Library of General Functions (LGF) for SIMATIC S7-1200/S7-1500

STEP 7 Basic/Professional V16 (TIA PORTAL)

https://support.industry.siemens.com/cs/ww/en/view/109479728

Siemens Industry Online Support

Legal information

Use of application examples

Application examples illustrate the solution of automation tasks through an interaction of several components in the form of text, graphics and/or software modules. The application examples are a free service by Siemens AG and/or a subsidiary of Siemens AG ("Siemens"). They are non-binding and make no claim to completeness or functionality regarding configuration and equipment. The application examples merely offer help with typical tasks; they do not constitute customer-specific solutions. You yourself are responsible for the proper and safe operation of the products in accordance with applicable regulations and must also check the function of the respective application example and customize it for your system.
Siemens grants you the non-exclusive, non-sublicensable and non-transferable right to have the application examples used by technically trained personnel. Any change to the application examples is your responsibility. Sharing the application examples with third parties or copying the application examples or excerpts thereof is permitted only in combination with your own products. The application examples are not required to undergo the customary tests and quality inspections of a chargeable product; they may have functional and performance defects as well as errors. It is your responsibility to use them in such a manner that any malfunctions that may occur do not result in property damage or injury to persons.

Disclaimer of liability

Siemens shall not assume any liability, for any legal reason whatsoever, including, without limitation, liability for the usability, availability, completeness and freedom from defects of the application examples as well as for related information, configuration and performance data and any damage caused thereby. This shall not apply in cases of mandatory liability, for example under the German Product Liability Act, or in cases of intent, gross negligence, or culpable loss of life, bodily injury or damage to health, non-compliance with a guarantee, fraudulent non-disclosure of a defect, or culpable breach of material contractual obligations. Claims for damages arising from a breach of material contractual obligations shall however be limited to the foreseeable damage typical of the type of agreement, unless liability arises from intent or gross negligence or is based on loss of life, bodily injury or damage to health. The foregoing provisions do not imply any change in the burden of proof to your detriment. You shall indemnify Siemens against existing or future claims of third parties in this connection except where Siemens is mandatorily liable.
By using the application examples you acknowledge that Siemens cannot be held liable for any damage beyond the liability provisions described.

Other information

Siemens reserves the right to make changes to the application examples at any time without notice. In case of discrepancies between the suggestions in the application examples and other Siemens publications such as catalogs, the content of the other documentation shall have precedence.
The Siemens terms of use (https://support.industry.siemens.com) shall also apply.

Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.
In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions constitute one element of such a concept.
Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the Internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.
For additional information on industrial security measures that may be implemented, please visit https://www.siemens.com/industrialsecurity.
Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customer's exposure to cyber threats.
To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed at: https://www.siemens.com/industrialsecurity.

Table of Contents

1 Library Overview

1.1 General Information

TIA Portal has an extensive number of ready-to-use commands (mathematical functions, times, counters, etc.). In addition, there are other useful basic functions.

These functions are provided in the form of a library and they can be used freely. The finished functions are freely customizable and can therefore be used universally.

The library described here is versioned and it will be continuously extended. You can find information on versioning in Chapter 4.3.1 Versioning.

1.2 Hardware and Software Requirements

Requirements for this library

In order to use the functionality of the library described here, the following hardware and software requirements must be met.

Hardware

All blocks (FB, FC, DB,…) in the library can be used universally with the following controllers:

- SIMATIC S7-1200 and SIMATIC S7-1200 F product family (from firmware V4.2)
- SIMATIC S7-1500 and SIMATIC S7-1500 F product family (from firmware V2.0)
- Simulation with SIMATIC S7-PLCSIM (from V14)

Software

- SIMATIC STEP 7 Basic/Professional (TIA PORTAL)

Note

In general, it is possible to open a library with STEP 7 Basic, although STEP 7 Professional elements (e.g. SIMATIC S7-1500 controller) are included. In this case you will be informed with a message when opening the library.

All elements (types and copy templates) can be used if they are supported by the hardware installed in the TIA Portal.

If you try to copy elements with STEP 7 Basic from the library that are not supported (e.g. SIMATIC S7-1500 controller), an error message is displayed.

2 Working with the Library

2.1 General Information

All blocks in the “LGF” library can be used freely in conjunction with SIMATIC S7-1200 and SIMATIC S7-1500 controllers.

Most of the blocks are stored as types in the library. This means that the blocks are versioned and can thus use all advantages.

- Central update function for library elements
- Versioning of library elements

Note You can find information on the general handling of libraries in the Guideline for Library Handling

https://support.industry.siemens.com/cs/ww/en/view/109747503

and in the Programming Guideline for S7-1200/1500 in the chapter “Libraries”.

https://support.industry.siemens.com/cs/ww/en/view/81318674

Note All blocks in the LGF were created in accordance with the Programming Style Guide.

https://support.industry.siemens.com/cs/ww/en/view/81318674

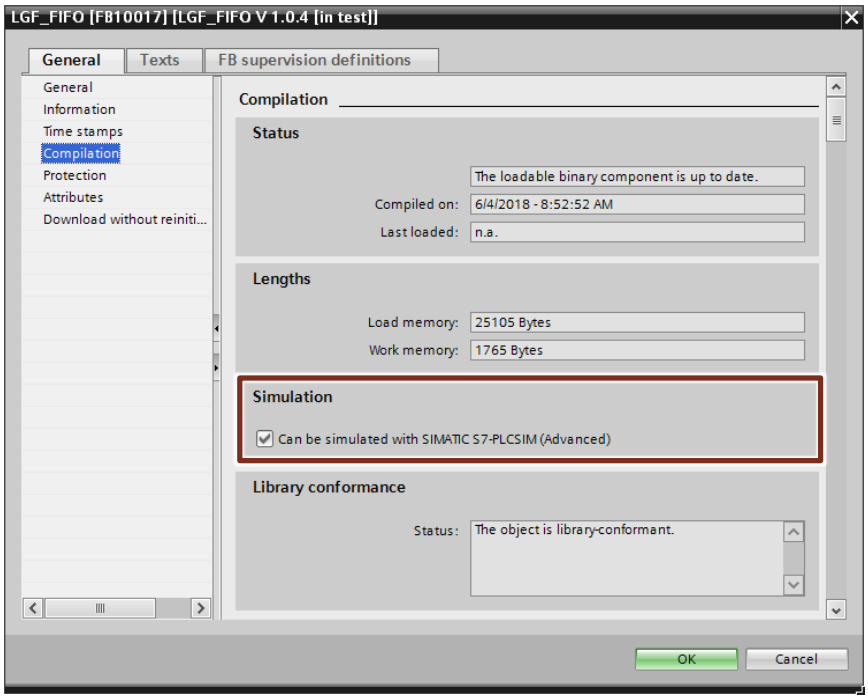
For more information on libraries, visit the TIA Portal:

- Libraries in the TIA Portal
  https://support.industry.siemens.com/cs/ww/en/view/109738702
- How do you open libraries in STEP 7 (TIA Portal)?
  https://support.industry.siemens.com/cs/ww/en/view/37364723
- Automate in less than 10 minutes TIA Portal: Time Savers – Global libraries
  https://support.industry.siemens.com/cs/ww/en/view/78529894
- Which elements from STEP 7 (TIA Portal) can be stored in a library as a type or as a copy template?
  https://support.industry.siemens.com/cs/ww/en/view/109476862
- How can you automatically open a global library when starting the TIA Portal V13 or higher and use it e.g. as a corporate library?
  https://support.industry.siemens.com/cs/ww/en/view/100451450
- Library with PLC data types for IO module / technology modules and PROFIdrive drives (LPD)
  https://support.industry.siemens.com/cs/ww/en/view/109482396

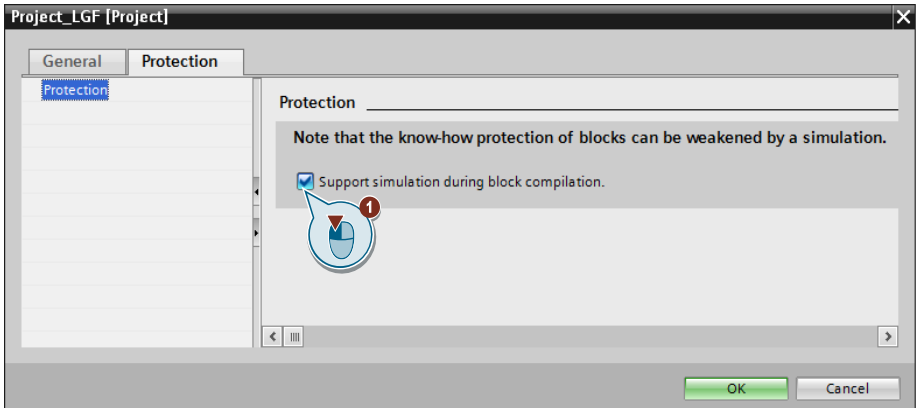
2.2 Simulability with SIMATIC S7-PLCSIM Advanced

Simulation with SIMATIC S7-PLCSIM Advanced is already activated in the properties of the LGF blocks.

Figure 2-1: Block properties



After compilation with SIMATIC S7-PLCSIM Advanced, proceed as follows so that the blocks can be simulated:

1. Open the properties of your project and enable the option “Support simulation during block compilation” in the “Protection” tab.
2. 

Note Blocks with activated simulatability take up more memory space in the PLC.

2.3 User-Defined Documentation (User Help)

In order to explain the principle of operation and use of the blocks to users of the LGF library, user-defined documentation has been created for each block. The user-defined documentation per block is available in German and English as a PDF file. The PDF files are stored in the following directories of the LGF library.

- German: “UserFiles\UserDocumentation\en-DE\Library Types”
- English: “UserFiles\UserDocumentation\en-US\Library Types”

The user-defined documentation for a block can be called up in the task card “Library” and in the library view with the key combination <Shift+F1>. The respective PDF is always opened with the standard program defined in Microsoft Windows.

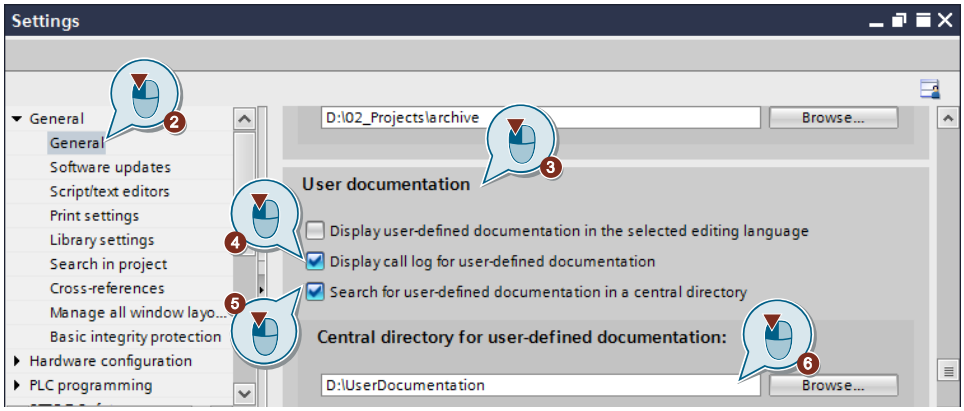
So that the user-defined documentation of the blocks can also be called up in the project navigation, you must copy the directories with the PDF files into the project directory “UserFiles”.

Note For the user-defined documentation, you will need SIMATIC STEP 7 Basic/Professional V15.1, update 1 or higher.

Central directory for user-defined documentation

Alternatively, you can store the user-defined documentation in a central directory for all projects. To define a central storage location for user help, proceed as follows:

1. In the “Extras” menu, select the “Settings” command.
2. Open the area “General > General”.
3. Navigate to the “User documentation” section.
4. Enable the check box “Display call log for user-defined documentation” to display a log of the call-up of the user-defined documentation in the Inspector window.
5. Enable the “Search for user-defined documentation in a central directory” check box to store user-defined documentation in a central directory for projects.
6. In the “Central directory for user-defined documentation” field, specify the path where you want to store cross-project documentation.



Note Do not change the names of the PDF, because the file name must precisely match the name of the object in the TIA Portal.

Note You can find more information on user-defined documentation in the “SIMATIC STEP 7 Basic/Professional V16 and SIMATIC WinCC V16” system manual:

https://support.industry.siemens.com/cs/ww/en/view/109773506/119453612171

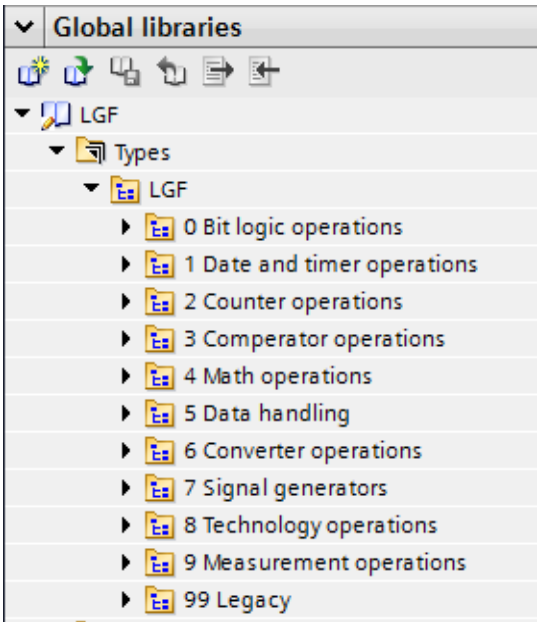
3 Explanation of the Blocks

The chapters below describe all blocks of the library, “Library General Functions”. The chapters have the same structure as the library itself.

All blocks are divided into application areas or categories:

- Bit logic operations
- Date and timer operations
- Counter operations
- Comparator operations
- Math operations
- Data handling
- Converter operations
- Signal generators
- Technology operations
- Measurement operations

Figure 3-1: Global Library (LGF)



3.0 Bit Logic Operations

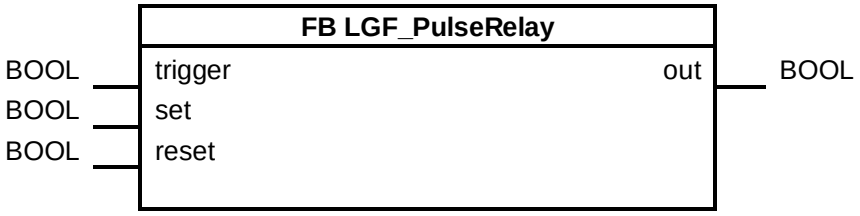
3.0.1 LGF_PulseRelay

Short description

This block corresponds to an impulse relay or a toggle flip-flop including set and reset input.

Block

Figure 3-2: FB LGF_PulseRelay



Input parameters

Table 3-1: Input parameters

Parameters	Data type	Description
trigger	BOOL	Each rising edge changes the boolean value of the output “out”.
set	BOOL	Each rising edge sets the boolean value of the output “out” to “TRUE”.
reset	BOOL	Each rising edge sets the boolean value of the output “out” to “FALSE”.

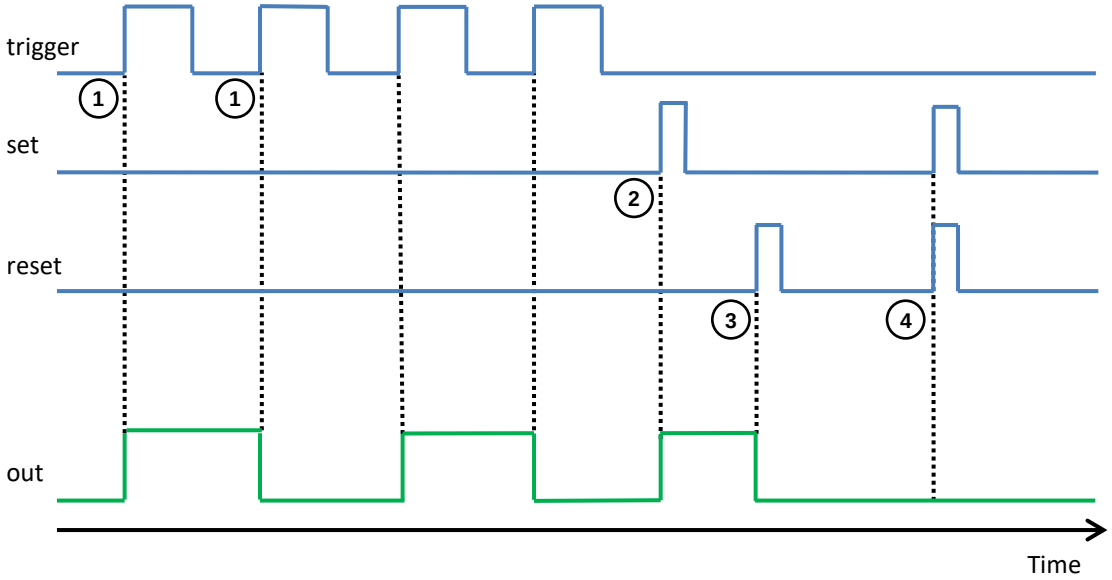
Output parameters

Table 3-2: Output parameters

Parameters	Data type	Description
out	BOOL	Signal output

Principle of operation

Figure 3-3: LGF_PulseRelay Signal diagram



1. Each rising edge of the input “trigger” changes the boolean value of the output “out”.
2. Each rising edge of the input “set” sets the boolean value of the output “out” to “TRUE”.
3. Each rising edge of the input “reset” sets the boolean value of the output “out” to “FALSE”.
4. If the inputs “set” and “reset” are set in the same cycle, the “reset” input has priority.

The block can also be used as a frequency divider. If the input “trigger” is supplied with a fixed frequency, the output “out” delivers half the frequency.

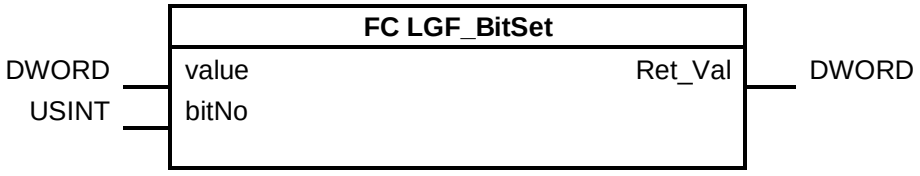
3.0.2 LGF_BitSet

Short description

This block sets a bit at a given position in a variable of the data type DWORD. Alternatively, Word and Byte can be used instead of DWord by converting the passed parameter with, for example, BYTE_TO_DWORD and the result with DWORD_TO_BYTE.

Block

Figure 3-4: FC LGF_BitSet



Input parameters

Table 3-3: Input parameters

Parameters	Data type	Description
value	DWORD	Variable at which the bit is set.
bitNo	USINT	Bit number to be set in the “value” parameter.

Output parameters

Table 3-4: Output parameters

Parameters	Data type	Description
Ret_Val	DWORD	Variable with set bit.

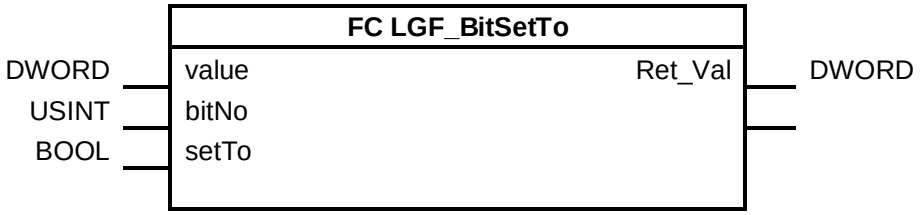
3.0.3 LGF_BitSetTo

Short description

This block sets a bit to TRUE or FALSE at a predefined position in a variable of the data type DWORD. Alternatively, Word and Byte can be used instead of DWord by converting the passed parameter with, for example, BYTE_TO_DWORD and the result with DWORD_TO_BYTE.

Block

Figure 3-5: FC LGF_BitSetTo



Input parameters

Table 3-5: Input parameters

Parameters	Data type	Description
value	DWORD	Variable at which the bit is set.
bitNo	USINT	Bit number to be set in the “value” parameter.
setTo	BOOL	Set bit to FALSE or TRUE

Output parameters

Table 3-6: Output parameters

Parameters	Data type	Description
Ret_Val	DWORD	Variable with set bit.

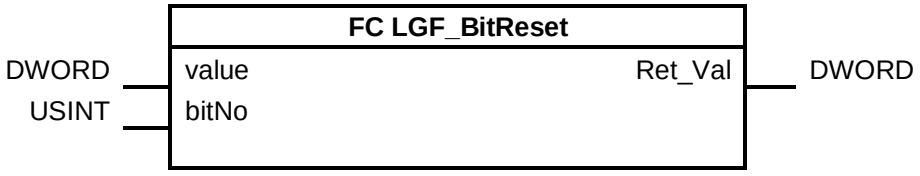
3.0.4 LGF_BitReset

Short description

This block resets a bit at a predefined position in a variable of the data type DWORD. Alternatively, Word and Byte can be used instead of DWord by converting the passed parameter with, for example, BYTE_TO_DWORD and the result with DWORD_TO_BYTE.

Block

Figure 3-6: FC LGF_ BitReset



Input parameters

Table 3-7: Input parameters

Parameters	Data type	Description
value	DWORD	Variable at which the bit is reset.
bitNo	USINT	Bit number to be reset in the “value” parameter.

Output parameters

Table 3-8: Output parameters

Parameters	Data type	Description
Ret_Val	DWORD	Variable with reset bit.

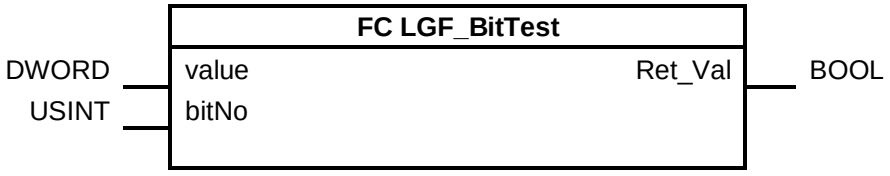
3.0.5 LGF_BitTest

Short description

This block checks whether a bit is TRUE or FALSE at a given position in a variable of the data type DWORD. Alternatively, Word and Byte can be used instead of DWord by converting the passed parameter with, for example, BYTE_TO_DWORD and the result with DWORD_TO_BYTE.

Block

Figure 3-7: FC LGF_BitTest



Input parameters

Table 3-9: Input parameters

Parameters	Data type	Description
value	DWORD	Variable at which the bit is checked.
bitNo	USINT	Bit number to be checked in the “value” parameter.

Output parameters

Table 3-10: Output parameters

Parameters	Data type	Description
Ret_Val	BOOL	Value of the checked bit.

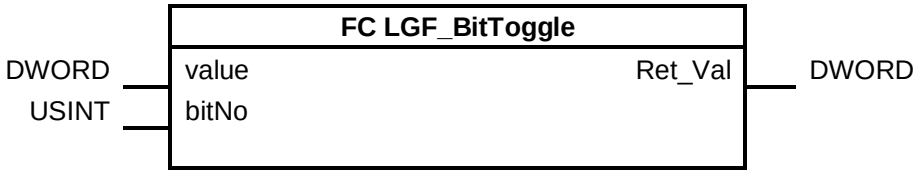
3.0.6 LGF_BitToggle

Short description

This block toggles a bit at a predefined position in a variable of the data type DWORD. Alternatively, Word and Byte can be used instead of DWord by converting the passed parameter with, for example, BYTE_TO_DWORD and the result with DWORD_TO_BYTE.

