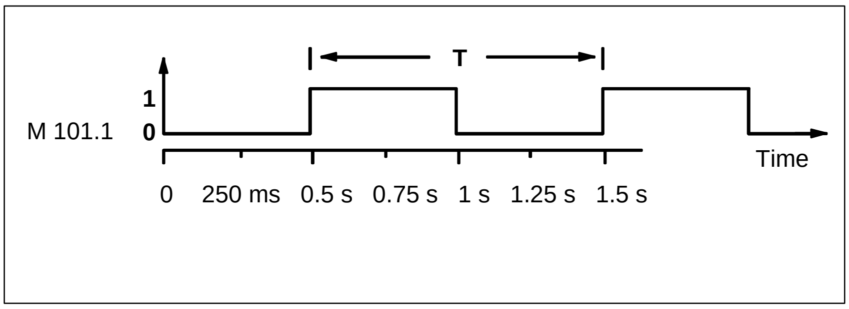
Bits of MB101/MB100	**Frequency in Hz**	**Duration**
M 101.0	2.0	0.5 s (250 ms on / 250 ms off)
M 101.1	1.0	1 s (0.5 s on / 0.5 s off)
M 101.2	0.5	2 s (1 s on / 1 s off)
M 101.3	0.25	4 s (2 s on / 2 s off)
M 101.4	0.125	8 s (4 s on / 4 s off)
M 101.5	0.0625	16 s (8 s on / 8 s off)
M 101.6	0.03125	32 s (16 s on / 16 s off)
M 101.7	0.015625	64 s (32 s on / 32 s off)
M 100.0	0.0078125	128 s (64 s on / 64 s off)
M 100.1	0.0039062	256 s (128 s on / 128 s off)
M 100.2	0.0019531	512 s (256 s on / 256 s off)
M 100.3	0.0009765	1024 s (512 s on / 512 s off)
M 100.4	0.0004882	2048 s (1024 s on / 1024 s off)
M 100.5	0.0002441	4096 s (2048 s on / 2048 s off)
M 100.6	0.000122	8192 s (4096 s on / 4096 s off)
M 100.7	0.000061	16384 s (8192 s on / 8192 s off)

Signal states of the Bits of Memory MB 101

Scan Cycle	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**	**Time Value in ms**
0	0	0	0	0	0	0	**0**	0	250
1	0	0	0	0	0	0	**0**	1	250
2	0	0	0	0	0	0	1	0	250
3	0	0	0	0	0	0	1	1	250
4	0	0	0	0	0	1	**0**	0	250
5	0	0	0	0	0	1	**0**	1	250
6	0	0	0	0	0	1	1	0	250
7	0	0	0	0	0	1	1	1	250
8	0	0	0	0	1	0	**0**	0	250
9	0	0	0	0	1	0	**0**	1	250
10	0	0	0	0	1	0	1	0	250
11	0	0	0	0	1	0	1	1	250
12	0	0	0	0	1	1	**0**	0	250

Signal state of Bit 1 of MB 101 (M 101.1)

