ABB low voltage AC drives

Application guide
Application programming for ACS850 and ACQ810 drives

Power and productivity
for a better world™

List of related manuals

Drive hardware manuals and guides	Code (English)	
ACS850-04 (0.37…45 kW) hardware manual	*3AUA0000045496*	*1)*
ACS850-04 (0.37…45 kW) quick installation guide	*3AUA0000045495*	*1)*
ACS850-04 (55…160 kW, 75…200 hp) hardware manual	*3AUA0000045487*	*1)*
ACS850-04 (55…160 kW, 75…200 hp) quick installation guide	*3AUA0000045488*	*1)*
ACS850-04 (200…500 kW, 250…600 hp) hardware manual	*3AUA0000026234*	*1)*
ACS850-04 (400…560 kW, 450…700 hp) hardware manual	*3AUA0000081249*	*1)*
ACQ810-04 (0.37…45 kW, 0.5…60 hp) hardware manual	*3AUA0000055160*	*1)*
ACQ810-04 (55…160 kW, 75…200 hp) hardware manual	*3AUA0000055161*	*1)*
ACQ810-04 (200…400 kW, 250…600 hp) hardware manual	*3AUA0000055155*	*1)*

Drive firmware manuals and guides		
ACS850 standard control program quick start-up guide	*3AUA0000045498*	*2)*
ACS850 standard control program firmware manual	*3AUA0000045497*	*3)*
ACQ810-04 drive modules start-up guide	*3AUA0000055159*	*2)*
ACQ810 standard pump control program firmware manual	*3AUA0000055144*	*3)*

Option manuals and guides		
ACS-CP-U control panel IP54 mounting platform kit (+J410) installation guide	*3AUA0000049072*	*1)*
Manuals and quick guides for I/O extension modules, fieldbus adapters, etc.		*1)*

Drive PC tools manuals	
DriveSPC user manual	*3AFE68836590*
DriveStudio user manual	*3AFE68749026*

1) Delivered as a printed copy with the drive if the order includes printed manuals.

2) Delivered as a printed copy with the control program.

3) Delivered as a printed copy with the control program if the order includes printed manuals.

All manuals are available in PDF format on the Internet. See section *Document library on the Internet* on the inside of the back cover.

Application guide

Application programming for ACS850 and ACQ810 drives

Table of contents

3AUA0000078664 Rev B
EN
EFFECTIVE: 2011-11-28

Table of contents

5. Examples of using standard function blocks

Further information

1

About the manual

What this chapter contains

This chapter describes the contents of this manual. It also contains information on the compatibility, safety, intended audience and the purpose of the manual.

Compatibility

This manual is compatible with ACS850 drives with the Standard control program and ACQ810 drives with the Standard pump control program.

Safety instructions

Follow all safety instructions delivered with the drive.

- Read the **complete safety instructions** before you install, commission, or use the drive. The complete safety instructions are given at the beginning of the drive *Hardware manual*.
- Read the **software function specific warnings and notes** before changing the default settings of the function. For each function, the warnings and notes are given in this manual in the section describing the related user-adjustable parameters.

Reader

The reader of the manual is expected to know the standard electrical wiring practices, electronic components, and electrical schematic symbols.

Purpose of the manual

The purpose of this manual is to provide the reader with the information needed in designing application programs for ACS850 and ACQ810 drives using the DriveSPC PC tool.

The manual is intended to be used together with the drive *Firmware manual*, which contains the basic information on the drive parameters.

Contents of the manual

The manual consists of the following chapters:

- *Drive programming* introduces drive programming and describes application programming using the DriveSPC PC tool.
- *Firmware function blocks* presents the firmware function blocks available.
- *Standard function blocks* presents the standards function blocks available.
- *Examples of using standard function blocks* contains examples of using standard function blocks in application programming.

2

Drive programming

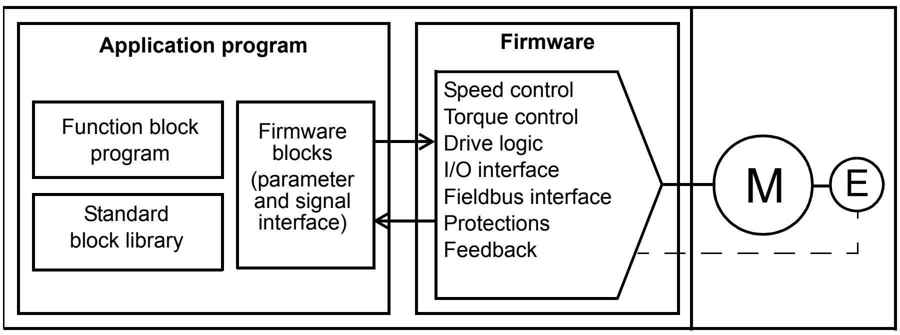
What this chapter contains

This chapter introduces drive programming and describes application programming using the DriveSPC PC tool.

General about drive programming

The drive control program is divided into two parts:
- firmware program
- application program.

Drive control program

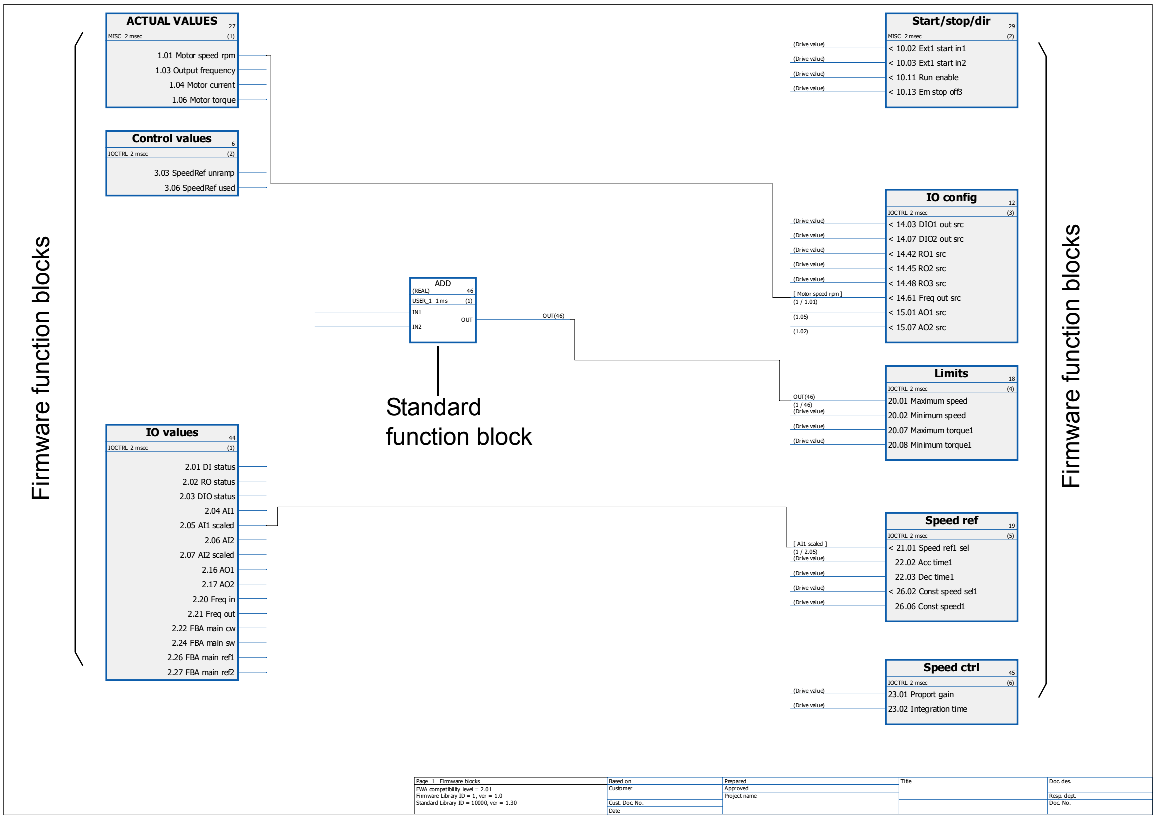


The firmware program performs the main control functions, including speed and torque control, drive logic (start/stop), I/O, feedback, communication and protection functions. Firmware functions are configured and programmed with parameters. Parameters can be set via the drive control panel, the DriveStudio PC tool or the fieldbus interface. For more information on programming via parameters, see the *Firmware manual*.

Application programming

The functions of the firmware program can be extended with application programming. The user can build an application program with firmware and standard functions blocks based on the IEC-61131 standard. ABB also offers customized application programs for specific applications; for more information, contact your local ABB representative.

Application programs are created with the DriveSPC PC tool. The following figure presents a view from DriveSPC.



■ Function blocks

An application program uses two types of function blocks: firmware function blocks and standard function blocks.

Firmware function blocks

The essential functions of the drive are represented as firmware function blocks in the DriveSPC PC tool. These blocks are part of the drive control firmware and act as an interface between the firmware and the application program. The inputs of the blocks correspond to drive parameters in groups 10…99 and can be modified via the application program; the outputs provide measured or calculated signals from groups 01…09. Note that not all parameters are accessible through the firmware function blocks.

The firmware function blocks available are presented in chapter *Firmware function blocks*.

Standard function blocks

Standard function blocks (for example, ADD, AND) are used to create an executable application program. The maximum size of an application program is approximately 30 standard function blocks, depending on the block types used. The standard function blocks available are presented in chapter *Standard function blocks*.

A standard function block library is always included in the drive delivery.

User parameters

User parameters can be created with the DriveSPC PC tool. User parameters can be added to any existing parameter group; the first available index is 70. Parameter groups 5 and 75...89 are available for user parameters starting from index 1. Using attributes, the parameters can be defined as write-protected, hidden, etc.

For more information, see *DriveSPC user manual* (3AFE68836590 [English]).

Application events

Application programmers can create their own application events (alarms and faults) by adding alarm and fault blocks; these blocks are managed through the Alarm and Fault Managers of the DriveSPC PC tool.

The operation of alarm and fault blocks is the same: when the block is enabled (by setting the Enable input to 1), an alarm or fault is generated by the drive.

Program execution

The application program is loaded to the permanent (non-volatile) memory of the memory unit (JMU). When the loading finishes, the drive control board is automatically reset and the downloaded program started. The program is executed in real time on the same Central Processing Unit (CPU of the drive control board) as the drive firmware. The program can be executed at the two dedicated time levels of 1 and 10 milliseconds, as well as other time levels between certain firmware tasks.

Note: Because the firmware and application programs use the same CPU, the programmer must ensure that the drive CPU is not overloaded. See parameter *01.21 Cpu usage*.

Application program licensing and protection

Note: This functionality is only available with DriveSPC version 1.5 and later.

The drive can be assigned an application licence consisting of an ID and password using the DriveSPC tool. Likewise, the application program created in DriveSPC can be protected by an ID and password. For instructions, see *DriveSPC user manual*.

If a protected application program is downloaded to a licensed drive, the IDs and passwords of the application and drive must match. A protected application cannot be downloaded to an unlicensed drive. On the other hand, an unprotected application can be downloaded to a licensed drive.

The ID of the application licence is displayed by DriveStudio in the drive software properties as APPL LICENCE. If the value is 0, no licence has been assigned to the drive.

Notes:

- The application licence can only be assigned to a complete drive, not a stand-alone control unit.
- The protected application can only be downloaded to a complete drive, not a stand-alone control unit.

Operation modes

The DriveSPC PC tool offers the following operation modes:

Off-line

When the off-line mode is used without a drive connection, the user can

- open an application program file (if it exists)
- modify and save the application program
- print the program pages.

When the off-line mode is used with a drive(s) connection, the user can

- connect the selected drive to DriveSPC
- upload an application program from the connected drive (an empty template which includes only the firmware blocks is available by default.)
- download the configured application program to the drive and start the program execution. The downloaded program contains the function block program and the parameter values set in DriveSPC.
- remove the program from the connected drive.

On-line

In the on-line mode, the user can

- modify firmware parameters (changes are stored directly to the drive memory)
- modify application program parameters (that is, parameters created in DriveSPC)
- monitor the actual values of all function blocks in real time.

3

Firmware function blocks

What this chapter contains

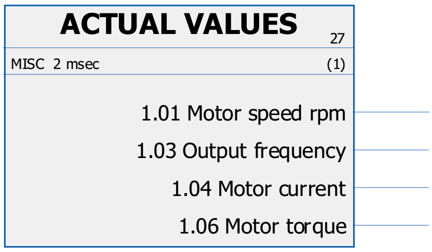
This chapter presents the firmware function blocks. The blocks are grouped according to parameter numbering in the drive firmware.

Note: The Speed ctrl block is not available with ACQ810 drives.

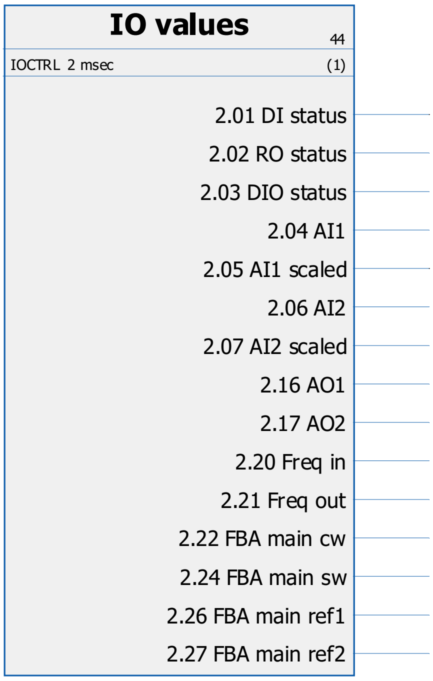
Note: Parameter 14.48 RO3 src does not exist with ACQ810 drives.

Description	Illustration

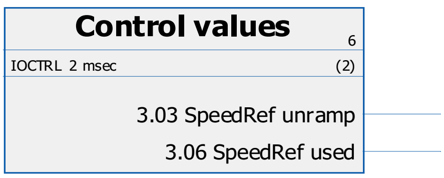
ACTUAL VALUES
(1)

Basic signals for monitoring the drive.	

IO values
(2)

Input and output signals.	

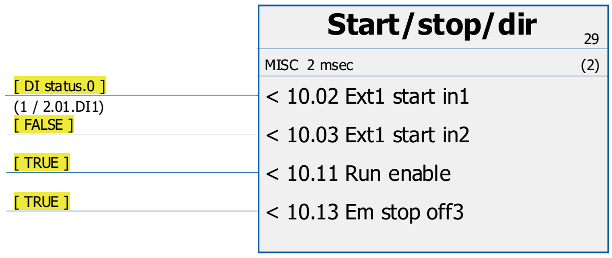
Control values
(3)

Speed reference values.	

Description	Illustration

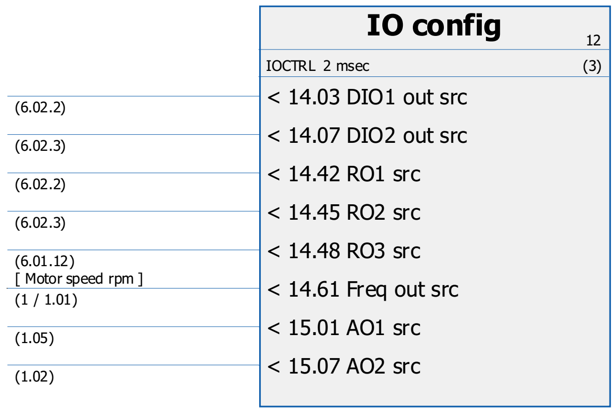
Start/stop/dir
(10)

Source selections for start/stop, run enable and emergency stop signals.



IO config
(13)

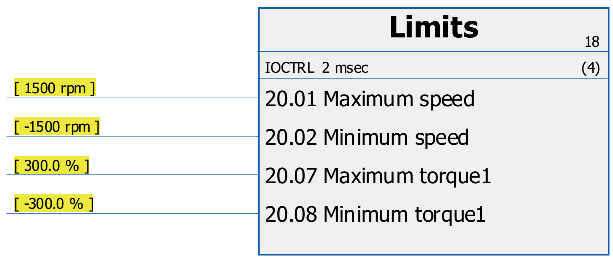
Configuration of digital input/outputs, relay outputs and analog outputs.



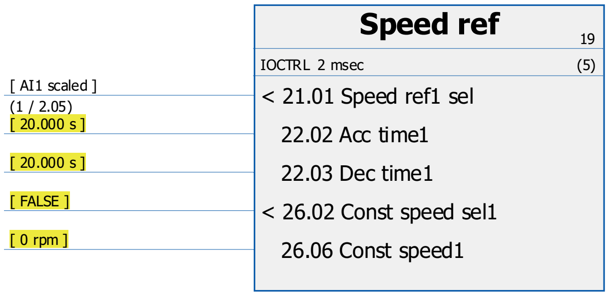
Note: Parameter 14.48 RO3 src does not exist with ACQ810 drives.

Limits
(20)

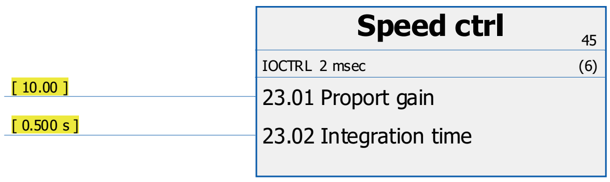
Drive operation limits.



<table>
<tr><th>Description</th><th>Illustration</th></tr>
<tr><td colspan="2">Speed ref
(21)</td></tr>
<tr><td>Speed reference source selection; acceleration/ deceleration and constant speed settings.</td><td>



</td></tr>
<tr><td colspan="2">Speed ctrl
(23)</td></tr>
<tr><td>Speed controller settings.</td><td>



Note: This block is not available with ACQ810 drives.</td></tr>
</table>

4

Standard function blocks

What this chapter contains

This chapter presents the standard function blocks. The blocks are grouped according to the grouping in the DriveSPC PC tool.

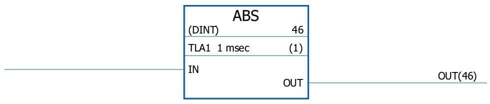
Note: The given execution times can vary depending on the drive application used.

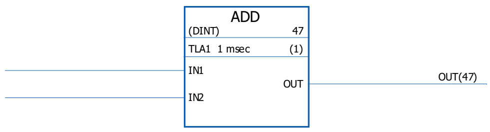
Terms

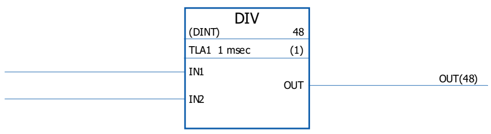
Data type	Description	Range
Boolean	Boolean	0 or 1
DINT	32-bit integer value (31 bits + sign)	-2147483648…2147483647
INT	16-bit integer value (15 bits + sign)	-32768…32767
PB	Packed Boolean	0 or 1 for each individual bit
REAL	$\underbrace{\text{16-bit value}}_{=\text{ integer value}}\ \underbrace{\text{16-bit value}}_{=\text{ fractional value}}$ (31 bits + sign)	-32768,99998…32767,9998
REAL24	$\underbrace{\text{8-bit value}}_{=\text{ integer value}}\ \underbrace{\text{24-bit value}}_{=\text{ fractional value}}$ (31 bits + sign)	-128,0…127,999

Alphabetical index

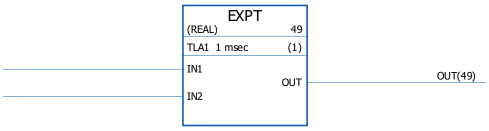
Arithmetic

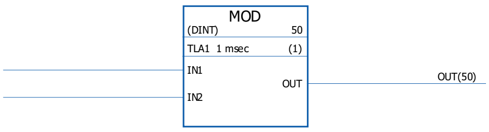
ABS (10001)	
Illustration	
Execution time	0.53 µs
Operation	The output (OUT) is the absolute value of the input (IN). OUT = | IN |
Inputs	The input data type is selected by the user. Input (IN): DINT, INT, REAL or REAL24
Outputs	Output (OUT): DINT, INT, REAL or REAL24

ADD (10000)	
Illustration	
Execution time	3.36 µs (when two inputs are used) + 0.52 µs (for every additional input). When all inputs are used, the execution time is 18.87 µs.
Operation	The output (OUT) is the sum of the inputs (IN1…IN32). OUT = IN1 + IN2 + … + IN32 The output value is limited to the maximum and minimum values defined by the selected data type range.
Inputs	The input data type and the number of the inputs (2…32) are selected by the user. Input (IN1…IN32): DINT, INT, REAL or REAL24
Outputs	Output (OUT): DINT, INT, REAL or REAL24

DIV (10002)	
Illustration	
Execution time	2.55 µs
Operation	The output (OUT) is input IN1 divided by input IN2. OUT = IN1/IN2 The output value is limited to the maximum and minimum values defined by the selected data type range. If the divider (IN2) is 0, the output is 0.