Block

Figure 3-8: FC LGF_BitToggle



Input parameters

Table 3-11: Input parameters

Parameters	Data type	Description
value	DWORD	Variable at which the bit is toggled.
bitNo	USINT	Bit number to be toggled in the “value” parameter.

Output parameters

Table 3-12: Output parameters

Parameters	Data type	Description
Ret_Val	DWORD	Variable with toggled bit.

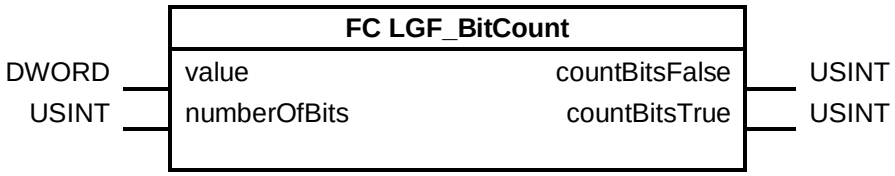
3.0.7 LGF_BitCount

Short description

This block counts the set (TRUE) and unset (FALSE) bits in a variable of the data type DWORD. Alternatively, Word and Byte can be used instead of DWord by converting the passed parameter with, for example, BYTE_TO_DWORD and the result with DWORD_TO_BYTE. The parameter “numberOfBits” defines how many bits are to be counted.

Block

Figure 3-9: FC LGF_BitCount



Input parameters

Table 3-13: Input parameters

Parameters	Data type	Description
value	DWORD	Variable at which the bit is toggled.
numberOfBits	USINT	Number of bits to be counted. Byte = 8, Word = 16, DWORD = 32

Output parameters

Table 3-14: Output parameters

Parameters	Data type	Description
countBitsFalse	USINT	Number of bits not set.
countBitsTrue	USINT	Number of bits set.

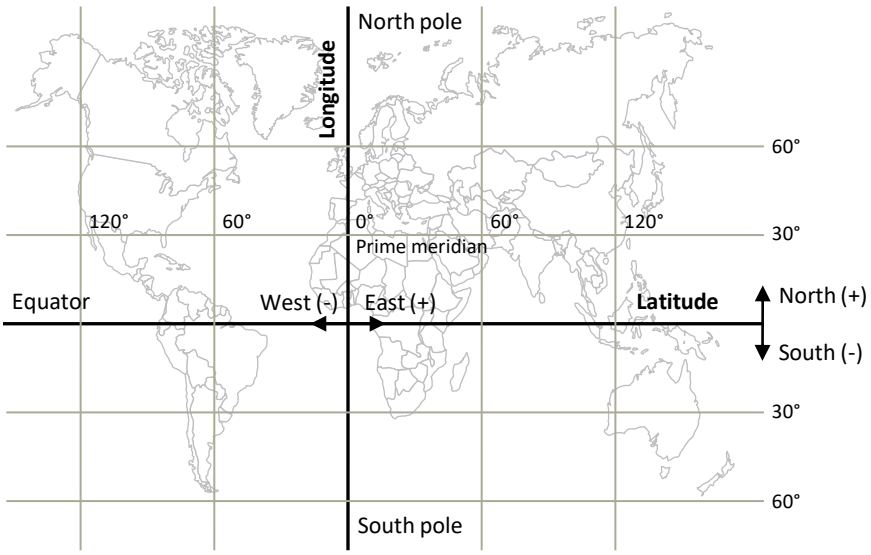
3.1 Date and Timer Operations

3.1.1 LGF_AstroClock

Short description

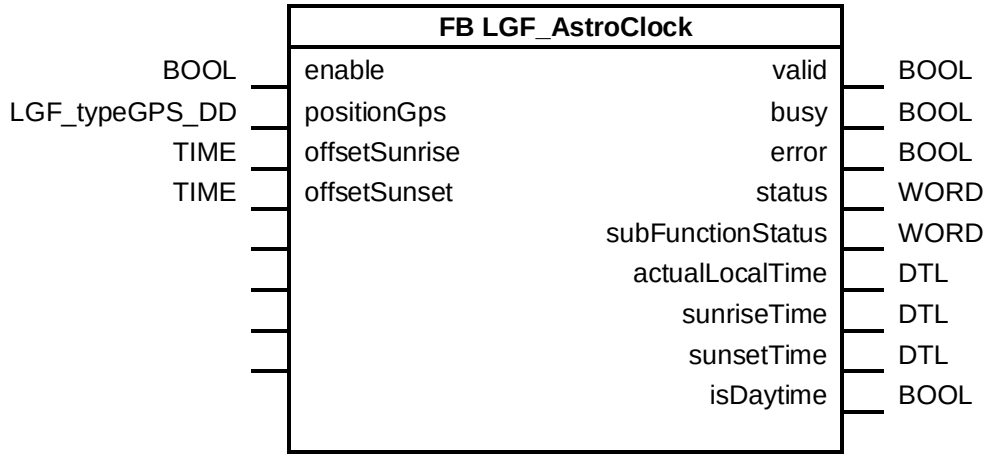
This block calculates the times of sunrise and sunset for a specific place on Earth. The exact position is transferred to the block in the form of geographical coordinates (longitude and latitude).

Figure 3-10: Earth with longitude and latitude



Block

Figure 3-11: FB LGF_AstroClock



Input parameters

Table 3-15: Input parameters

Parameters	Data type	Description
enable	BOOL	TRUE: Activates the functionality of the FB
positionGps	LGF_typeGPS_DD	GPS position to calculate the time of sunrise and sunset
offsetSunrise	TIME	Offset to sunrise
offsetSunset	TIME	Offset to sunset

Output parameters

Table 3-16: Output parameters

Parameters	Data type	Description
valid	BOOL	TRUE: Output values at FB valid
busy	BOOL	TRUE: FB is not finished and new output values can be expected.
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).
subFunctionStatus	WORD	Status or return value of the called FCs and system blocks.
actualLocalTime	DTL	Current time (local time)
sunrise	DTL	Sunrise time (local time)
sunset	DTL	Sunset time (local time)
isDaytime	BOOL	TRUE: If the local time of the controller is between “sunrise” and “sunset”.

Status and error displays

Table 3-17: Status/error codes

status	Meaning	Remedy / notes
16#0000	No error	-
16#7001	Block is being processed	-
16#8204	Wrong Latitude DD value	Check the actual value at the input.
16#8205	Wrong Longitude DD value	Check the actual value at the input.
16#8601	Error in “RD_SYS_T” command.	Check the error code in “subFunctionStatus”
16#8602	Error in “RD_LOC_T” command.	Check the error code in “subFunctionStatus”

Note The status of called commands is output in “subFunctionStatus”. In this case, the output value in “status” indicates which command caused the error. In this case, refer to the TIA Portal Online Help section for information on the respective commands.

LGF_typeGPS_DD data type

Table 3-18: LGF_typeGPS_DD data type

Parameters	Data type	Description
latitude	REAL	Degrees latitude with decimal places (Unit: degree decimal), North = positive; South = negative valid value range [-90.00000..90.00000]
longitude	REAL	Longitude in degrees with decimal places (Unit: degree decimal), East = positive; West = negative valid range [-180.0000..180.0000]

Principle of operation

If processes must run automatically depending on the change between day and night, the function of an astronomical clock is required. Examples of this would be switching outdoor lighting on and off or opening and closing roller shutters.

If these processes are to be executed with a time delay i.e. a defined time before or after sunrise or sunset an offset is required in each case.

Note For precise execution of the function, it must be ensured that system time and local time of the SIMATIC controller are set correctly.

Based on the system time/local time of the SIMATIC controller and the set coordinates, the block calculates the times for sunrise and sunset. The offset times are added to the sunrise and sunset and output on the “sunrise” and “sunset” outputs. If the system time of the SIMATIC controller is between these values, the output “daytime” is set to the value “TRUE”.

Note Since the times for sunrise and sunset change daily, it is possible that the “daytime” output remains permanently on “TRUE” or “FALSE” over a longer period of time:

- with correspondingly large offset values
- for a place on the other side of the Arctic Circle

The input of the GPS coordinate values is checked for valid values. If there are invalid values, an appropriate error code is output to “status” (see Table 3-17).

If there is an invalid coordinate value for a formal parameter, the outputs “sunrise” and “sunset” are set to the value DTL#1970-01-01-00:00:00.

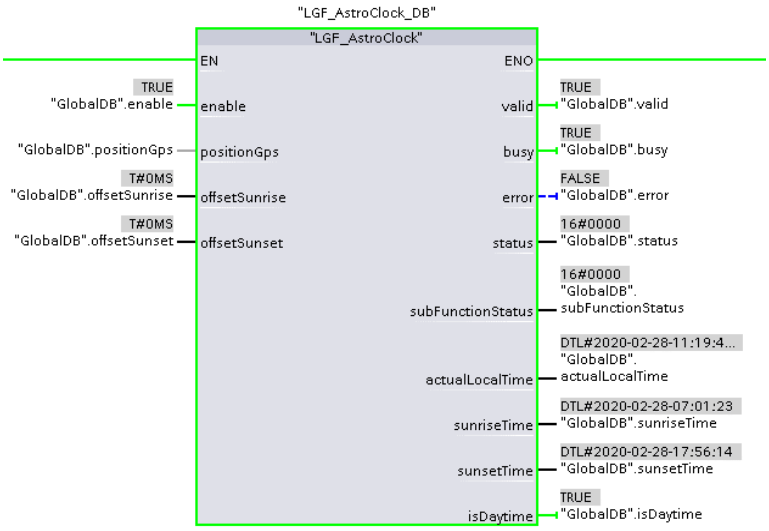
Example

The following example illustrates the block's functionality.

Table 3-19: Geographical coordinates for Nuremberg-Moorenbrunn, date, and system time

Longitude:	+ 11.07675°		
Latitude:	+ 49.45203°		
Date:	02/28/2020	Local time:	11:22:37
		Sonnenaufgang	07:01:23
		Sunset	17:56:14

Figure 3-12: FB LGF_AstroClock, Observation of the block online with the parameters and the actual parameters via the Observation Table



Name	...	Anzeigeformat	Beobachtungswert
"GlobalDB".enable		BOOL	TRUE
"GlobalDB".positionGps.latitude		Gleitpunktzahl	49.45203
"GlobalDB".positionGps.longitude		Gleitpunktzahl	11.07675
"GlobalDB".offsetSunrise		Zeit	T#0MS
"GlobalDB".offsetSunset		Zeit	T#0MS
"GlobalDB".valid		BOOL	TRUE
"GlobalDB".busy		BOOL	TRUE
"GlobalDB".error		BOOL	FALSE
"GlobalDB".status		Hex	16#0000
"GlobalDB".subFunctionStatus		Hex	16#0000
"GlobalDB".actualLocalTime		DATE_AND_TIME	DTL#2020-02-28-11:22:37.220143296
"GlobalDB".sunriseTime		DATE_AND_TIME	DTL#2020-02-28-07:01:23
"GlobalDB".sunsetTime		DATE_AND_TIME	DTL#2020-02-28-17:56:14
"GlobalDB".isDaytime		BOOL	TRUE

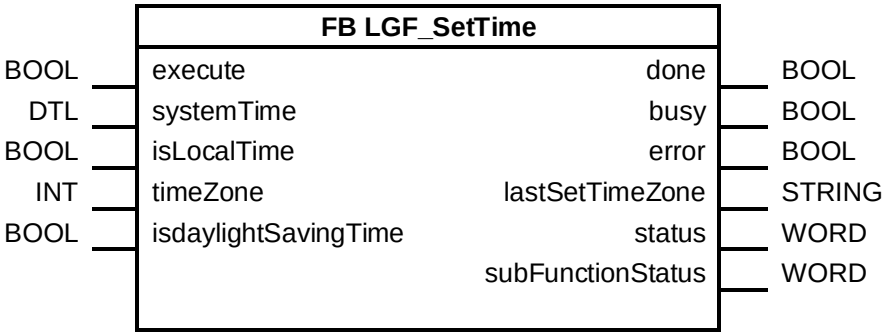
3.1.2 LGF_SetTime

Short description

This block combines the functions of system time, local time, and set time zone.

Block

Figure 3-13: FB LGF_SetTime



Input parameters

Table 3-20: Input parameters

Parameters	Data type	Description
execute	BOOL	A rising edge starts the action once
systemTime	DTL	System time to be set in the CPU
isLocalTime	BOOL	TRUE: System time is local time FALSE: System time is UTC time
timeZone	INT	Defined time zone (format [+-HHMM] Examples: • UTC -12:00 [-1200] • UTC -03:30 [-330] • UTC [0] • UTC +13:00 [1300]
daylightSavingTime	BOOL	TRUE: Daylight saving time changeover active (local time + 60 min) - from last Sunday in March at 02:00 - until last Sunday in October at 03:00 FALSE: no daylight saving time changeover **Note**: For other local time zones, you must adjust the static variable “statTimeZone” in the interface of the block. (see Adjusting parameters in the “statTimeZone” variable)

Output parameters

Table 3-21: Output parameters

Parameters	Data type	Description
done	BOOL	TRUE: The function was completed successfully
busy	BOOL	TRUE: FB is not finished and new output values can be expected.
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.
lastSetTimeZone	STRING	Time zone that was last set by the block
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).
subFunctionStatus	WORD	Status or return value of the called FCs and system blocks.

Status and error displays

Table 3-22: Status/error codes

status	Meaning	Remedy / notes
16#0000	No error	-
16 #7000	No job is currently being processed	-
16#7001	First call after new job (rising edge at “execute”)	-
16#7002	Subsequent call during active processing without further details	-
16#8600	Error due to an undefined state in the State Machine	-
16#8601	Error due to an undefined time zone	Check the input value.
16#8201	Error in “WR_LOC_T” command.	Check the error code in “subFunctionStatus”
16#8202	Error in “WR_SYS_T” command.	Check the error code in “subFunctionStatus”
16#8203	Error command “SET_TIMEZONE”.	Check the error code in “subFunctionStatus”

Note The status of called commands is output in “subFunctionStatus”. In this case, the output value in “status” indicates which command caused the error. In this case, refer to the TIA Portal Online Help section for information on the respective commands.

Principle of operation

This block combines the functions of system time, local time, and set time zone.

The following time zones are possible on the “timeZone” input.

Input “timeZone”	Time zone
-1200	(UTC -12:00) Eniwetok, Kwajalein
-1100	(UTC -11:00) Midway Island
-1000	(UTC -10:00) Hawaii
-930	(UTC -09:30) (French) Polynesia
-900	(UTC -09:00) Alaska
-800	(UTC -08:00) Tijuana, Los Angeles, Seattle, Vancouver
-700	(UTC -07:00) Arizona, Denver, Salt Lake City, Calgary
-600	(UTC -06:00) Chicago, Dallas, Kansas City, Winnipeg
-500	(UTC -05:00) Eastern Time (USA & Canada)
-400	(UTC -04:00) La Paz, Georgetown
-330	(UTC -03:30) Newfoundland
-300	(UTC -03:00) Brasilia, Buenos Aires
-200	(UTC -02:00) Mid-Atlantic
-100	(UTC -01:00) Azores, Cape Verde Is.
0	(UTC) Dublin, Edinburgh, Lisbon, London
100	(UTC +01:00) Berlin, Bern, Brussels, Rome, Stockholm, Vienna
200	(UTC +02:00) Athens, Istanbul, Minsk, Bucharest
300	(UTC +03:00) Moscow, St. Petersburg, Baghdad, Kuwait, Riyadh
330	(UTC +03:30) Iran: Tehran
400	(UTC +04:00) Abu Dhabi, Muscat
430	(UTC +04:30) Afghanistan: Kabul
500	(UTC +05:00) Islamabad, Karachi, Tashkent
530	(UTC +05:30) India, Sri Lanka
545	(UTC +05:45) Nepal
600	(UTC +06:00) Astana, Almaty, Dhaka, Colombo
630	(UTC +06:30) Coco Island, Mayanmar
700	(UTC +07:00) Bangkok, Hanoi, Jakarta
800	(UTC +08:00) Beijing, Chongqing, Hong Kong, Urumqi
830	(UTC +08:30) North Korea old
845	(UTC +08:45) Western Australia: Eucla
900	(UTC +09:00) Yakutsk, Osaka, Sapporo, Tokyo, Seoul
930	(UTC +09:30) Australia: Northern Territory, South Australia
1000	(UTC +10:00) Brisbane, Canberra, Melbourne, Sydney
1030	(UTC +10:30) Australia: Lord Howe Island
1100	(UTC +11:00) Vladivostok, Magadan, Solomon Is., New Caledonia
1200	(UTC +12:00) Auckland, Wellington
1245	(UTC +12:45) Chatham Islands
1300	(UTC +13:00) Tonga, Samoa
1400	(UTC +14:00) Kiribati

Note Daylight saving time/standard time

The parameters (time difference, start summer time, start winter time) must be adapted to the desired time zone in the static variable “statTimeZone”.

Adjusting parameters in the “statTimeZone” variable

The static variable “statTimeZone” in the block interface is of the system data type TimeTransformationRule. In this system data type, the parameters for the local time zone and the summer/winter time changeover are stored.

The default values of the static variable “statTimeZone” are set to Central European Summer Time in the block interface:

- Time difference: 60 min
- Start summer time: last Sunday in March, 02:00 a.m.
- Start winter time: last Sunday in October, 03:00 a.m.

The following Figure shows the settings for the summer/winter time changeover of Central European Summer Time.

The parameter “Bias” is determined by the input parameter “timeZone”. The parameter “DaylightBias” depends on the input parameter “daylightSavingTime” and is either “0” or “60”.

For other time zones, the parameters for summer/winter time changeover must be adjusted (marked below).

Figure 3-14

Name	Data type	Default value
Static		
statTimeZone	TimeTransformationRule	
Bias	Int	0
DaylightBias	Int	60
DaylightStartMonth	USInt	3
DaylightStartWeek	USInt	5
DaylightStartWeekday	USInt	1
DaylightStartHour	USInt	2
DaylightStartMinute	USInt	0
StandardStartMonth	USInt	10
StandardStartWeek	USInt	5
StandardStartWeekday	USInt	1
StandardStartHour	USInt	3
StandardStartMinute	USInt	0
TimeZoneName	String[80]	'not even set ...

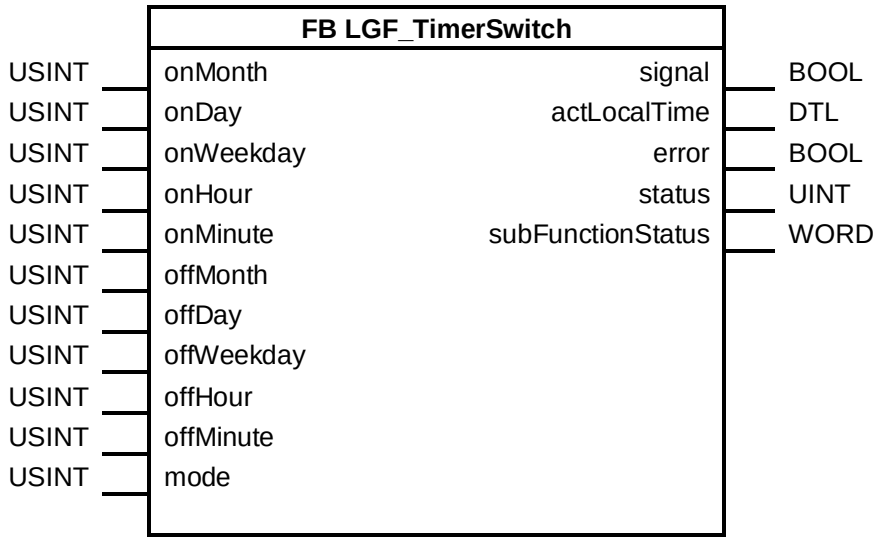
3.1.3 LGF_TimerSwitch

Short description

This block is a timer. It is possible to define daily, weekly, monthly, yearly time switch points and time switch points for working days or weekend days.

Block

Figure 3-15: FB LGF_TimerSwitch



Input parameters

Table 3-23: Input parameters

Parameters	Data type	Description
onMonth	USINT	Month in which the signal will be set.
onDay	USINT	Day on which the signal will be set.
onWeekday	USINT	Day of the week on which the signal will be set. (Sunday = 1)
onHour	USINT	Hour in which the signal will be set.
onMinute	USINT	Minute in which the signal will be set.
offMonth	USINT	Month in which the signal will be reset.
offDay	USINT	Day on which the signal will be reset.
offWeekday	USINT	Day of the week on which the signal will be reset. (Sunday = 1)
offHour	USINT	Hour in which the signal will be reset.
offMinute	USINT	Minute in which the signal will be reset.
mode	USINT	Specifies the mode (see Principle of operation).

Output parameters

Table 3-24: Output parameters

Parameters	Data type	Description
signal	BOOL	Output signal
actLocalTime	DTL	Current local time
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).
subFunctionStatus	WORD	Status or return value of the called FCs and system blocks.

Status and error displays

Table 3-25: Status/error codes

status	Meaning	Remedy / notes
16#0000	No error	-
16#8200	No valid mode was transferred at the input “mode”.	Permitted values “1”, “2”, “3”, “4”, “5”, “6”
16#8600	Error in “RD_LOC_T” command.	Check the error code in “subFunctionStatus”

Note The status of called commands is output in “subFunctionStatus”. In this case, the output value in “status” indicates which command caused the error. In this case, refer to the TIA Portal Online Help section for information on the respective commands.

Principle of operation

The block offers various timer types, which are determined in the “mode” parameter:

- Daily timer (mode = 1)
- Weekly timer (mode = 2)
- Monthly timer (mode = 3)
- Yearly timer (mode = 4)
- Weekdays, Monday to Friday (mode = 5)
- Weekend, Saturday and Sunday (mode = 6)

Depending on the mode, the following formal parameters must be interconnected:

Table 3-26: Required formal parameters for the respective mode

Mode	Mode	Required formal parameters
1	Daily timer	• onHour / offHour • onMinute / offMinute
2	Weekly timer	• onWeekday / offWeekday • onHour / offHour • onMinute / offMinute
3	Monthly timer	• onDay / offDay • onHour / offHour • onMinute / offMinute
4	Yearly timer	• onMonth / offMonth • onDay / offDay • onHour / offHour • onMinute / offMinute
5	Weekdays	• onHour / offHour • onMinute / offMinute
6	Weekend	• onHour / offHour • onMinute / offMinute

If the set start time equals the current local time of the controller, the output “signal” is set to “TRUE”. If the set switch-off time equals the current local time of the controller, the “signal” output is reset again.

Note Please note that the block can be used in the “Monthly timer” modes (mode = 3) or “yearly timer” (mode = 4) the block only switches if the days that you specify at the input parameters, “onDay” and “offDay”, actually occur in this month.

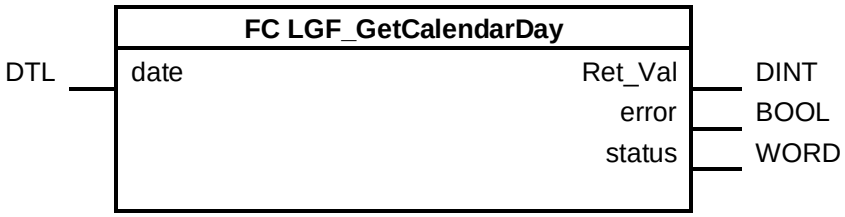
3.1.4 LGF_GetCalendarDay

Short description

This function uses the specified date to calculate the number of days that have passed since the beginning of the year.