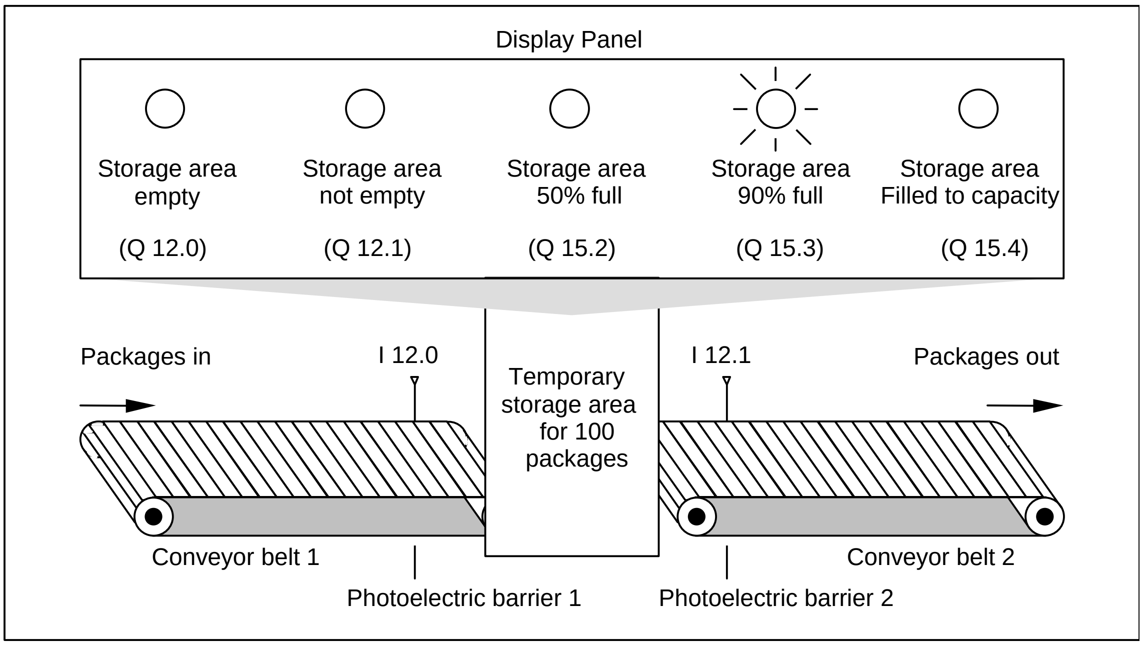
Frequency = 1/T = 1/1 s = 1 Hz



B.4 Example: Counter and Comparison Instructions

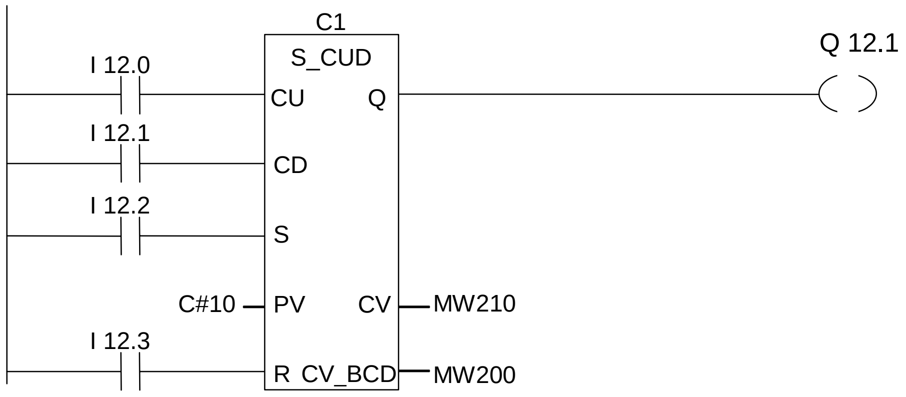
Storage Area with Counter and Comparator

The following figure shows a system with two conveyor belts and a temporary storage area in between them. Conveyor belt 1 delivers packages to the storage area. A photoelectric barrier at the end of conveyor belt 1 near the storage area determines how many packages are delivered to the storage area. Conveyor belt 2 transports packages from the temporary storage area to a loading dock where trucks take the packages away for delivery to customers. A photoelectric barrier at the end of conveyor belt 2 near the storage area determines how many packages leave the storage area to go to the loading dock. A display panel with five lamps indicates the fill level of the temporary storage area.