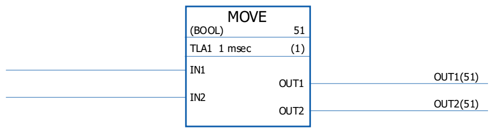
Inputs	The input data type is selected by the user. Input (IN1, IN2): INT, DINT, REAL, REAL24
Outputs	Output (OUT): INT, DINT, REAL, REAL24

EXPT (10003)	
Illustration	
Execution time	81.90 ìs
Operation	The output (OUT) is input IN1 raised to the power of the input IN2: OUT = IN1IN2 If input IN1 is 0, the output is 0. The output value is limited to the maximum value defined by the selected data type range. **Note:** The execution of the EXPT function is slow.
Inputs	The input data type is selected by the user. Input (IN1): REAL, REAL24 Input (IN2): REAL
Outputs	Output (OUT): REAL, REAL24

MOD (10004)	
Illustration	
Execution time	1.67 µs
Operation	The output (OUT) is the remainder of the division of the inputs IN1 and IN2. OUT = remainder of IN1/IN2 If input IN2 is zero, the output is zero.
Inputs	The input data type is selected by the user. Input (IN1, IN2): INT, DINT
Outputs	Output (OUT): INT, DINT

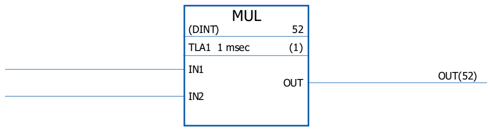
MOVE

(10005)

Illustration	
Execution time	2.10 µs (when two inputs are used) + 0.42 µs (for every additional input). When all inputs are used, the execution time is 14.55 µs.
Operation	Copies the input values (IN1…32) to the corresponding outputs (OUT1…32).
Inputs	The input data type and number of inputs (2…32) are selected by the user. Input (IN1…IN32): INT, DINT, REAL, REAL24, Boolean
Outputs	Output (OUT1…OUT32): INT, DINT, REAL, REAL24, Boolean

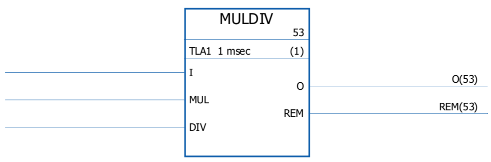
MUL

(10006)

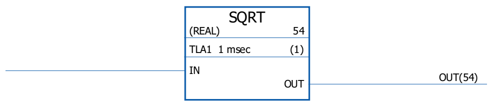
Illustration	
Execution time	3.47 µs (when two inputs are used) + 2.28 µs (for every additional input). When all inputs are used, the execution time is 71.73 µs.
Operation	The output (OUT) is the product of the inputs (IN). O = IN1 × IN2 × … × IN32 The output value is limited to the maximum and minimum values defined by the selected data type range.
Inputs	The input data type and the number of inputs (2…32) are selected by the user. Input (IN1…IN32): INT, DINT, REAL, REAL24
Outputs	Output (OUT): INT, DINT, REAL, REAL24

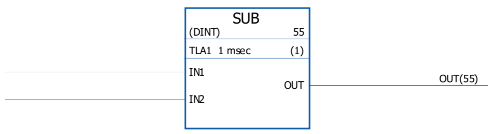
MULDIV

(10007)

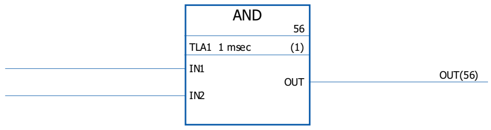
Illustration	
Execution time	7.10 µs

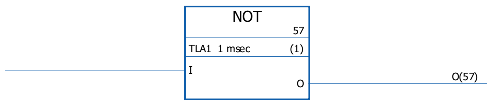
Operation	The output (O) is the product of input IN and input MUL divided by input DIV. Output = (I × MUL) / DIV O = whole value. REM = remainder value. Example: I = 2, MUL = 16 and DIV = 10: (2 × 16) / 10 = 3.2, i.e. O = 3 and REM = 2 The output value is limited to the maximum and minimum values defined by the data type range.
Inputs	Input (I): DINT Multiplier input (MUL): DINT Divider input (DIV): DINT
Outputs	Output (O): DINT Remainder output (REM): DINT

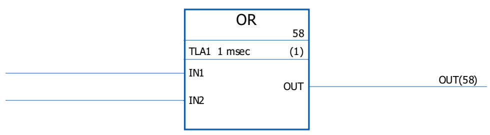
SQRT (10008)	
Illustration	
Execution time	2.09 µs
Operation	Output (OUT) is the square root of the input (IN). OUT = sqrt(IN) Output is 0 if the input value is negative
Inputs	The input data type is selected by the user. Input (IN): REAL, REAL24
Outputs	Output (OUT): REAL, REAL24

SUB (10009)	
Illustration	
Execution time	2.33 µs
Operation	Output (OUT) is the difference between the input signals (IN): OUT = IN1 - IN2 The output value is limited to the maximum and minimum values defined by the selected data type range.
Inputs	The input data type is selected by the user. Input (IN1, IN2): INT, DINT, REAL, REAL24
Outputs	Output (OUT): INT, DINT, REAL, REAL24

Bitstring

<table>
<tr><th colspan="2">AND
(10010)</th></tr>
<tr><td>Illustration</td><td>
</td></tr>
<tr><td>Execution time</td><td>1.55 µs (when two inputs are used) + 0.60 µs (for every additional input). When all inputs are used, the execution time is 19.55 µs.</td></tr>
<tr><td>Operation</td><td>The output (OUT) is 1 if all the connected inputs (IN1…IN32) are 1. Otherwise the output is 0.
Truth table:
<table>
<tr><th>IN1</th><th>IN2</th><th>OUT</th></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>1</td></tr>
</table>
The inputs can be inverted.</td></tr>
<tr><td>Inputs</td><td>The number of inputs is selected by the user.
Input (IN1…IN32): Boolean</td></tr>
<tr><td>Outputs</td><td>Output (OUT): Boolean</td></tr>
</table>

<table>
<tr><th colspan="2">NOT
(10011)</th></tr>
<tr><td>Illustration</td><td>
</td></tr>
<tr><td>Execution time</td><td>0.32 µs</td></tr>
<tr><td>Operation</td><td>The output (O) is 1 if the input (I) is 0. The output is 0 if the input is 1.</td></tr>
<tr><td>Inputs</td><td>Input (I): Boolean</td></tr>
<tr><td>Outputs</td><td>Output (O): Boolean</td></tr>
</table>

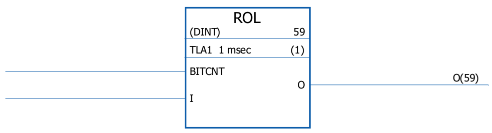
<table>
<tr><th colspan="2">OR
(10012)</th></tr>
<tr><td>Illustration</td><td>
</td></tr>
<tr><td>Execution time</td><td>1.55 µs (when two inputs are used) + 0.60 µs (for every additional input). When all inputs are used, the execution time is 19.55 µs.</td></tr>
</table>

Operation	The output (OUT) is 0, if all connected inputs (IN) are 0. Otherwise the output is 1. Truth table: **IN1 / IN2 / OUT** 0 / 0 / 0 0 / 1 / 1 1 / 0 / 1 1 / 1 / 1 The inputs can be inverted.
Inputs	The number of inputs (2…32) is selected by the user. Input (IN1…IN32): Boolean
Outputs	Output (OUT): Boolean

ROL

(10013)

Illustration



Execution time: 1.28 µs

Operation: Input bits (I) are rotated to the left by the number (N) of bits defined by BITCNT. The N most significant bits (MSB) of the input are stored as the N least significant bits (LSB) of the output.

Example: If BITCNT = 3

3 MSB

I	1 1 1 0 0 0 0 0 1 1 1 0 0 1 0 1 1 1 0 1 0 0 1 1 0 0 1 1 0 1 0 1	
O	0 0 0 0 0 1 1 1 0 0 1 0 1 1 1 0 1 0 0 1 1 0 0 1 1 0 1 0 1	1 1 1

3 LSB

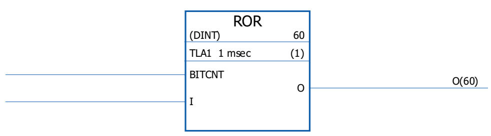
Inputs: The input data type is selected by the user.
Number of bits input (BITCNT): INT, DINT
Input (I): INT, DINT

Outputs: Output (O): INT, DINT

ROR

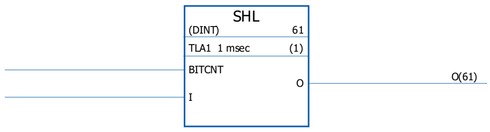
(10014)

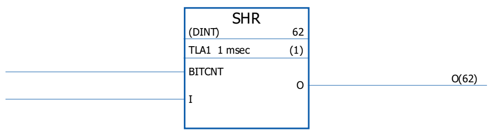
Illustration



Execution time: 1.28 µs

Operation	Input bits (I) are rotated to the right by the number (N) of bits defined by BITCNT. The N least significant bits (LSB) of the input are stored as the N most significant bits (MSB) of the output. Example: If BITCNT = 3 3 LSB **I** 1 1 1 0 0 0 0 0 1 1 1 0 0 1 0 1 1 1 0 1 0 0 1 1 0 0 1 1 0 1 0 1 **O** 1 0 1 1 1 1 0 0 0 0 0 1 1 1 0 0 1 0 1 1 1 0 1 0 0 1 1 0 0 1 1 0 3 MSB
Inputs	The input data type is selected by the user. Number of bits input (BITCNT): INT, DINT Input (I): INT, DINT
Outputs	Output (O): INT, DINT

SHL (10015)	
Illustration	
Execution time	0.80 µs
Operation	Input bits (I) are rotated to the left by the number (N) of bits defined by BITCNT. The N most significant bits (MSB) of the input are lost and the N least significant bits (LSB) of the output are set to 0. Example: If BITCNT = 3 3 MSB **I** 1 1 1 0 0 0 0 0 1 1 1 0 0 1 0 1 1 1 0 1 0 0 1 1 0 0 1 1 0 1 0 1 **O** 0 0 0 0 0 1 1 1 0 0 1 0 1 1 1 0 1 0 0 1 1 0 0 1 1 0 1 0 1 0 0 0 3 LSB
Inputs	The input data type is selected by the user. Number of bits (BITCNT): INT; DINT Input (I): INT, DINT
Outputs	Output (O): INT; DINT

SHR (10016)	
Illustration	
Execution time	0.80 µs

Operation

Input bits (I) are rotated to the right by the number (N) of bits defined by BITCNT. The N least significant bits (LSB) of the input are lost and the N most significant bits (MSB) of the output are set to 0.

Example: If BITCNT = 3

3 LSB

I	1 1 1 0 0 0 0 0 1 1 1 0 0 1 0 1 1 1 0 1 0 0 1 1 0 0 1 1 0 1 0 1
O	0 0 0 1 1 1 0 0 0 0 0 1 1 1 0 0 1 0 1 1 1 0 1 0 0 1 1 0 0 1 1 0

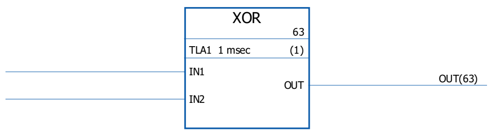
3 MSB

Inputs	The input data type is selected by the user. Number of bits (BITCNT): INT; DINT Input (I): INT, DINT
Outputs	Output (O): INT; DINT

XOR

(10017)

Illustration



Execution time

1.24 µs (when two inputs are used) + 0.72 µs (for every additional input). When all inputs are used, the execution time is 22.85 µs.

Operation

The output (OUT) is 1 if one of the connected inputs (IN1…IN32) is 1. Output is zero if all the inputs have the same value.

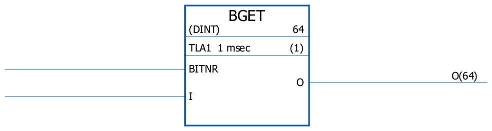
Example:

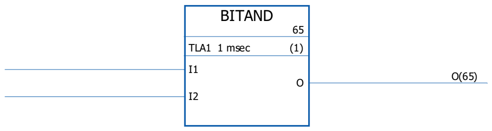
IN1	IN2	OUT
0	0	0
0	1	1
1	0	1
1	1	0

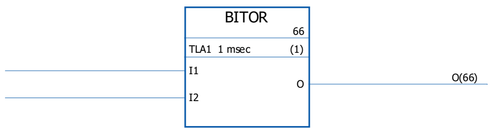
The inputs can be inverted.

Inputs	The number of inputs (2…32) is selected by the user. Input (IN1…IN32): Boolean
Outputs	Output (OUT): Boolean

Bitwise

BGET (10034)	
Illustration	
Execution time	0.88 µs
Operation	The output (O) is the value of the selected bit (BITNR) of the input (I). BITNR: Bit number (0 = bit number 0, 31 = bit number 31) If bit number is not in the range of 0…31 (for DINT) or 0…15 (for INT), the output is 0.
Inputs	The input data type is selected by the user. Number of the bit (BITNR): DINT Input (I): DINT, INT
Outputs	Output (O): Boolean

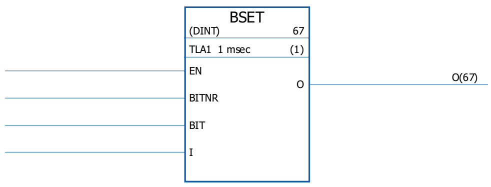
<table>
<tr><th colspan="2">BITAND
(10035)</th></tr>
<tr><td>Illustration</td><td></td></tr>
<tr><td>Execution time</td><td>0.32 µs</td></tr>
<tr><td>Operation</td><td>The output (O) bit value is 1 if the corresponding bit values of the inputs (I1 and I2) are 1. Otherwise the output bit value is 0.
Example:
<table>
<tr><td>I1</td><td>1 1 1 0 0 0 0 0 1 1 1 0 0 1 0 1 1 1 0 1 0 0 1 1 0 0 1 1 0 1 0 1</td></tr>
<tr><td>I2</td><td>0 0 0 0 0 1 1 1 0 0 1 0 1 1 1 0 1 0 0 1 1 0 0 1 1 0 1 0 1 1 1 1</td></tr>
<tr><td>O</td><td>0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 1 0 1</td></tr>
</table></td></tr>
<tr><td>Inputs</td><td>Input (I1, I2): DINT</td></tr>
<tr><td>Outputs</td><td>Output (O): DINT</td></tr>
</table>

BITOR (10036)	
Illustration	
Execution time	0.32 µs

<table>
<tr><td>Operation</td><td>The output (O) bit value is 1 if the corresponding bit value of any of the inputs (I1 or I2) is 1. Otherwise the output bit value is 0.
Example:
<table>
<tr><td>I1</td><td>1 1 1 0 0 0 0 0 1 1 1 0 0 1 0 1 1 1 0 1 0 0 1 1 0 0 1 1 0 1 0 1</td></tr>
<tr><td>I2</td><td>0 0 0 0 0 1 1 1 0 0 1 0 1 1 1 0 1 0 0 1 1 0 0 1 1 0 1 0 1 1 1 1</td></tr>
<tr><td>O</td><td>1 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 0 1 1 1 0 1 1 1 1 1 1</td></tr>
</table>
</td></tr>
<tr><td>Inputs</td><td>Input (I1, I2): DINT</td></tr>
<tr><td>Outputs</td><td>Output (O): DINT</td></tr>
</table>

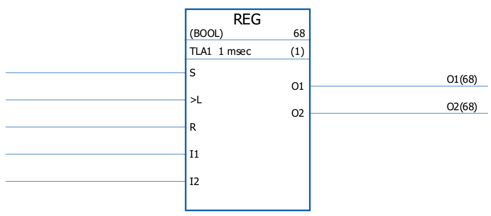
BSET

(10037)

Illustration	
Execution time	1.36 µs
Operation	The value of a selected bit (BITNR) of the input (I) is set as defined by the bit value input (BIT). The function must be enabled by the enable input (EN). BITNR: Bit number (0 = bit number 0, 31 = bit number 31) If BITNR is not in the range of 0…31 (for DINT) or 0…15 (for INT) or if EN is reset to zero, the input value is stored to the output as it is (i.e. no bit setting occurs). Example: EN = 1, BITNR = 3, BIT = 0 IN = 0000 0000 1111 1111 O = 0000 0000 1111 0111
Inputs	The input data type is selected by the user. Enable input (EN): Boolean Number of the bit (BITNR): DINT Bit value input (BIT): Boolean Input (I): INT, DINT
Outputs	Output (O): INT, DINT

REG

(10038)

Illustration	

Execution time	2.27 µs (when two inputs are used) + 1.02 µs (for every additional input). When all inputs are used, the execution time is 32.87 µs.
Operation	The input (I1…I32) value is stored to the corresponding output (O1…O32) if the load input (L) is set to 1 or the set input (S) is 1. When the load input is set to 1, the input value is stored to the output only once. When the set input is 1, the input value is stored to the output every time the block is executed. The set input overrides the load input. If the reset input (R) is 1, all connected outputs are 0. Example:

S	**R**	**L**	**I**	**$O1_{previous}$**	**O1**
0	0	0	10	15	15
0	0	0->1	20	15	20
0	1	0	30	20	0
0	1	0->1	40	0	0
1	0	0	50	0	50
1	0	0->1	60	50	60
1	1	0	70	60	0
1	1	0->1	80	0	0

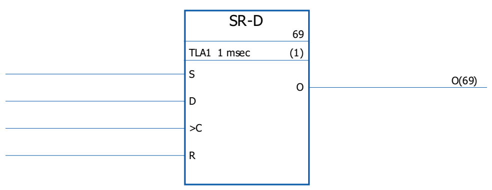
$O1_{previous}$ is the previous cycle output value.

Inputs	The input data type and number of inputs (1…32) are selected by the user. Set input (S): Boolean Load input (L): Boolean Reset input (R): Boolean Input (I1…I32): Boolean, INT, DINT, REAL, REAL24
Outputs	Output (O1…O32): Boolean, INT, DINT, REAL, REAL24

SR-D

(10039)

Illustration



Execution time	1.04 µs

<table>
<tr><td>Operation</td><td>
When clock input (C) is set to 1, the data input (D) value is stored to the output (O). When reset input (R) is set to 1, the output is set to 0.
If only set (S) and reset (R) inputs are used, SR-D block acts as an SR block:
The output is 1 if the set input (S) is 1. The output will retain the previous output state if the set input (S) and reset input (R) are 0. The output is 0 if the set input is 0 and the reset input is 1.
Truth table:
<table>
<tr><th>S</th><th>R</th><th>D</th><th>C</th><th>$O_{previous}$</th><th>O</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0 (= Previous output value)</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0 -> 1</td><td>0</td><td>0 (= Data input value)</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0 (= Previous output value)</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0 -> 1</td><td>0</td><td>1 (= Data input value)</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>0 (Reset)</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0 -> 1</td><td>0</td><td>0 (Reset)</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0 (Reset)</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0 -> 1</td><td>0</td><td>0 (Reset)</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1 (= Set value)</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>0 -> 1</td><td>1</td><td>0 (= Data input value) for one execution cycle, then changes to 1 according to the set input (S = 1).</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>1 (= Set value)</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>0 -> 1</td><td>1</td><td>1 (= Data input value)</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>0</td><td>1</td><td>0 (Reset)</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>0 -> 1</td><td>0</td><td>0 (Reset)</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0 (Reset)</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>0 -> 1</td><td>0</td><td>0 (Reset)</td></tr>
<tr><td colspan="6">$O_{previous}$ is the previous cycle output value.</td></tr>
</table>
</td></tr>
<tr><td>Inputs</td><td>Set input (S): Boolean
Data input (D): Boolean
Clock input (C): Boolean
Reset input (R): Boolean</td></tr>
<tr><td>Outputs</td><td>Output (O): Boolean</td></tr>
</table>

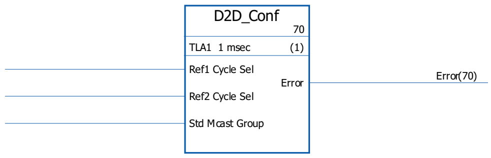
Communication (ACS850 only)

See also section *Drive-to-drive communication (ACS850 only)* on page 95.

D2D_Conf

(10092)

Illustration



Execution time

-

Operation

Defines handling interval for drive-to-drive references 1 and 2, and the address (group number) for standard (non-chained) multicast messages.

The values of the Ref1/2 Cycle Sel inputs correspond to the following intervals:

Value	Handling interval
0	Default (500 µs for reference 1; 2 ms for reference 2)
1	250 µs
2	500 µs
3	2 ms

Note: Negative value of Ref2 Cycle Sel disables the handling of Ref2 (if disabled in the master, it must be disabled in all follower drives as well).

Allowable values for the Std Mcast Group input are 0 (= multicasting not used) and 1…62 (multicast group).

An unconnected input, or an input in an error state, is interpreted as having the value 0.

The error codes indicated by the Error output are as follows:

Bit	Description
0	REF1_CYCLE_ERR: Value of input Ref1 Cycle Sel out of range
1	REF2_CYCLE_ERR: Value of input Ref2 Cycle Sel out of range
2	STD_MCAST_ERR: Value of input Std Mcast Group out of range

Inputs

Drive-to-drive reference 1 handling interval (Ref1 Cycle Sel): INT

Drive-to-drive reference 2 handling interval (Ref2 Cycle Sel): INT

Standard multicast address (Std Mcast Group): INT

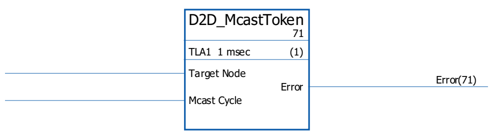
Outputs

Error output (Error): PB

D2D_McastToken

(10096)

Illustration



Execution time

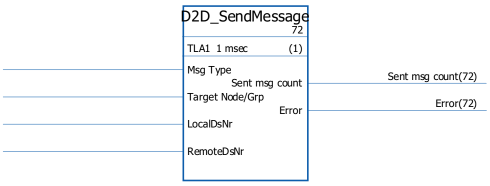
-

Operation	Configures the transmission of token messages sent to a follower. Each token authorizes the follower to send one message to another follower or group of followers. For the message types, see block *D2D_SendMessage*. **Note:** This block is only supported in the master. The Target Node input defines the node address the master sends the tokens to; the range is 1…62. The Mcast Cycle specifies the interval between token messages in the range of 2…1000 milliseconds. Setting this input to 0 disables the sending of tokens. The error codes indicated by the Error output are as follows:
Inputs	Token recipient (Target Node): INT Token interval (Mcast Cycle): INT
Outputs	Error output (Error): DINT

Bit	Description
0	D2D_MODE_ERR: Drive is not master
5	TOO_SHORT_CYCLE: Token interval is too short, causing overloading
6	INVALID_INPUT_VAL: An input value is out of range
7	GENERAL_D2D_ERR: Drive-to-drive communication driver failed to initialize message

D2D_SendMessage

(10095)

Illustration	
Execution time	-

Operation

Configures the transmission between the dataset tables of drives.

The Msg Type input defines the message type as follows:

Value	Message type
0	Disabled
1	Master P2P: The master sends the contents of a local dataset (specified by LocalDsNr input) to the dataset table (dataset number specified by RemoteDsNr input) of a follower (specified by Target Node/Grp input). The follower replies by sending the next dataset (RemoteDsNr + 1) to the master (LocalDsNr + 1). The node number of a drive is defined by parameter *57.03*. **Note:** Only supported in the master drive.
2	Read Remote: The master reads a dataset (specified by RemoteDsNr input) from a follower (specified by Target Node/Grp input) and stores it into local dataset table (dataset number specified by LocalDsNr input). The node number of a drive is defined by parameter *57.03*. **Note:** Only supported in the master drive.
3	Follower P2P: The follower sends the contents of a local dataset (specified by LocalDsNr input) to the dataset table (dataset number specified by RemoteDsNr input) of another follower (specified by Target Node/Grp input). The node number of a drive is defined by parameter *57.03*. **Note:** Only supported in a follower drive. A token from the master drive is required for the follower to be able to send the message. See block *D2D_McastToken*.
4	Standard Multicast: The drive sends the contents of a local dataset (specified by LocalDsNr input) to the dataset table (dataset number specified by RemoteDsNr input) of a group of followers (specified by Target Node/Grp input). Which multicast group a drive belongs to is defined by the Std Mcast Group input of the *D2D_Conf* block. A token from the master drive is required for a follower to be able to send the message. See the block *D2D_McastToken*.
5	Broadcast: The drive sends the contents of a local dataset (specified by LocalDsNr input) to the dataset table (dataset number specified by RemoteDsNr input) of all followers. A token from the master drive is required for a follower to be able to send the message. See block *D2D_McastToken*. **Note:** With this message type, the Target Node/Grp input must be connected in DriveSPC even if not used.

The Target Node/Grp input specifies the target drive or multicast group of drives depending on message type. See the message type explanations above.

Note: The input must be connected in DriveSPC even if not used.