Block

Figure 3-16: FC LGF_GetCalendarDay



Input parameters

Table 3-27: Input parameters

Parameters	Data type	Description
inDate	DTL	Calculation date for the days since 1 January.

Output parameters

Table 3-28: Output parameters

Parameters	Data type	Description
Ret_Val	DINT	Days past since January 1st.
error	BOOL	Error display. FALSE: no error. TRUE: Error in block, “status” outputs error code.
status	WORD	“status” outputs the status/error code (see Table below).

Status and error displays

Table 3-29: Status/error codes

Status	Meaning	Remedy
16#0000	No error.	-
16#8201	Date on input “inDate” below the lower limit.	Select “inDate” greater than 1971-01-01.

Principle of operation

The function is used in the functions “LGF_GetCalendarWeek_ISO” and “LGF_GetCalendarWeek_US”.

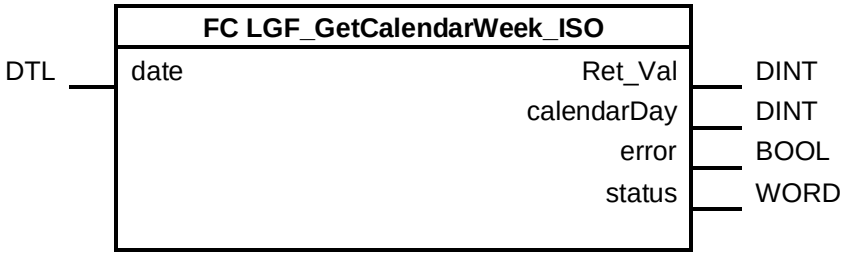
3.1.5 LGF_GetCalendarWeek_ISO

Short description

This function uses the specified date to calculate the calendar week and the number of days that have passed since the beginning of the year for ISO 8601 European countries.

Block

Figure 3-17: FC LGF_GetCalendarWeek_ISO



Input parameters

Table 3-30: Input parameters

Parameters	Data type	Description
inDate	DTL	Date used to calculate the calendar week and days since 1 January.

Output parameters

Table 3-31: Output parameters

Parameters	Data type	Description
Ret_Val	DINT	Number of the calendar week.
calendarDay	DINT	Days past since January 1st.
error	BOOL	Error display. FALSE: no error. TRUE: Error in block, “status” outputs error code.
status	WORD	“status” outputs the status/error code (see Table below).

Status and error displays

Table 3-32: Status/error codes

Status	Meaning	Remedy
16#0000	No error.	-
16#8201	Date on input “inDate” below the lower limit. “status” is passed by LGF_GetCalendarDay and is not declared in the function.	Select “inDate” greater than 1971-01-01.

Counting method for European countries in accordance with ISO 8601

- Calendar weeks have 7 days, start on a Monday, and they are counted continuously throughout the year
- Calendar week 1 of a year is the week that contains the first Thursday.
- Each year has either 52 or 53 calendar weeks.
- A year has 53 calendar weeks if the following characteristics apply:
  - A common year begins on a Thursday and ends on a Thursday.
  - A leap year begins either on a Wednesday and ends on a Thursday or it begins on a Thursday and ends on a Friday.
- The 29th, 30th and 31st December can belong to the calendar week 1 of the following year.
- The 1st, 2nd, and 3rd January can still belong to the last calendar week of the previous year.

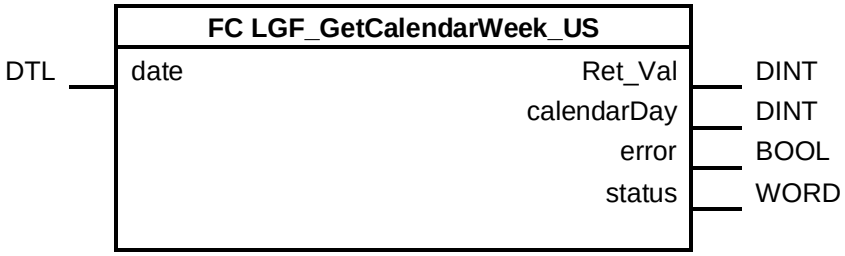
3.1.6 LGF_GetCalendarWeek_US

Short description

This function uses the specified date to calculate the calendar week and the number of days that have passed since the beginning of the year for the USA and many other countries.

Block

Figure 3-18: FC LGF_GetCalendarWeek_US



Input parameters

Table 3-33: Input parameters

Parameters	Data type	Description
inDate	DTL	Date used to calculate the calendar week and days since 1 January.

Output parameters

Table 3-34: Output parameters

Parameters	Data type	Description
Ret_Val	DINT	Number of the calendar week.
calendarDay	DINT	Days past since January 1st.
error	BOOL	Error display. FALSE: no error. TRUE: Error in block, “status” outputs error code.
status	WORD	“status” outputs the status/error code (see Table below).

Status and error displays

Table 3-35: Status/error codes

Status	Meaning	Remedy
16#0000	No error.	-
16#8201	Date on input “inDate” below the lower limit. “status” is passed by LGF_GetCalendarDay and is not declared in the function.	Select “inDate” greater than 1971-01-01.

Counting method for the USA and many other countries

- Calendar weeks have 7 days, start on a Sunday and are counted continuously throughout the year
- Calendar week 1 of a year is the week that contains January 1.
- Each year has either 52 or 53 calendar weeks.
- A year has 53 calendar weeks if the following characteristics apply:
  - A common year begins on a Saturday and ends on a Saturday.
  - A leap year begins either on a Saturday and ends on a Sunday or it begins on a Friday and ends on a Saturday.
- The days after the last Saturday in December can belong to the first calendar week of the following year.

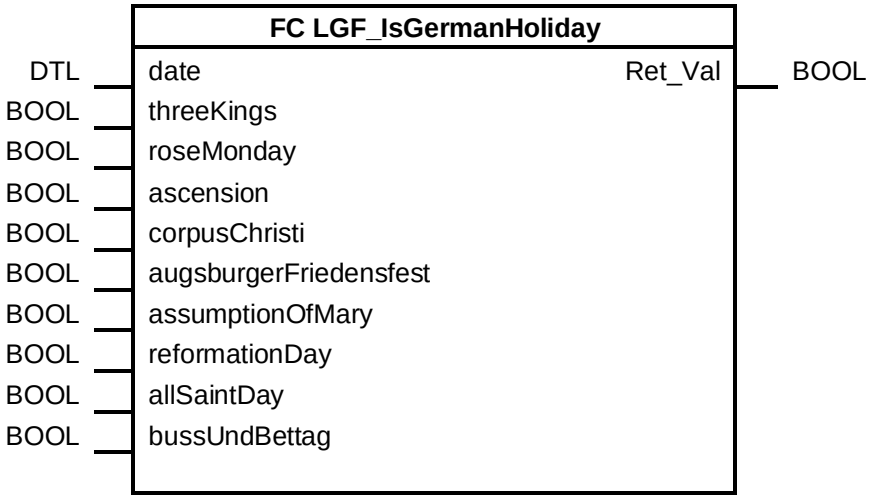
3.1.7 LGF_IsGermanHoliday

Short description

The FC LGF_IsGermanHoliday determines whether a given date is a public holiday. All public holidays in Germany are taken into account. Holidays that are not uniform nationwide can be switched on and off.

Block

Figure 3-19: FB LGF_IsGermanHoliday



Input parameters

Table-36: Input parameters

Parameters	Data type	Description
date	DTL	Date that is checked for a holiday.
threeKings	BOOL	Three Kings' Day
roseMonday	BOOL	Rosenmontag
ascension	BOOL	Ascension
corpusChristi	BOOL	Feast of Corpus Christi
augsburgerFriedensfest	BOOL	Augsburg Peace Festival
assumptionOfMary	BOOL	Assumption of Mary
refomationDay	BOOL	Reformation Day
allSaintDay	BOOL	All Saints' Day
bussUndBettag	BOOL	Day of Prayer and Repentance

Output parameters

Table 3: Output parameters

Parameters	Data type	Description
Ret_Val	BOOL	If TRUE, the date at the input parameter "date" is a public holiday.

Principle of operation

The block calculates the public holiday calendar of the year for a given date and displays whether the given date is a public holiday. Optionally, holidays that are not uniform nationwide, such as Epiphany (threeKings), can be taken into account via the appropriate input parameters in the block.

3.2 Counter Operations

3.2.1 LGF_CountFallnDWord

Short description

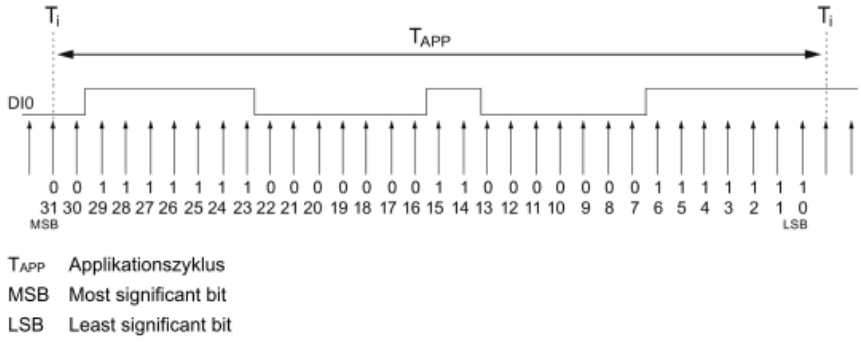
This block analyzes a variable of the type DWORD and outputs how often a 1-0 sequence (falling edge) occurs in the variable.

Application example

Excerpt from the manual of the technology module TM Timer DIDQ 16x24V

With the oversampling function, the technology module records the state of the respective digital input per application cycle (e.g. OB61) at 32 points in time with a uniform time interval. The 32 states are jointly returned as 32-bit values in the checkback interface.

Figure 3-20: Example of an oversampling of DI0 on TM Timer DIDQ 16x24V



T_{APP}	Applikationszyklus
MSB	Most significant bit
LSB	Least significant bit

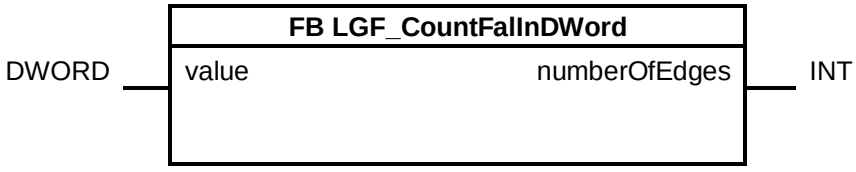
The LGF_CountFallnDWord block is used, in this case, to count how often a falling edge occurs.

SIMATIC ET 200MP/S7-1500 Technology Module TM Timer DIDQ 16x24V (6ES7552-1AA00-0AB0)

https://support.industry.siemens.com/cs/ww/en/view/95153313

Block

Figure 3-21: FB LGF_CountFallnDWord



Input parameters

Table 3-37: Input parameters

Parameters	Data type	Description
value	DWORD	Double word in which the falling edges are counted

Output parameters

Table 3-38: Output parameters

Parameters	Data type	Description
numberOfEdges	INT	Number of falling edges in the double word

Principle of operation

In a variable of the data type DWORD, the block counts the falling edges (1-0 transitions) from left to right. The output “countFalInDWord” outputs the number of falling edges.

So that falling edges at the variable limit are also detected, the input “value” is copied to the static variable “statDWordPrevCycle” at the end of the evaluation and evaluated in the next cycle.

Example

The following example illustrates the block’s functionality. In this case, it is assumed that a signal of unknown length is continuously sampled in the form of double words (DWORD) per cycle.

Within this signal, the 1-0 sequences (falling edges) must be counted and output continuously. To detect the falling edge on variable limits, as in this example, the input “statDWordPrevCycle” must be interconnected with the double word of the previous sampling.

Table 3-39: Example

DWORD previous cycle (statDWordPrevCycle)	DWORD actual cycle (value)
1001_0000_0001_1010_1001_0000_0001_1011	**0**01**0**_1**0**1**0**_0001_1111_**0**1**0**0_0011_1**0**00_01**0**1

Number of 1-0 sequences (falling edges): “Ret_Val” = 8

3.2.2 LGF_CountRisInDWord

Short description

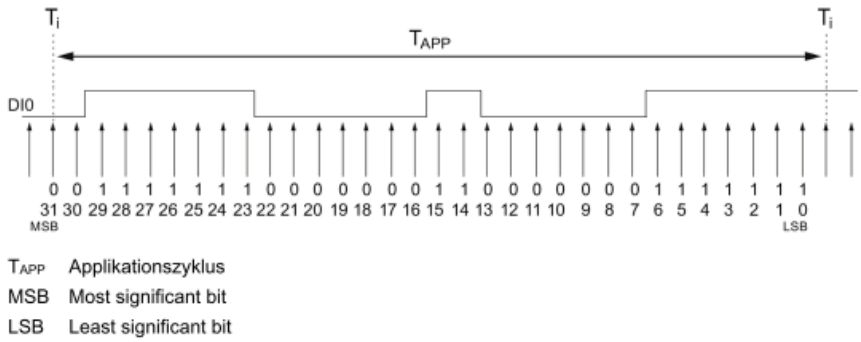
This block analyzes a variable of the type DWORD and outputs how often a 0-1 sequence (rising edge) occurs in the variable.

Application example:

Excerpt from the manual of the technology module TM Timer DIDQ 16x24V:

With the oversampling function, the technology module records the state of the respective digital input per application cycle (e.g. OB61) at 32 points in time with a uniform time interval. The 32 states are jointly returned as 32-bit values in the checkback interface.

Figure 3-22: Example of an oversampling of DI0 on TM Timer DIDQ 16x24V



T_{APP} Applikationszyklus
MSB Most significant bit
LSB Least significant bit

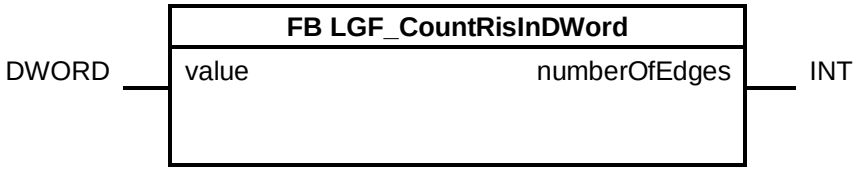
The block LGF_CountRisInDWordFB is used in this case to count how often a rising edge occurs.

SIMATIC ET 200MP/S7-1500 Technology Module TM Timer DIDQ 16x24V (6ES7552-1AA00-0AB0)

https://support.industry.siemens.com/cs/ww/en/view/95153313

Block

Figure 3-23: FB LGF_CountRisInDWord



Input parameters

Table 3-40: Input parameters

Parameters	Data type	Description
value	DWORD	Double word in which the rising edges are counted

Output parameters

Table 3-41: Output parameters

Parameters	Data type	Description
numberOfEdges	INT	Number of rising edges in the double word

Principle of operation

In a variable of the data type DWORD, the block counts the rising edges (0-1 transitions) from left to right. The output “countRisInDWord” outputs the number of rising edges.

So that rising edges at the variable limit are also detected, the input “value” is copied to the static variable “statDWordPrevCycle” at the end of the evaluation and evaluated in the next cycle.

Example

The following example illustrates the block’s functionality. In this case, it is assumed that a signal of unknown length is continuously sampled in the form of double words (DWORD) per cycle.

Within this signal, the 0-1 sequences (rising edges) must be counted and output continuously. In order to detect the rising edge at variable limits, as in this example, the input “statDWordPrevCycle” must be interconnected with the double word of the previous sampling.

Table 3-42: Example

DWORD previous cycle (statDWordPrevCycle)	DWORD actual cycle (value)
1001_0000_0001_1010_1001_0000_0001_1010	**1**0**1**0_**1**0**1**0_000**1**_1111_0**1**00_00**1**1_1000_0**1**0**1**

Number of 0-1 sequences (rising edges): “Ret_Val” = 9

3.3 Comparator Operations

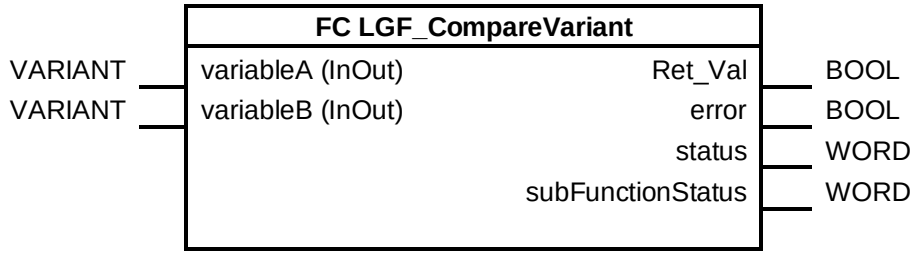
3.3.1 LGF_CompareVariant

Short description

This block compares two structured (array, PLC data type) actual parameters and outputs whether they are of the same type and have the same values.

Block

Figure 3-24: FC LGF_CompareVariant



Input/output parameters (InOut)

Table 3-43: Input/output parameters (InOut)

Parameters	Data type	Description
variable1	VARIANT	First comparison variable with any data type
variable2	VARIANT	Second comparison variable with any data type

Output parameters

Table 3-44: Output parameters

Parameters	Data type	Description
Ret_Val	BOOL	FALSE: Values of comparison variables or PLC data types are different. TRUE: Values of the comparison variables are equal and PLC data types are identical.
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).
subFunctionStatus	WORD	Status or return value of the called FCs and system blocks.

Status and error displays

Table 3-45: Status/error codes

status	Meaning	Remedy / notes
16#0000	No error	-
16#8201	Error, the input types must match.	-
16#8202	Error, the input types have different lengths after serialization.	“subFunctionStatus” provides an indicator for the different length.
16#8601	Error in “Serialize” “variableA” command.	Check the error code in “subFunctionStatus”
16#8602	Error in “Serialize” “variableB” command.	Check the error code in “subFunctionStatus”

Note The status of called commands is output in “subFunctionStatus”. In this case, the output value in “status” indicates which command caused the error. In this case, refer to the TIA Portal Online Help section for information on the respective commands.

Principle of operation

This block compares two (structured) actual parameters and shows whether they equate to the same value.

Note The following differences cannot be detected with the comparison method (byte level):

- Variables of the data type “Struct” cannot be compared.
- For strings, there may be differences in the range between the actual length and the maximum length.
- With REAL numbers in the structure, a disparity can also be displayed for “same” variables.
- Variables of the type “ARRAY of BOOL” cannot be checked for equality with the function, because the command used, “CountOfElements”, also counts the filling elements (e.g. 8 is returned with an ARRAY[0..1] of BOOL).

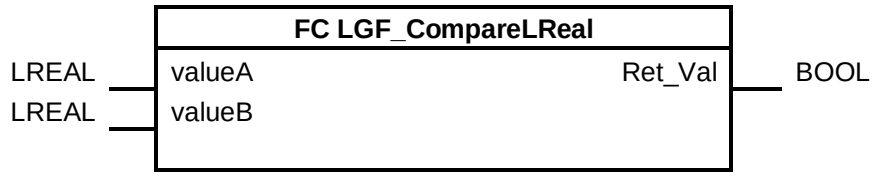
3.3.2 LGF_CompareLReal

Short description

This function checks two LREAL numbers for equality using an approximation formula.

Block

Figure 3-25: FC LGF_CompareLReal



Input parameters

Table 3-46: Input parameters

Parameters	Data type	Description
valueA	LREAL	First LREAL number to be compared.
valueB	LREAL	Second LREAL number to be compared.

Output parameters

Table 3-47: Output parameters

Parameters	Data type	Description
Ret_Val	BOOL	FALSE: not the same TRUE: same

Principle of operation

The comparison of the LREAL numbers is based on an accuracy of 1.0E-12. The difference between the two input values must be smaller than the “PRECISION” accuracy multiplied by one of the two input values.

Note If your application requires a different accuracy when comparing the numbers, adapt the “PRECISION” constant in the function to your requirements.

Or you may use the FC LGF_CompareLRealByPrecision.

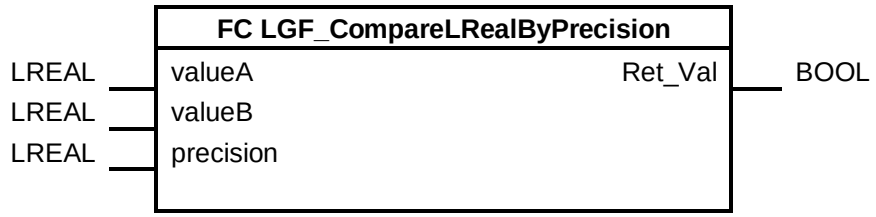
3.3.3 LGF_CompareLRealByPrecision

Short description

This function checks two LREAL numbers for equality using an approximation formula. The accuracy is specified by the “precision” input variable.

Block

Figure 3-26: FC LGF_CompareLRealByPrecision



Input parameters

Table 3-48: Input parameters

Parameters	Data type	Description
valueA	LREAL	First LREAL number to be compared.
valueB	LREAL	Second LREAL number to be compared.
precision	LREAL	Accuracy with which the two values are compared.

Output parameters

Table 3-49: Output parameters

Parameters	Data type	Description
Ret_Val	BOOL	FALSE: not the same TRUE: same

Principle of operation

The comparison of the LREAL numbers is based on an accuracy of 1.0E-12. The difference between the two input values must be smaller than the “precision” accuracy value multiplied by one of the two input values.

3.4 Math Operations

3.4.1 LGF_MatrixAddition

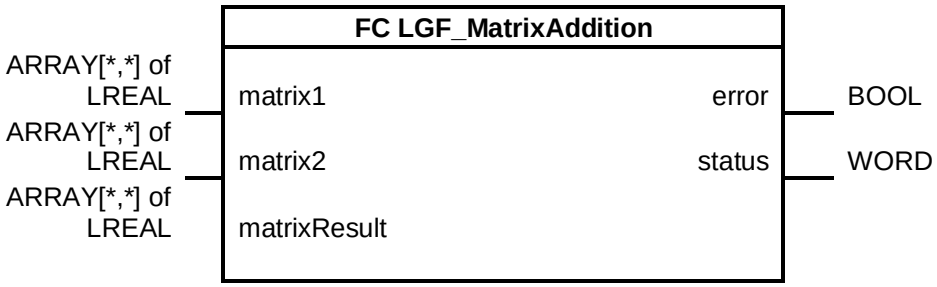
Short description

This block adds two matrices of equal size of the data type ARRAY[*,*] of LREAL.

$$\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix} + \begin{pmatrix} b_{11} & \cdots & b_{1n} \\ \vdots & \ddots & \vdots \\ b_{m1} & \cdots & b_{mn} \end{pmatrix} = \begin{pmatrix} a_{11} + b_{11} & \cdots & a_{1n} + b_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} + b_{m1} & \cdots & a_{mn} + b_{mn} \end{pmatrix}$$

Block

Figure 3-27: FC LGF_MatrixAddition



Input/output parameters (InOut)

Table 3-50: Input/output parameters (InOut)

Parameters	Data type	Description
matrix1	ARRAY[*,*] of LREAL	First summand (matrix)
matrix2	ARRAY[*,*] of LREAL	Second summand (matrix)
matrix Result	ARRAY[*,*] of LREAL	Sum (matrix)

Output parameters

Table 3-51: Output parameters

Parameters	Data type	Description
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).