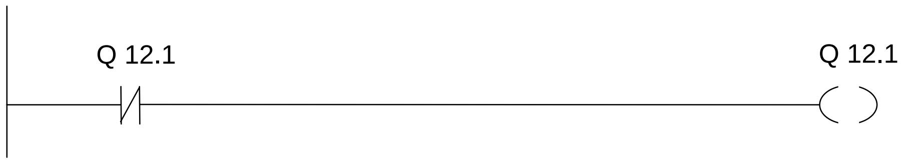


Ladder Logic Program that Activates the Indicator Lamps on the Display Panel

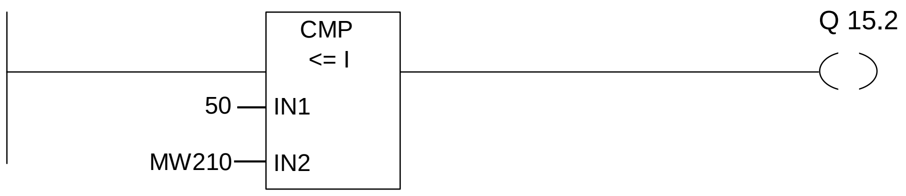
Network 1: Counter C1 counts up at each signal change from ”0” to ”1” at input CU and counts down at each signal change from ”0” to ”1” at input CD. With a signal change from ”0” to ”1” at input S, the counter value is set to the value PV. A signal change from ”0” to ”1” at input R resets the counter value to ”0”. MW200 contains the current counter value of C1. Q12.1 indicates ”storage area not empty”.



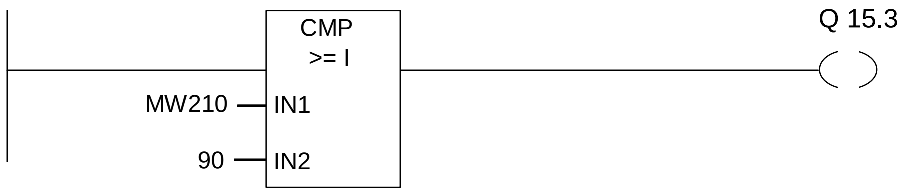
Network 2: Q12.0 indicates ”storage area empty”.



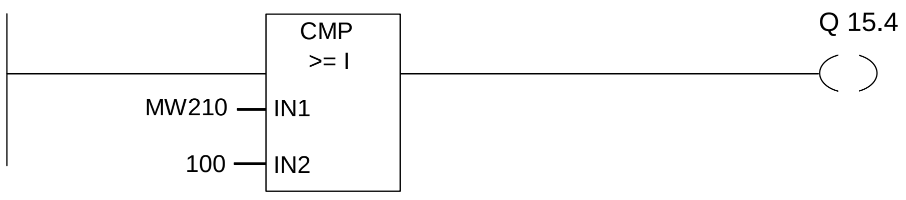
Network 3: If 50 is less than or equal to the counter value (in other words if the current counter value is greater than or equal to 50), the indicator lamp for ”storage area 50% full” is lit.



Network 4: Network 4: If the counter value is greater than or equal to 90, the indicator lamp for ”storage area 90% full” is lit.



Network 5: If the counter value is greater than or equal to 100, the indicator lamp for ”storage area full” is lit.



B.5 Example: Integer Math Instructions

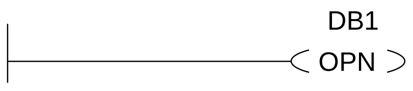
Solving a Math Problem

The sample program shows you how to use three integer math instructions to produce the same result as the following equation:

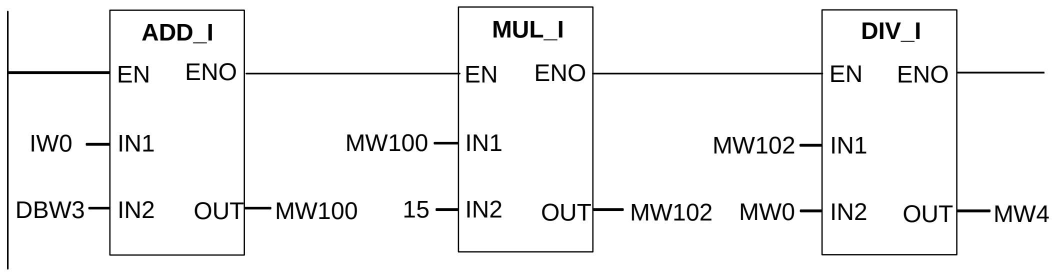
MW4 = ((IW0 + DBW3) x 15) / MW0

Ladder Logic Program

Network 1: Open Data Block DB1.