The LocalDsNr input specifies the number of the local dataset used as the source or the target of the message.

The RemoteDsNr input specifies the number of the remote dataset used as the target or the source of the message.

The Sent msg count output is a wrap-around counter of successfully sent messages.

The error codes indicated by the Error output are as follows:

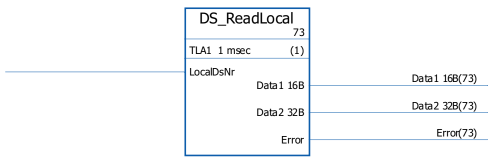
Bit	Description
0	D2D_MODE_ERR: Drive-to-drive communication not activated, or message type not supported in current drive-to-drive mode (master/follower)
1	LOCAL_DS_ERR: LocalDsNr input out of range (16…199)
2	TARGET_NODE_ERR: Target Node/Grp input out of range (1…62)
3	REMOTE_DS_ERR: Remote dataset number out of range (16…199)
4	MSG_TYPE_ERR: Msg Type input out of range (0…5)
5…6	Reserved
7	GENERAL_D2D_ERR: Unspecified error in D2D driver
8	RESPONSE_ERR: Syntax error in received response
9	TRA_PENDING: Message has not yet been sent
10	REC_PENDING: Response has not yet been received
11	REC_TIMEOUT: No response received
12	REC_ERROR: Frame error in received message
13	REJECTED: Message has been removed from transmit buffer
14	BUFFER_FULL: Transmit buffer full

Inputs	Message type (Msg Type): INT Target node or multicast group (Target Node/Grp): INT Local dataset number (LocalDsNr): INT Remote dataset number (RemoteDsNr): INT
Outputs	Successfully sent messages counter (Sent msg count): DINT Error output (Error): PB

DS_Read_Local

(10094)

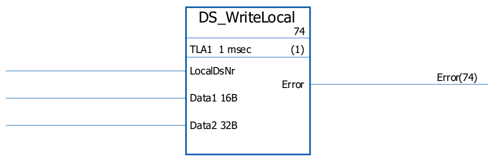
Illustration



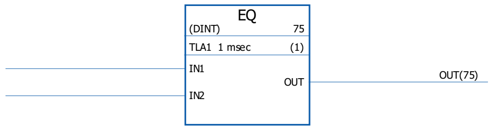
Execution time	-
Operation	Reads the dataset defined by the LocalDsNr input from the local dataset table. One dataset contains one 16-bit and one 32-bit word which are directed to the Data1 16B and Data2 32B outputs respectively. The LocalDsNr input defines the number of the dataset to be read. The error codes indicated by the Error output are as follows:

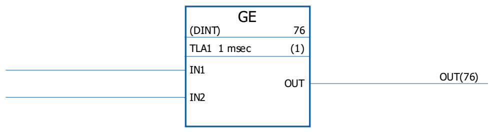
Bit	Description
1	LOCAL_DS_ERR: LocalDsNr out of range (16…199)

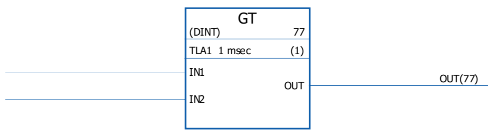
Inputs	Local dataset number (LocalDsNr): INT
Outputs	Contents of dataset (Data1 16B): INT Contents of dataset (Data2 32B): DINT Error output (Error): DINT

<table>
<tr><th colspan="2">DS_WriteLocal
(10093)</th></tr>
<tr><td>Illustration</td><td>
</td></tr>
<tr><td>Execution time</td><td>-</td></tr>
<tr><td>Operation</td><td>Writes data into the local dataset table. Each dataset contains 48 bits; the data is input through the Data1 16B (16 bits) and Data2 32B (32 bits) inputs. The dataset number is defined by the LocalDsNr input.
The error codes indicated by the Error output are as follows:
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>1</td><td>LOCAL_DS_ERR: LocalDsNr out of range (16…199)</td></tr>
</table></td></tr>
<tr><td>Inputs</td><td>Local dataset number (LocalDsNr): INT
Contents of dataset (Data1 16B): INT
Contents of dataset (Data2 32B): DINT</td></tr>
<tr><td>Outputs</td><td>Error output (Error): DINT</td></tr>
</table>

Comparison

EQ (10040)	
Illustration	
Execution time	0.89 μs (when two inputs are used) + 0.43 μs (for every additional input). When all inputs are used, the execution time is 13.87 μs.
Operation	The output (OUT) is 1 if all the connected input values are equal (IN1 = IN2 = … = IN32). Otherwise the output is 0.
Inputs	The input data type and the number of inputs (2…32) are selected by the user. Input (IN1…IN32): INT, DINT, REAL, REAL24
Outputs	Output (OUT): Boolean

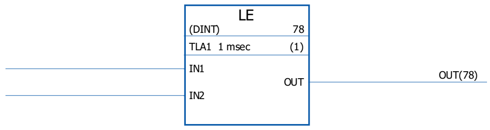
GE >= (10041)	
Illustration	
Execution time	0.89 μs (when two inputs are used) + 0.43 μs (for every additional input). When all inputs are used, the execution time is 13.87 μs.
Operation	The output (OUT) is 1 if (IN1 > IN2) & (IN2 > IN3) & … & (IN31 > IN32). Otherwise the output is 0.
Inputs	The input data type and the number of inputs (2…32) are selected by the user. Input (IN1…IN32): INT, DINT, REAL, REAL24
Outputs	Output (OUT): Boolean

GT > (10042)	
Illustration	
Execution time	0.89 μs (when two inputs are used) + 0.43 μs (for every additional input). When all inputs are used, the execution time is 13.87 μs.
Operation	The output (OUT) is 1 if (IN1 > IN2) & (IN2 > IN3) & … & (IN31 > IN32). Otherwise the output is 0.
Inputs	The input data type and the number of inputs (2…32) are selected by the user. Input (IN1…IN32): INT, DINT, REAL, REAL24

Outputs	Output (OUT): Boolean

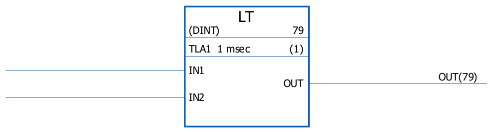
LE <=

(10043)

Illustration	
Execution time	0.89 µs (when two inputs are used) + 0.43 µs (for every additional input). When all inputs are used, the execution time is 13.87 µs.
Operation	Output (OUT) is 1 if (IN1 ≤ IN2) & (IN2 ≤ IN3) & … & (IN31 ≤ IN32). Otherwise the output is 0.
Inputs	The input data type and the number of inputs (2…32) are selected by the user. Input (IN1…IN32): INT, DINT, REAL, REAL24
Outputs	Output (OUT): Boolean

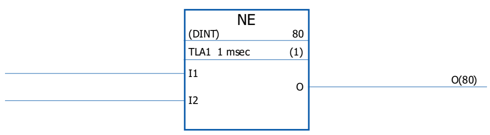
LT <

(10044)

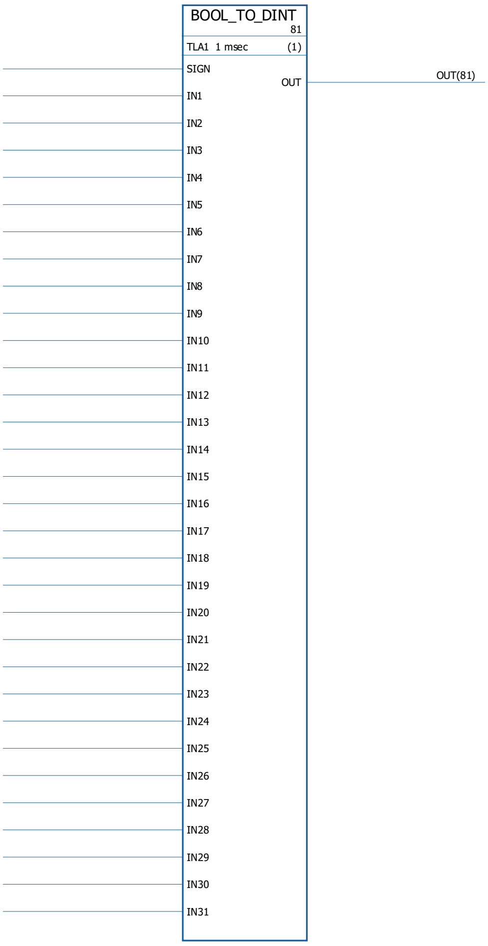
Illustration	
Execution time	0.89 µs (when two inputs are used) + 0.43 µs (for every additional input). When all inputs are used, the execution time is 13.87 µs.
Operation	Output (OUT) is 1 if (IN1 < IN2) & (IN2 < IN3) & … & (IN31 < IN32). Otherwise the output is 0.
Inputs	The input data type and the number of inputs (2…32) are selected by the user. Input (IN1…IN32): INT, DINT, REAL, REAL24
Outputs	Output (OUT): Boolean

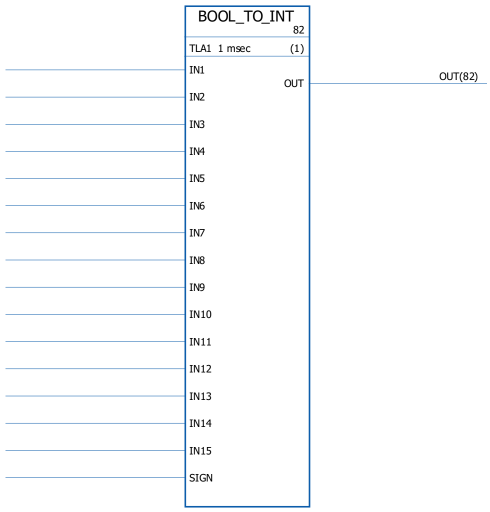
NE <>

(10045)

Illustration	
Execution time	0.44 µs
Operation	The output (O) is 1 if I1 <> I2. Otherwise the output is 0.
Inputs	The input data type is selected by the user. Input (I1, I2): INT, DINT, REAL, REAL24
Outputs	Output (O): Boolean

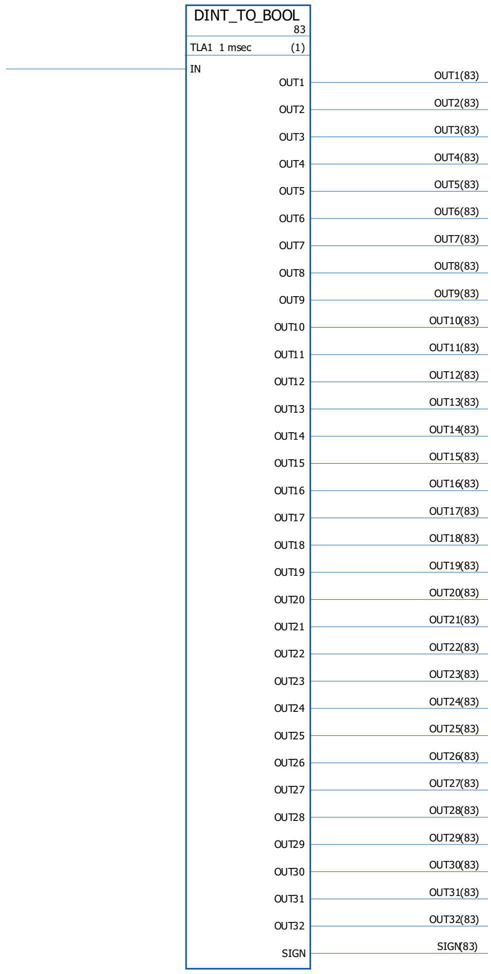
Conversion

BOOL_TO_DINT (10018)	
Illustration	
Execution time	13.47 µs
Operation	The output (OUT) value is a 32-bit integer value formed from the boolean input (IN1…IN31 and SIGN) values. IN1 = bit 0 and IN31 = bit 30. Example: IN1 = 1, IN2 = 0, IN3…IN31 = 1, SIGN = 1 OUT = 1111 1111 1111 1111 1111 1111 1111 1101 SIGN IN31,Ä¶I
Inputs	Sign input (SIGN): Boolean Input (IN1…IN31): Boolean
Outputs	Output (OUT): DINT (31 bits + sign)

BOOL_TO_INT (10019)	
Illustration	
Execution time	5.00 µs
Operation	The output (OUT) value is a 16-bit integer value formed from the boolean input (IN1…IN15 and SIGN) values. IN1 = bit 0 and IN15 = bit 14. Example: IN1…IN15 = 1, SIGN = 0 OUT = 0111 1111 1111 1111 SIGN IN15,Ä¶I
Inputs	Input (IN1…IN15): Boolean Sign input (SIGN): Boolean
Outputs	Output (OUT): DINT (15 bits + sign)

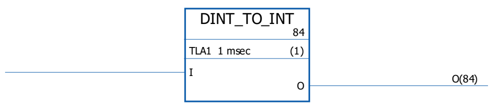
DINT_TO_BOOL

(10020)

Illustration	
Execution time	11.98 µs
Operation	The boolean output (OUT1…OUT32) values are formed from the 32-bit integer input (IN) value. Example: IN = $\underbrace{0}_{\text{SIGN}}\ \underbrace{111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1100}_{\text{OUT32,Ä¶OU}}$
Inputs	Input (IN): DINT
Outputs	Output (OUT1…OUT32): Boolean Sign output (SIGN): Boolean

DINT_TO_INT

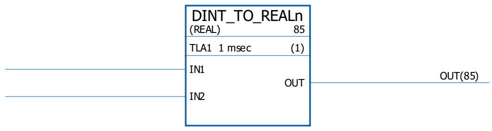
(10021)

Illustration	

<table>
<tr><td>Execution time</td><td>0.53 µs</td></tr>
<tr><td>Operation</td><td>The output (O) value is a 16-bit integer value of the 32-bit integer input (I) value.
Examples:
<table>
<tr><td>I (31 bits + sign)</td><td>O (15 bits + sign)</td></tr>
<tr><td>2147483647</td><td>32767</td></tr>
<tr><td>-2147483648</td><td>-32767</td></tr>
<tr><td>0</td><td>0</td></tr>
</table></td></tr>
<tr><td>Inputs</td><td>Input (I): DINT</td></tr>
<tr><td>Outputs</td><td>Output (O): INT</td></tr>
</table>

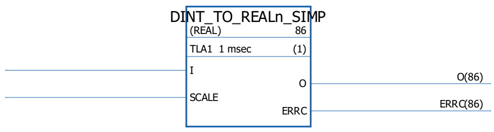
DINT_TO_REALn

(10023)

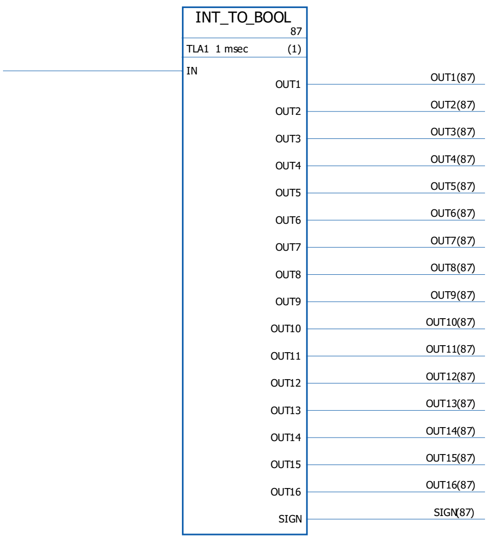
Illustration	
Execution time	7.25 µs
Operation	The output (OUT) is the REAL/REAL24 equivalent of the input (IN). Input IN1 is the integer value and input IN2 is the fractional value. If one (or both) of the input values is negative, the output value is negative. Example (from DINT to REAL): When IN1 = 2 and IN2 = 3276, OUT = 2.04999. The output value is limited to the maximum value of the selected data type range.
Inputs	Input (IN1, IN2): DINT
Outputs	The output data type is selected by the user. Output (OUT): REAL, REAL24

DINT_TO_REALn_SIMP

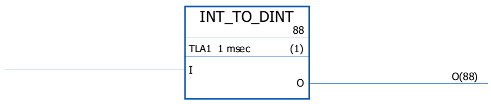
(10022)

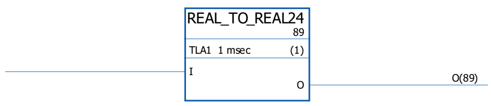
Illustration	
Execution time	6.53 µs

<table>
<tr><td>Operation</td><td>The output (O) is the REAL/REAL24 equivalent of the input (I) divided by the scale input (SCALE).
Error codes indicated at the error output (ERRC) are as follows:
<table>
<tr><th>Error code</th><th>Description</th></tr>
<tr><td>0</td><td>No error</td></tr>
<tr><td>1001</td><td>The calculated REAL/REAL24 value exceeds the minimum value of the selected data type range. The output is set to the minimum value.</td></tr>
<tr><td>1002</td><td>The calculated REAL/REAL24 value exceeds the maximum value of the selected data type range. The output is set to the maximum value.</td></tr>
<tr><td>1003</td><td>The SCALE input is 0. The output is set to 0.</td></tr>
<tr><td>1004</td><td>Incorrect SCALE input, i.e. the scale input is < 0 or is not a factor of 10.</td></tr>
</table>
Example (from DINT to REAL24):
When I = 205 and SCALE = 100, I/SCALE = 205 /100 = 2.05 and O = 2.04999.</td></tr>
<tr><td>Inputs</td><td>Input (I): DINT
Scale input (SCALE): DINT</td></tr>
<tr><td>Outputs</td><td>The output data type is selected by the user.
Output (O): REAL, REAL24
Error output (ERRC): DINT</td></tr>
</table>

<table>
<tr><th colspan="2">INT_TO_BOOL
(10024)</th></tr>
<tr><td>Illustration</td><td>
</td></tr>
<tr><td>Execution time</td><td>4.31 µs</td></tr>
<tr><td>Operation</td><td>The boolean output (OUT1…OUT16) values are formed from the 16-bit integer input (IN) value.
Example:
IN = 0111 1111 1111 1111
SIGN OUT16,Ä¶OU</td></tr>
</table>

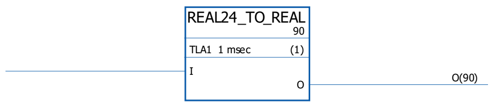
Inputs	Input (IN): INT
Outputs	Output (OUT1…OUT16): Boolean Sign output (SIGN): Boolean

<table>
<tr><th colspan="2">INT_TO_DINT
(10025)</th></tr>
<tr><td>Illustration</td><td>
</td></tr>
<tr><td>Execution time</td><td>0.33 µs</td></tr>
<tr><td>Operation</td><td>The output (O) value is a 32-bit integer value of the 16-bit integer input (I) value.
<table>
<tr><th>I</th><th>O</th></tr>
<tr><td>32767</td><td>32767</td></tr>
<tr><td>-32767</td><td>-32767</td></tr>
<tr><td>0</td><td>0</td></tr>
</table></td></tr>
<tr><td>Inputs</td><td>Input (I): INT</td></tr>
<tr><td>Outputs</td><td>Output (O): DINT</td></tr>
</table>

<table>
<tr><th colspan="2">REAL_TO_REAL24
(10026)</th></tr>
<tr><td>Illustration</td><td>
</td></tr>
<tr><td>Execution time</td><td>1.35 µs</td></tr>
<tr><td>Operation</td><td>Output (O) is the REAL24 equivalent of the REAL input (I).
The output value is limited to the maximum value of the data type.
Example:
$I = \underbrace{0000\ 0000\ 0010\ 0110}_{\text{Integer value}}\ \underbrace{1111\ 1111\ 1111\ 1111}_{\text{Fractional value}}$
$O = \underbrace{0010\ 0110}_{\text{Integer value}}\ \underbrace{1111\ 1111\ 1111\ 1111\ 0000\ 0000}_{\text{Fractional value}}$</td></tr>
<tr><td>Inputs</td><td>Input (I): REAL</td></tr>
<tr><td>Outputs</td><td>Output (O): REAL24</td></tr>
</table>

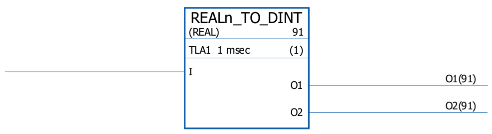
REAL24_TO_REAL

(10027)

Illustration	
Execution time	1.20 µs
Operation	Output (O) is the REAL equivalent of the REAL24 input (I). The output value is limited to the maximum value of the data type range. Example: $I = \underbrace{0010\ 0110}_{\text{Integer value}}\ \underbrace{1111\ 1111\ 1111\ 1111\ 0000\ 0000}_{\text{Fractional value}}$ $O = \underbrace{0000\ 0000\ 0010\ 0110}_{\text{Integer value}}\ \underbrace{1111\ 1111\ 1111\ 1111}_{\text{Fractional value}}$
Inputs	Input (I): REAL24
Outputs	Output (O): REAL

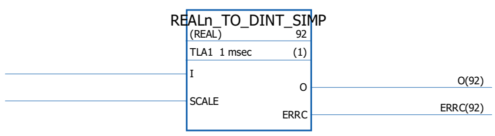
REALn_TO_DINT

(10029)

Illustration	
Execution time	6.45 µs
Operation	Output (O) is the 32-bit integer equivalent of the REAL/REAL24 input (I). Output O1 is the integer value and output O2 is the fractional value. The output value is limited to the maximum value of the data type range. Example (from REAL to DINT): When I = 2.04998779297, O1 = 2 and O2 = 3276.
Inputs	The input data type is selected by the user. Input (I): REAL, REAL24
Outputs	Output (O1, O2): DINT

REALn_TO_DINT_SIMP

(10028)

Illustration	
Execution time	5.54 µs

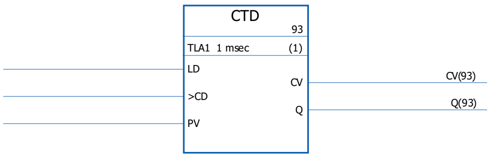
<table>
<tr><td>Operation</td><td>Output (O) is the 32-bit integer equivalent of the REAL/REAL24 input (I) multiplied by the scale input (SCALE).
Error codes are indicated by the error output (ERRC) as follows:
<table>
<tr><th>Error code</th><th>Description</th></tr>
<tr><td>0</td><td>No error</td></tr>
<tr><td>1001</td><td>The calculated integer value exceeds the minimum value. The output is set to the minimum value.</td></tr>
<tr><td>1002</td><td>The calculated integer value exceeds the maximum value. The output is set to the maximum value.</td></tr>
<tr><td>1003</td><td>Scale input is 0. The output is set to 0.</td></tr>
<tr><td>1004</td><td>Incorrect scale input, i.e. scale input is < 0 or is not a factor of 10.</td></tr>
</table>
Example (from REAL to DINT):
When I = 2.04998779297and SCALE = 100, O = 204.</td></tr>
<tr><td>Inputs</td><td>The input data type is selected by the user.
Input (I): REAL, REAL24
Scale input (SCALE): DINT</td></tr>
<tr><td>Outputs</td><td>Output (O): DINT
Error output (ERRC): DINT</td></tr>
</table>

Counters

CTD

(10047)

Illustration



Execution time

0.92 µs

Operation

The counter output (CV) value is decreased by 1 if the counter input (CD) value changes from 0 -> 1 and the load input (LD) value is 0. If the load input value is 1, the preset input (PV) value is stored as the counter output (CV) value. If the counter output has reached its minimum value -32768, the counter output remains unchanged.