Status and error displays

Table 3-52: Status/error codes

status	Meaning	Remedy / notes
16#0000	No error	-
16#8200	Lower-limit rows(Dim1) of the arrays of Matrix1 and Matrix2 are different.	All arrays must have the same low limit, for example: Array[**0**..2, **0**..2] of LREAL
16#8201	Lower-limit rows(Dim1) of the arrays of Matrix1 and Result Matrix are different.	All arrays must have the same low limit, for example: Array[**0**..2, **0**..2] of LREAL
16#8202	Lower-limit columns(Dim2) of the arrays of Matrix1 and Matrix2 are different.	All arrays must have the same low limit, for example: Array[**0**..2, **0**..2] of LREAL
16#8203	Lower-limit columns(Dim2) of the arrays of Matrix1 and Result Matrix are different.	All arrays must have the same low limit, for example: Array[**0**..2, **0**..2] of LREAL
16#8204	Upper-limit rows(Dim1) of the arrays of Matrix1 and Matrix2 are different.	All arrays must have the same high limit, for example: Array[**0**..2, **0**..2] of LREAL
16#8205	Upper-limit rows(Dim1) of the arrays of Matrix1 and Result Matrix are different.	All arrays must have the same high limit, for example: Array[**0**..2, **0**..2] of LREAL
16#8206	Upper-limit columns(Dim2) of the arrays of Matrix1 and Matrix2 are different.	All arrays must have the same high limit, for example: Array[**0**..2, **0**..2] of LREAL
16#8207	Upper-limit columns(Dim2) of the arrays of Matrix1 and Result Matrix are different.	All arrays must have the same high limit, for example: Array[**0**..2, **0**..2] of LREAL

Principle of operation

The block adds two matrices of the same size. The individual fields of the two incoming matrices are read, added and then output in the matrix “matrix Result”.

Note Note that all input and output matrices must have the same low and high limits and therefore the same number of columns and rows.

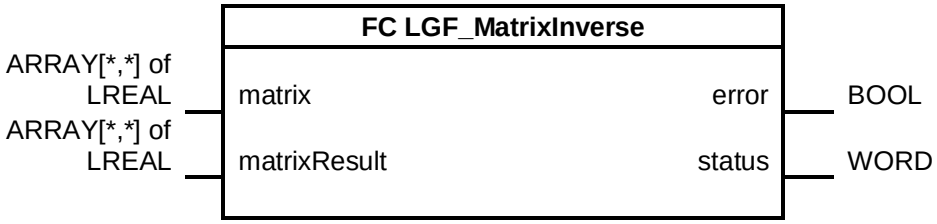
3.4.2 LGF_MatrixInverse

Short description

This function block inverts a square matrix of the data type ARRAY[*,*] of LREAL.

Block

Figure 3-28: FC LGF_MatrixInverse



Input/output parameters (InOut)

Table 3-53: Input/output parameters (InOut)

Parameters	Data type	Description
matrix	ARRAY[*,*] of LREAL	Square input matrix (Array[0..x, 0..x] of REAL)
matrix Result	ARRAY[*,*] of LREAL	Inverted matrix

Output parameters

Table 3-54: Output parameters

Parameters	Data type	Description
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).

Status and error displays

Table 3-55: Status/error codes

status	Meaning	Remedy / notes
16#0000	No error	-
16#8200	Input matrix is not square.	The number of rows must be equal to the number of columns.
16#8201	Application of the algorithm for the input matrix is not possible.	First element ($a_{1,1}$) of the input matrix must not be zero.
16#8202	Lower-limit rows(Dim1) of the arrays of Matrix1 and Result Matrix are different.	All arrays must have the same low limit, for example: Array[**0**..2, **0**..2] of LREAL
16#8203	Lower-limit columns(Dim2) of the arrays of Matrix1 and Result Matrix are different.	All arrays must have the same low limit, for example: Array[**0**..2, **0**..2] of LREAL
16#8204	Upper-limit rows(Dim1) of the arrays of Matrix1 and Result Matrix are different.	All arrays must have the same high limit, for example: Array[**0**..2, **0**..2] of LREAL
16#8205	Upper-limit rows(Dim1) of the arrays of Matrix1 and Result Matrix are different.	All arrays must have the same high limit, for example: Array[**0**..2, **0**..2] of LREAL

Principle of operation

The block inverts a square matrix of any size according to the Shipley-Coleman method.

Note

Note that the input matrix must be square. This means that the number of rows must be equal to the number of columns.

The output matrix must be the same size and have the same array boundaries as the input matrix.

3.4.3 LGF_MatrixMultiplication

Short description

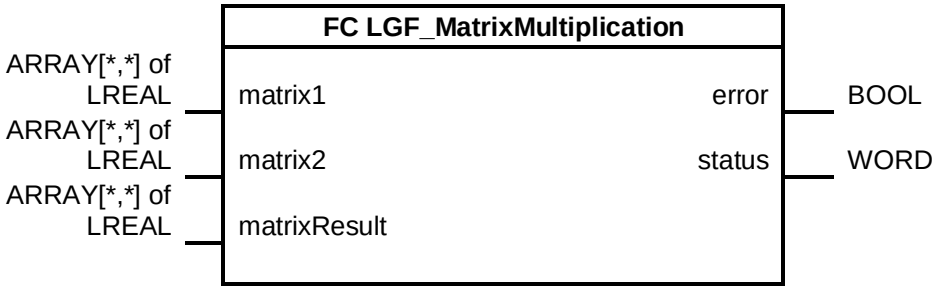
This block multiplies two matrices of the data type ARRAY[*,*] of LREAL.

Example for 2x2 matrix:

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} * \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} = \begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{pmatrix}$$

Block

Figure 3-29: FC LGF_MatrixMultiplication



Input/output parameters (InOut)

Table 3-56: Input/output parameters (InOut)

Parameters	Data type	Description
matrix1	ARRAY[*,*] of LREAL	First factor: Matrix to multiply
matrix2	ARRAY[*,*] of LREAL	Second factor: Matrix to multiply
matrix Result	ARRAY[*,*] of LREAL	Product: The resulting matrix

Output parameters

Table 3-57: Output parameters

Parameters	Data type	Description
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).

Status and error displays

Table 3-58: Status/error codes

status	Meaning	Remedy / notes
16#0000	No error	-
16#8200	Lower-limit columns(Dim2) of the array of Matrix1 and lower-limit rows(Dim1) of the array of Matrix2 are different.	-
16#8201	Upper-limit columns(Dim2) of the array of Matrix1 and upper-limit rows(Dim1) of the array of Matrix2 are different.	-
16#8202	Lower-limit rows(Dim1) of the arrays of Matrix1 and Result Matrix are different.	-
16#8203	Upper-limit columns(Dim2) of the arrays of Matrix2 and Result Matrix are different.	-
16#8204	Upper-limit rows(Dim1) of the arrays of Matrix1 and Result Matrix are different.	-
16#8205	Upper-limit columns(Dim2) of the arrays of Matrix1 and Result Matrix are different.	-

Principle of operation

The block multiplies two matrices of variable size. The individual elements of the two incoming matrices are read, multiplied, and then output in the “matrix Result” matrix.

Note

Note that the number of columns in the first matrix must be equal to the number of rows in the second matrix.

The size of the initial matrix (m * n) results from the number of rows (m) of “matrix1” and the number of columns (n) of “matrix2”.

3.4.4 LGF_MatrixScalarMultiplication

Short description

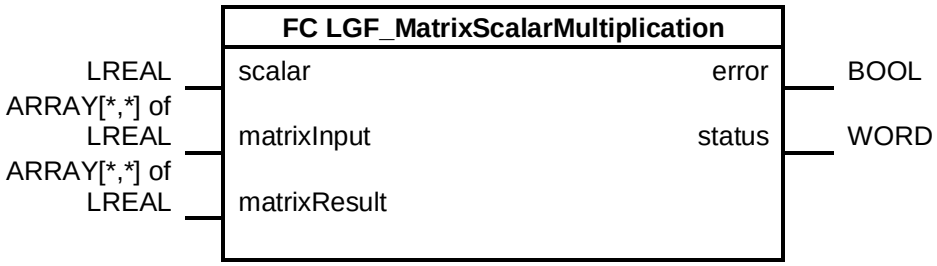
This function block multiplies a matrix of the data type ARRAY[*,*] of LREAL with a scalar.

Example for 2x2 matrix:

$$b * \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} = \begin{pmatrix} b * a_{11} & b * a_{12} \\ b * a_{21} & b * a_{22} \end{pmatrix}$$

Block

Figure 3-30: FC LGF_MatrixScalarMultiplication



Input parameters

Table 3-59: Input parameters

Parameters	Data type	Description
scalar	LREAL	Skalar

Input/output parameters (InOut)

Table 3-60: Input/output parameters (InOut)

Parameters	Data type	Description
matrixInput	ARRAY[*,*] of LREAL	Matrix to multiply
matrix Result	ARRAY[*,*] of LREAL	Product: The resulting matrix

Output parameters

Table 3-61: Output parameters

Parameters	Data type	Description
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).

Status and error displays

Table 3-62: Status/error codes

status	Meaning	Remedy / notes
16#0000	No error	-
16#8201	Lower-limit rows(Dim1) of the arrays of Matrix Input and Result Matrix are different.	All arrays must have the same low limit, for example: Array[**0**..2, **0**..2] of LREAL
16#8202	Upper-limit rows(Dim1) of the arrays of Matrix1 and Result Matrix are different.	All arrays must have the same high limit, for example: Array[**0**..2, **0**..2] of LREAL
16#8203	Lower-limit columns(Dim2) of the arrays of Matrix Input and Result Matrix are different.	All arrays must have the same low limit, for example: Array[**0**..2, **0**..2] of LREAL
16#8204	Upper-limit columns(Dim2) of the arrays of Matrix Input and Result Matrix are different.	All arrays must have the same high limit, for example: Array[**0**..2, **0**..2] of LREAL

Principle of operation

The block multiplies a matrix of variable size by a scalar. A matrix is multiplied by a scalar, thereby multiplying each matrix element by the scalar. The result is output in the “matrixResult” matrix.

Note Note that the input and output matrix must have the same number of columns and rows.

3.4.5 LGF_MatrixSubtraction

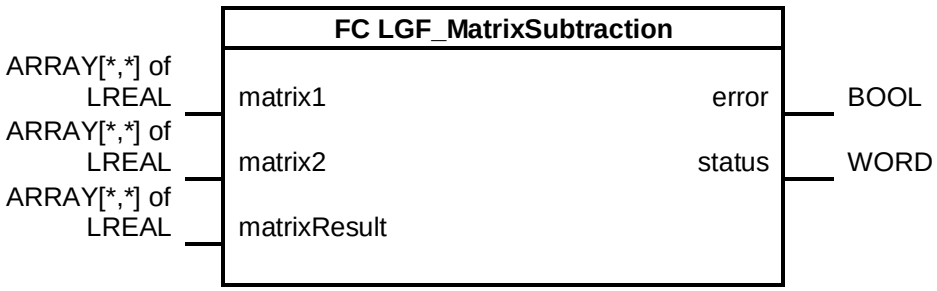
Short description

This function block subtracts a matrix of the data type ARRAY[*,*] of LREAL from another one.

$$\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix} - \begin{pmatrix} b_{11} & \cdots & b_{1n} \\ \vdots & \ddots & \vdots \\ b_{m1} & \cdots & b_{mn} \end{pmatrix} = \begin{pmatrix} a_{11} - b_{11} & \cdots & a_{1n} - b_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} - b_{m1} & \cdots & a_{mn} - b_{mn} \end{pmatrix}$$

Block

Figure 3-31: FC LGF_MatrixSubtraction



Input/output parameters (InOut)

Table 3-63: Input/output parameters (InOut)

Parameters	Data type	Description
matrix1	ARRAY[*,*] of LREAL	Minuend: “Matrix2” is subtracted from this matrix.
matrix2	ARRAY[*,*] of LREAL	Subtrahend: This matrix is subtracted from “matrix1”.
matrix Result	ARRAY[*,*] of LREAL	Difference: The resulting matrix

Output parameters

Table 3-64: Output parameters

Parameters	Data type	Description
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).

Status and error displays

Table 3-65: Status/error codes

status	Meaning	Remedy / notes
16#0000	No error	-
16#8200	Lower limit rows(Dim1) of the arrays of Matrix1 and Matrix2 are different.	All arrays must have the same low limit, for example: Array[**0**..2, **0**..2] of LREAL
16#8201	Lower-limit rows(Dim1) of the arrays of Matrix1 and Result Matrix are different.	All arrays must have the same low limit, for example: Array[**0**..2, **0**..2] of LREAL
16#8202	Lower-limit columns(Dim2) of the arrays of Matrix1 and Matrix2 are different.	All arrays must have the same low limit, for example: Array[**0**..2, **0**..2] of LREAL
16#8203	Lower-limit columns(Dim2) of the arrays of Matrix1 and Result Matrix are different.	All arrays must have the same low limit, for example: Array[**0**..2, **0**..2] of LREAL
16#8204	Upper-limit rows(Dim1) of the arrays of Matrix1 and Matrix2 are different.	All arrays must have the same high limit, for example: Array[**0**..2, **0**..2] of LREAL
16#8205	Upper-limit rows(Dim1) of the arrays of Matrix1 and Result Matrix are different.	All arrays must have the same high limit, for example: Array[**0**..2, **0**..2] of LREAL
16#8206	Upper-limit columns(Dim2) of the arrays of Matrix1 and Matrix2 are different.	All arrays must have the same high limit, for example: Array[**0**..2, **0**..2] of LREAL
16#8207	Upper-limit columns(Dim2) of the arrays of Matrix1 and Result Matrix are different.	All arrays must have the same high limit, for example: Array[**0**..2, **0**..2] of LREAL

Principle of operation

The block subtracts two matrices of variable size. The individual fields of the two matrices are read, subtracted and then output in the matrix “matrix Result”.

Note Note that all input and output matrices must have the same number of columns and rows.

3.4.6 LGF_MatrixTranspose

Short description

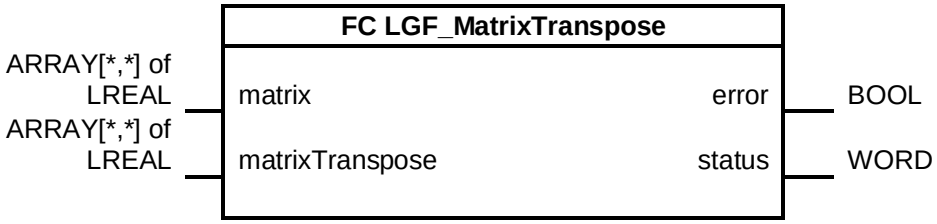
This block transposes a matrix of the data type ARRAY[*,*] of LREAL.

Condition: Input matrix (m x n) = output matrix (n x m)

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix}; \quad A^T = \begin{pmatrix} a_{11} & \cdots & a_{m1} \\ \vdots & \ddots & \vdots \\ a_{1n} & \cdots & a_{mn} \end{pmatrix}$$

Block

Figure 3-32: FC LGF_MatrixTranspose



Input/output parameters (InOut)

Table 3-66: Input/output parameters (InOut)

Parameters	Data type	Description
matrix	ARRAY[*,*] of LREAL	Matrix to be transposed
matrix Result	ARRAY[*,*] of LREAL	Transposed matrix

Output parameters

Table 3-67: Output parameters

Parameters	Data type	Description
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).

Status and error displays

Table 3-68: Status/error codes

status	Meaning	Remedy / notes
16#0000	No error	-
16#8200	Lower-limit rows(Dim1) of the array of Matrix1 and lower-limit columns(Dim2) of the array of Result Matrix are different.	-
16#8201	Lower-limit columns(Dim1) of the array of Matrix1 and lower-limit rows(Dim2) of the array of Result Matrix are different.	-
16#8202	Upper-limit rows(Dim1) of the array of Matrix1 and upper-limit columns(Dim2) of the array of Result Matrix are different.	-
16#8203	Upper-limit columns(Dim1) of the array of Matrix1 and lower-limit rows(Dim2) of the array of Result Matrix are different.	-

Principle of operation

A matrix is transposed by making columns out of the rows.

Note Note that the number of rows of the input matrix must be equal to the number of columns of the output matrix. Also, the number of columns of the input matrix must be equal to the number of rows of the output matrix.

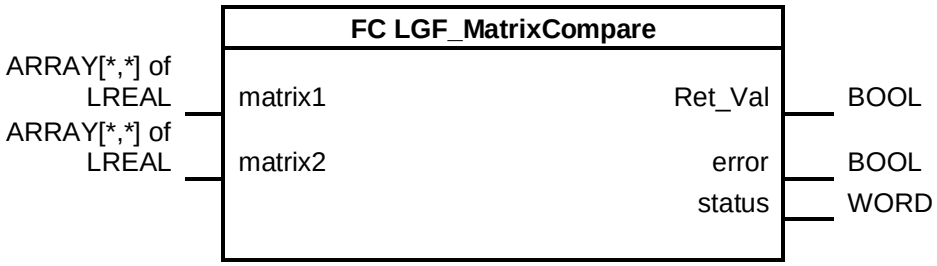
3.4.7 LGF_MatrixCompare

Short description

This function block compares two matrices of the data type ARRAY[*,*] of LREAL of equal size.

Block

Figure 3-33: FC LGF_MatrixCompare



Input/output parameters (InOut)

Table 3-69: Input/output parameters (InOut)

Parameters	Data type	Description
matrix1	ARRAY[*,*] of LREAL	First summand (matrix)
matrix2	ARRAY[*,*] of LREAL	Second summand (matrix)

Output parameters

Table 3-70: Output parameters

Parameters	Data type	Description
Ret_Val	BOOL	TRUE: Both matrices are identical.
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).

Status and error displays

Table 3-71: Status/error codes

status	Meaning	Remedy / notes
16#0000	No error	-
16#8200	Lower limit rows(Dim1) of the arrays of Matrix1 and Matrix2 are different.	All arrays must have the same low limit, for example: Array[**0**..2, **0**..2] of LREAL
16#8201	Lower-limit columns(Dim2) of the arrays of Matrix1 and Matrix2 are different.	All arrays must have the same low limit, for example: Array[**0**..2, **0**..2] of LREAL
16#8202	Upper-limit rows(Dim1) of the arrays of Matrix1 and Matrix2 are different.	All arrays must have the same high limit, for example: Array[**0**..2, **0**..2] of LREAL
16#8206	Upper-limit columns(Dim2) of the arrays of Matrix1 and Matrix2 are different.	All arrays must have the same high limit, for example: Array[**0**..2, **0**..2] of LREAL

Principle of operation

The block compares two matrices of the same size. If both matrices are identical, the return value of the function is set to TRUE.

Note Note that all input matrices must have the same lower and upper limit, and, therefore, the same number of columns and rows.

3.4.8 LGF_StoreMinMax

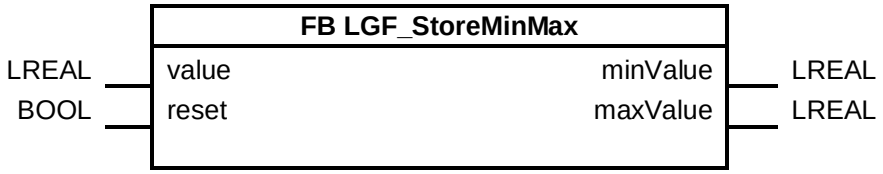
Short description

This block reads-in a value of a variable at each call and outputs the maximum and minimum value that has been read in since the first call.

The evaluation can be reset if necessary. The block supports the data type LREAL.

Block

Figure 3-34: FB LGF_StoreMinMax



Input parameters

Table 3-72: Input parameters

Parameters	Data type	Description
value	LREAL	Variable whose value is checked for minimum and maximum.
reset	BOOL	The block is reset and the evaluation starts over again.

Output parameters

Table 3-73: Output parameters

Parameters	Data type	Description
minValue	LREAL	Minimum value since first call or since activation of the input “reset”.
maxValue	LREAL	Maximum values since first call, or since activation of the input “reset”.

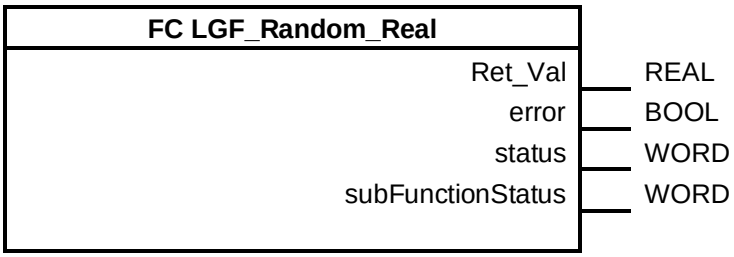
3.4.9 LGF_Random_Real

Short description

This function generates a random value between 0.0 and 1.0 for each call. The random number has the data type REAL.

Block

Figure 3-35: FC LGF_Random_Real



Output parameters

Table 3-74: Output parameters

Parameters	Data type	Description
Ret_Val	REAL	Random number
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).
subFunctionStatus	WORD	Status or return value of the called FCs and system blocks.

Status and error displays

Table 3-75: Status/error codes

status	Meaning	Remedy / notes
16#0000	No error	-
16#8600	Error in “RD_SYS_T” command.	Check the error code in “subFunctionStatus”

Note The status of called commands is output in “subFunctionStatus”. In this case, the output value in “status” indicates which command caused the error. In this case, refer to the TIA Portal Online Help section for information on the respective commands.

Principle of operation

The function generates random values in the range of 0.0 ≤ Ret_Val ≤ 1.0.

Background information

The random value is formed from the nanoseconds of the current system time of the CPU. The byte order of this value is inverted and then converted to a floating point number.

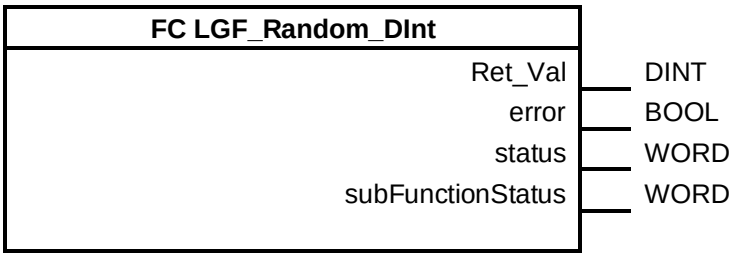
3.4.10 LGF_Random_DInt

Short description

This function generates a random value with each call. The random number has the data type DINT.

Block

Figure 3-36: FC LGF_Random_DInt



Output parameters

Table 3-76: Output parameters

Parameters	Data type	Description
Ret_Val	DINT	Random number
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).
subFunctionStatus	WORD	Status or return value of the called FCs and system blocks.

Status and error displays

Table 3-77: Status/error codes

status	Meaning	Remedy / notes
16#0000	No error	-
16#8600	Error in “RD_SYS_T” command.	Check the error code in “subFunctionStatus”

Note The status of called commands is output in “subFunctionStatus”. In this case, the output value in “status” indicates which command caused the error. In this case, refer to the TIA Portal Online Help section for information on the respective commands.