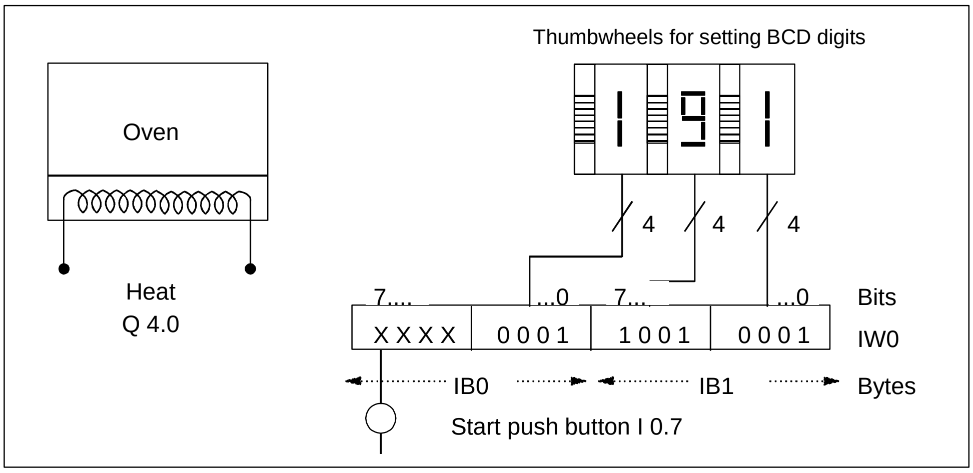
Network 2: Input word IW0 is added to shared data word DBW3 (data block must be defined and opened) and the sum is loaded into memory word MW100. MW100 is then multiplied by 15 and the answer stored in memory word MW102. MW102 is divided by MW0 with the result stored in MW4.



B.6 Example: Word Logic Instructions

Heating an Oven

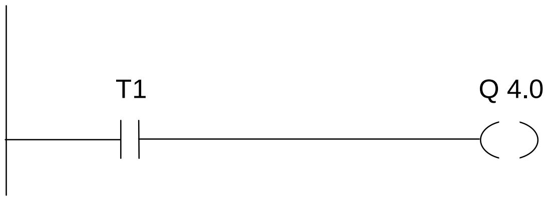
The operator of the oven starts the oven heating by pushing the start push button. The operator can set the length of time for heating by using the thumbwheel switches shown in the figure. The value that the operator sets indicates seconds in binary coded decimal (BCD) format.



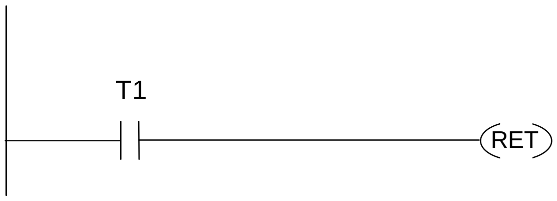
System Component	Absolute Address
Start Push Button	I 0.7
Thumbwheel for ones	I 1.0 to I 1.3
Thumbwheel for tes	I 1.4 to I 1.7
Thumbwheel for hundreds	I 0.0 to I 0.3
Heating starts	Q 4.0