The status output (Q) is 1 if the counter output (CV) value $\leq$ 0.

Example:

LD	CD	PV	Q	CV_{prev}	CV
0	1 -> 0	10	0	5	5
0	0 -> 1	10	0	5	5 - 1 = 4
1	1 -> 0	-2	1	4	-2
1	0 -> 1	1	0	-2	1
0	0 -> 1	5	1	1	1 -1 = 0
1	1 -> 0	-32768	1	0	-32768
0	0 -> 1	10	1	-32768	-32768

CV_{prev} is the previous cycle counter output value.

Inputs

Load input (LD): Boolean

Counter input (CD): Boolean

Preset input (PV): INT

Outputs

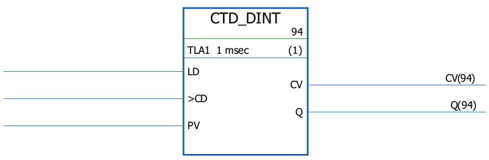
Counter output (CV): INT

Status output (Q): Boolean

CTD_DINT

(10046)

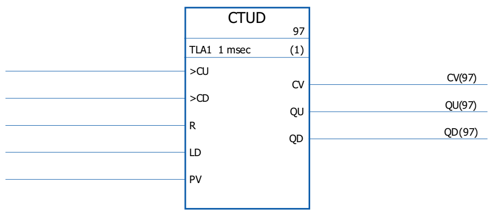
Illustration



Execution time

0.92 µs

<table>
<tr><td>Operation</td><td>
The counter output (CV) value is decreased by 1 if the counter input (CD) value changes from 0 -> 1 and the load input (LD) value is 0. If the load input (LD) value is 1, the preset input (PV) value is stored as the counter output (CV) value. If the counter output has reached its minimum value -2147483648, the counter output remains unchanged.
The status output (Q) is 1 if the counter output (CV) value < 0.
Example:
<table>
<tr><th>LD</th><th>CD</th><th>PV</th><th>Q</th><th>CV_{prev}</th><th>CV</th></tr>
<tr><td>0</td><td>1 -> 0</td><td>10</td><td>0</td><td>5</td><td>5</td></tr>
<tr><td>0</td><td>0 -> 1</td><td>10</td><td>0</td><td>5</td><td>5 - 1 = 4</td></tr>
<tr><td>1</td><td>1 -> 0</td><td>-2</td><td>1</td><td>4</td><td>-2</td></tr>
<tr><td>1</td><td>0 -> 1</td><td>1</td><td>0</td><td>-2</td><td>1</td></tr>
<tr><td>0</td><td>0 -> 1</td><td>5</td><td>1</td><td>1</td><td>1 -1 = 0</td></tr>
<tr><td>1</td><td>1 -> 0</td><td>-2147483648</td><td>1</td><td>0</td><td>-2147483648</td></tr>
<tr><td>0</td><td>0 -> 1</td><td>10</td><td>1</td><td>-2147483648</td><td>-2147483648</td></tr>
<tr><td colspan="6">CV_{prev} is the previous cycle counter output value.</td></tr>
</table>
</td></tr>
<tr><td>Inputs</td><td>Load input (LD): Boolean
Counter input (CD): Boolean
Preset input (PV): DINT</td></tr>
<tr><td>Outputs</td><td>Counter output (CV): DINT
Status output (Q): Boolean</td></tr>
</table>

CTU (10049)	
Illustration	
Execution time	0.92 µs

Operation

The counter output (CV) value is increased by 1 if the counter input (CU) value changes from 0 -> 1 and the reset input (R) value is 0. If the counter output has reached its maximum value 32767, the counter output remains unchanged.

The counter output (CV) is reset to 0 if the reset input (R) is 1.

The status output (Q) is 1 if the counter output (CV) value ≥ preset input (PV) value.

Example:

R	**CU**	**PV**	**Q**	**CV_{prev}**	**CV**
0	1 -> 0	20	0	10	10
0	0 -> 1	11	1	10	10 + 1 = 11
1	1 -> 0	20	0	11	0
1	0 -> 1	5	0	0	0
0	0 -> 1	20	0	0	0 + 1 = 1
0	0 -> 1	30	1	32767	32767

CV_{prev} is the previous cycle counter output value.

Inputs

Counter input (CU): Boolean
Reset input (R): Boolean
Preset input (PV): INT

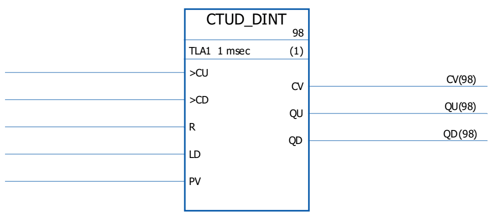
Outputs

Counter output (CV): INT
Status output (Q): Boolean

CTU_DINT

(10048)

Illustration



Execution time

0.92 µs

Operation

The counter output (CV) value is increased by 1 if the counter input (CU) value changes from 0 -> 1 and the reset input (R) value is 0. If the counter output has reached its maximum value 2147483647, the counter output remains unchanged.

The counter output (CV) is reset to 0 if the reset input (R) is 1.

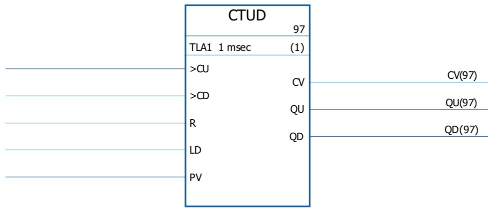
The status output (Q) is 1 if the counter output (CV) value ≥ preset input (PV) value.

Example:

R	**CU**	**PV**	**Q**	**CV_{prev}**	**CV**
0	1 -> 0	20	0	10	10
0	0 -> 1	11	1	10	10 + 1 = 11
1	1 -> 0	20	0	11	0
1	0 -> 1	5	0	0	0
0	0 -> 1	20	0	0	0 + 1 = 1
0	0 -> 1	30	1	2147483647	2147483647

CV_{prev} is the previous cycle counter output value.

Inputs	Counter input (CU): Boolean Reset input (R): Boolean Preset input (PV): DINT
Outputs	Counter output (CV): DINT Status output (Q): Boolean

CTUD (10051)	
Illustration	
Execution time	1.40 µs
Operation	The counter output (CV) value is increased by 1 if the counter input (CU) value changes from 0 -> 1 and the reset input (R) is 0 and the load input (LD) is 0. The counter output (CV) value is decreased by 1 if the counter input (CD) changes from 0 -> 1 and the load input (LD) is 0 and the reset input (R) is 0. If the load input (LD) is 1, the preset input (PV) value is stored as the counter output (CV) value. The counter output (CV) is reset to 0 if the reset input (R) is 1. If the counter output has reached its minimum or maximum value, -32768 or +32767, the counter output remains unchanged until it is reset (R) or until the load input (LD) is set to 1. The up counter status output (QU) is 1 if the counter output (CV) value ≥ preset input (PV) value. The down counter status output (QD) is 1 if the counter output (CV) value ≤ 0.

Example:

CU	CD	R	LD	PV	QU	QD	CV_{prev}	CV
0 -> 0	0 -> 0	0	0	2	0	1	0	0
0 -> 0	0 -> 0	0	1	2	1	0	0	2
0 -> 0	0 -> 0	1	0	2	0	1	2	0
0 -> 0	0 -> 0	1	1	2	0	1	0	0
0 -> 0	0 -> 1	0	0	2	0	1	0	0 - 1 = -1
0 -> 0	1 -> 1	0	1	2	1	0	-1	2
0 -> 0	1 -> 1	1	0	2	0	1	2	0
0 -> 0	1 -> 1	1	1	2	0	1	0	0
0 -> 1	1 -> 0	0	0	2	0	0	0	0 + 1 = 1
1 -> 1	0 -> 0	0	1	2	1	0	1	2
1 -> 1	0 -> 0	1	0	2	0	1	2	0
1 -> 1	0 -> 0	1	1	2	0	1	0	0
1 -> 1	0 -> 1	0	0	2	0	1	0	0 - 1 = -1
1 -> 1	1 -> 1	0	1	2	1	0	-1	2
1 -> 1	1 -> 1	1	0	2	0	1	2	0
1 -> 1	1 -> 1	1	1	2	0	1	0	0

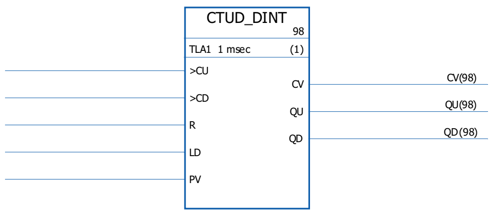
CV_{prev} is the previous cycle counter output value.

Inputs	Up counter input (CU): Boolean Down counter input (CD): Boolean Reset input (R): Boolean Load input (LD): Boolean Preset input (PV): INT
Outputs	Counter output (CV): INT Up counter status output (QU): Boolean Down counter status output (QD): Boolean

CTUD_DINT

(10051)

Illustration



Execution time	1.40 µs

Operation

The counter output (CV) value is increased by 1 if the counter input (CU) changes from 0 -> 1 and the reset input (R) is 0 and the load input (LD) is 0.

The counter output (CV) value is decreased by 1 if the counter input (CD) changes from 0 -> 1 and the load input (LD) is 0 and the reset input (R) is 0.

If the counter output has reached its minimum or maximum value, -2147483648 or +2147483647, the counter output remains unchanged until it is reset (R) or until the load input (LD) is set to 1.

If the load input (LD) value is 1, the preset input (PV) value is stored as the counter output (CV) value.

The counter output (CV) is reset to 0 if the reset input (R) is 1.

The up counter status output (QU) is 1 if the counter output (CV) value ≥ preset input (PV) value.

The down counter status output (QD) is 1 if the counter output (CV) value ≤ 0.

Example:

CU	CD	R	LD	PV	QU	QD	CV_{prev}	CV
0 -> 0	0 -> 0	0	0	2	0	1	0	0
0 -> 0	0 -> 0	0	1	2	1	0	0	2
0 -> 0	0 -> 0	1	0	2	0	1	2	0
0 -> 0	0 -> 0	1	1	2	0	1	0	0
0 -> 0	0 -> 1	0	0	2	0	1	0	0 - 1 = -1
0 -> 0	1 -> 1	0	1	2	1	0	-1	2
0 -> 0	1 -> 1	1	0	2	0	1	2	0
0 -> 0	1 -> 1	1	1	2	0	1	0	0
0 -> 1	1 -> 0	0	0	2	0	0	0	0 + 1 = 1
1 -> 1	0 -> 0	0	1	2	1	0	1	2
1 -> 1	0 -> 0	1	0	2	0	1	2	0
1 -> 1	0 -> 0	1	1	2	0	1	0	0
1 -> 1	0 -> 1	0	0	2	0	1	0	0 - 1 = -1
1 -> 1	1 -> 1	0	1	2	1	0	-1	2
1 -> 1	1 -> 1	1	0	2	0	1	2	0
1 -> 1	1 -> 1	1	1	2	0	1	0	0
CV_{prev} is the previous cycle counter output value.								

Inputs

Up counter input (CU): Boolean

Down counter input (CD): Boolean

Reset input (R): Boolean

Load input (LD): Boolean

Preset input (PV): DINT

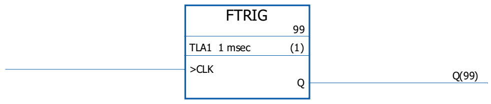
Outputs

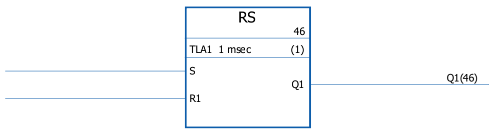
Counter output (CV): DINT

Up counter status output (QU): Boolean

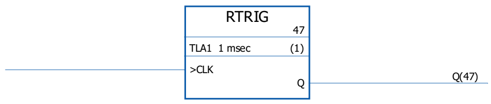
Down counter status output (QD): Boolean

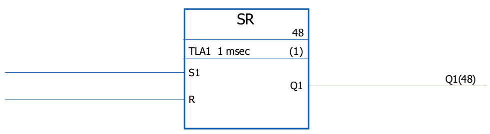
Edge & bistable

<table>
<tr><th colspan="2">FTRIG
(10030)</th></tr>
<tr><td>Illustration</td><td>
</td></tr>
<tr><td>Execution time</td><td>0.38 µs</td></tr>
<tr><td>Operation</td><td>The output (Q) is set to 1 when the clock input (CLK) changes from 1 to 0. The output is set back to 0 with the next execution of the block. Otherwise the output is 0.
<table>
<tr><th>$CLK_{previous}$</th><th>CLK</th><th>Q</th></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>1 (for one execution cycle time, returns to 0 at the next execution)</td></tr>
<tr><td>1</td><td>1</td><td>0</td></tr>
<tr><td colspan="3">$CLK_{previous}$ is the previous cycle output value.</td></tr>
</table></td></tr>
<tr><td>Inputs</td><td>Clock input (CLK): Boolean</td></tr>
<tr><td>Outputs</td><td>Output (Q): Boolean</td></tr>
</table>

<table>
<tr><th colspan="2">RS
(10032)</th></tr>
<tr><td>Illustration</td><td>
</td></tr>
<tr><td>Execution time</td><td>0.38 µs</td></tr>
<tr><td>Operation</td><td>The output (Q1) is 1 if the set input (S) is 1 and the reset input (R1) is 0. The output will retain the previous output state if the set input (S) and the reset input (R1) are 0. The output is 0 if the reset input is 1.
Truth table:
<table>
<tr><th>S</th><th>R1</th><th>$Q1_{previous}$</th><th>Q1</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>0</td></tr>
<tr><td colspan="4">$Q_{previous}$ is the previous cycle output value.</td></tr>
</table></td></tr>
</table>

Inputs	Set input (S): Boolean Reset input (R1): Boolean
Outputs	Output (Q1): Boolean

<table>
<tr><td colspan="2">RTRIG
(10031)</td></tr>
<tr><td>Illustration</td><td>
</td></tr>
<tr><td>Execution time</td><td>0.38 µs</td></tr>
<tr><td>Operation</td><td>The output (Q) is set to 1 when the clock input (CLK) changes from 0 to 1. The output is set back to 0 with the next execution of the block. Otherwise the output is 0.
<table>
<tr><th>$CLK_{previous}$</th><th>CLK</th><th>Q</th></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>0</td></tr>
<tr><td colspan="3">$CLK_{previous}$ is the previous cycle output value.</td></tr>
</table>
Note: The output (Q) is 1 after the first execution of the block after cold restart when the clock input (CLK) is 1. Otherwise the output is always 0 when the clock input is 1.</td></tr>
<tr><td>Inputs</td><td>Clock input (CLK): Boolean</td></tr>
<tr><td>Outputs</td><td>Output (Q): Boolean</td></tr>
</table>

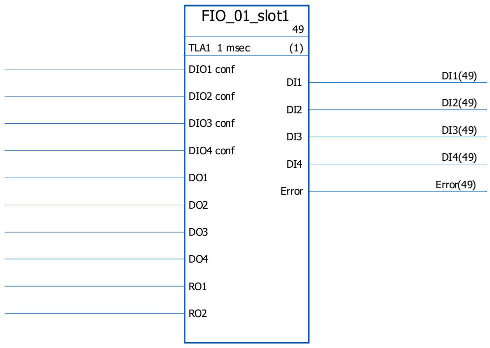
<table>
<tr><td colspan="2">SR
(10033)</td></tr>
<tr><td>Illustration</td><td>
</td></tr>
<tr><td>Execution time</td><td>0.38 µs</td></tr>
</table>

<table>
<tr><td>Operation</td><td>The output (Q1) is 1 if the set input (S1) is 1. The output will retain the previous output state if the set input (S1) and the reset input (R) are 0. The output is 0 if the set input is 0 and the reset input is 1.
Truth table:
<table>
<tr><th>S1</th><th>R</th><th>$Q1_{previous}$</th><th>Q1</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td></tr>
<tr><td colspan="4">$Q1_{previous}$ is the previous cycle output value.</td></tr>
</table>
</td></tr>
<tr><td>Inputs</td><td>Set input (S1): Boolean
Reset input (R): Boolean</td></tr>
<tr><td>Outputs</td><td>Output (Q1): Boolean</td></tr>
</table>

Extensions (ACS850 only)

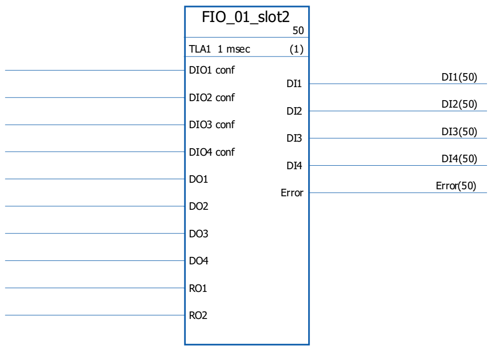
FIO_01_slot1

(10084)

Illustration	
Execution time	8.6 µs
Operation	The block controls the four digital inputs/outputs (DIO1…DIO4) and two relay outputs (RO1, RO2) of a FIO-01 Digital I/O Extension mounted on slot 1 of the drive control unit. The state of a DIOx conf input of the block determines whether the corresponding DIO on the FIO-01 is an input or an output (0 = input, 1 = output). If the DIO is an output, the DOx input of the block defines its state. The RO1 and RO2 inputs define the state of the relay outputs of the FIO-01 (0 = not energized, 1 = energized). The DIx outputs show the state of the DIOs.
Inputs	Digital input/output mode selection (DIO1 conf … DIO4 conf): Boolean Digital output state selection (DO1…DO4): Boolean Relay output state selection (RO1, RO2): Boolean
Outputs	Digital input/output state (DI1…DI4): Boolean Error output (Error): DINT (0 = No error; 1 = Application program memory full)

FIO_01_slot2

(10085)

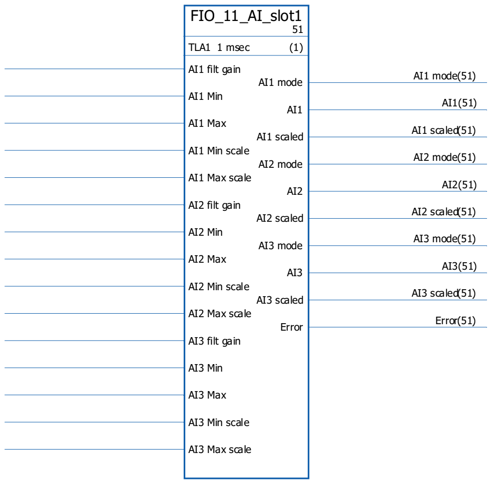
Illustration	
Execution time	8.6 µs

Operation	The block controls the four digital inputs/outputs (DIO1…DIO4) and two relay outputs (RO1, RO2) of a FIO-01 Digital I/O Extension mounted on slot 2 of the drive control unit. The state of a DIOx conf input of the block determines whether the corresponding DIO on the FIO-01 is an input or an output (0 = input, 1 = output). If the DIO is an output, the DOx input of the block defines its state. The RO1 and RO2 inputs define the state of the relay outputs of the FIO-01 (0 = not energised, 1 = energised). The DIx outputs show the state of the DIOs.
Inputs	Digital input/output mode selection (DIO1 conf … DIO4 conf): Boolean Digital output state selection (DO1…DO4): Boolean Relay output state selection (RO1, RO2): Boolean
Outputs	Digital input/output state (DI1…DI4): Boolean Error output (Error): DINT (0 = No error; 1 = Application program memory full)

FIO_11_AI_slot1

(10088)

Illustration



Execution time	11.1 µs

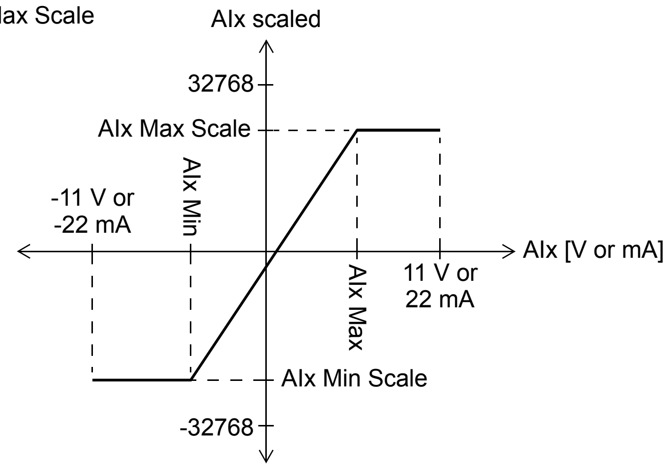
Operation

The block controls the three analogue inputs (AI1…AI3) of a FIO-11 Analog I/O Extension mounted on slot 1 of the drive control unit.

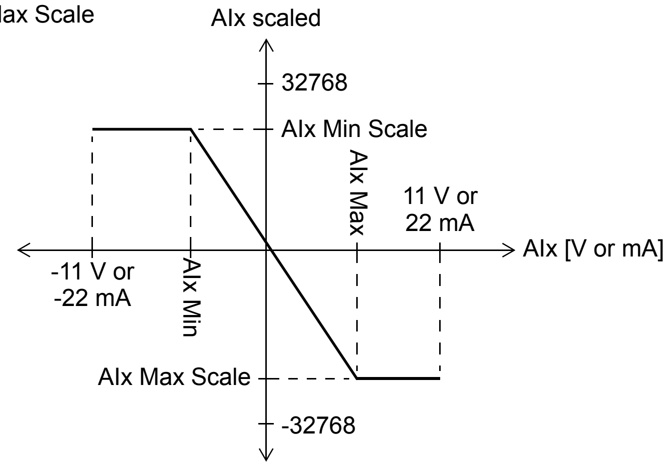
The block outputs both the unscaled (AIx) and scaled (AIx scaled) actual values of each analogue input. The scaling is based on the relationship between the ranges AIx min … AIx max and AIx min scale … AIx max scale.

AIx Min must be smaller than AIx Max; AIx Max Scale can be greater or smaller than AIx Min Scale.