Principle of operation

The function generates random values in the range of -2147483648 ≤ Ret_Val ≤ 2147483647.

Background information

The random value is formed from the nanoseconds of the current system time of the CPU. The byte order of this value is inverted and then converted to DINT.

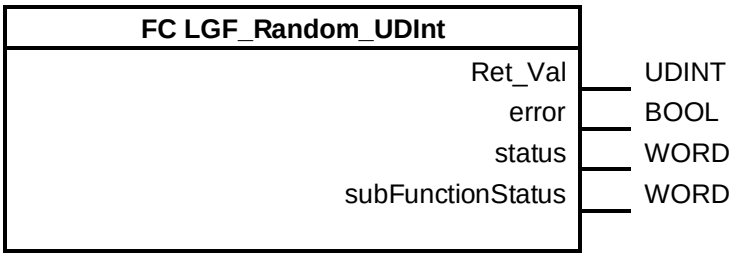
3.4.11 LGF_Random_UDInt

Short description

This function generates a random value with each call. The random number has the data type UDINT.

Block

Figure 3-37: FC LGF_Random_UDInt



Output parameters

Table 3-78: Output parameters

Parameters	Data type	Description
Ret_Val	UDINT	Random number
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).
subFunctionStatus	WORD	Status or return value of the called FCs and system blocks.

Status and error displays

Table 3-79: Status/error codes

status	Meaning	Remedy / notes
16#0000	No error	-
16#8600	Error in “RD_SYS_T” command.	Check the error code in “subFunctionStatus”

Note The status of called commands is output in “subFunctionStatus”. In this case, the output value in “status” indicates which command caused the error. In this case, refer to the TIA Portal Online Help section for information on the respective commands.

Principle of operation

The function generates random values in the range of 0 ≤ Ret_Val ≤ 4294967295.

Background information

The random value is formed from the nanoseconds of the current system time of the CPU. The byte order of this value is inverted and then converted to UDINT.

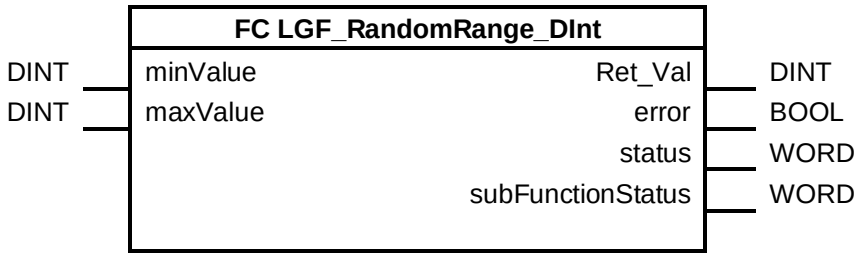
3.4.12 LGF_RandomRange_DInt

Short description

This block generates a “random” value between a defined maximum and minimum value for each call. The random number has the data type DINT.

Block

Figure 3-38: FC LGF_RandomRange_DInt



Input parameters

Table 3-80: Input parameters

Parameters	Data type	Description
minValue	DINT	Defines the low limit of the random number.
maxValue	DINT	Defines the high limit of the random number.

Output parameters

Table 3-81: Output parameters

Parameters	Data type	Description
Ret_Val	DINT	Random number
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).
subFunctionStatus	WORD	Status or return value of the called FCs and system blocks.

Status and error displays

Table 3-82: Status/error codes

status	Meaning	Remedy / notes
16#0000	No error	-
16#8200	“minValue” is greater than “maxValue”.	-
16#8600	Error in “RD_SYS_T” command.	Check the error code in “subFunctionStatus”

Note The status of called commands is output in “subFunctionStatus”. In this case, the output value in “status” indicates which command caused the error. In this case, refer to the TIA Portal Online Help section for information on the respective commands.

Principle of operation

The block generates random values that are between the specified “minValue” value and the “maxValue” value. This random value is output via the “Ret_Val”.

Background information

The random value is formed from the nanoseconds of the current system time of the CPU. The byte order of this value is inverted and then converted into a normalized DINT integer.

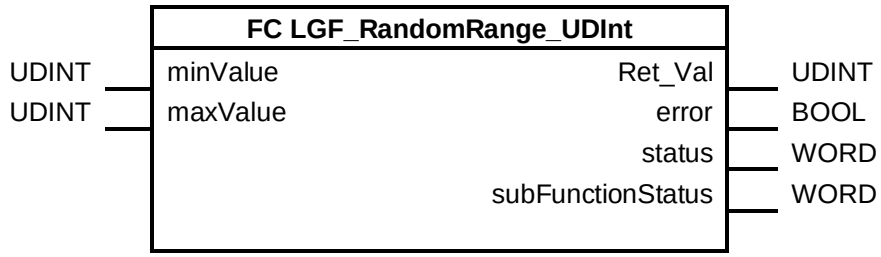
3.4.13 LGF_RandomRange_UDInt

Short description

This block generates a “random” value between a defined maximum and minimum value for each call. The random number has the data type UDINT.

Block

Figure 3-39: FC LGF_RandomRange_UDInt



Input parameters

Table 3-83: Input parameters

Parameters	Data type	Description
minValue	UDINT	Defines the low limit of the random number.
maxValue	UDINT	Defines the high limit of the random number.

Output parameters

Table 3-84: Output parameters

Parameters	Data type	Description
Ret_Val	UDINT	Random number
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).
subFunctionStatus	WORD	Status or return value of the called FCs and system blocks.

Status and error displays

Table 3-85: Status/error codes

status	Meaning	Remedy / notes
16#0000	No error	-
16#8200	“minValue” is greater than “maxValue”.	-
16#8600	Error in “RD_SYS_T” command.	Check the error code in “subFunctionStatus”

Note The status of called commands is output in “subFunctionStatus”. In this case, the output value in “status” indicates which command caused the error. In this case, refer to the TIA Portal Online Help section for information on the respective commands.

Principle of operation

The block generates random values that are between the specified “minValue” value and the “maxValue” value. This random value is output via the “Ret_Val”.

Background information

The random value is formed from the nanoseconds of the current system time of the CPU. The byte order of this value is inverted and then converted into a normalized integer UDINT.

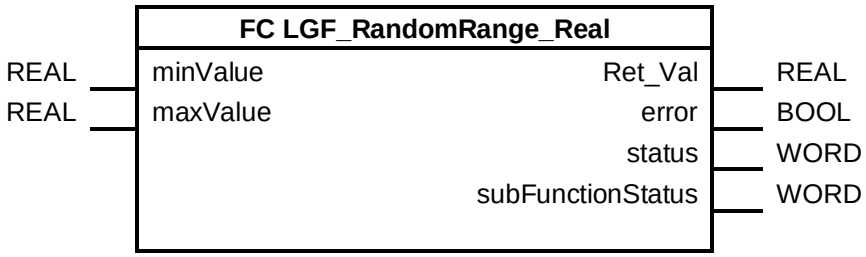
3.4.14 LGF_RandomRange_Real

Short description

This block generates a “random” value between a defined maximum and minimum value for each call. The random number has the data type REAL.

Block

Figure 3-40: FC LGF_RandomRange_Real



Input parameters

Table 3-86: Input parameters

Parameters	Data type	Description
minValue	REAL	Defines the low limit of the random number.
maxValue	REAL	Defines the high limit of the random number.

Output parameters

Table 3-87: Output parameters

Parameters	Data type	Description
Ret_Val	REAL	Random number
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).
subFunctionStatus	WORD	Status or return value of the called FCs and system blocks.

Status and error displays

Table 3-88: Status/error codes

status	Meaning	Remedy / notes
16#0000	No error	-
16#8200	“minValue” is greater than “maxValue”.	-
16#8600	Error in “RD_SYS_T” command.	Check the error code in “subFunctionStatus”

Note The status of called commands is output in “subFunctionStatus”. In this case, the output value in “status” indicates which command caused the error. In this case, refer to the TIA Portal Online Help section for information on the respective commands.

Principle of operation

The block generates random values that are between the specified “minValue” value and the “maxValue” value. This random value is output via the “Ret_Val”.

Background information

The random value is formed from the nanoseconds of the current system time of the CPU. The byte sequence of this value is inverted and then transformed into a normalized floating point number.

3.4.15 LGF_SearchMinMax

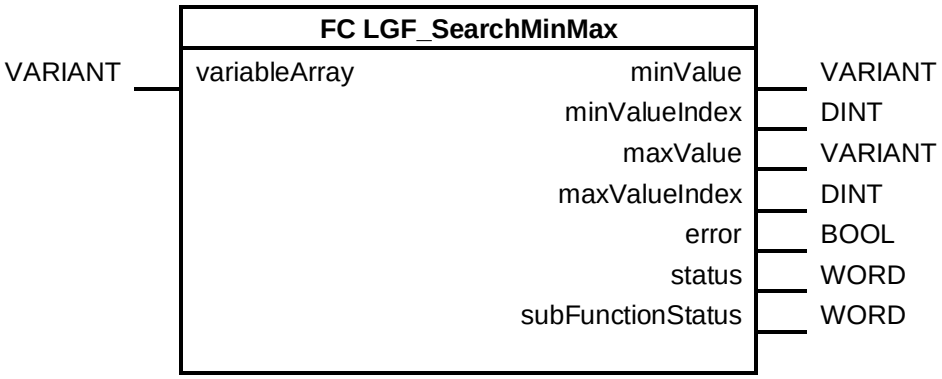
Short description

This block searches in an array for the maximum and minimum value, and for the respective index in the array.

The following data types of the array elements are supported:
Int, DInt, UInt, UDInt, USInt, SInt, and Real.

Block

Figure 3-41: FC LGF_SearchMinMax



Input parameters

Table 3-89: Input parameters

Parameters	Data type	Description
variableArray	VARIANT	Array in whose fields the maximum and minimum are searched.

Output parameters

Table 3-90: Output parameters

Parameters	Data type	Description
minValue	VARIANT	Smallest found value.
minValueIndex	DINT	Start index of the array plus minArrayIndex results in the array index of the smallest value. The index starts with 0.
maxValue	VARIANT	Largest found value.
maxValueIndex	DINT	Start index of the array plus maxArrayIndex results in the array index of the largest value. The index starts with 0.
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).
subFunctionStatus	WORD	Status or return value of the called FCs and system blocks.

Status and error displays

Table 3-91: Status/error codes

status	Meaning	Remedy / notes
16#0000	No error	-
16#8200	At input “variableArray” the actual parameter is not an array.	-
16#8201	The data type of the elements in the array is not supported.	Only the data types Int, UInt, DInt, UDInt, USInt, SInt and Real are supported.
16#8202	The elements of the array do not have the same data type as the outputs “minValue” and “maxValue”.	-
16#8203	Error in “MOVE_BLK_VARIANT” command.	Check the error code in “subFunctionStatus”

Note The status of called commands is output in “subFunctionStatus”. In this case, the output value in “status” indicates which command caused the error. In this case, refer to the TIA Portal Online Help section for information on the respective commands.

Principle of operation

An array of any size is connected via the “variableArray” input. After a data type query in the block, the elements are copied one after the other into a variable of the appropriate type and compared. The smallest and largest values, as well as their corresponding index are output to the array.

Note If there are several identical min. or max. values, the index of the first min. or max. value is output.

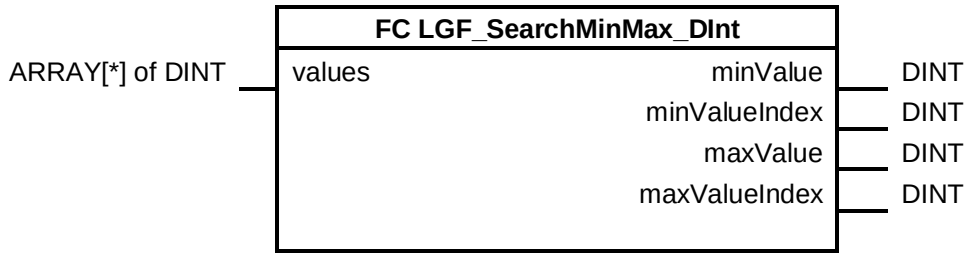
3.4.16 LGF_SearchMinMax_DInt

Short description

This function block searches, in an array of the data type DINT, for the maximum and minimum value and the respective index in the array.

Block

Figure 3-42: FC LGF_SearchMinMax_DInt



Input/output parameters (InOut)

Table 3-92: Input/output parameters (InOut)

Parameters	Data type	Description
values	ARRAY[*] of DINT	Array in whose fields the maximum and minimum are searched.

Output parameters

Table 3-93: Output parameters

Parameters	Data type	Description
minValue	DINT	Smallest found value.
minValueIndex	DINT	Start index of the array plus minArrayIndex results in the array index of the smallest value. The index starts with 0.
maxValue	DINT	Largest found value.
maxValueIndex	DINT	Start index of the array plus maxArrayIndex results in the array index of the largest value. The index starts with 0.

Principle of operation

An array of any size is connected via the “values” input. The elements are then compared in turn. The smallest and largest values, as well as their corresponding index are output to the array.

Note If there are several identical min. or max. values, the index of the first min. or max. value is output.

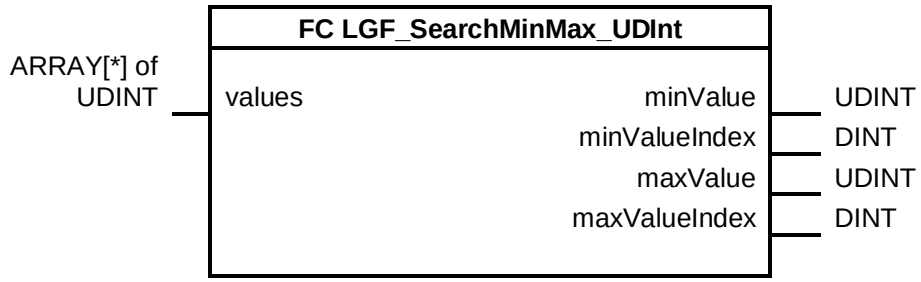
3.4.17 LGF_SearchMinMax_UDInt

Short description

This function block searches, in an array of the data type UDINT, for the maximum and minimum value and the respective index in the array.

Block

Figure 3-43: FC LGF_SearchMinMax_UDInt



Input/output parameters (InOut)

Table 3-94: Input/output parameters (InOut)

Parameters	Data type	Description
values	ARRAY[*] of UDINT	Array in whose fields the maximum and minimum are searched.

Output parameters

Table 3-95: Output parameters

Parameters	Data type	Description
minValue	UDINT	Smallest found value.
minValueIndex	DINT	Start index of the array plus minArrayIndex results in the array index of the smallest value. The index starts with 0.
maxValue	UDINT	Largest found value.
maxValueIndex	DINT	Start index of the array plus maxArrayIndex results in the array index of the largest value. The index starts with 0.

Principle of operation

An array of any size is connected via the “values” input. The elements are then compared in turn. The smallest and largest values, as well as their corresponding index are output to the array.

Note If there are several identical min. or max. values, the index of the first min. or max. value is output.

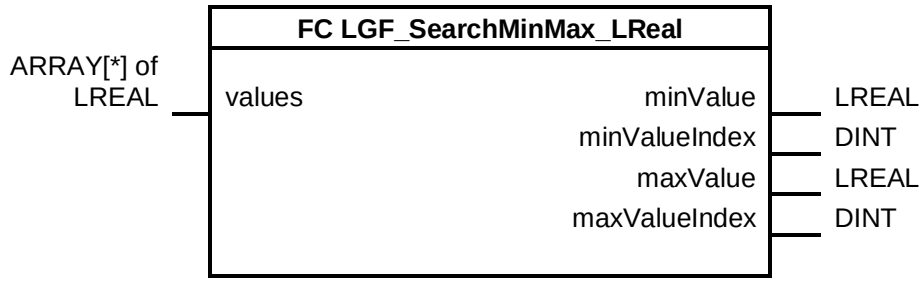
3.4.18 LGF_SearchMinMax_LReal

Short description

This function block searches, in an array of the data type LREAL, for the maximum and minimum value and the respective index in the array.

Block

Figure 3-44: FC LGF_SearchMinMax_LReal



Input/output parameters (InOut)

Table 3-96: Input/output parameters (InOut)

Parameters	Data type	Description
values	ARRAY[*] of LREAL	Array in whose fields the maximum and minimum are searched.

Output parameters

Table 3-97: Output parameters

Parameters	Data type	Description
minValue	LREAL	Smallest found value.
minValueIndex	DINT	Start index of the array plus minArrayIndex results in the array index of the smallest value. The index starts with 0.
maxValue	LREAL	Largest found value.
maxValueIndex	DINT	Start index of the array plus maxArrayIndex results in the array index of the largest value. The index starts with 0.

Principle of operation

An array of any size is connected via the “values” input. The elements are then compared in turn. The smallest and largest values, as well as their corresponding index are output to the array.

Note If there are several identical min. or max. values, the index of the first min. or max. value is output.

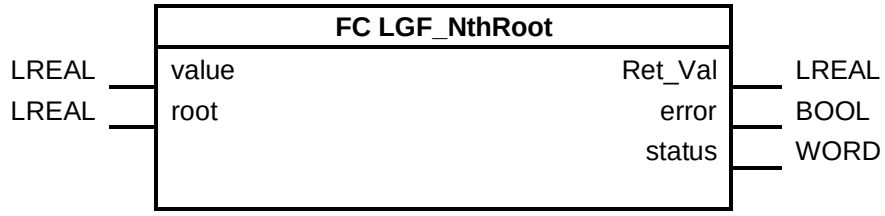
3.4.19 LGF_NthRoot

Short description

This block calculates the nth root of a variable.

Block

Figure 3-45: FC LGF_NthRoot



Input parameters

Table 3-98: Input parameters

Parameters	Data type	Description
value	LREAL	Variable from which the root should be calculated.
root	LREAL	Root (e.g. 3 as 3rd root)

Output parameters

Table 3-99: Output parameters

Parameters	Data type	Description
Ret_Val	LREAL	Output of the result
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).

Status and error displays

Table 3-100: Status/error codes

status	Meaning	Remedy / notes
16#0000	No error	-
16#8200	Error: Negative value for variable “value” not allowed.	-

Principle of operation

The block calculates the nth root from a number.

The root is defined as follows:

$$Ret\_Val = \sqrt[root]{number} = number^{\frac{1}{root}}$$

STEP 7 (TIA Portal) results in the following formula:

$$Ret_{Val} = number ** (1/root)$$

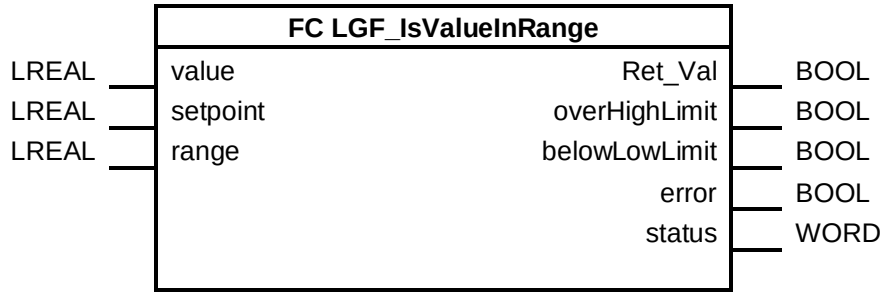
3.4.20 LGF_IsValueInRange

Short description

The function checks whether a value is within a defined value range. The value range is defined with a set point and a range around this set point. The function calculates the low limit and high limit of the value range.

Block

Figure 3-46: FC LGF_IsValueInRange



Input parameters

Table 3-101: Input parameters

Parameters	Data type	Description
value	LREAL	Value to be checked to determine whether it is within the defined value range
setpoint	LREAL	Set point
range	LREAL	Area in which the set point is within the range

Output parameters

Table 3-102: Output parameters

Parameters	Data type	Description
Ret_Val	BOOL	“TRUE” if the “value” is in the value range (range of the set point).
overHighLimit	BOOL	“TRUE” if the “value” is greater than the upper limit value (“setpoint” + 0.5 * “range”).
belowLowLimit	BOOL	“TRUE”, if the “value” is less than the lower limit value (“setpoint” - 0.5 * “range”).
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).

Status and error displays

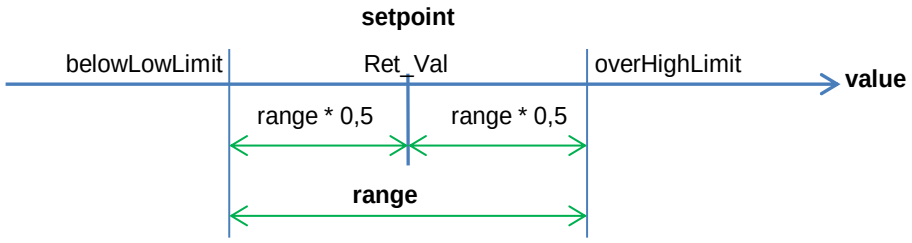
Table 3-103: Status/error codes

Status	Meaning	Remedy / notes
16#0000	No error	-
16#8401	Error in the calculation of the limit values	-

Principle of operation

The “setpoint” and “range” variables define a range of values. The function checks whether the “value” is below, in or above the value range. The outputs “belowLowLimit”, “Ret_Val”, or “overHighLimit” show where the “value” is located.

Figure 3-47: Principle of operation



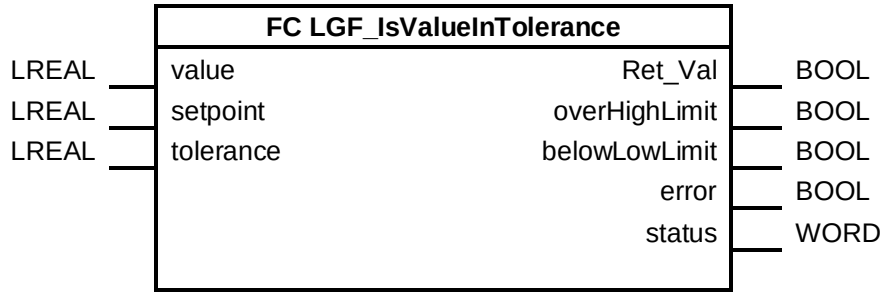
3.4.21 LGF_IsValueInTolerance

Short description

The function checks whether a value is within a defined value range. The value range is defined with a set point, as well as a tolerance range, around the set point in percent (%). The function calculates the low limit and high limit of the value range.

Block

Figure 3-48: FC LGF_IsValueInTolerance



Input parameters

Table 3-104: Input parameters

Parameters	Data type	Description
value	LREAL	Value to be checked to determine whether it is within the defined value range
setpoint	LREAL	Set point
tolerance	LREAL	Tolerance range around the set point in percent (%)

Output parameters

Table 3-105: Output parameters

Parameters	Data type	Description
Ret_Val	BOOL	“TRUE” if the “value” is in the value range (range of the set point).
overHighLimit	BOOL	“TRUE” if the “value” is greater than the upper limit value (“setpoint” * (1.0 + “tolerance” / 100%).
belowLowLimit	BOOL	“TRUE”, if the “value” is less than the lower limit value (“setpoint” * (1.0 - “tolerance” / 100%).
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).