Ladder Logic Program

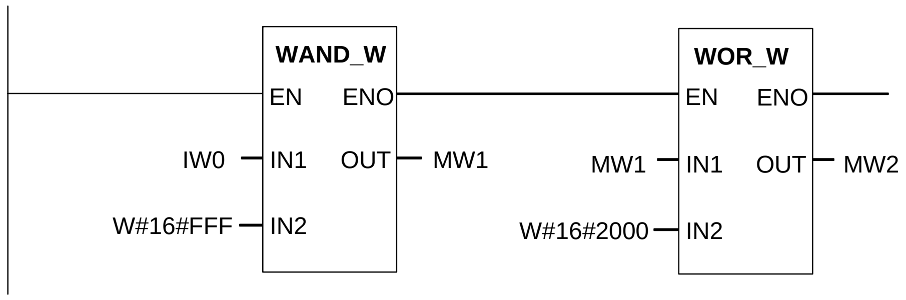
Network 1: If the timer is running, then turn on the heater.



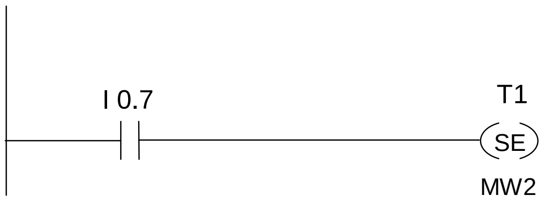
Network 2: If the timer is running, the **Return** instruction ends the processing here.



Network 3: Mask input bits I 0.4 through I 0.7 (that is, reset them to 0). These bits of the thumbwheel inputs are not used. The 16 bits of the thumbwheel inputs are combined with W#16#0FFF according to the **(Word) And Word** instruction. The result is loaded into memory word MW1. In order to set the time base of seconds, the preset value is combined with W#16#2000 according to the **(Word) Or Word** instruction, setting bit 13 to 1 and resetting bit 12 to 0.



Network 4: Start timer T 1 as an extended pulse timer if the start push button is pressed, loading as a preset value memory word MW2 (derived from the logic above).



C Working with Ladder Logic

C.1 EN/ENO Mechanism

The enable (EN) and enable output (ENO) of FBD/LAD boxes is achieved by means of the BR bit.

If EN and ENO are connected, the following applies:

ENO = EN AND NOT (box error)

If no error occurs (box error = 0), ENO = EN.

The EN/ENO mechanism is used for:

- Math instructions,
- Transfer and conversion instructions,
- Shift and rotate instructions,
- Block calls.

This mechanism is **not** used for:

- Comparisons,
- Counters,
- Timers.

Around the actual instructions in the box, additional STL instructions are generated for the EN/ENO mechanism with dependency on the existing preceding and subsequent logic operations. The four possible cases are shown using the example of an adder:

1. Adder with EN and with ENO Connected
2. Adder with EN and without ENO Connected
3. Adder without EN and with ENO Connected
4. Adder without EN and without ENO Connected

Note on Creating Your Own Blocks

If you want to program blocks which you want to call in FBD or LAD, you must ensure that the BR bit is set when the block is exited. The fourth example shows that this is not automatically the case. You cannot use the BR as a memory bit because it is constantly overwritten by the EN/ENO mechanism. Instead, use a temporary variable in which you save any errors which occur. Initialize this variable with 0. At each point in the block at which you think an unsuccessful instruction represents an error for the whole block, set this variable using the assistance of the EN/ENO mechanism. A NOT and a SET coil will be sufficient for this. At the end of the block program the following network:

```
end:   AN error

       SAVE
```

Ensure that this network is processed in every case, which means you must not use BEC within the block and skip this network.

C.1.1 Adder with EN and with ENO Connected

If the adder has an EN and an ENO connected, the following STL instructions are triggered:

```
1   A   I   0.0         // EN connection
2   JNB _001            // Shift RLO into BR and jump if RLO = 0
3   L     in1           // Box parameter
4   L     in2           // Box parameter
5   +I                  // Actual addition
6   T     out           // Box parameter
7   AN    OV            // Error recognition
8   SAVE                // Save error in BR
9   CLR                 // First check
10  _001: A     BR      // Shift BR into RLO
11  =     Q   4.0
```

Following line 1 the RLO contains the result of the preceding logic operation. The JNB instruction copies the RLO into the BR bit and sets the first check bit.