AIx Min Scale < AIx Max Scale



AIx Min Scale > AIx Max Scale

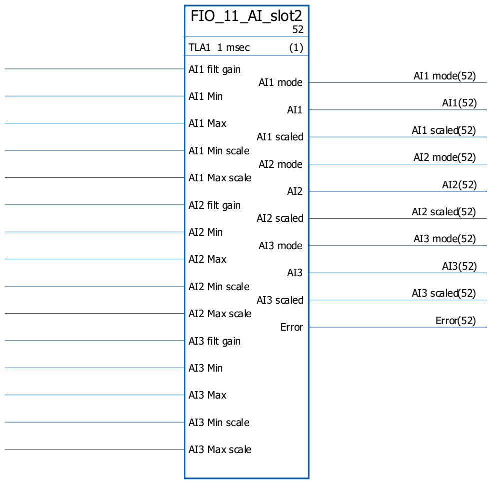
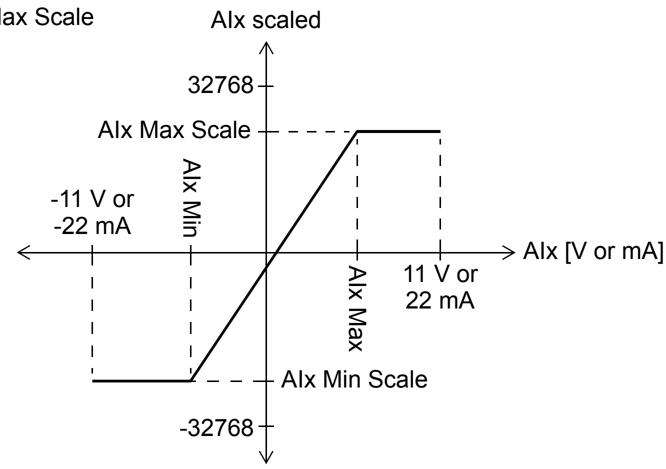


The AIx filt gain inputs determine a filtering time for each input as follows:

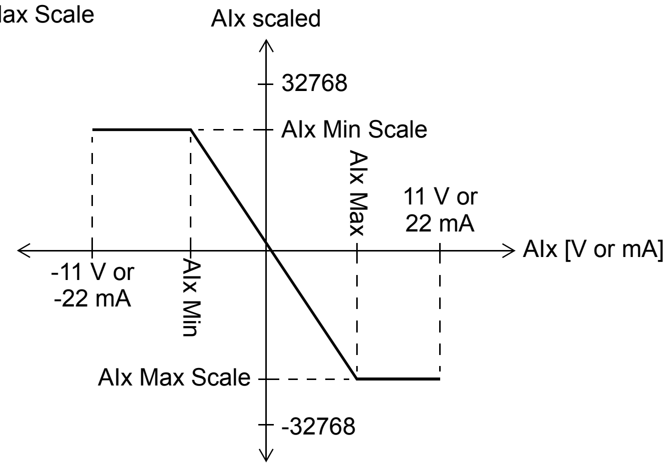
AIx filt gain	Filtering time	Notes
0	No filtering	
1	125 µs	Recommended setting
2	250 µs	
3	500 µs	
4	1 ms	
5	2 ms	
6	4 ms	
7	7.9375 ms	

The AIx mode outputs show whether the corresponding input is voltage (0) or current (1). The voltage/current selection is made using the hardware switches on the FIO-11.

Inputs	Analogue input filter gain selection (AI1 filt gain … AI3 filt gain): INT Minimum value of input signal (AI1 Min … AI3 Min): REAL (≥ -11 V or -22 mA) Maximum value of input signal (AI1 Max … AI3 Max): REAL (≤ 11 V or 22 mA) Minimum value of scaled output signal (AI1 Min scale … AI3 Min scale): REAL Maximum value of scaled output signal (AI1 Max scale … AI3 Max scale): REAL
Outputs	Analogue input mode (voltage or current) (AI1 mode … AI3 mode): Boolean Value of analogue input (AI1 … AI3): REAL Scaled value of analogue input (AI1 scaled … AI3 scaled): REAL Error output (Error): DINT (0 = No error; 1 = Application program memory full)

FIO_11_AI_slot2 (10089)	
Illustration	
Execution time	11.1 µs
Operation	The block controls the three analogue inputs (AI1…AI3) of a FIO-11 Analog I/O Extension mounted on slot 2 of the drive control unit. The block outputs both the unscaled (AIx) and scaled (AIx scaled) actual values of each analogue input. The scaling is based on the relationship between the ranges AIx min … AIx max and AIx min scale … AIx max scale. AIx Min must be smaller than AIx Max; AIx Max Scale can be greater or smaller than AIx Min Scale. AIx Min Scale < AIx Max Scale 

AIx Min Scale > AIx Max Scale



The AIx filt gain inputs determine a filtering time for each input as follows:

AIx filt gain	Filtering time	Notes
0	No filtering	
1	125 µs	Recommended setting
2	250 µs	
3	500 µs	
4	1 ms	
5	2 ms	
6	4 ms	
7	7.9375 ms	

The AIx mode outputs show whether the corresponding input is voltage (0) or current (1). The voltage/current selection is made using the hardware switches on the FIO-11.

Inputs

Analogue input filter gain selection (AI1 filt gain … AI3 filt gain): INT
Minimum value of input signal (AI1 Min … AI3 Min): REAL (≥ -11 V or -22 mA)
Maximum value of input signal (AI1 Max … AI3 Max): REAL (≤ 11 V or 22 mA)
Minimum value of scaled output signal (AI1 Min scale … AI3 Min scale): REAL
Maximum value of scaled output signal (AI1 Max scale … AI3 Max scale): REAL

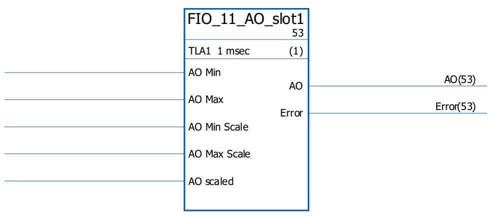
Outputs

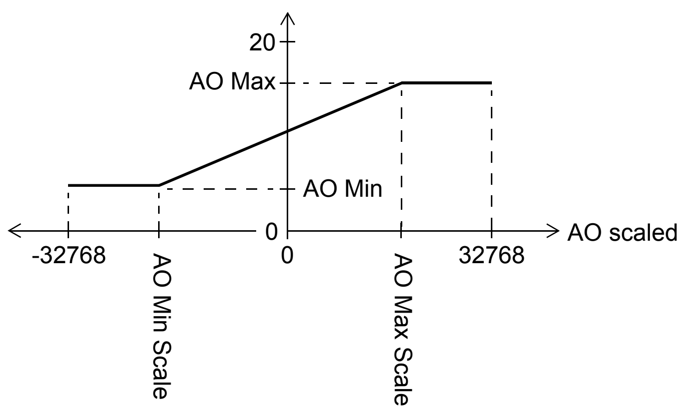
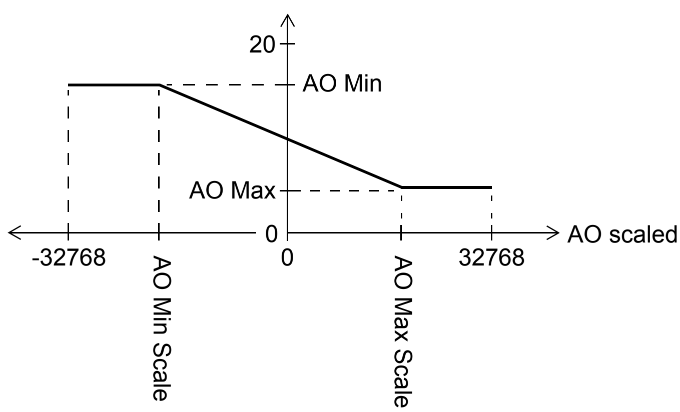
Analogue input mode (voltage or current) (AI1 mode … AI3 mode): Boolean
Value of analogue input (AI1 … AI3): REAL
Scaled value of analogue input (AI1 scaled … AI3 scaled): REAL
Error output (Error): DINT (0 = No error; 1 = Application program memory full)

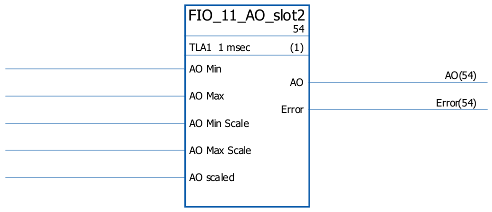
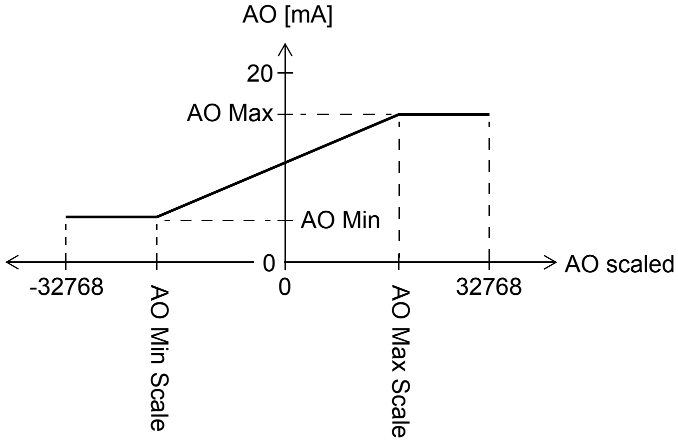
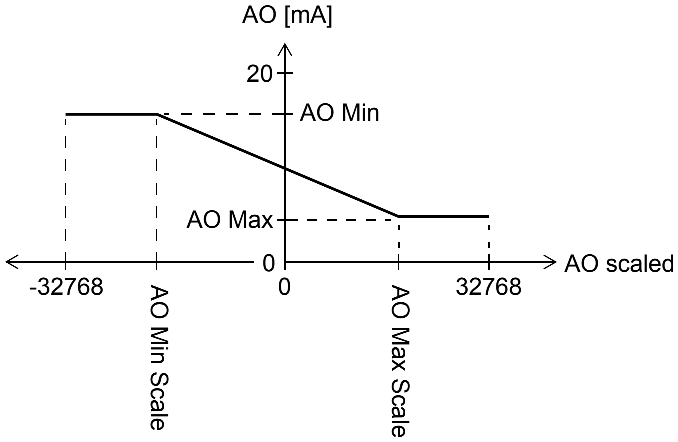
FIO_11_AO_slot1

(10090)

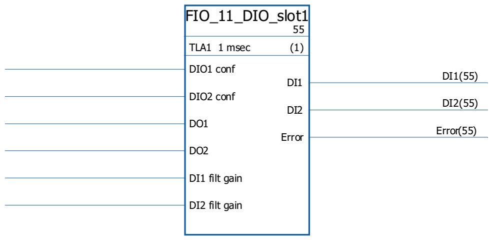
Illustration

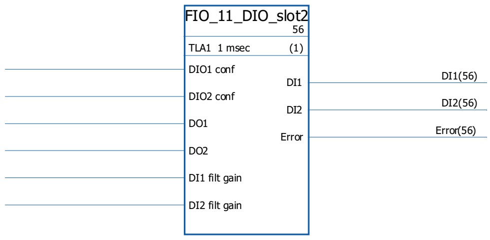


Execution time	4.9 µs
Operation	The block controls the analogue output (AO1) of a FIO-11 Analog I/O Extension mounted on slot 1 of the drive control unit. The block converts the input signal (AO scaled) to a 0…20 mA signal (AO) that drives the analogue output; the input range AO Min Scale … AO Max Scale corresponds to the current signal range of AO Min … AO Max. AO Min Scale must be smaller than AO Max Scale; AO Max can be greater or smaller than AO Min. AO Min < AO Max  AO Min > AO Max 
Inputs	Minimum current signal (AO Min): REAL (0…20 mA) Maximum current signal (AO Max): REAL (0…20 mA) Minimum input signal (AO Min Scale): REAL Maximum input signal (AO Max Scale): REAL Input signal (AO scaled): REAL
Outputs	Analogue output current value (AO): REAL Error output (Error): DINT (0 = No error; 1 = Application program memory full)

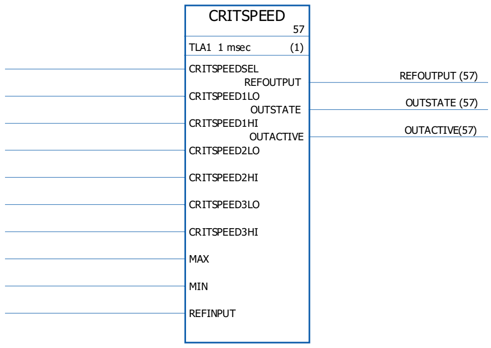
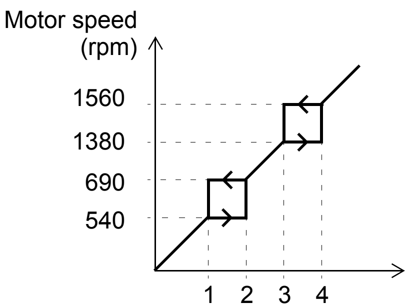
FIO_11_AO_slot2 (10091)	
Illustration	
Execution time	4.9 µs
Operation	The block controls the analogue output (AO1) of a FIO-11 Analog I/O Extension mounted on slot 2 of the drive control unit. The block converts the input signal (AO scaled) to a 0…20 mA signal (AO) that drives the analogue output; the input range AO Min Scale … AO Max Scale corresponds to the current signal range of AO Min … AO Max. AO Min Scale must be smaller than AO Max Scale; AO Max can be greater or smaller than AO Min. AO Min < AO Max  AO Min > AO Max 

Inputs	Minimum current signal (AO Min): REAL (0…20 mA) Maximum current signal (AO Max): REAL (0…20 mA) Minimum input signal (AO Min Scale): REAL Maximum input signal (AO Max Scale): REAL Input signal (AO scaled): REAL
Outputs	Analogue output current value (AO): REAL Error output (Error): DINT (0 = No error; 1 = Application program memory full)

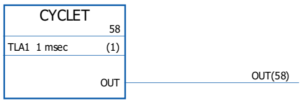
<table>
<tr><td colspan="2">FIO_11_DIO_slot1
(10086)</td></tr>
<tr><td>Illustration</td><td>
</td></tr>
<tr><td>Execution time</td><td>6.0 µs</td></tr>
<tr><td>Operation</td><td>The block controls the two digital inputs/outputs (DIO1, DIO2) of a FIO-11 Digital I/O Extension mounted on slot 1 of the drive control unit.
The state of a DIOx conf input of the block determines whether the corresponding DIO on the FIO-11 is an input or an output (0 = input, 1 = output). If the DIO is an output, the DOx input of the block defines its state.
The DIx outputs show the state of the DIOs.
The DIx filt gain inputs determine a filtering time for each input as follows:
<table>
<tr><th>DIx filt gain</th><th>Filtering time</th></tr>
<tr><td>0</td><td>7.5 µs</td></tr>
<tr><td>1</td><td>195 µs</td></tr>
<tr><td>2</td><td>780 µs</td></tr>
<tr><td>3</td><td>4.680 ms</td></tr>
</table></td></tr>
<tr><td>Inputs</td><td>Digital input/output mode selection (DIO1 conf, DIO2 conf): Boolean
Digital output state selection (DO1, DO2): Boolean
Digital input filter gain selection (DI1 filt gain, DI2 filt gain): INT</td></tr>
<tr><td>Outputs</td><td>Digital input/output state (DI1, DI2): Boolean
Error output (Error): DINT (0 = No error; 1 = Application program memory full)</td></tr>
</table>

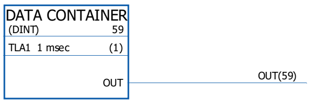
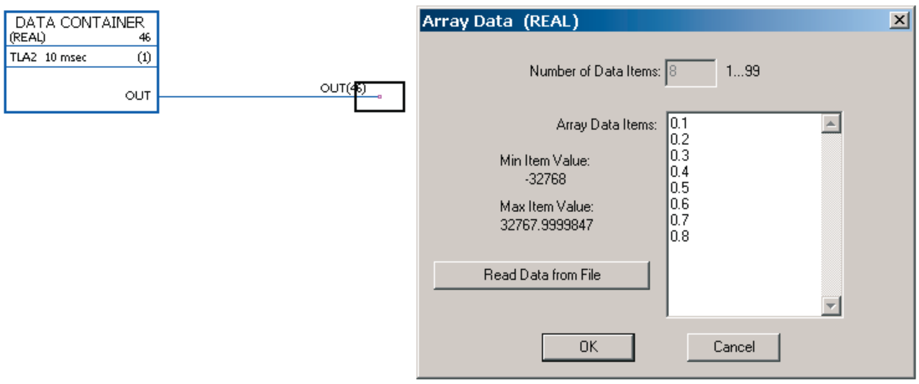
<table>
<tr><th colspan="2">FIO_11_DIO_slot2
(10087)</th></tr>
<tr><td>Illustration</td><td>
</td></tr>
<tr><td>Execution time</td><td>6.0 µs</td></tr>
<tr><td>Operation</td><td>The block controls the two digital inputs/outputs (DIO1, DIO2) of a FIO-11 Digital I/O Extension mounted on slot 2 of the drive control unit.
The state of a DIOx conf input of the block determines whether the corresponding DIO on the FIO-11 is an input or an output (0 = input, 1 = output). If the DIO is an output, the DOx input of the block defines its state.
The DIx outputs show the state of the DIOs.
The DIx filt gain inputs determine a filtering time for each input as follows:
<table>
<tr><th>DIx filt gain</th><th>Filtering time</th></tr>
<tr><td>0</td><td>7.5 µs</td></tr>
<tr><td>1</td><td>195 µs</td></tr>
<tr><td>2</td><td>780 µs</td></tr>
<tr><td>3</td><td>4.680 ms</td></tr>
</table></td></tr>
<tr><td>Inputs</td><td>Digital input/output mode selection (DIO1 conf, DIO2 conf): Boolean
Digital output state selection (DO1, DO2): Boolean
Digital input filter gain selection (DI1 filt gain, DI2 filt gain): INT</td></tr>
<tr><td>Outputs</td><td>Digital input/output state (DI1, DI2): Boolean
Error output (Error): DINT (0 = No error; 1 = Application program memory full)</td></tr>
</table>

Feedback & algorithms

<table>
<tr><th colspan="2">CRITSPEED
(10068)</th></tr>
<tr><td>Illustration</td><td>
</td></tr>
<tr><td>Execution time</td><td>4.50 µs</td></tr>
<tr><td>Operation</td><td>A critical speeds function block is available for applications where it is necessary to avoid certain motor speeds or speed bands because of e.g. mechanical resonance problems. The user can define three critical speeds or speed bands.
Example: An application has vibrations in the range of 540 to 690 rpm and 1380 to 1560 rpm. To make the drive made to jump over the vibration speed ranges:
- activate the critical speeds function (CRITSPEEDSEL = 1),
- set the critical speed ranges as in the figure below.


<table>
<tr><td>1</td><td>CRITSPEED1LO = 540 rpm</td></tr>
<tr><td>2</td><td>CRITSPEED1HI = 690 rpm</td></tr>
<tr><td>3</td><td>CRITSPEED2LO = 1380 rpm</td></tr>
<tr><td>4</td><td>CRITSPEED2HI = 1560 rpm</td></tr>
</table>
Output OUTACTIVE is set to 1 when the output reference (REFOUTPUT) is different from the input reference (REFINPUT).
The output is limited by the defined minimum and maximum limits (MIN and MAX).
Output OUTSTATE indicates in which critical speed range the operation point is.</td></tr>
<tr><td>Inputs</td><td>Critical speed activation input (CRITSPEEDSEL): Boolean
Minimum/maximum critical speed range input (CRITSPEEDNLO / CRITSPEEDNHI): REAL
Maximum/minimum input (MAX/MIN): REAL
Reference input (REFINPUT): REAL</td></tr>
<tr><td>Outputs</td><td>Reference output (REFOUTPUT): REAL
Output state (OUTSTATE): REAL
Output active (OUTACTIVE): Boolean</td></tr>
</table>

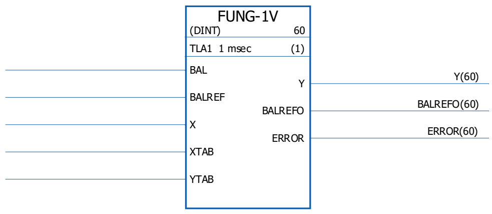
CYCLET (10074)	
Illustration	
Execution time	0.00 µs
Operation	Output (OUT) is the time level of the CYCLET function block.
Inputs	-
Outputs	Output (OUT): DINT. 1 = 1 µs

DATA CONTAINER (10073)	
Illustration	
Execution time	0.00 µs
Operation	Output (OUT) is an array of data with values 1…99. The array can be used by the XTAB and YTAB tables in block *FUNG-1V*. The array is defined by selecting "Define Pin Array Data" on the output pin in DriveSPC. Each value in the array must be on a separate row. Data can also be read from an *.arr file. Example: 
Inputs	-
Outputs	The output data type and the number of coordinate pairs are selected by the user. Output (OUT): DINT, INT, REAL or REAL24

FUNG-1V

(10072)

Illustration



Execution time

9.29 µs

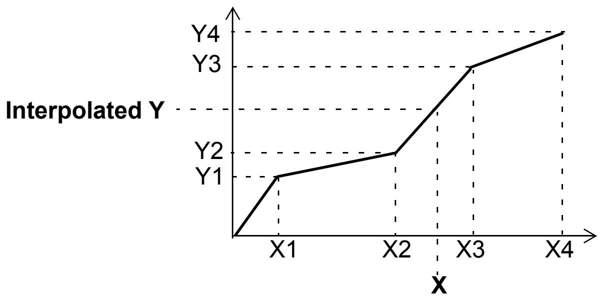
Operation

The output (Y) at the value of the input (X) is calculated with linear interpolation from a piecewise linear function.

$Y = Y_k + (X - X_k)(Y_{k+1} - Y_k) / (X_{k+1} - X_k)$

The piecewise linear function is defined by the X and Y vector tables (XTAB and YTAB). For each X-value in the XTAB table, there is a corresponding Y-value in the YTAB table. The values in XTAB and YTAB must be in ascending order (i.e. from low to high).

XTAB and YTAB values are defined with the DriveSPC tool.



X table (XTAB)	Y table (YTAB)
X1	Y1
X2	Y2
X3	Y3
…	…
X9	Y9

The balancing function (BAL) permits the output signal to track an external reference and gives a smooth return to the normal operation. If BAL is set to 1, output Y is set to the value of the balance reference input (BALREF). The X value which corresponds to this Y value is calculated with linear interpolation and it is indicated by the balance reference output (BALREFO).

If the X input is outside the range defined by the XTAB table, the output Y is set to the highest or lowest value in the YTAB table.

If BALREF is outside the range defined by the YTAB table when balancing is activated (BAL: 0 -> 1), the output Y is set to the value of the BALREF input and the BALREFO output is set to the highest or lowest value in the XTAB table.

The ERROR output is set to 1 when the number of the XTAB and YTAB inputs are different. When ERROR is 1, the FUNG-1V block will not function. XTAB and YTAB tables can be defined in the *DATA CONTAINER* block or the *REG-G* block.

Inputs

The input data type is selected by the user.

Balance input (BAL): Boolean

Balance reference input (BALREF): DINT, INT, REAL, REAL24.