Status and error displays

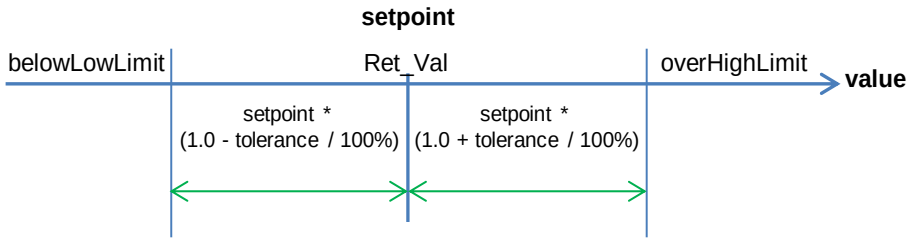
Table 3-106: Status/error codes

Status	Meaning	Remedy / notes
16#0000	No error	-
16#8401	Error in the calculation of the limit values	-

Principle of operation

The “setpoint” and “tolerance” percentage variables define a value range. The function checks whether the “value” is below, in or above the value range. The outputs “belowLowLimit”, “Ret_Val”, or “overHighLimit” show where the “value” is located.

Figure 3-49: Principle of operation



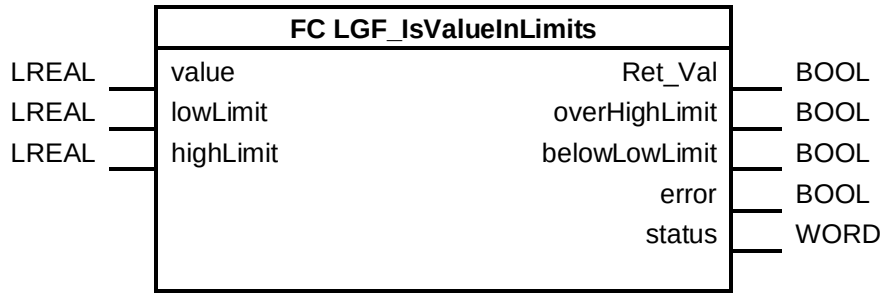
3.4.22 LGF_IsValueInLimits

Short description

The function checks whether a value is within a defined value range. The value range is defined with a lower and an upper limit.

Block

Figure 3-50: FC LGF_IsValueInLimits



Input parameters

Table 3-107: Input parameters

Parameters	Data type	Description
value	LREAL	Value to be checked to determine whether it is within the defined value range
lowLimit	LREAL	Lower limit of the value range
highLimit	LREAL	Upper limit of the value range

Output parameters

Table 3-108: Output parameters

Parameters	Data type	Description
Ret_Val	BOOL	“TRUE”, if the “value” is in the value range between lower and upper limit.
overHighLimit	BOOL	“TRUE” if “value” is greater than the upper limit value “highLimit”.
belowLowLimit	BOOL	“TRUE”, if “value” is smaller than the lower limit value “lowLimit”.
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).

Status and error displays

Table 3-109: Status/error codes

Status	Meaning	Remedy / notes
16#0000	No error	-
16#8401	Error, upper limit value smaller than lower limit value	-

Principle of operation

The variables “lowLimit” and “highLimit” define a value range. The function checks whether the “value” is below, in or above the value range. The outputs “belowLowLimit”, “Ret_Val”, or “overHighLimit” show where the “value” is located.

Figure 3-51: Principle of operation

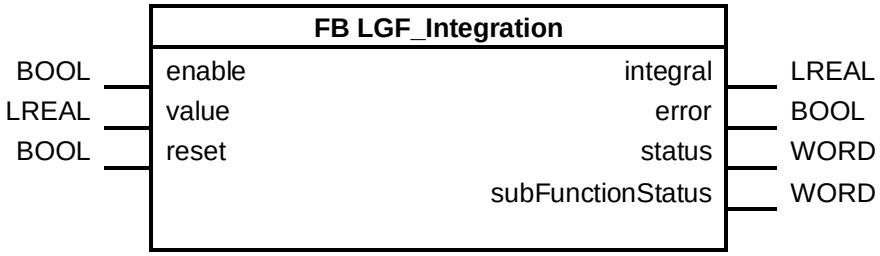
3.4.23 LGF_Integration

Short description

This block approximately calculates the area under a function curve. The function curve is transferred as an analog value (LREAL) which varies over time. The integral value is output on the output.

Block

Figure 3-52: FB LGF_Integration



Input parameters

Table 3-110: Input parameters

Parameters	Data type	Description
enable	BOOL	Activation of the integral calculation. If this input is set to the value “FALSE”, the integral calculation is stopped and the “integral” output shows the last calculated value.
value	LREAL	Analog value of the continuous function curve
reset	BOOL	Sets the output “integral” to “0.0”.

Output parameters

Table 3-111: Output parameters

Parameters	Data type	Description
integral	LREAL	Integrated value
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).
subFunctionStatus	WORD	Status or return value of the called FCs and system blocks.

Status and error displays

Table 3-112: Status/error codes

status	Meaning	Remedy / notes
16#0000	No error	-
16#8600	Error in “RD_SYS_T” command.	Check the error code in “subFunctionStatus”

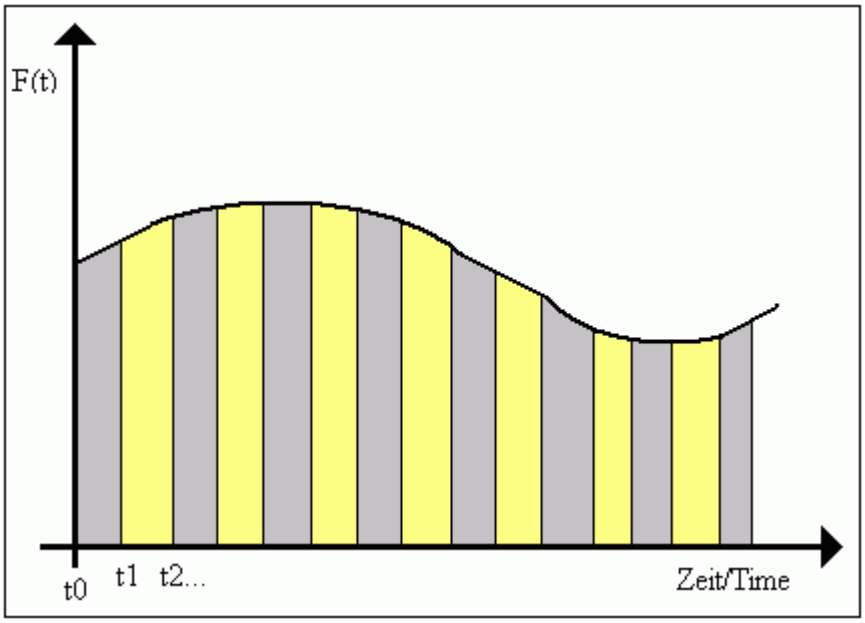
Note The status of called commands is output in “subFunctionStatus”. In this case, the output value in “status” indicates which command caused the error. In this case, refer to the TIA Portal Online Help section for information on the respective commands.

Principle of operation

The integral calculation includes the summation of those trapezoidal areas that span between the last two function values on the “value” input and the time. The elapsed time is calculated via the system time of the CPU. This trapezoidal area is identical to the product of the mean value of the two process values and the time interval.

$$A = \frac{1}{2} * \left(F_{(t1)} + F_{(t0)}\right) * (t1 - t0) + \frac{1}{2} * \left(F_{(t2)} + F_{(t1)}\right) * (t2 - t1) + \cdots$$

Figure 3-53: Principle of operation



To start the integral calculation for the input value on the “value” parameter, you must

- set the parameter “enable” to the value “TRUE”,
- set the parameter “reset” to the value “FALSE “.

If the parameter “enable” is set to the value “FALSE”, the integral calculation is stopped and the output “integral” outputs the last calculated value.

If the parameter “reset” is set to the value “TRUE”, the output “integral” is reset to “0.0”.

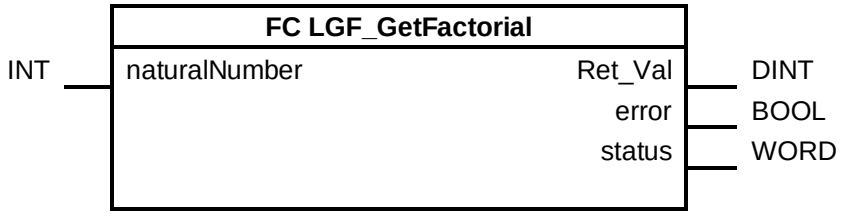
3.4.24 LGF_GetFactorial

Short description

The function calculates the faculty of a natural number.

Block

Figure 3-54: LGF_Factorial



Input parameters

Table 3-113: Input parameters

Parameters	Data type	Description
naturalNumber	INT	Natural number (0..12)

Output parameters

Table 3-114: Output parameters

Parameters	Data type	Description
Ret_Val	DINT	Calculated faculty
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).

Status and error displays

Table 3-115: Status/error codes

Status	Meaning	Remedy / notes
16#0000	No error	-
16#8101	Wrong value range for input parameter “naturalNumber”.	Enter a value between 0 and 12.

Principle of operation

The function calculates the faculty of a natural number. The permissible value range of the input parameter “naturalNumber” is between 0 and 12.

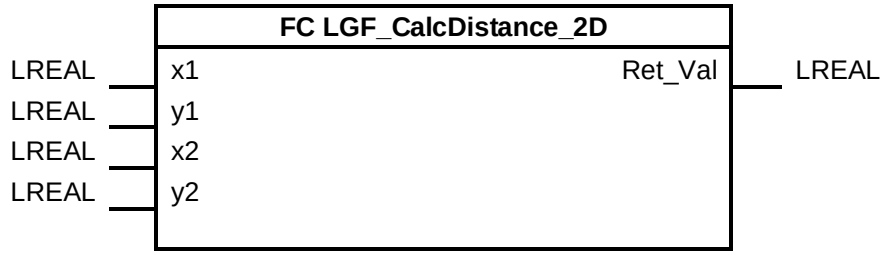
3.4.25 LGF_CalcDistance_2D

Short description

The function calculates the distance between two points in the plane.

Block

Figure 3-55: FC LGF_CalcDistance_2D



Input parameters

Table 3-116: Input parameters

Parameters	Data type	Description
x1	LREAL	X coordinate point 1
y1	LREAL	Y coordinate point 1
x2	LREAL	X coordinate point 2
y2	LREAL	Y-coordinate point 2

Output parameters

Table 3-117: Output parameters

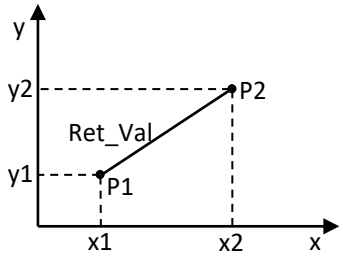
Parameters	Data type	Description
Ret_Val	LREAL	Calculated distance

Principle of operation

The block calculates the distance between two points in a Cartesian coordinate system. The distance is calculated with the following formula:

$$Ret\_Val = \sqrt[root]{(x2 - x1)^2 + (y2 - y1)^2}$$

Figure 3-56: Graphical representation



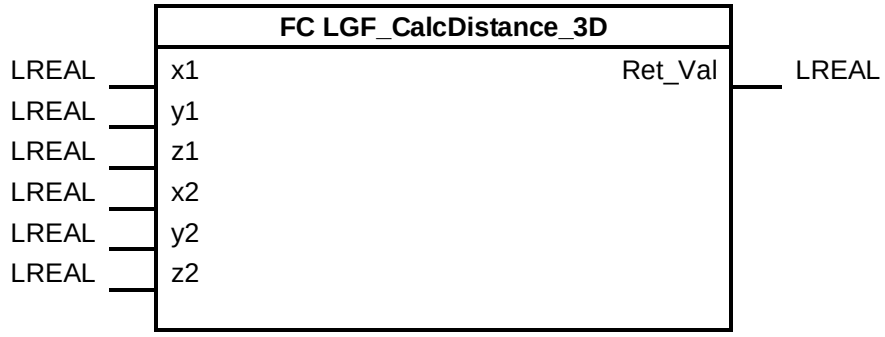
3.4.26 LGF_CalcDistance_3D

Short description

The function calculates the distance between two points in space.

Block

Figure 3-57: FC LGF_CalcDistance_3D



Input parameters

Table 3-118: Input parameters

Parameters	Data type	Description
x1	LREAL	X coordinate point 1
y1	LREAL	Y coordinate point 1
z1	LREAL	Z-coordinate point 1
x2	LREAL	X coordinate point 2
y2	LREAL	Y-coordinate point 2
z2	LREAL	Z-coordinate point 2

Output parameters

Table 3-119: Output parameters

Parameters	Data type	Description
Ret_Val	LREAL	Calculated distance

Principle of operation

The block calculates the distance between two points in a Cartesian coordinate system. The distance is calculated with the following formula:

$$Ret\_Val = \sqrt[root]{(x2 - x1)^2 + (y2 - y1)^2 + (z2 - z1)^2}$$

3.5 Data handling

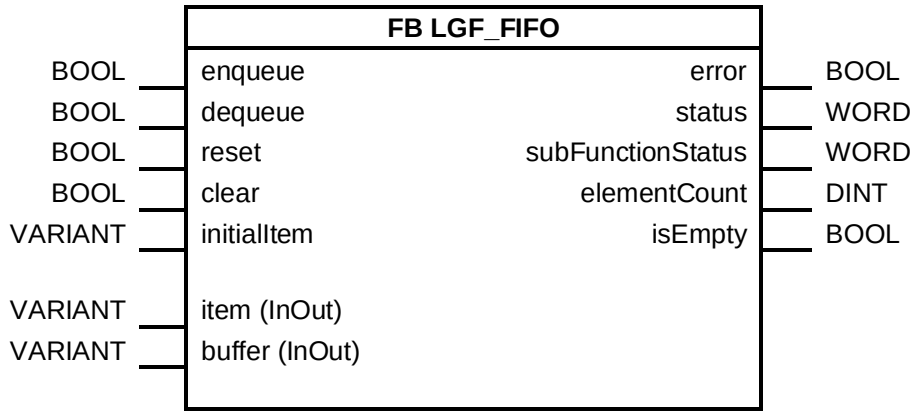
3.5.1 LGF_FIFO

Short description

This block stores incoming data and outputs the oldest unprocessed data.

Block

Figure 3-58: FB LGF_FIFO



Input parameters

Table 3-120: Input parameters

Parameters	Data type	Description
enqueue	BOOL	Row data into the buffer
dequeue	BOOL	Remove data from the buffer
reset	BOOL	Initialize buffer (reset index and counter)
clear	BOOL	Empty the buffer and initialize with the initial value “initialItem” (Reset index and counter).
initialItem	VARIANT	Value for initializing the buffer (usually: 0)

Input/output parameters (InOut)

Table 3-121: Input/output parameters (InOut)

Parameters	Data type	Description
item	VARIANT	Value that will be returned from the buffer or written into the ring buffer.
buffer	VARIANT	Buffer (Array of…)

Output parameters

Table 3-122: Output parameters

Parameters	Data type	Description
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).
subFunctionStatus	WORD	Status or return value of the called FCs and system blocks.
elementCount	DINT	Number of elements in the buffer
isEmpty	BOOL	TRUE: Buffer is empty

Status and error displays

Table 3-123: Status/error codes

status	Meaning	Remedy / notes
16#0000	No error	-
16 #7000	Initial value	-
16#8001	The buffer is empty.	-
16#8002	The buffer is full.	-
16#8200	There is no array on the “buffer” input.	-
16#8201	The data type of the InOut parameter “item” does not equal the data type of the array elements of the input “buffer”.	-
16#8202	The data type of the “initialItem” input does not correspond to the data type of the InOut parameter “item”.	-
16#8601	The variable “nextEmptyItemIndex” does not lie within the array boundaries.	-
16#8602	The variable “firstItemIndex” is not within the array boundaries.	-
16#8610	Error in “MOVE_BLK_VARIANT” command. (clearing buffer)	Check the error code in “subFunctionStatus”
16#8611	Error in “MOVE_BLK_VARIANT” command. (return first entry of buffer)	Check the error code in “subFunctionStatus”
16#8612	Error in “MOVE_BLK_VARIANT” command. (replace item by initial value)	Check the error code in “subFunctionStatus”
16#8613	Error in “MOVE_BLK_VARIANT” command. (write entry to buffer)	Check the error code in “subFunctionStatus”

Note The status of called commands is output in “subFunctionStatus”. In this case, the output value in “status” indicates which command caused the error. In this case, refer to the TIA Portal Online Help section for information on the respective commands.

Note In computer science, the queue is also based on the FIFO principle.

Principle of operation

With the “enqueue” input, a new item is stored to the InOut parameter “item” in the next free position in the buffer. The output “elementCount” is incremented by the value “1”.

With the “dequeue” input, the next element to be processed is output to the InOut parameter “item”, and this field in the buffer is replaced by the value in the parameter “initialItem”. The output “elementCount” is decremented by the value “1”.

With the “reset” input, the buffer is initialized and the index and counter are reset. The “elementCount” output is set to “0” and the “isEmpty” output is set to TRUE.

With the “clear” input, the buffer is emptied and initialized with the initial value “initialItem”. Index and counter are reset. The “elementCount” output is set to “0” and the “isEmpty” output is set to TRUE.

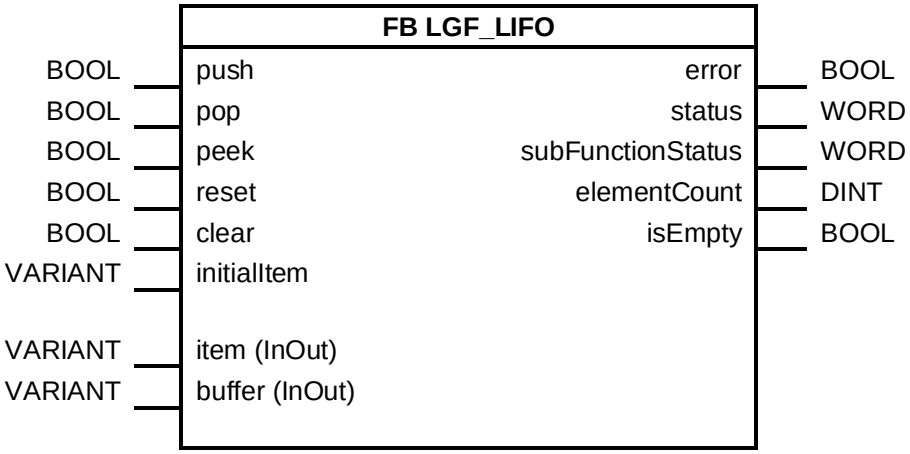
3.5.2 LGF_LIFO

Short description

This block stores incoming data and outputs the latest/most recent not-yet-processed data.

Block

Figure 3-59: FB LGF_LIFO



Input parameters

Table 3-124: Input parameters

Parameters	Data type	Description
push	BOOL	Sending data to the buffer
pop	BOOL	Get data from the buffer
peek	BOOL	View data in buffer (the buffer is not changed)
reset	BOOL	Initialize buffer (reset index and counter)
clear	BOOL	Empty the buffer and initialize with the initial value “initialItem” (Reset index and counter).
initialItem	VARIANT	Value for initializing the buffer (usually: 0)

Input/output parameters (InOut)

Table 3-125: Input/output parameters (InOut)

Parameters	Data type	Description
item	VARIANT	Value that will be returned from the buffer or written into the ring buffer.
buffer	VARIANT	Buffer (Array of…)

Output parameters

Table 3-126: Output parameters

Parameters	Data type	Description
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).
subFunctionStatus	WORD	Status or return value of the called FCs and system blocks.
elementCount	DINT	Number of elements in the buffer
isEmpty	BOOL	TRUE: Buffer is empty

Status and error displays

Table 3-127: Status/error codes

status	Meaning	Remedy / notes
16#0000	No error	-
16 #7000	Initial value	-
16#8001	The buffer is empty.	-
16#8002	The buffer is full.	-
16#8200	There is no array on the “buffer” input.	-
16#8201	The data type of the InOut parameter “item” does not equal the data type of the array elements of the input “buffer”.	-
16#8202	The data type of the “initialItem” input does not correspond to the data type of the InOut parameter “item”.	-
16#8601	The variable “nextEmptyItemIndex” does not lie within the array boundaries.	-
16#8602	The variable “firstItemIndex” is not within the array boundaries.	-
16#8610	Error in “MOVE_BLK_VARIANT” command. (clearing buffer)	Check the error code in “subFunctionStatus”
16#8611	Error in “MOVE_BLK_VARIANT” command. (return first entry of buffer)	Check the error code in “subFunctionStatus”
16#8612	Error in “MOVE_BLK_VARIANT” command. (replace item by initial value)	Check the error code in “subFunctionStatus”
16#8613	Error in “MOVE_BLK_VARIANT” command. (write entry to buffer)	Check the error code in “subFunctionStatus”

Note The status of called commands is output in “subFunctionStatus”. In this case, the output value in “status” indicates which command caused the error. In this case, refer to the TIA Portal Online Help section for information on the respective commands.

Note I computer science the stack is also based on the LIFO principle.

Principle of operation

With the “push” input, a new item is stored at the InOut parameter “item” at the next free position in the buffer. The output “elementCount” is incremented by the value “1”.

With the “pop” input, the latest/most recent item is output to the InOut parameter “item”, and this field in the buffer is replaced by the value at the parameter “initialItem”. The output “elementCount” is decremented by the value “1”.

The “peek” input allows the last entry in the buffer to be read out. The buffer is not changed.

With the “reset” input, the buffer is initialized and the index and counter are reset. The “elementCount” output is set to “0” and the “isEmpty” output is set to TRUE.

With the “clear” input, the buffer is emptied and initialized with the initial value “initialItem”. Index and counter are reset. The “elementCount” output is set to “0” and the “isEmpty” output is set to TRUE.

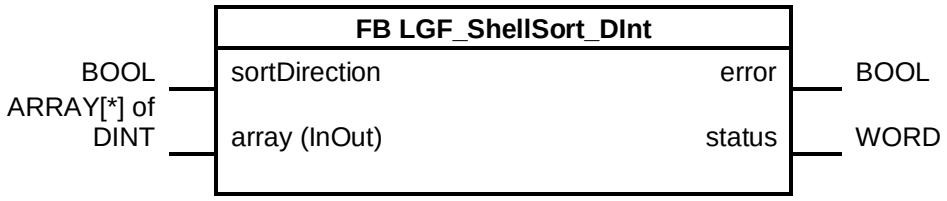
3.5.3 LGF_ShellSort_DInt

Short description

This block sorts an array of type “DInt” with any number of elements (max. 1000) in ascending or descending order.

Block

Figure 3-60: FB LGF_ShellSort_DInt



Input parameters

Table 3-128: Input parameters

Parameters	Data type	Description
sortDirection	BOOL	FALSE: sort in ascending order (default) TRUE: sort in descending order

Input/output parameters (InOut)

Table 3-129: Input/output parameters (InOut)

Parameters	Data type	Description
array	ARRAY[*] of DINT	Array to sort.

Output parameters

Table 3-130: Output parameters

Parameters	Data type	Description
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).

Status and error displays

Table 3-131: Status/error codes

status	Meaning	Remedy / notes
16#0000	No error	-
16#8200	Actual parameter at the “array” input has only one element.	At the “array” input, connect an array with at least two elements.
16#8201	Actual parameter at the “array” input has too many elements.	Connect an array with less than 1000 elements at input “array”.