- If the RLO = 0, the program jumps to line 10 and resumes with A BR. The addition is not executed. In line 10 the BR is copied into the RLO again and 0 is thus assigned to the output.
- If the RLO = 1, the program does not jump, meaning the addition is executed. In line 7 the program evaluates whether an error occurred during addition, this is then stored in BR in line 8. Line 9 sets the first check bit. Now the BR bit is copied back into the RLO in line 10 and thus the output shows whether the addition was successful or not.
  The BR bit is not changed by lines 10 and 11, so it also shows whether the addition was successful.

C.1.2 Adder with EN and without ENO Connected

If the adder has an EN but no ENO connected, the following STL instructions are triggered:

```
1  A     I      0.0  // EN connection
2  JNB _001          // Shift RLO into BR and jump if RLO = 0
3  L     in1         // Box parameter
4  L     in2         // Box parameter
5  +I                // Actual addition
6  T     out         // Box parameter
7  _001:  NOP   0
```

Following line 1 the RLO contains the result of the preceding logic operation. The JNB instruction copies the RLO into the BR bit and sets the first check bit.

- If the RLO = 0, the program jumps to line 7 and the addition is not executed. The RLO and BR are 0.
- If RLO was 1, the program does not jump, meaning the addition is executed. The program does not evaluate whether an error occurred during addition. The RLO and BR are 1.

C.1.3 Adder without EN and with ENO Connected

If the adder has no EN but an ENO connected, the following STL instructions are triggered:

```
1  L     in1     // Box parameter

2  L     in2     // Box parameter

3  +I            // Actual addition

4  T     out     // Box parameter

5  AN    OV      // Error recognition

6  SAVE          // Save error in BR

7  CLR           // First check

8  A     BR      // Shift BR into RLO

9  =  Q   4.0
```

The addition is executed in every case. In line 5 the program evaluates whether an error occurred during addition, this is then stored in BR in line 6. Line 7 sets the first check bit. Now the BR bit is copied back into the RLO in line 8 and thus the output shows whether the addition was successful or not.

The BR bit is not changed by lines 8 and 9, so it also shows whether the addition was successful.

C.1.4 Adder without EN and without ENO Connected

If the adder has no EN and no ENO connected, the following STL instructions are triggered:

```
L     in1     // Box parameter
L     in2     // Box parameter
+I            // Actual addition
T     out     // Box parameter
NOP 0
```

The addition is executed. The RLO and the BR bit remain unchanged.

C.2 Parameter Transfer

The parameters of a block are transferred as a value. With function blocks a copy of the actual parameter value in the instance data block is used in the called block. With functions a copy of the actual value lies in the local data stack. Pointers are not copied. Prior to the call the INPUT values are copied into the instance DB or to the L stack. After the call the OUTPUT values are copied back into the variables. Within the called block you can only work on a copy. The STL instructions required for this are in the calling block and remain hidden from the user.

Note

If memory bits, inputs, outputs or peripheral I/Os are used as actual address of a function they are treated in a different way than the other addresses. Here, updates are carried out directly, not via L Stack.

Exception:

If the corresponding formal parameter is an input parameter of the data type BOOL, the current parameters are updated via the L stack.

Caution

When programming the called block, ensure that the parameters declared as OUTPUT are also written. Otherwise the values output are random! With function blocks the value will be the value from the instance DB noted by the last call, with functions the value will be the value which happens to be in the L stack.

Note the following points:

- Initialize all OUTPUT parameters if possible.
- Try not to use any Set and Reset instructions. These instructions are dependent on the RLO. If the RLO has the value 0, the random value will be retained.
- If you jump within the block, ensure that you do not skip any locations where OUTPUT parameters are written. Do not forget BEC and the effect of the MCR instructions.

Index