X value input (X): DINT, INT, REAL, REAL24

X table input (XTAB): DINT, INT, REAL, REAL24

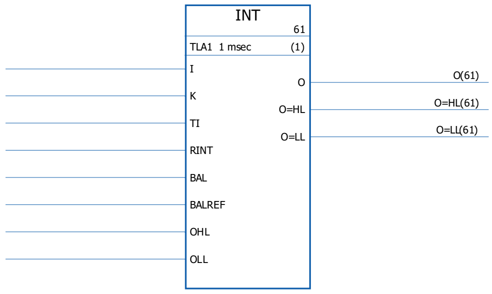
Y table input (YTAB): DINT, INT, REAL, REAL24

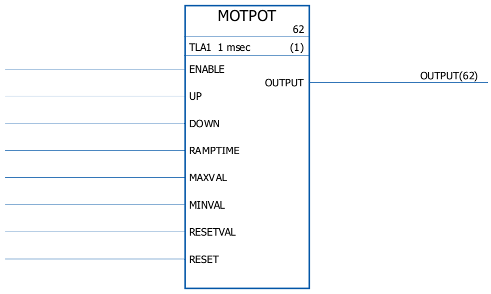
Outputs

Y value output (Y): DINT, INT, REAL, REAL24

Balance reference output (BALREFO): DINT, INT, REAL, REAL24

Error output (ERROR): Boolean

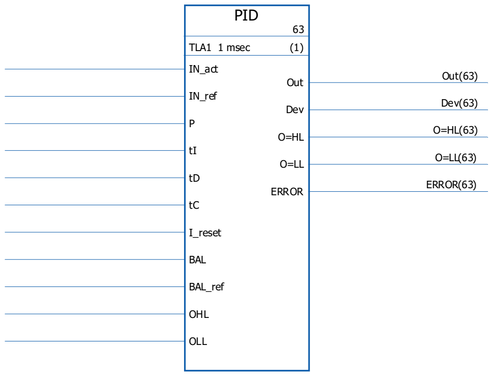
INT (10065)	
Illustration	
Execution time	4.73 µs
Operation	The output (O) is the integrated value of the input (I): $O(t) = K/TI (\int I(t)\,dt)$ Where TI is the integration time constant and K is the integration gain. The step response for the integration is: $O(t) = K \times I(t) \times t/TI$ The transfer function for the integration is: $G(s) = K\ 1/sTI$ The output value is limited according to the defined minimum and maximum limits (OLL and OHL). If the value is below the minimum value, output O = LL is set to 1. If the value exceeds the maximum value, output O = HL is set to 1. The output (O) retains its value when the input signal I(t) = 0. The integration time constant is limited to value 2147483 ms. If the time constant is negative, zero time constant is used. If the ratio between the cycle time and the integration time constant Ts/TI < 1, Ts/TI is set to 1. The integrator is cleared when the reset input (RINT) is set to 1. If BAL is set to 1, output O is set to the value of the input BALREF. When BAL is set back to 0, normal integration operation continues.
Inputs	Input (I): REAL Gain input (K): REAL Integration time constant input (TI): DINT, 0…2147483 ms Integrator reset input (RINT): Boolean Balance input (BAL): Boolean Balance reference input (BALREF): REAL Output high limit input (OHL): REAL Output low limit input (OLL): REAL
Outputs	Output (O): REAL High limit output (O=HL): Boolean Low limit output (O=LL): Boolean

MOTPOT (10067)	
Illustration	
Execution time	2.92 µs
Operation	The motor potentiometer function controls the rate of change of the output from the minimum to the maximum value and vice versa. The function is enabled by setting the ENABLE input to 1. If the up input (UP) is 1, the output reference (OUTPUT) is increased to the maximum value (MAXVAL) with the defined ramp time (RAMPTIME). If the down input (DOWN) is 1, the output value is decreased to the minimum value (MINVAL) with the defined ramp time. If the up and down inputs are activated/deactivated simultaneously, the output value is not increased/decreased. If the RESET input is 1, the output will be reset to the value defined by the reset value input (RESETVAL) or to the value defined by the minimum input (MINVAL), whichever is higher. If the ENABLE input is 0, the output is zero. Digital inputs are normally used as up and down inputs.
Inputs	Function enable input (ENABLE): Boolean Up input (UP): Boolean Down input (DOWN): Boolean Ramp time input (RAMPTIME): REAL (seconds) (i.e. the time required for the output to change from the minimum to the maximum value or from the maximum to the minimum value) Maximum reference input (MAXVAL): REAL Minimum reference input (MINVAL): REAL Reset value input (RESETVAL): REAL Reset input (RESET): Boolean
Outputs	Output (OUTPUT) REAL

PID

(10075)

Illustration



Execution time

15.75 μs

Operation

The PID controller can be used for closed-loop control systems. The controller includes anti-windup correction and output limitation.

The PID controller output (Out) before limitation is the sum of the proportional (U_P), integral (U_I) and derivative (U_D) terms:

$Out_{unlimited}(t) = U_P(t) + U_I(t) + U_D(t)$

$U_P(t) = P \times Dev(t)$

$U_I(t) = P/tI \times [\int Dev(\tau)d\tau + tC \times (Out(t) - Out_{unlimited}(t))]$

$U_D(t) = P \times tD \times d(Dev(t))/dt$

Integrator:

The integral term can be cleared by setting I_reset to 1. Note that the anti-windup correction is simultaneously disabled. When I_reset is 1, the controller acts as a PD controller.

If integration time constant tI is 0, the integral term will not be updated.

Smooth return to normal operation is guaranteed after errors or abrupt input value changes. This is achieved by adjusting the integral term so that the output will retain its previous value during these situations.

Limitation:

The output is limited by the defined minimum and maximum values, OLL and OHL:

If the actual value of the output reaches the specified minimum limit, output O=LL is set to 1.

If the actual value of the output reaches the specified maximum limit, output O=HL is set to 1.

Smooth return to normal operation after limitation is requested if and only if the anti-windup correction is not used, i.e. when tI = 0 or tC = 0.

Error codes:

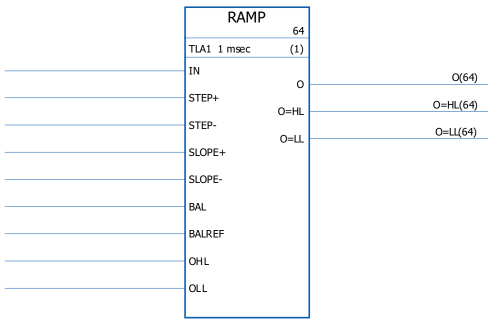
Error codes are indicated by the error output (ERROR) as follows

Error code	Description
1	The minimum limit (OLL) exceeds the maximum limit (OHL).
2	Overflow with Up, Ui, or Ud calculation

	Balancing: The balancing function (BAL) permits the output signal to track an external reference and gives a smooth return to the normal operation. If BAL is set to 1, the output (Out) is set to the value of the balance reference input (BAL_ref). Balance reference is limited by the defined minimum and maximum limits (OLL and OHL). Anti-windup: Anti-windup correction time constant is defined by input tC, which defines the time after which the difference between the unlimited and limited outputs is subtracted from the I-term during limitation. If tC = 0 or tI = 0, anti-windup correction is disabled.
Inputs	Actual input (IN_act): REAL Reference input (IN_ref): REAL Proportional gain input (P): REAL Integration time constant input (tI): REAL. 1 = 1 ms Derivation time constant input (tD): REAL. 1 = 1 ms Antiwind-up correction time constant input (tC): IQ6. 1 = 1 ms Integrator reset input (I_reset): Boolean Balance input (BAL): Boolean Balance reference input (BAL_ref): REAL Output high limit input (OHL): REAL Output low limit input (OLL): REAL
Outputs	Output (Out): REAL Deviation output (Dev): REAL (= actual -reference = IN_act - IN_ref) High limit output (O=HL): Boolean Low limit output (O=LL): Boolean Error code output (ERROR): INT32

RAMP

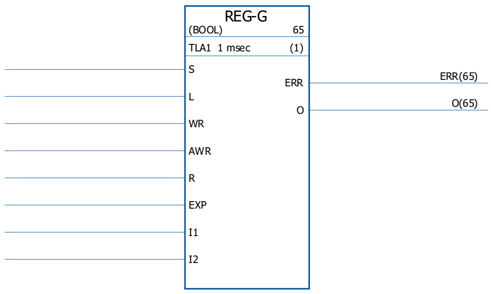
(10066)

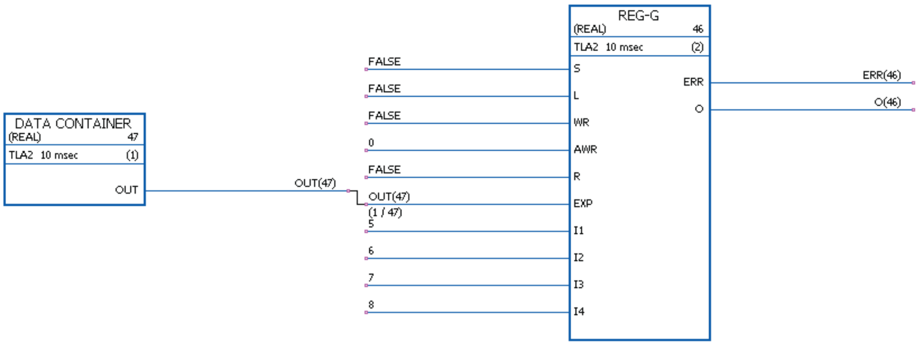
Illustration	
Execution time	4.23 µs

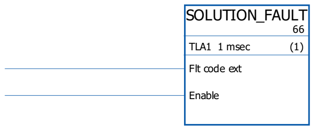
Operation	Limits the rate of the change of the signal. The input signal (IN) is connected directly to the output (O) if the input signal does not exceed the defined step change limits (STEP+ and STEP-). If the input signal change exceeds these limits, the output signal change is limited by the maximum step change (STEP+/STEP- depending on the direction of rotation). After this, the output signal is accelerated/decelerated by the defined ramp value (SLOPE+/SLOPE-) per second until the input and output signal values are equal. The output is limited by the defined minimum and maximum values (OLL and OHL). If the actual value of the output falls below the specified minimum limit (OLL), output O=LL is set to 1. If the actual value of the output exceeds the specified maximum limit (OHL), output O=HL is set to 1. If the balancing input (BAL) is set to 1, the output (O) is set to the value of the balance reference input (BAL_ref). Balancing reference is also limited by the minimum and maximum values (OLL and OHL).
Inputs	Input (IN): REAL Maximum positive step change input (STEP+): REAL Maximum negative step change input (STEP-): REAL Ramp-up value per second input (SLOPE+): REAL Ramp-down value per second input (SLOPE-): REAL Balance input (BAL): Boolean Balance reference input (BALREF): REAL Output high limit input (OHL): REAL Output low limit input (OLL): REAL
Outputs	Output (O): REAL High limit output (O=HL): Boolean Low limit output (O=LL): Boolean

REG-G

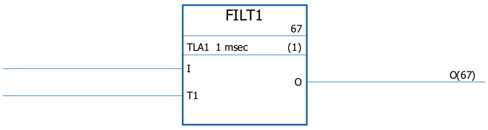
(10102)

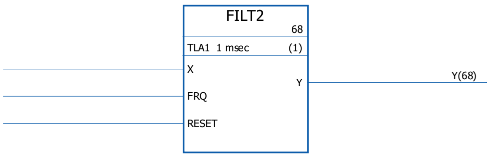
Illustration	
Execution time	-

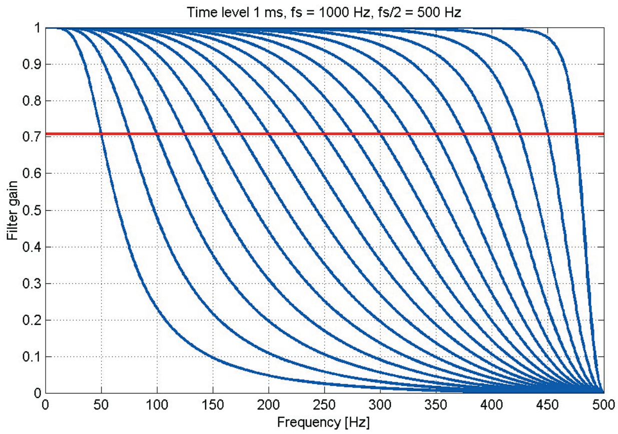
Operation	Combines the array (group of variables) (if any) on the EXP input with the values of the I1…I32 pins to produce an output array. The data type of the arrays can be INT, DINT, REAL16, REAL24 or Boolean. The output array consists of the data from the EXP input and the values of the I1…In (in this order). When input S is 1, data is continuously assembled into the output array. The element acts as a latch when input S is 0; the latest data assembled then remains at the output. If S is 0 and L changes state from 0 to 1, the array from the EXP input and the values of the I1…In inputs are copied to output O during this program cycle. If S or R is 1, L has no effect. WR and AWR are used to change individual cells of the output array. AWR indicates the input whose value is moved to the output array. If AWR is 0, only the array from input EXP is moved to the output. If AWR is not 0, the corresponding I input is moved to the output. This is performed when WR goes from 0 to 1. When input R is 1, the output array is cleared and all further data entry is prevented. R overrides both S and L. If WR is 1, the address at AWR is checked and if it is illegal (negative or greater than the number of inputs), the error output (ERR) is set to 2. Otherwise ERR is 0. Whenever an error is detected, ERR is set within one cycle. No place in the register is affected when an error occurs. Example:  In the diagram, the DATA CONTAINER block includes an array with values [1,2,3,4]. At start, the output array is [0,0,0,0,0,0,0,0]. When WR changes to 1 and returns to 0, the AWR value of 0 means that only EXP is moved into the output array, which now reads [1,2,3,4,0,0,0,0]. After this, AWR is changed to 3, meaning that inputs EXP and I3 are moved to the output. After a WR switch, the output array is [1,2,3,4,0,0,7,0].
Inputs	Set (S): Boolean, INT, DINT, REAL, REAL24 Load (L): Boolean, INT, DINT, REAL, REAL24 Write (WR): Boolean, INT, DINT, REAL, REAL24 Write address (AWR): INT Reset (R): Boolean Expander (EXP): IArray Data input (I1…I32): Boolean, INT, DINT, REAL, REAL24
Outputs	Error (ERR): INT Array data output (O): OC1

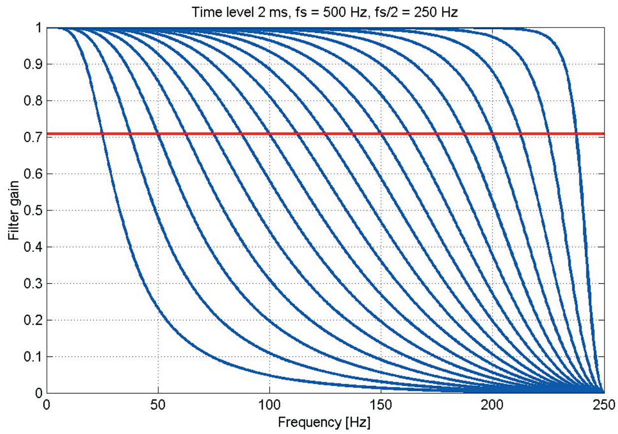
SOLUTION_FAULT (10097)	
Illustration	
Execution time	-
Operation	When the block is enabled (by setting the Enable input to 1), a fault (F-0317 SOLUTION FAULT) is generated by the drive. The value of the Flt code ext input is recorded by the fault logger.
Inputs	Fault code extension (Flt code ext): DINT Generate fault (Enable): Boolean
Outputs	-

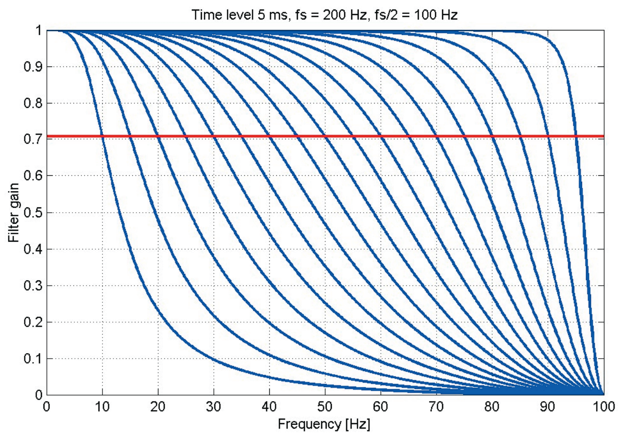
Filters

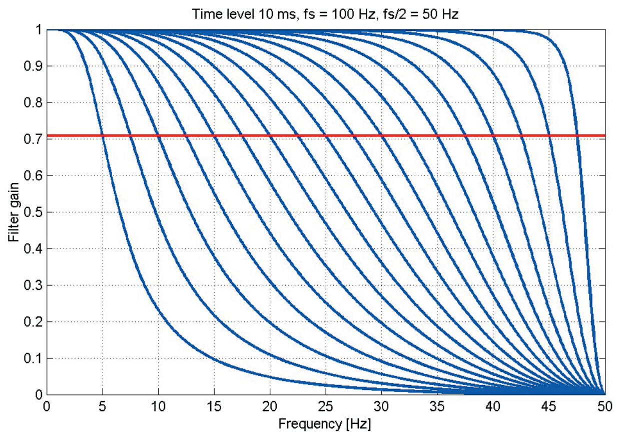
FILT1 (10069)	
Illustration	
Execution time	7.59 µs
Operation	The output (O) is the filtered value of the input (I) value and the previous output value (O_{prev}). The FILT1 block acts as 1st order low pass filter. **Note:** Filter time constant (T1) must be selected so that T1/Ts < 32767. If the ratio exceeds 32767, it is considered as 32767. Ts is the cycle time of the program in ms. If T1 < Ts, the output value is the input value. The step response for a single pole low pass filter is: $O(t) = I(t) \times (1 - e^{-t/T1})$ The transfer function for a single pole low pass filter is: $G(s) = 1/(1 + sT1)$
Inputs	Input (I): REAL Filter time constant input (T1): DINT, 1 = 1 ms
Outputs	Output (O): REAL

FILT2 (10070)	
Illustration	
Execution time	6.30 µs
Operation	The output (Y) is the filtered value of the input (X). The FILT2 block acts as a 2nd order low pass filter. When the RESET input value is set to 1, the input is connected to the output without filtering. **Notes:** • The -3 dB cutoff frequency (FRQ) is limited to its maximum value (16383 Hz). • The frequency of the input signal must be less than half of sampling frequency (fs) – any higher frequencies are aliased to the allowable range. The sampling frequency is defined by the time level of the block; for example, 1 ms corresponds to a sampling frequency of 1000 Hz. The following diagrams show the frequency responses for 1, 2, 5 and 10 ms time levels. The -3 dB cutoff level is represented as the horizontal line at 0.7 gain.


Time level 1 ms, fs = 1000 Hz, fs/2 = 500 Hz
Filter gain
Frequency [Hz]


Time level 2 ms, fs = 500 Hz, fs/2 = 250 Hz
Filter gain
Frequency [Hz]

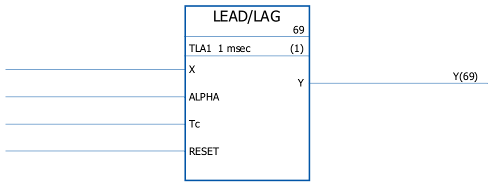

Time level 5 ms, fs = 200 Hz, fs/2 = 100 Hz
Filter gain
Frequency [Hz]


Time level 10 ms, fs = 100 Hz, fs/2 = 50 Hz
Filter gain
Frequency [Hz]

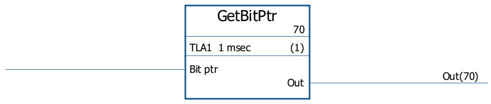
Inputs	Input (X): REAL -3 dB cutoff frequency input (FRQ): DINT (0…16383 Hz) Reset input (RESET): Boolean
Outputs	Output (Y): REAL

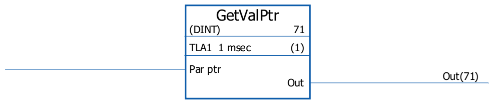
LEAD/LAG

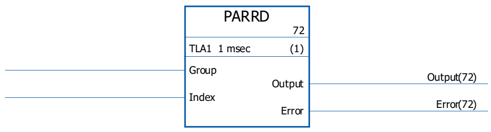
(10071)

Illustration	
Execution time	5.55 µs
Operation	The output (Y) is the filtered value of the input (X). When ALPHA > 1, the function block acts as a lead filter. When ALPHA < 1, the function block acts as a lag filter. When ALPHA = 1, no filtering occurs. The transfer function for a lead/lag filter is: $(1 + \mathrm{ALPHA}T_c s) / (1 + T_c s)$ When RESET input is 1, the input value (X) is connected to the output (Y). If ALPHA or Tc < 0, the negative input value is set to zero before filtering.
Inputs	Input (X): REAL Lead/Lag filter type input (ALPHA): REAL Time constant input (Tc): REAL Reset input (RESET): Boolean
Outputs	Output (Y): REAL

Parameters

GetBitPtr (10099)	
Illustration	
Execution time	-
Operation	Reads the status of one bit within a parameter value cyclically. The Bit ptr input specifies the parameter group, index and bit to be read. The output (Out) provides the value of the bit.
Inputs	Parameter group, index and bit (Bit ptr): DINT
Outputs	Bit status (Out): DINT

GetValPtr (10098)	
Illustration	
Execution time	-
Operation	Reads the value of a parameter cyclically. The Par ptr input specifies the parameter group and index to be read. The output (Out) provides the value of the parameter.
Inputs	Parameter group and index (Par ptr): DINT
Outputs	Parameter value (Out): DINT

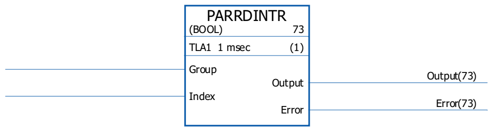
<table>
<tr><th colspan="2">PARRD
(10082)</th></tr>
<tr><td>Illustration</td><td></td></tr>
<tr><td>Execution time</td><td>6.00 µs</td></tr>
<tr><td>Operation</td><td>Reads the scaled value of a parameter (specified by the Group and Index inputs). If the parameter is a pointer parameter, the Output pin provides the number of the source parameter instead of its value.
Error codes are indicated by the error output (Error) as follows:
<table>
<tr><th>Error code</th><th>Description</th></tr>
<tr><td>0</td><td>No error</td></tr>
<tr><td><> 0</td><td>Error</td></tr>
</table>
See also blocks PARRDINTR and PARRDPTR.</td></tr>
</table>

Inputs	Parameter group input (Group): DINT Parameter index input (Index): DINT
Outputs	Output (Output): DINT Error output (Error): DINT

PARRDINTR

(10101)

Illustration



Execution time

-

Operation

Reads the internal (non-scaled) value of a parameter (specified by the Group and Index inputs). The value is provided by the Output pin.