Note The status of called commands is output in “subFunctionStatus”. In this case, the output value in “status” indicates which command caused the error. In this case, refer to the TIA Portal Online Help section for information on the respective commands.

Principle of operation

The block sorts according to the shell sort procedure. Note that the execution time of the block depends significantly on how many elements the array to be sorted has. The overview below shows several measured values of the block depending on the number of array elements.

Table 3-132: Execution times of the block “LGF_ShellSort…”

Number of array elements	SIMATIC S7-1212C DC/DC/DC	SIMATIC S7-1516-3 PN/DP
100	approx. 11-16 ms	approx. 1-2 ms
1000	approx. 185-205 ms	approx. 10-12 ms

Note The block is executed synchronously and is not split over several PLC cycles. Thus the execution time has a direct effect on the PLC cycle time. Note this behavior for your project of the controller used and adjust the monitoring time of the controller if necessary.

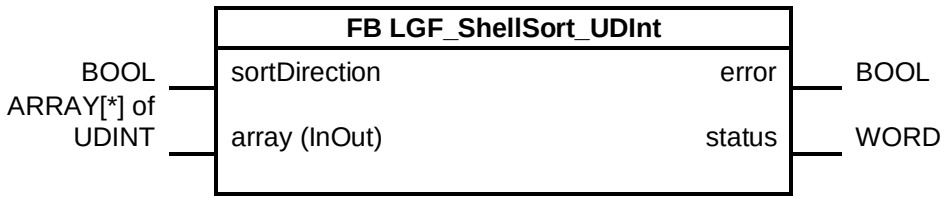
3.5.4 LGF_ShellSort_UDInt

Short description

This block sorts an array of the type “UDInt” with any number of elements (max. 1000) in ascending or descending order.

Block

Figure 3-61: FB LGF_ShellSort_UDInt



Input parameters

Table 3-133: Input parameters

Parameters	Data type	Description
sortDirection	BOOL	FALSE: sort in ascending order (default) TRUE: sort in descending order

Input/output parameters (InOut)

Table 3-134: Input/output parameters (InOut)

Parameters	Data type	Description
array	ARRAY[*] of UDINT	Array to sort.

Output parameters

Table 3-135: Output parameters

Parameters	Data type	Description
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).

Status and error displays

Table 3-136: Status/error codes

status	Meaning	Remedy / notes
16#0000	No error	-
16#8200	Actual parameter at the “array” input has only one element.	At the “array” input, connect an array with at least two elements.
16#8201	Actual parameter at the “array” input has too many elements.	Connect an array with less than 1000 elements at input “array”.

Note The status of called commands is output in “subFunctionStatus”. In this case, the output value in “status” indicates which command caused the error. In this case, refer to the TIA Portal Online Help section for information on the respective commands.

Principle of operation

The block sorts according to the shell sort procedure. Note that the execution time of the block depends significantly on how many elements the array to be sorted has. The overview below shows several measured values of the block depending on the number of array elements.

Table 3-137: Execution times of the block “LGF_ShellSort…”

Number of array elements	SIMATIC S7-1212C DC/DC/DC	SIMATIC S7-1516-3 PN/DP
100	approx. 11-16 ms	approx. 1-2 ms
1000	approx. 185-205 ms	approx. 10-12 ms

Note The block is executed synchronously and is not split over several PLC cycles. Thus the execution time has a direct effect on the PLC cycle time. Note this behavior for your project of the controller used and adjust the monitoring time of the controller if necessary.

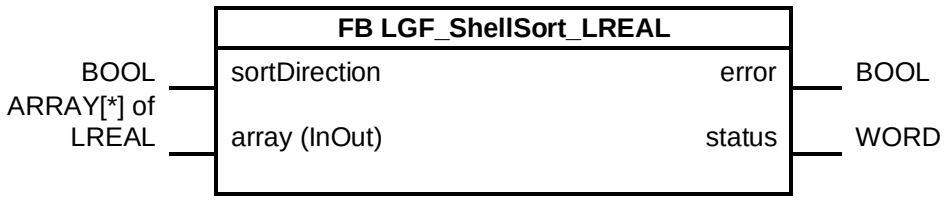
3.5.5 LGF_ShellSort_LReal

Short description

This block sorts an array, of the type “LREAL”, with any number of elements (max. 1000) in ascending or descending order.

Block

Figure 3-62: FB LGF_ShellSort_LREAL



Input parameters

Table 3-138: Input parameters

Parameters	Data type	Description
sortDirection	BOOL	FALSE: sort in ascending order (default) TRUE: sort in descending order

Input/output parameters (InOut)

Table 3-139: Input/output parameters (InOut)

Parameters	Data type	Description
array	ARRAY[*] of LREAL	Array to sort.

Output parameters

Table 3-140: Output parameters

Parameters	Data type	Description
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).

Status and error displays

Table 3-141: Status/error codes

status	Meaning	Remedy / notes
16#0000	No error	-
16#8200	Actual parameter at the “array” input has only one element.	At the “array” input, connect an array with at least two elements.
16#8201	Actual parameter at the “array” input has too many elements.	Connect an array with less than 1000 elements at input “array”.

Note The status of called commands is output in “subFunctionStatus”. In this case, the output value in “status” indicates which command caused the error. In this case, refer to the TIA Portal Online Help section for information on the respective commands.

Principle of operation

The block sorts according to the shell sort procedure. Note that the execution time of the block depends significantly on how many elements the array to be sorted has. The overview below shows several measured values of the block depending on the number of array elements.

Table 3-142: Execution times of the block “LGF_ShellSort…”

Number of array elements	SIMATIC S7-1212C DC/DC/DC	SIMATIC S7-1516-3 PN/DP
100	approx. 11-16 ms	approx. 1-2 ms
1000	approx. 185-205 ms	approx. 10-12 ms

Note The block is executed synchronously and is not split over several PLC cycles. Thus the execution time has a direct effect on the PLC cycle time. Note this behavior for your project of the controller used and adjust the monitoring time of the controller if necessary.

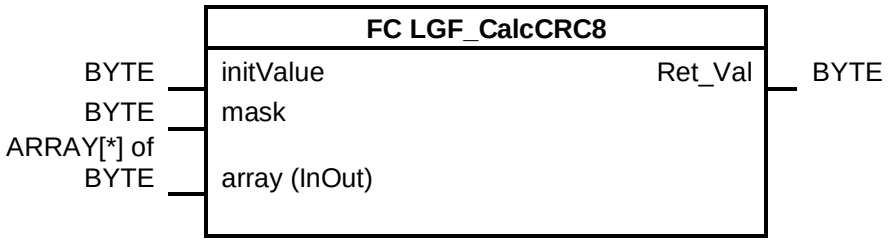
3.5.6 LGF_CalcCRC8

Short description

The CRC calculation is used for error detection at data transmission. The result of a calculation returns a CRC value via the data sent. The receiver detects a faulty transmission due to the unequal CRC value. The function “LGF_CalcCRC8” uses 8 bits as the generator polynomial (mask).

Block

Figure 3-63: FC LGF_CalcCRC8



Input parameters

Table-143: Input parameters

Parameters	Data type	Description
initValue	BYTE	Start value with which the calculation is executed. If you do not need a start value, assign 0x0 to the parameter.
mask	BYTE	Generator polynomial with which the calculation is executed.

Input/output parameters (InOut)

Table-144: Input/output parameters (InOut)

Parameters	Data type	Description
array	ARRAY[*] of BYTE	Data stream for which the CRC value will be calculated.

Output parameters

Table-145: Output parameters

Parameters	Data type	Description
Ret_Val	BYTE	Calculated CRC value (return value of the function).

Principle of operation

The block calculates the CRC value from a data stream of any size. The data stream is composed of the individual elements of the array at the input/output parameter “array”. The start value “initValue” and the generator polynomial “mask” can be freely selected.

Note Various online tools are available for calculating the CRC values. The function of the block was tested with the following online tool, since it supports the input parameters “mask” (“Polynomial”) and “initValue” (“Initial Value”):

http://www.sunshine2k.de/coding/javascript/crc/crc_js.html

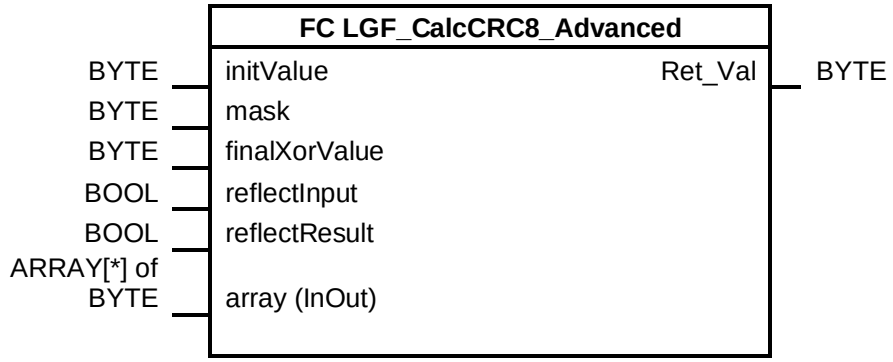
3.5.7 LGF_CalcCRC8Advanced

Short description

The CRC calculation is used for error detection at data transmission. The result of a calculation returns a CRC value via the data sent. The receiver detects a faulty transmission due to the unequal CRC value. The “LGF_CalcCRC8Advanced” function uses 8 bits as the generator polynomial (mask) and the parameters “finalXorValue”, “reflectInput”, and “reflectResult”.

Block

Figure 3-64: FC LGF_CalcCRC8Advanced



Input parameters

Table-146: Input parameters

Parameters	Data type	Description
initValue	BYTE	Start value with which the calculation is executed. If you do not need a start value, assign 0x0 to the parameter.
mask	BYTE	Generator polynomial with which the calculation is executed.
finalXorValue	BYTE	Value with which another XOR operation is performed at the end
reflectInput	BOOL	If the value is TRUE, the sequence of the bits within the input byte is mirrored. The sequence 0...7 becomes 7...0.
reflectResult	BOOL	If the value is TRUE, the order of the bits within the result is mirrored. The sequence 0...7 becomes 7...0.

Input/output parameters (InOut)

Table-147: Input/output parameters (InOut)

Parameters	Data type	Description
array	ARRAY[*] of BYTE	Data stream for which the CRC value will be calculated.

Output parameters

Table-148: Output parameters

Parameters	Data type	Description
Ret_Val	BYTE	Calculated CRC value (return value of the function).

Principle of operation

The block calculates the CRC value from a data stream of any size. The data stream is composed of the individual elements of the array at the input/output parameter “array”. The start value “initValue” and the generator polynomial “mask” can be freely selected.

Via the boolean input parameters “reflectInput” and “reflectResult”, you may optionally mirror the bits of the input data or the CRC value. An XOR operation is also performed with the CRC value at the end and the value “finalXorValue”.

Note Various online tools are available for calculating the CRC values. The function of the block was tested with the following online tool, since it supports the same input parameters:

http://www.sunshine2k.de/coding/javascript/crc/crc_js.html

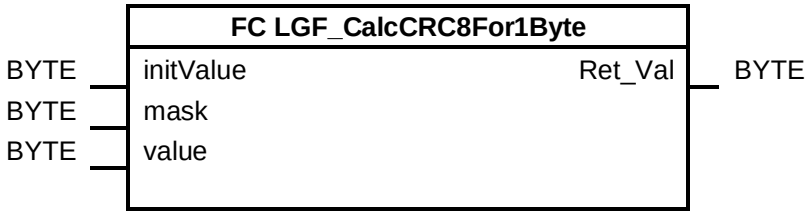
3.5.8 LGF_CalcCRC8For1Byte

Short description

The CRC calculation is used for error detection at data transmission. The result of the calculation provides a CRC value over the transmitted data (BYTE). The receiver detects a faulty transmission due to the unequal CRC value. The function “LGF_CalcCRC8For1Byte” uses 8 bits as the generator polynomial (mask).

Block

Figure 3-65: FC LGF_CalcCRC8For1Byte



Input parameters

Table-149: Input parameters

Parameters	Data type	Description
initValue	BYTE	Start value with which the calculation is executed. If you do not need a start value, assign 0x0 to the parameter.
mask	BYTE	Generator polynomial with which the calculation is executed.
value	BYTE	Data byte for which the CRC value will be calculated.

Output parameters

Table-150: Output parameters

Parameters	Data type	Description
Ret_Val	BYTE	Calculated CRC value (return value of the function).

Principle of operation

The block calculates the CRC value from a data byte “value”. The start value “initValue” and the generator polynomial “mask” can be freely selected.

Note Various online tools are available for calculating the CRC values. The function of the block was tested with the following online tool, since it supports the input parameters “mask” (“Polynomial”) and “initValue” (“Initial Value”):

http://www.sunshine2k.de/coding/javascript/crc/crc_js.html

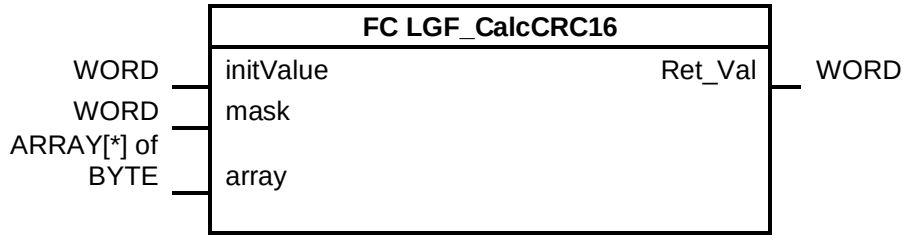
3.5.9 LGF_CalcCRC16

Short description

The CRC calculation is used for error detection at data transmission. The result of a calculation returns a CRC value via the data sent. The receiver detects a faulty transmission due to the unequal CRC value. The function “LGF_CalcCRC16” uses 16 bits as the generator polynomial (mask).

Block

Figure 3-66: FC LGF_CalcCRC16



Input parameters

Table-151: Input parameters

Parameters	Data type	Description
initValue	WORD	Start value with which the calculation is executed. If you do not need a start value, assign 0x0 to the parameter.
mask	WORD	Generator polynomial with which the calculation is executed.

Input/output parameters (InOut)

Table-152: Input/output parameters (InOut)

Parameters	Data type	Description
array	ARRAY[*] of BYTE	Data stream for which the CRC value will be calculated.

Output parameters

Table-153: Output parameters

Parameters	Data type	Description
Ret_Val	WORD	Calculated CRC value (return value of the function).

Principle of operation

The block calculates the CRC value from a data stream of any size. The data stream is composed of the individual elements of the array at the input/output parameter “array”. The start value “initValue” and the generator polynomial “mask” can be freely selected.

Note Various online tools are available for calculating the CRC values. The function of the block was tested with the following online tool, since it supports the input parameters “mask” (“Polynomial”) and “initValue” (“Initial Value”):

http://www.sunshine2k.de/coding/javascript/crc/crc_js.html

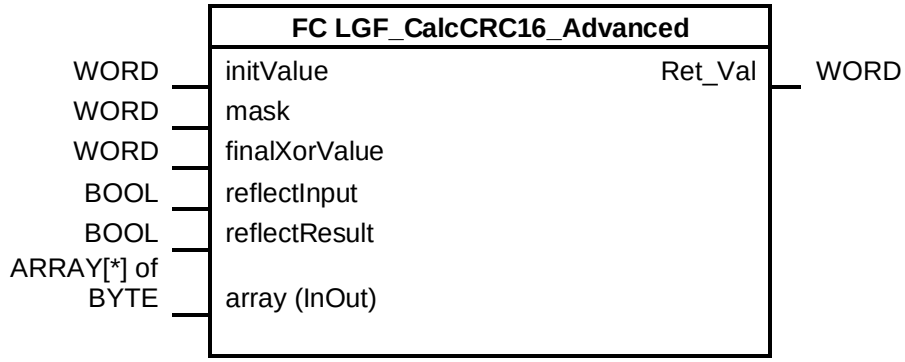
3.5.10 LGF_CalcCRC16Advanced

Short description

The CRC calculation is used for error detection at data transmission. The result of a calculation returns a CRC value via the data sent. The receiver detects a faulty transmission due to the unequal CRC value. The function "LGF_CalcCRC16Advanced" uses 16 bits as the generator polynomial (mask) and the parameters "finalXorValue", "reflectInput", and "reflectResult.

Block

Figure 3-67: FC LGF_CalcCRC16Advanced



Input parameters

Table-154: Input parameters

Parameters	Data type	Description
initValue	WORD	Start value with which the calculation is executed. If you do not need a start value, assign 0x0 to the parameter.
mask	WORD	Generator polynomial with which the calculation is executed.
finalXorValue	WORD	Value with which another XOR operation is performed at the end
reflectInput	BOOL	If the value is TRUE, the sequence of the bits within the input byte is mirrored. The sequence 0...7 becomes 7...0.
reflectResult	BOOL	If the value is TRUE, the order of the bits within the result is mirrored. The sequence 0…15 becomes 15…0.

Input/output parameters (InOut)

Table-155: Input/output parameters (InOut)

Parameters	Data type	Description
array	ARRAY[*] of BYTE	Data stream for which the CRC value will be calculated.

Output parameters

Table-156: Output parameters

Parameters	Data type	Description
Ret_Val	WORD	Calculated CRC value (return value of the function).

Principle of operation

The block calculates the CRC value from a data stream of any size. The data stream is composed of the individual elements of the array at the input/output parameter “array”. The start value “initValue” and the generator polynomial “mask” can be freely selected.

Via the boolean input parameters “reflectInput” and “reflectResult”, you may optionally mirror the bits of the input data or the CRC value. An XOR operation is also performed with the CRC value at the end and the value “finalXorValue”.

Note Various online tools are available for calculating the CRC values. The function of the block was tested with the following online tool, since it supports the same input parameters:

http://www.sunshine2k.de/coding/javascript/crc/crc_js.html

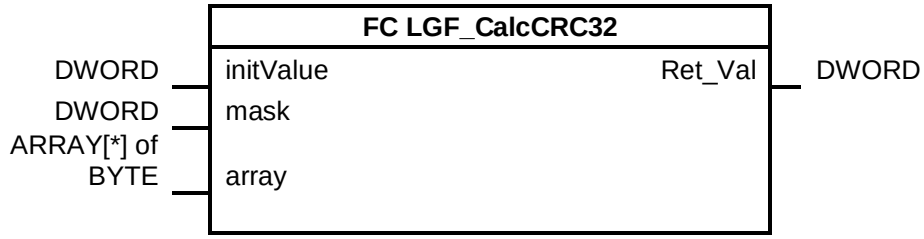
3.5.11 LGF_CalcCRC32

Short description

The CRC calculation is used for error detection at data transmission. The result of a calculation returns a CRC value via the data sent. The receiver detects a faulty transmission due to the unequal CRC value. The function “LGF_CalcCRC32” uses 32 bits as the generator polynomial (mask).

Block

Figure 3-68: FC LGF_CalcCRC32



Input parameters

Table-157: Input parameters

Parameters	Data type	Description
initValue	DWORD	Start value with which the calculation is executed. If you do not need a start value, assign 0x0 to the parameter.
mask	DWORD	Generator polynomial with which the calculation is executed.

Input/output parameters (InOut)

Table-158: Input/output parameters (InOut)

Parameters	Data type	Description
array	ARRAY[*] of BYTE	Data stream for which the CRC value will be calculated.

Output parameters

Table-159: Output parameters

Parameters	Data type	Description
Ret_Val	DWORD	Calculated CRC value (return value of the function).

Principle of operation

The block calculates the CRC value from a data stream of any size. The data stream is composed of the individual elements of the array at the input/output parameter “array”. The start value “initValue” and the generator polynomial “mask” can be freely selected.

Note Various online tools are available for calculating the CRC values. The function of the block was tested with the following online tool, since it supports the input parameters “mask” (“Polynomial”) and “initValue” (“Initial Value”):

http://www.sunshine2k.de/coding/javascript/crc/crc_js.html

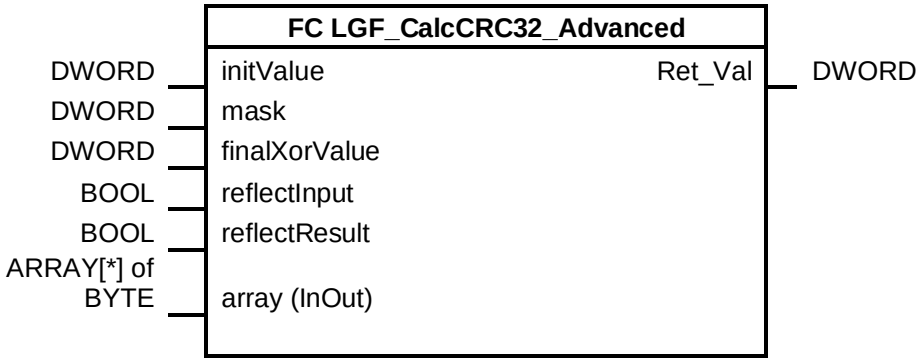
3.5.12 LGF_CalcCRC32Advanced

Short description

The CRC calculation is used for error detection at data transmission. The result of a calculation returns a CRC value via the data sent. The receiver detects a faulty transmission due to the unequal CRC value. The function "LGF_CalcCRC8Advanced" uses 32 bits as the generator polynomial (mask) and the parameters "finalXorValue", "reflectInput", and "reflectResult.

Block

Figure 3-69: FC LGF_CalcCRC32Advanced



Input parameters

Table-160: Input parameters

Parameters	Data type	Description
initValue	DWORD	Start value with which the calculation is executed. If you do not need a start value, assign 0x0 to the parameter.
mask	DWORD	Generator polynomial with which the calculation is executed.
finalXorValue	DWORD	Value with which another XOR operation is performed at the end
reflectInput	BOOL	If the value is TRUE, the sequence of the bits within the input byte is mirrored. The sequence 0...7 becomes 7...0.
reflectResult	BOOL	If the value is TRUE, the order of the bits within the result is mirrored. The sequence 0…31 becomes 31…0.

Input/output parameters (InOut)

Table-161: Input/output parameters (InOut)

Parameters	Data type	Description
array	ARRAY[*] of BYTE	Data stream for which the CRC value will be calculated.

Output parameters

Table-162: Output parameters

Parameters	Data type	Description
Ret_Val	DWORD	Calculated CRC value (return value of the function).

Principle of operation

The block calculates the CRC value from a data stream of any size. The data stream is composed of the individual elements of the array at the input/output parameter “array”. The start value “initValue” and the generator polynomial “mask” can be freely selected.

Via the boolean input parameters “reflectInput” and “reflectResult”, you may optionally mirror the bits of the input data or the CRC value. An XOR operation is also performed with the CRC value at the end and the value “finalXorValue”.

Note Various online tools are available for calculating the CRC values. The function of the block was tested with the following online tool, since it supports the same input parameters:

http://www.sunshine2k.de/coding/javascript/crc/crc_js.html

3.5.13 LGF_IsParityEven

Short description

The block checks whether the parity of the input variable of type DWORD is even. If the number of bits that are assigned “1” in the sequence is even, the return value is set to “TRUE”.

Block

Figure 3-70: FC LGF_IsParityEven

Input parameters

Table-163: Input parameters

Parameters	Data type	Description
doubleWord	DWORD	Variable for which the parity is to be determined.