Error codes are indicated by the error output (Error) as follows:

Error code	Description
0	No error or busy
<> 0	Error

Note: Using this block may cause incompatibility issues when upgrading the application to another firmware version.

Inputs

Parameter group (Group): DINT
Parameter index (Index): DINT

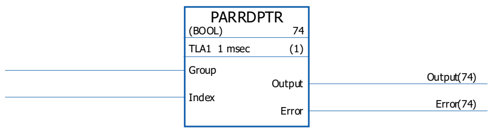
Outputs

Output (Output): Boolean, INT, DINT, REAL, REAL24
Error output (Error): DINT

PARRDPTR

(10100)

Illustration



Execution time

-

Operation

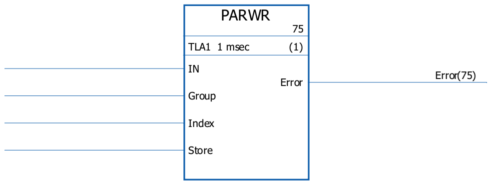
Reads the internal (non-scaled) value of the source of a pointer parameter. The pointer parameter is specified using the Group and Index inputs.

The value of the source selected by the pointer parameter is provided by the Output pin.

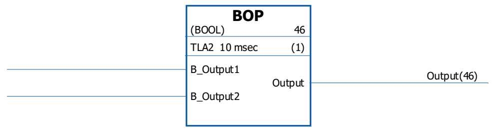
Error codes are indicated by the error output (Error) as follows:

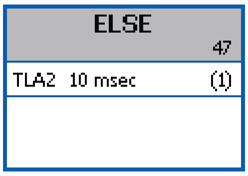
Error code	Description
0	No error or busy
<> 0	Error

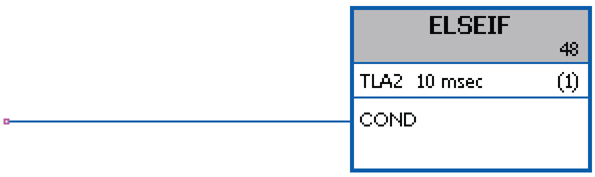
Inputs	Parameter group (Group): DINT Parameter index (Index): DINT
Outputs	Output (Output): Boolean, INT, DINT, REAL, REAL24 Error output (Error): DINT

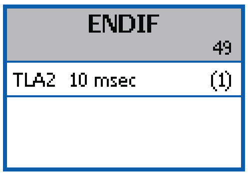
<table>
<tr><td colspan="2">PARWR
(10080)</td></tr>
<tr><td>Illustration</td><td>
</td></tr>
<tr><td>Execution time</td><td>14.50 µs</td></tr>
<tr><td>Operation</td><td>The input value (IN) is written to the defined parameter (Group and Index).
The new parameter value is stored to the flash memory if the store input (Store) is 1. Note: Cyclic parameter value storing can damage the memory unit. Parameter values should be stored only when necessary.
Error codes are indicated by the error output (Error) as follows:
<table>
<tr><th>Error code</th><th>Description</th></tr>
<tr><td>0</td><td>No error</td></tr>
<tr><td><> 0</td><td>Error</td></tr>
</table></td></tr>
<tr><td>Inputs</td><td>Input (IN): DINT
Parameter group input (Group): DINT
Parameter index input (Index): DINT
Store input (Store): Boolean</td></tr>
<tr><td>Outputs</td><td>Error output (Error): DINT</td></tr>
</table>

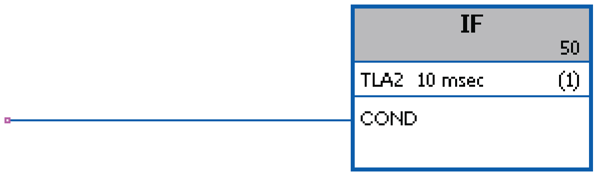
Program structure

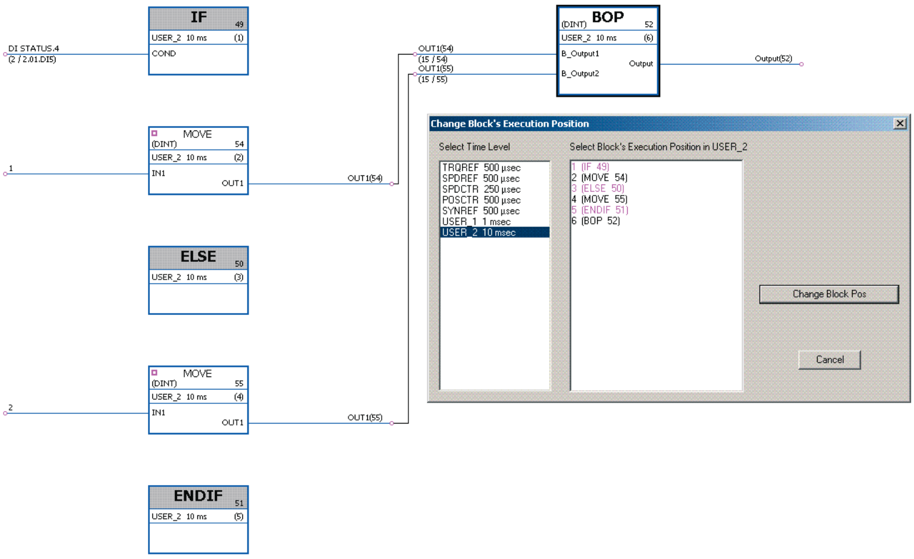
BOP (10105)	
Illustration	
Execution time	-
Operation	The BOP (Bundle OutPut) block collects the outputs of several different sources. The sources are connected to the B_Output pins. The B_Output pin that changed last is relayed to the Output pin. The block is intended for use with conditional IF-ENDIF structures. See the example under the *IF* block.
Inputs	Values from different conditional branches (B_Output1…B_OutputN): INT, DINT, Boolean, REAL, REAL24
Outputs	Output from currently active branch of a IF-ELSEIF structure or latest updated input value (Output): INT, DINT, Boolean, REAL, REAL24

ELSE	
Illustration	
Execution time	-
Operation	See the description of the *IF* block.
Inputs	-
Outputs	-

ELSEIF	
Illustration	
Execution time	-
Operation	See the description of the *IF* block.
Inputs	Input (COND): Boolean
Outputs	-

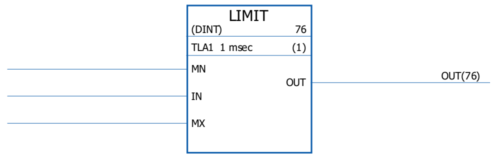
ENDIF	
Illustration	
Execution time	-
Operation	See the description of the *IF* block.
Inputs	-
Outputs	-

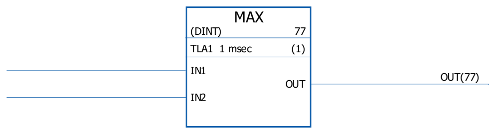
IF (10103)	
Illustration	
Execution time	-

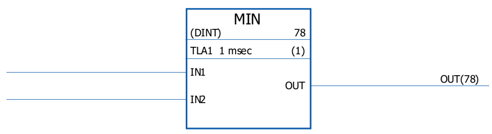
<table>
<tr><td>Operation</td><td>
The IF, ELSE, ELSEIF and ENDIF blocks define, by Boolean logic, which parts of the application program are executed.
If the condition input (COND) is true, the blocks between the IF block and the next ELSEIF, ELSE or ENDIF block (in execution order) are run. If the condition input (COND) is false, the blocks between the IF block and the next ELSEIF, ELSE or ENDIF block are skipped.
The outputs of the “branches” are collected and selected by using the BOP block.
Example:
Bit 4 of 02.01 DI status (digital input DI5) controls the branching of the application program. If the input is 0, the blocks between the IF and ELSE blocks are skipped but the blocks between ELSE and ENDIF are run. If the input is 1, the blocks between IF and ELSE are run. The program execution then jumps to the block that follows ENDIF, which is a BOP. The BOP block outputs the value from the branch that was executed. If the digital input is 0, the BOP block output is 2; if the digital input is 1, the BOP block output is 1.


</td></tr>
<tr><td>Inputs</td><td>Input (COND): Boolean</td></tr>
<tr><td>Outputs</td><td>-</td></tr>
</table>

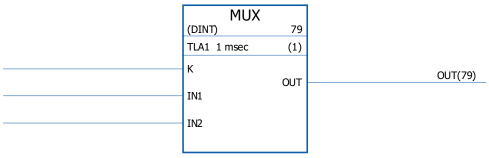
Selection

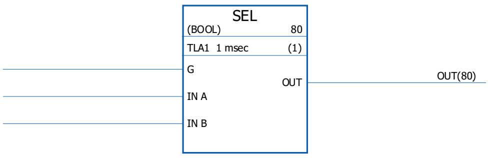
LIMIT (10052)	
Illustration	
Execution time	0.53 µs
Operation	The output (OUT) is the limited input (IN) value. Input is limited according to the minimum (MN) and maximum (MX) values.
Inputs	The input data type is selected by the user. Minimum input limit (MN): INT, DINT, REAL, REAL24 Input (IN): INT, DINT, REAL, REAL24 Maximum input limit (MX): INT, DINT, REAL, REAL24
Outputs	Output (OUT): INT, DINT, REAL, REAL24

MAX (10053)	
Illustration	
Execution time	0.81 µs (when two inputs are used) + 0.53 µs (for every additional input). When all inputs are used, the execution time is 16.73 µs.
Operation	The output (OUT) is the highest input value (IN).
Inputs	The input data type and the number of inputs (2…32) are selected by the user. Input (IN1…IN32): INT, DINT, REAL, REAL24
Outputs	Output (OUT): INT, DINT, REAL, REAL24

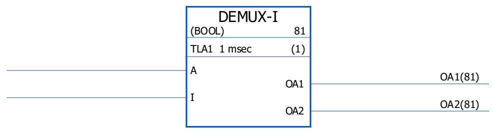
MIN (10054)	
Illustration	
Execution time	0.81 µs (when two inputs are used) + 0.52 µs (for every additional input). When all inputs are used, the execution time is 16.50 µs.
Operation	The output (OUT) is the lowest input value (IN).
Inputs	The input data type and the number of inputs (2…32) are selected by the user. Input (IN1…IN32): INT, DINT, REAL, REAL24

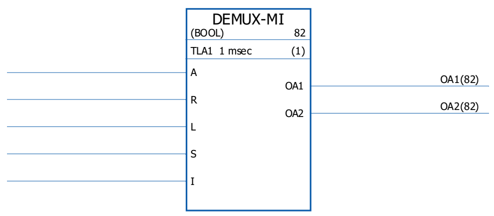
Outputs	Output (OUT): INT, DINT, REAL, REAL24

MUX (10055)	
Illustration	
Execution time	0.70 µs
Operation	The value of an input (IN) selected by the address input (K) is stored to the output (OUT). If the address input is 0, negative or exceeds the number of the inputs, the output is 0.
Inputs	The input data type and number of inputs (2…32) are selected by the user. Address input (K): DINT Input (IN1…IN32): INT, DINT, REAL, REAL24
Outputs	Output (OUT): INT, DINT, REAL, REAL24

SEL (10056)	
Illustration	
Execution time	1.53 µs
Operation	The output (OUT) is the value of the input (IN) selected by the selection input (G). If G = 0: OUT = IN A. If G = 1: OUT = IN B.
Inputs	The input data type is selected by the user. Selection input (G): Boolean Input (IN A, IN B): Boolean, INT, DINT, REAL, REAL24
Outputs	Output (OUT): Boolean, INT, DINT, REAL, REAL24

Switch & Demux

DEMUX-I (10061)	
Illustration	
Execution time	1.38 µs (when two outputs are used) + 0.30 µs (for every additional output). When all outputs are used, the execution time is 10.38 µs.
Operation	Input (I) value is stored to the output (OA1…OA32) selected by the address input (A). All other outputs are 0. If the address input is 0, negative or exceeds the number of the outputs, all outputs are 0.
Inputs	The input data type is selected by the user. Address input (A): DINT Input (I): INT, DINT, Boolean, REAL, REAL24
Outputs	The number of the output channels (1…32) is selected by the user. Output (OA1…OA32): INT, DINT, REAL, REAL24, Boolean

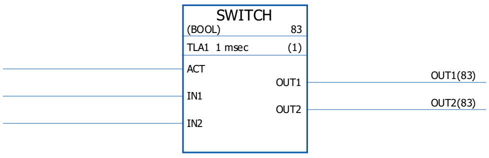
DEMUX-MI (10062)	
Illustration	
Execution time	0.99 µs (when two outputs are used) + 0.25 µs (for every additional output). When all outputs are used, the execution time is 8.4 µs.
Operation	The input (I) value is stored to the output (OA1…OA32) selected by the address input (A) if the load input (L) or the set input (S) is 1. When the load input is set to 1, the input (I) value is stored to the output only once. When the set input is set to 1, the input (I) value is stored to the output every time the block is executed. The set input overrides the load input. If the reset input (R) is 1, all connected outputs are 0. If the address input is 0, negative or exceeds the number of the outputs, all outputs are 0. Example:

S	L	R	A	I	OA1	OA2	OA3	OA4
1	0	0	2	150	0	150	0	0
0	0	0	2	120	0	150	0	0
0	1	0	3	100	0	150	100	0
1	0	0	1	200	200	150	100	0
1	1	0	4	250	200	150	100	250
1	1	1	2	300	0	0	0	0

Inputs	The input data type is selected by the user. Address input (A): DINT Reset input (R): Boolean Load input (L): Boolean Set input (S): Boolean Input (I): DINT, INT, REAL, REAL24, Boolean
Outputs	The number of the output channels (1…32) is selected by the user. Output (OA1…OA32): DINT, INT, REAL, REAL24, Boolean

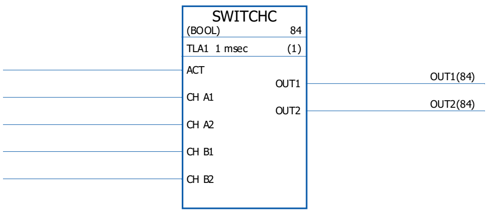
SWITCH

(10063)

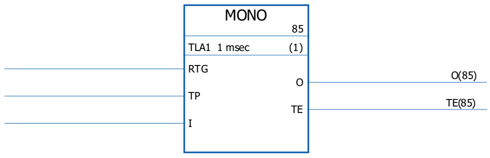
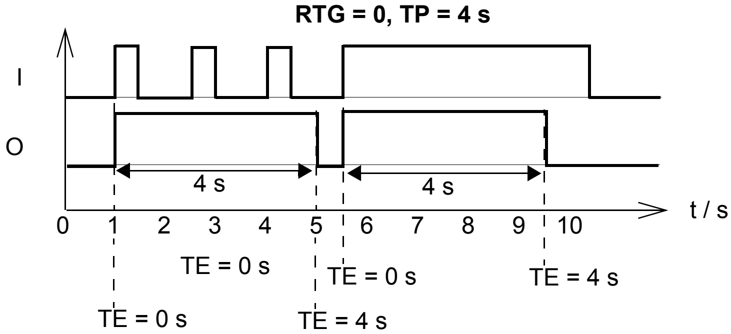
Illustration	
Execution time	0.68 µs (when two inputs are used) + 0.50 µs (for every additional input). When all inputs are used, the execution time is 15.80 µs.
Operation	The output (OUT) is equal to the corresponding input (IN) if the activate input (ACT) is 1. Otherwise the output is 0.
Inputs	The input data type and the number of inputs (1…32) are selected by the user. Activate input (ACT): Boolean Input (IN1…IN32): INT, DINT, REAL, REAL24, Boolean
Outputs	Output (OUT1…OUT32): INT, DINT, REAL, REAL24, Boolean

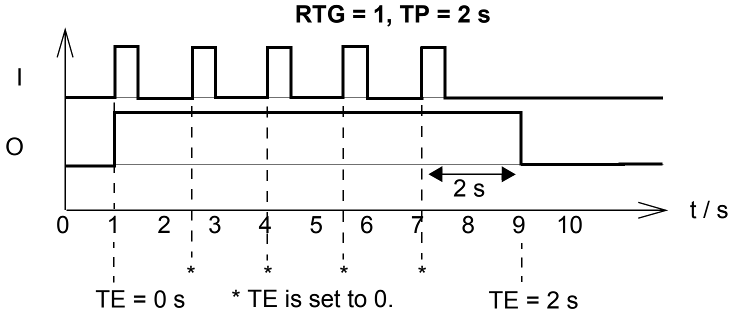
SWITCHC

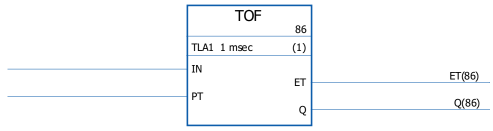
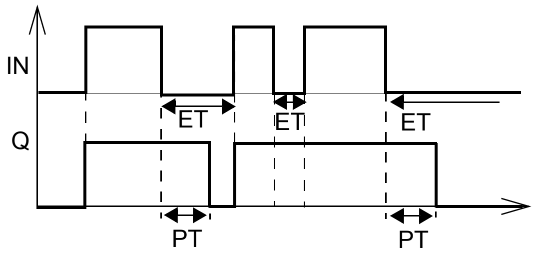
(10064)

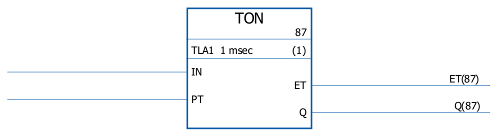
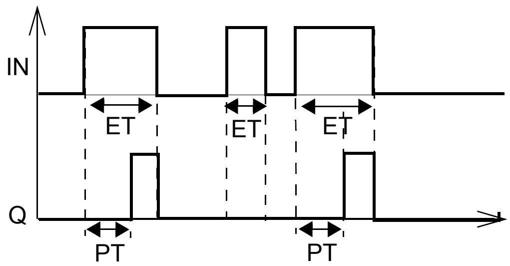
Illustration	
Execution time	1.53 µs (when two inputs are used) + 0.73 µs (for every additional input). When all inputs are used, the execution time is 23.31 µs.
Operation	The output (OUT) is equal to the corresponding channel A input (CH A1…32) if the activate input (ACT) is 0. The output is equal to the corresponding channel B input (CH B1…32) if the activate input (ACT) is 1.
Inputs	The input data type and the number of inputs (1…32) are selected by the user. Activate input (ACT): Boolean Input (CH A1…CH A32, CH B1…CH B32): INT, DINT, REAL, REAL24, Boolean
Outputs	Output (OUT1…OUT32): INT, DINT, REAL, REAL24, Boolean

Timers

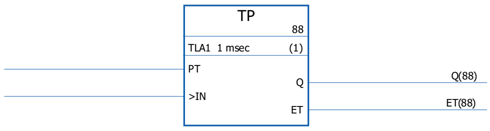
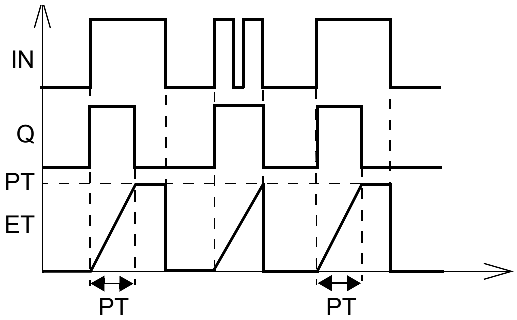
<table>
<tr><th colspan="2">MONO
(10057)</th></tr>
<tr><td>Illustration</td><td>
</td></tr>
<tr><td>Execution time</td><td>1.46 µs</td></tr>
<tr><td>Operation</td><td>The output (O) is set to 1 and the timer is started, if the input (I) is set to 1. The output is reset to 0 when the time defined by the time pulse input (TP) has elapsed. Elapsed time (TE) count starts when the output is set to 1 and stops when the output is set to 0.
If RTG is 0, a new input pulse during the time defined by TP has no effect on the function. The function can be restarted only after the time defined by TP has elapsed.
If RTG is 1, a new input pulse during the time defined by TP restarts the timer and sets the elapsed time (TE) to 0.
Example 1: MONO is not re-triggable, i.e. RTG = 0.
RTG = 0, TP = 4 s


Example 2: MONO is re-triggable, i.e. RTG = 1.
RTG = 1, TP = 2 s

</td></tr>
<tr><td>Inputs</td><td>Re-trigger input (RTG): Boolean
Time pulse input (TP): DINT (1 = µs)
Input (I): Boolean</td></tr>
<tr><td>Outputs</td><td>Output (O): Boolean
Time elapsed output (TE): DINT (1 = 1 µs)</td></tr>
</table>

TOF (10058)	
Illustration	
Execution time	1.10 µs
Operation	The output (Q) is set to 1, when the input (IN) is set to 1. The output is reset to zero when the input has been 0 for a time defined by the pulse time input (PT). Elapsed time count (ET) starts when the input is set to 0 and stops when the input is set to 1. Example: 
Inputs	Input (IN): Boolean Pulse time input (PT): DINT (1 = 1 µs)
Outputs	Elapsed time output (ET): DINT (1 = 1 µs) Output (Q): Boolean

TON (10059)	
Illustration	
Execution time	1.22 µs
Operation	The output (Q) is set to 1 when the input (IN) has been 1 for a time defined by the pulse time input (PT). The output is set to 0, when the input is set to 0. Elapsed time count (ET) starts when the input is set to 1 and stops when the input is set to 0. Example: 

Inputs	Input (IN): Boolean Pulse time input (PT): DINT (1 = 1 µs)
Outputs	Elapsed time output (ET): DINT (1 = 1 µs) Output (Q): Boolean

TP (10060)	
Illustration	
Execution time	1.46 µs
Operation	The output (Q) is set to 1 when the input (IN) is set to 1. The output is set to 0, when it has been 1 for a time defined by the pulse time input (PT). Elapsed time count (ET) starts when the input is set to 1 and stops when the input is set to 0. 
Inputs	Pulse time input (PT): DINT (1 = 1 µs) Input (IN): Boolean
Outputs	Output (Q): Boolean Elapsed time output (ET): DINT (1 = 1 µs)

5

Examples of using standard function blocks

What this chapter contains

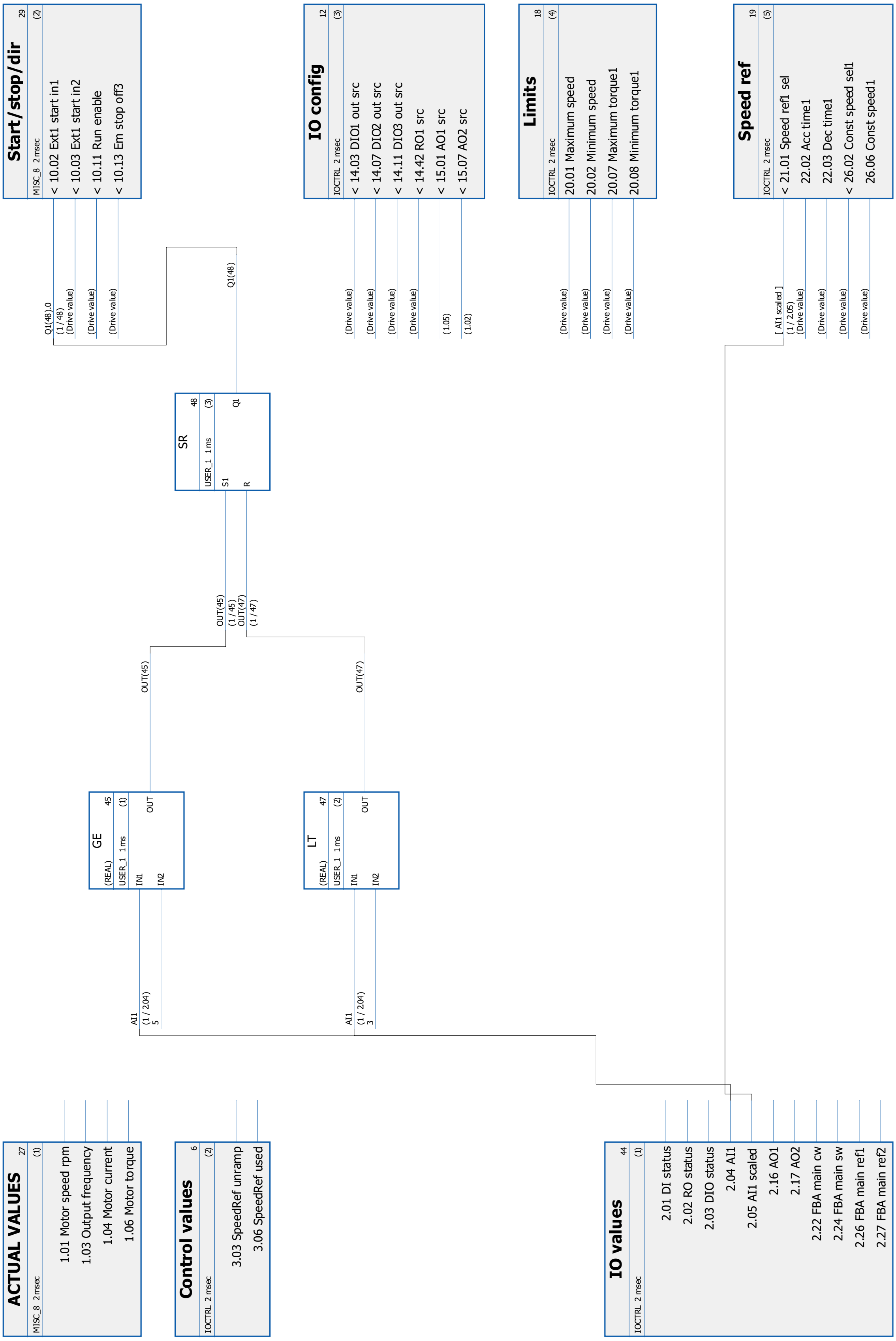
This chapter contains examples of using standard function blocks for
- start/stop
- relay output and digital input/output control
- drive-to-drive communication.

Start/stop using analog input

This example presents an application program, where
- the speed reference is given via analog input AI1
- the drive starts when AI1 is higher than 5 mA
- the drive stops when AI1 is lower than 3 mA.

Additional information

- Actual signal *02.04 AI1* displays AI1 as measured.
- The program is executed at the dedicated time level of 1 ms.


Start/stop/dir
MISC_8 2 msec 29 (2)
< 10.02 Ext1 start in1
< 10.03 Ext1 start in2
< 10.11 Run enable
< 10.13 Em stop off3
IO config
IOCTRL 2 msec 12 (3)
< 14.03 DIO1 out src
< 14.07 DIO2 out src
< 14.11 DIO3 out src
< 14.42 RO1 src
< 15.01 AO1 src
< 15.07 AO2 src
Limits
IOCTRL 2 msec 18 (4)
20.01 Maximum speed
20.02 Minimum speed
20.07 Maximum torque1
20.08 Minimum torque1
Speed ref
IOCTRL 2 msec 19 (5)
< 21.01 Speed ref1 sel
22.02 Acc time1
22.03 Dec time1
< 26.02 Const speed sel1
26.06 Const speed1
Q1(48).0
(1 / 48)
(Drive value)
Q1(48)
(1.05)
(1.02)
[AI1 scaled]
(1 / 2.05)
SR
USER_1 1 ms 48 (3)
S1
R
Q1
OUT(45)
(1 / 45)
OUT(47)
(1 / 47)
GE
(REAL) USER_1 1 ms 45 (1)
IN1
IN2
OUT
LT
(REAL) USER_1 1 ms 47 (2)
IN1
IN2
OUT
AI1
(1 / 2.04)
5
AI1
(1 / 2.04)
3
ACTUAL VALUES
MISC_8 2 msec 27 (1)
1.01 Motor speed rpm
1.03 Output frequency
1.04 Motor current
1.06 Motor torque
Control values
IOCTRL 2 msec 6 (2)
3.03 SpeedRef unramp
3.06 SpeedRef used
IO values
IOCTRL 2 msec 44 (1)
2.01 DI status
2.02 RO status
2.03 DIO status
2.04 AI1
2.05 AI1 scaled
2.16 AO1
2.17 AO2
2.22 FBA main cw
2.24 FBA main sw
2.26 FBA main ref1
2.27 FBA main ref2

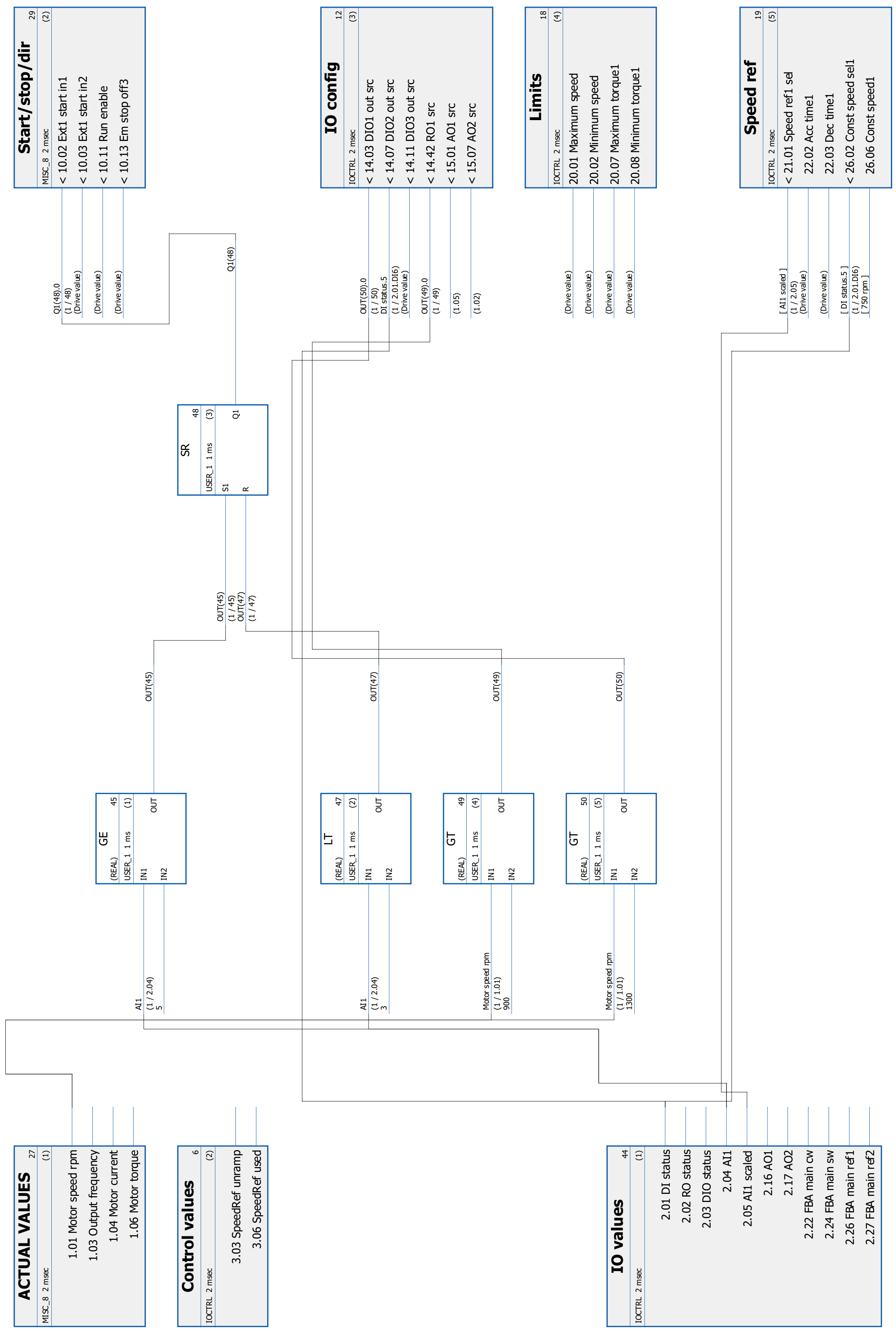
Relay output and digital input/output control

This example comprises the program presented in the previous example (page *91*) as well as the following additions:

- Relay output RO1 is activated when the speed is higher than 900 rpm.
- Digital input/output DIO1 is activated when the speed is higher than 1300 rpm.
- Digital input/output DIO2 is activated when constant speed 1 (750 rpm) is activated by digital input DI6.

Additional information

- Actual signal *02.04 AI1* displays AI1 as measured.
- Actual signal *02.01 DI status* bit 5 displays DI6.
- Actual signal *01.01 Motor speed rpm* displays the speed.
- The program is executed at the dedicated time level of 1 ms.

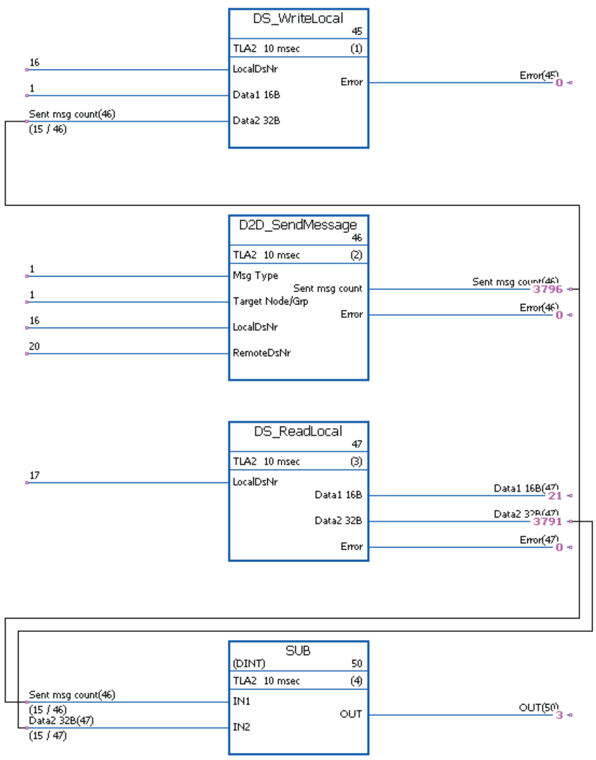

Start/stop/dir
MISC_8 2 msec
< 10.02 Ext1 start in1
< 10.03 Ext1 start in2
< 10.11 Run enable
< 10.13 Em stop off3
IO config
IOCTRL 2 msec
< 14.03 DIO1 out src
< 14.07 DIO2 out src
< 14.11 DIO3 out src
< 14.42 RO1 src
< 15.01 AO1 src
< 15.07 AO2 src
Limits
IOCTRL 2 msec
20.01 Maximum speed
20.02 Minimum speed
20.07 Maximum torque1
20.08 Minimum torque1
Speed ref
IOCTRL 2 msec
< 21.01 Speed ref1 sel
22.02 Acc time1
22.03 Dec time1
< 26.02 Const speed sel1
26.06 Const speed1
SR
USER_1 1 ms
S1
R
Q1
GE
LT
GT
GT
(REAL)
USER_1 1 ms
IN1
IN2
OUT
ACTUAL VALUES
MISC_8 2 msec
1.01 Motor speed rpm
1.03 Output frequency
1.04 Motor current
1.06 Motor torque
Control values
IOCTRL 2 msec
3.03 SpeedRef unramp
3.06 SpeedRef used
IO values
IOCTRL 2 msec
2.01 DI status
2.02 RO status
2.03 DIO status
2.04 AI1
2.05 AI1 scaled
2.16 AO1
2.17 AO2
2.22 FBA main cw
2.24 FBA main sw
2.26 FBA main ref1
2.27 FBA main ref2

Drive-to-drive communication (ACS850 only)

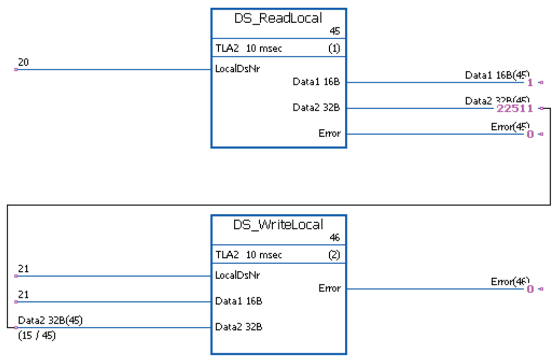
For the descriptions of the drive-to-drive standard function blocks, see section *Communication (ACS850 only)* on page 31.

Example of master point-to-point messaging

Master



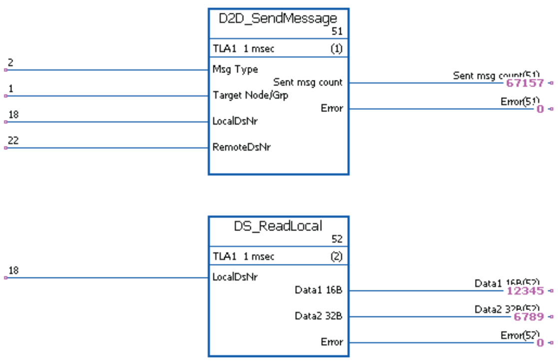
Follower (node 1)



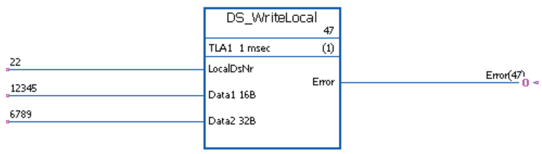
1. The master sends a constant (1) and the value of the message counter into follower dataset 20. Data is prepared to and sent from dataset 16.
2. The follower sends the received counter value and a constant (21) as a reply to the master.
3. The master calculates the difference of the latest message number and received data.

Example of read remote messaging

Master



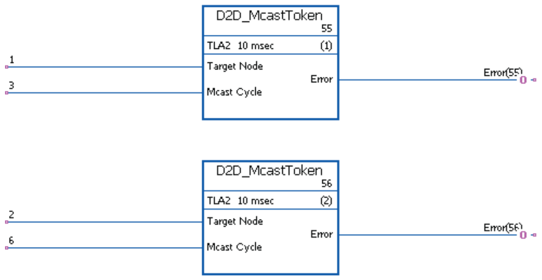
Follower (node 1)



1. The master reads the contents of the follower dataset 22 into its own dataset 18. Data is accessed using the *DS_Read_Local* block.
2. In the follower, constant data is prepared into dataset 22.

Example of releasing tokens for follower-to-follower communication

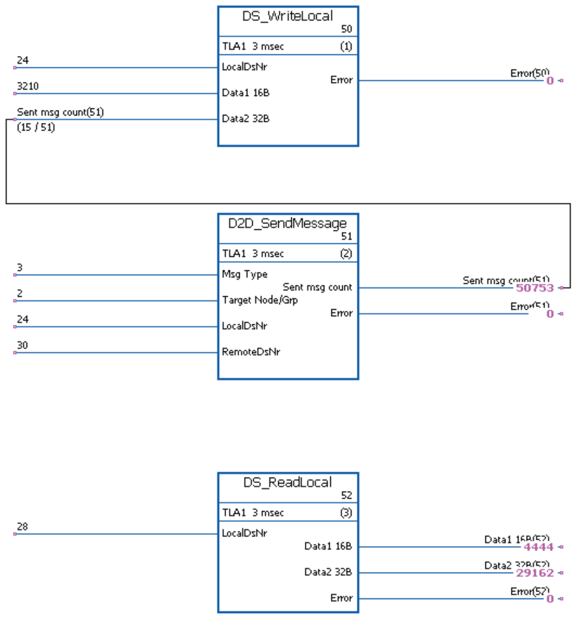
Master



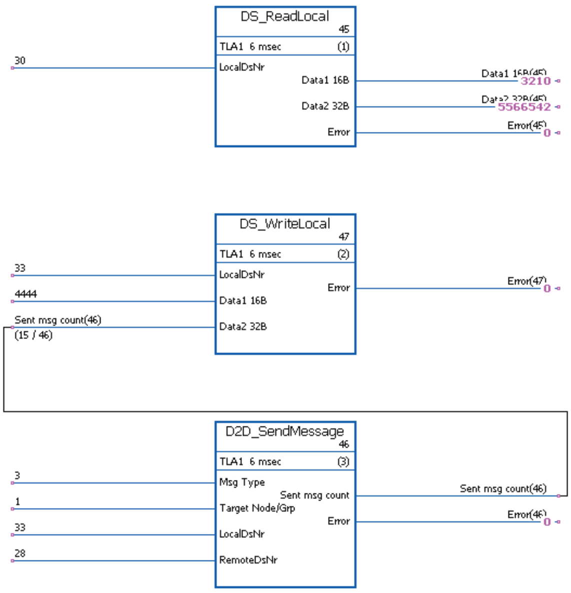
1. This drive-to-drive link consists of three drives (master and two followers).
2. The master operates as a “chairman”. Follower 1 (node 1) is allowed to send one message every 3 milliseconds. Follower 2 (node 2) is allowed to send one message every 6 milliseconds.

Example of follower point-to-point messaging

Follower 1 (node 1)



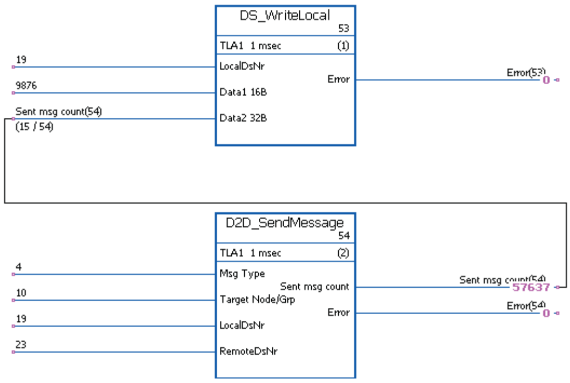
Follower 2 (node 2)



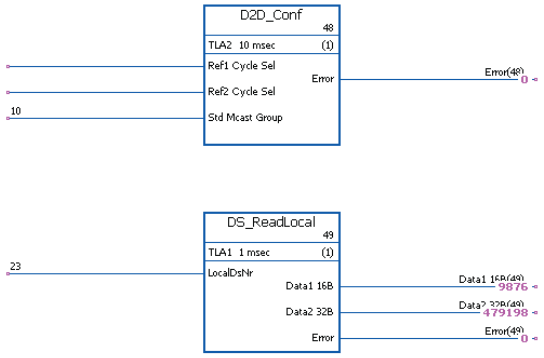
1. Follower 1 writes local dataset 24 to follower 2 dataset 30 (3 ms interval).
2. Follower 2 writes local dataset 33 to follower 1 dataset 28 (6 ms interval).
3. In addition, both followers read received data from local datasets.

Example of standard multicast messaging

Master



Follower(s) in Std Mcast Group 10

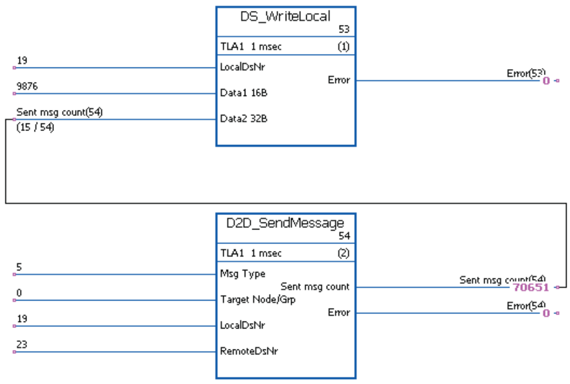


1. The master sends a constant (9876) and the value of the message counter to all followers in standard multicast group 10. The data is prepared into and sent from master dataset 19 to follower dataset 23.
2. Received data is read from dataset 23 of the receiving followers.

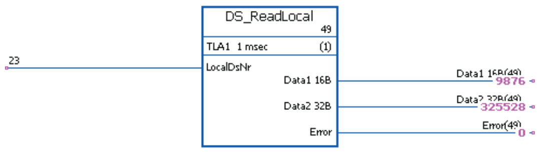
Note: The example application shown for Master above also applies to the sending follower in standard follower-to-follower multicasting.

Example of broadcast messaging

Master



Follower(s)



1. The master sends a constant (9876) and the value of the message counter to all followers. The data is prepared into and sent from master dataset 19 to follower dataset 23.
2. Received data is read from dataset 23 of the followers.

Note: The example application shown for Master above also applies to the sending follower in follower-to-follower broadcasting.

Further information

Product and service inquiries

Address any inquiries about the product to your local ABB representative, quoting the type designation and serial number of the unit in question. A listing of ABB sales, support and service contacts can be found by navigating to www.abb.com/drives and selecting *Sales, Support and Service network*.

Product training

For information on ABB product training, navigate to www.abb.com/drives and select *Training courses*.

Providing feedback on ABB Drives manuals

Your comments on our manuals are welcome. Go to www.abb.com/drives and select *Document Library – Manuals feedback form (LV AC drives)*.

Document library on the Internet

You can find manuals and other product documents in PDF format on the Internet. Go to www.abb.com/drives and select *Document Library.* You can browse the library or enter selection criteria, for example a document code, in the search field.

Contact us

ABB Oy
Drives
P.O. Box 184
FI-00381 HELSINKI
FINLAND
Telephone +358 10 22 11
Fax +358 10 22 22681
www.abb.com/drives

ABB Inc.
Automation Technologies
Drives & Motors
16250 West Glendale Drive
New Berlin, WI 53151
USA
Telephone 262 785-3200
1-800-HELP-365
Fax 262 780-5135
www.abb.com/drives

ABB Beijing Drive Systems Co. Ltd.
No. 1, Block D, A-10 Jiuxianqiao Beilu
Chaoyang District
Beijing, P.R. China, 100015
Telephone +86 10 5821 7788
Fax +86 10 5821 7618
www.abb.com/drives

3AUA0000078664 Rev B EN 2011-11-28