Output parameters

Table-164: Output parameters

Parameters	Data type	Description
Ret_Val	BOOL	TRUE: When the number of bits that are assigned “1” is even

3.5.14 LGF_IsParityOdd

Short description

The block checks whether the parity of the input variable of type DWORD is odd. The return value is set to “TRUE” if the number of bits that are assigned “1” in the sequence is odd.

Block

Figure 3-71: FC LGF_IsParityOdd

Input parameters

Table-165: Input parameters

Parameters	Data type	Description
doubleWord	DWORD	Variable for which the parity is to be determined.

Output parameters

Table-166: Output parameters

Parameters	Data type	Description
Ret_Val	BOOL	TRUE: When the number of bits that are assigned “1” is odd

3.6 Converter operations

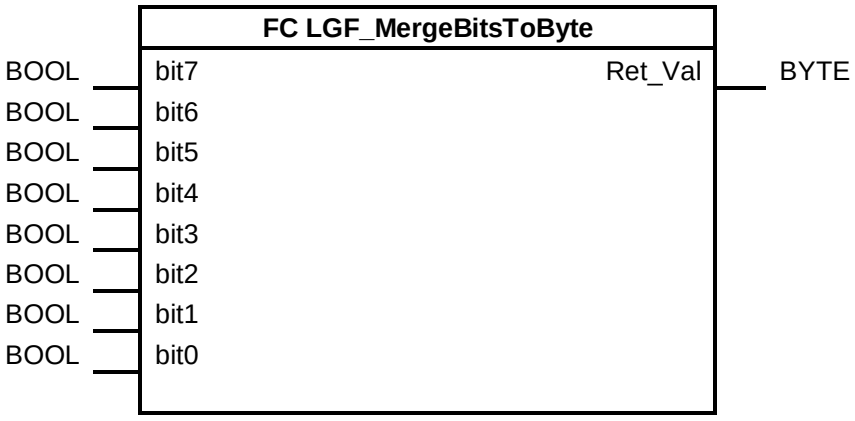
3.6.1 LGF_MergeBitsToByte

Short description

This block converts 8 Boolean variables into one Byte variable.

Block

Figure 3-72: FC LGF_MergeBitsToByte



Input parameters

Table 3-167: Input parameters

Parameters	Data type	Description
bit7 … bit0	BOOL	Bit variables, whose bits are combined to form a bit sequence

Output parameters

Table 3-168: Output parameters

Parameters	Data type	Description
Ret_Val	BYTE	Composite bit sequence stored in a variable

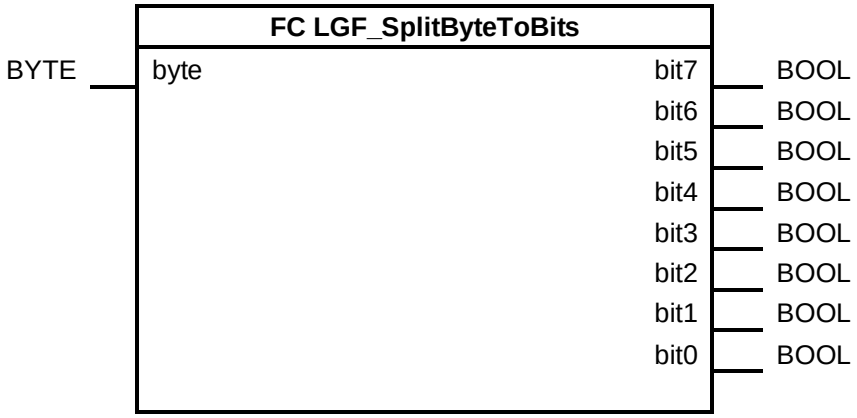
3.6.2 LGF_SplitByteToBits

Short description

This block converts a Byte variable into 8 Boolean variables.

Block

Figure 3-73: FC LGF_SplitByteToBits



Input parameters

Table 3-169: Input parameters

Parameters	Data type	Description
byte	BYTE	Bit sequence to be parsed

Output parameters

Table 3-170: Output parameters

Parameters	Data type	Description
bit7 … bit0	BOOL	Bit variables, in which the individual bits are stored

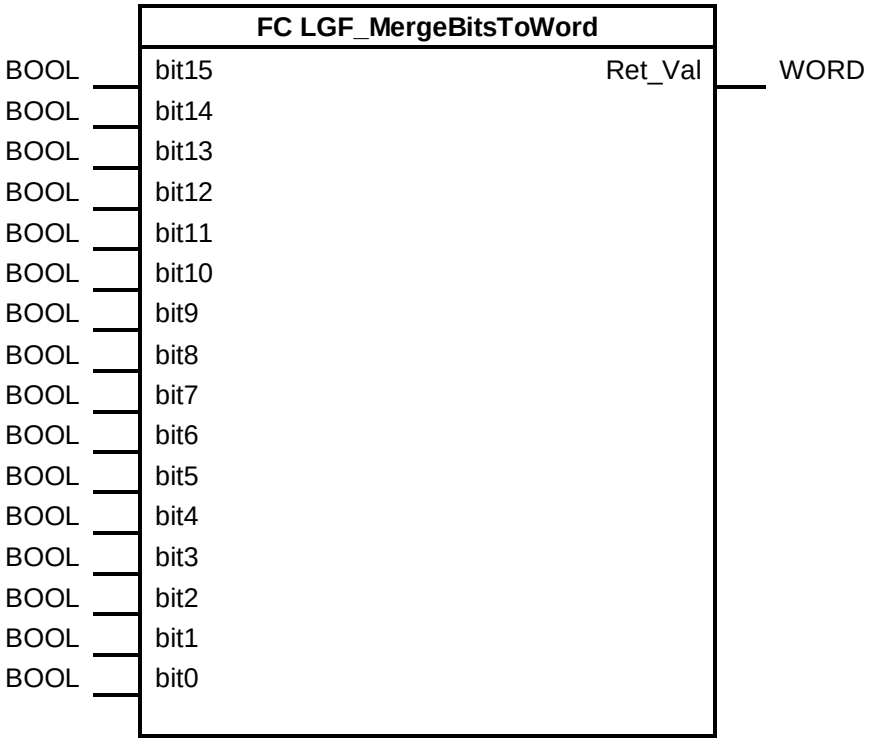
3.6.3 LGF_MergeBitsToWord

Short description

This block converts 16 BOOL variables into a WORD variable.

Block

Figure 3-74: FC LGF_MergeBitsToWord



Input parameters

Table 3-171: Input parameters

Parameters	Data type	Description
bit15 … bit0	BOOL	Bit variables, whose bits are combined to form a bit sequence

Output parameters

Table 3-172: Output parameters

Parameters	Data type	Description
Ret_Val	WORD	Composite bit sequence stored in a variable

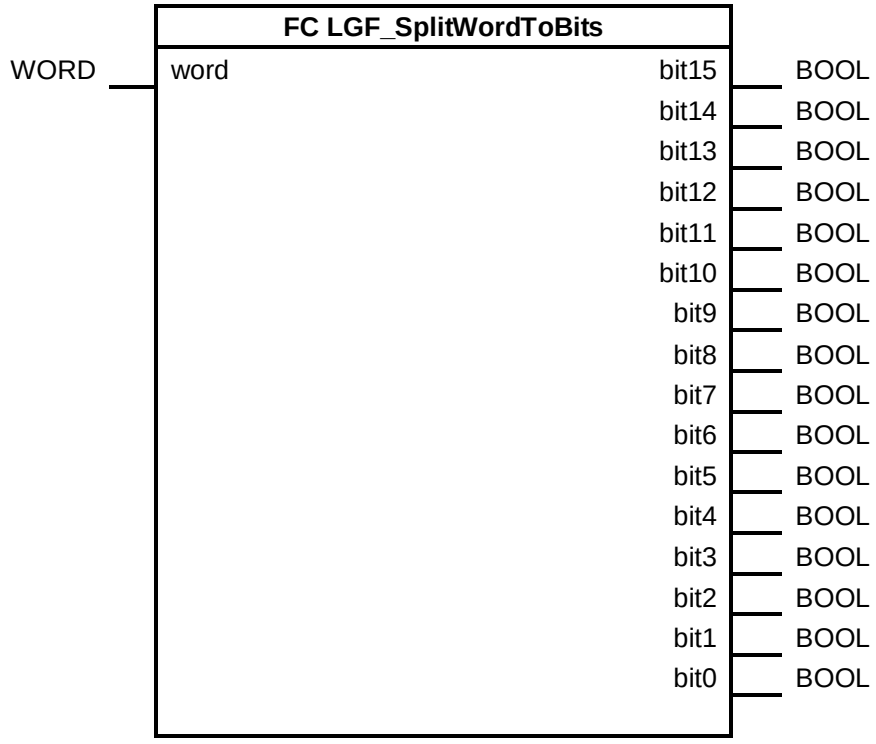
3.6.4 LGF_SplitWordToBits

Short description

This block converts a WORD variable into 16 BOOL variables.

Block

Figure 3-75: FC LGF_SplitWordToBits



Input parameters

Table 3-173: Input parameters

Parameters	Data type	Description
word	WORD	Bit sequence to be parsed

Output parameters

Table 3-174: Output parameters

Parameters	Data type	Description
bit15 … bit0	BOOL	Bit variables, in which the individual bits are stored

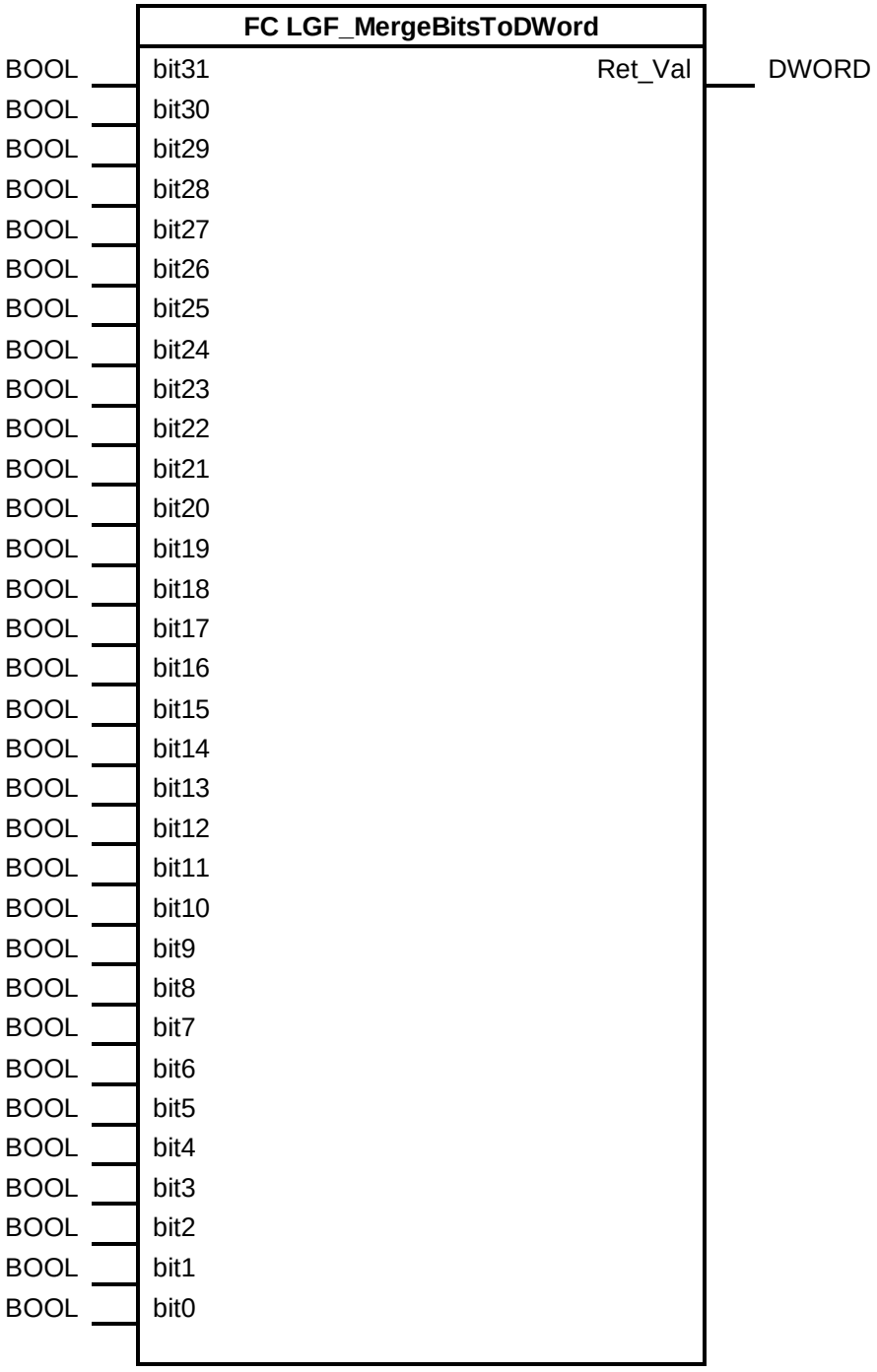
3.6.5 LGF_MergeBitsToDWord

Short description

This block converts 32 Boolean variables into a DWord variable.

Block

Figure 3-76: FC LGF_MergeBitsToDWord



Input parameters

Table 3-175: Input parameters

Parameters	Data type	Description
bit31 … bit0	BOOL	Bit variables, whose bits are combined to form a bit sequence

Output parameters

Table 3-176: Output parameters

Parameters	Data type	Description
Ret_Val	DWORD	Composite bit sequence stored in a variable

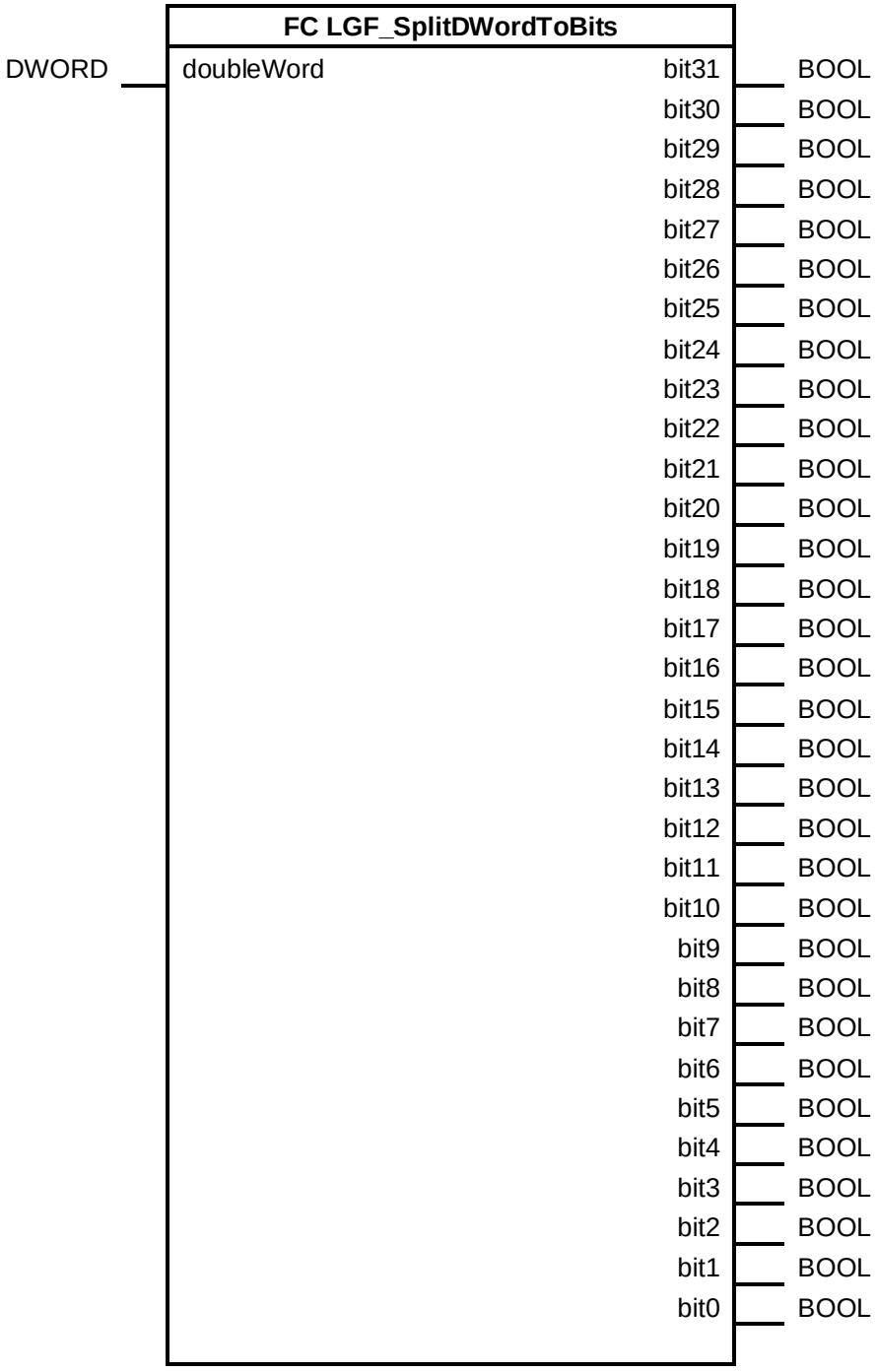
3.6.6 LGF_SplitDWordToBits

Short description

This block converts a DWord variable into 32 Boolean variables.

Block

Figure 3-77: FC LGF_SplitDWordToBits



Input parameters

Table 3-177: Input parameters

Parameters	Data type	Description
doubleWord	DWORD	Bit sequence to be parsed

Output parameters

Table 3-178: Output parameters

Parameters	Data type	Description
bit31 … bit0	BOOL	Bit variables, in which the individual bits are stored

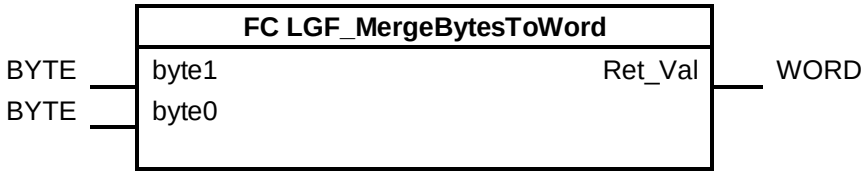
3.6.7 LGF_MergeBytesToWord

Short description

This block converts 2 Byte variables into a Word variable.

Block

Figure 3-78: FC LGF_MergeBytesToWord



Input parameters

Table 3-179: Input parameters

Parameters	Data type	Description
byte1 … byte0	BYTE	Byte variables, whose Bytes are combined into a word

Output parameters

Table 3-180: Output parameters

Parameters	Data type	Description
Ret_Val	WORD	Assembled Byte sequence, stored in a variable

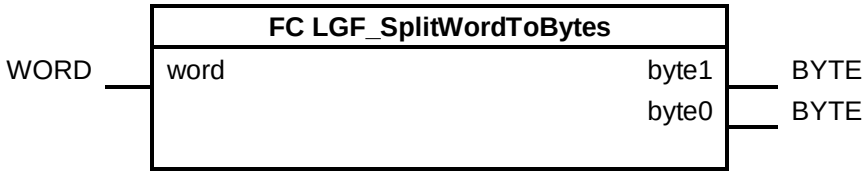
3.6.8 LGF_SplitWordToBytes

Short description

This block converts a Word variable into 2 Byte variables.

Block

Figure 3-79: FC LGF_SplitWordToBytes



Input parameters

Table 3-181: Input parameters

Parameters	Data type	Description
word	WORD	Byte sequence that is parsed

Output parameters

Table 3-182: Output parameters

Parameters	Data type	Description
byte1 … byte0	BYTE	Byte variables, in which the individual Bytes are stored

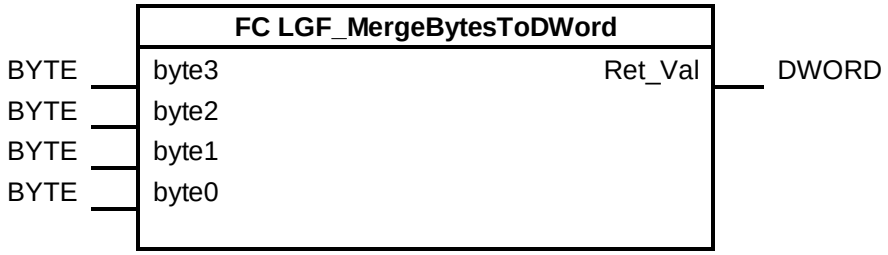
3.6.9 LGF_MergeBytesToDWord

Short description

This block converts 4 Byte variables into a DWord variable.

Block

Figure 3-80: FC LGF_MergeBytesToDWord



Input parameters

Table 3-183: Input parameters

Parameters	Data type	Description
byte3 … byte0	BYTE	Byte variables, whose Bytes are combined to a DWord

Output parameters

Table 3-184: Output parameters

Parameters	Data type	Description
Ret_Val	DWORD	Assembled Byte sequence, stored in a variable

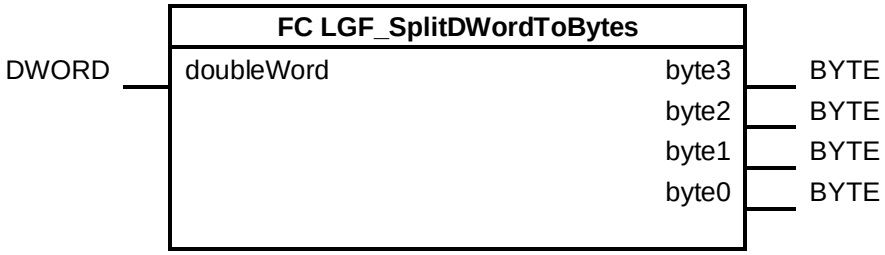
3.6.10 LGF_SplitDWordToBytes

Short description

This block converts a DWord variable into 4 Byte variables.

Block

Figure 3-81: FC LGF_SplitDWordToBytes



Input parameters

Table 3-185: Input parameters

Parameters	Data type	Description
doubleWord	DWORD	Byte sequence that is parsed

Output parameters

Table 3-186: Output parameters

Parameters	Data type	Description
byte3 … byte0	BYTE	Byte variables, in which the individual Bytes are stored

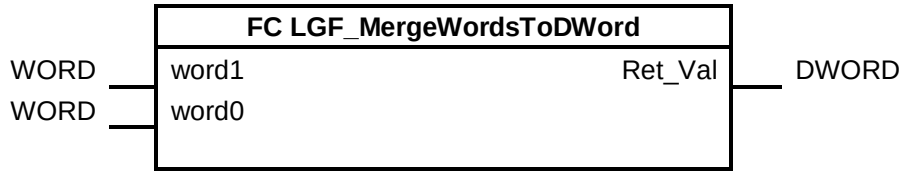
3.6.11 LGF_MergeWordsToDWord

Short description

This block converts 2 Word variables into one DWord variable.

Block

Figure 3-82: FC LGF_MergeWordsToDWord



Input parameters

Table 3-187: Input parameters

Parameters	Data type	Description
word1 … word0	WORD	Word variables, whose words are merged into a DWord

Output parameters

Table 3-188: Output parameters

Parameters	Data type	Description
Ret_Val	DWORD	A compound word order stored in a variable

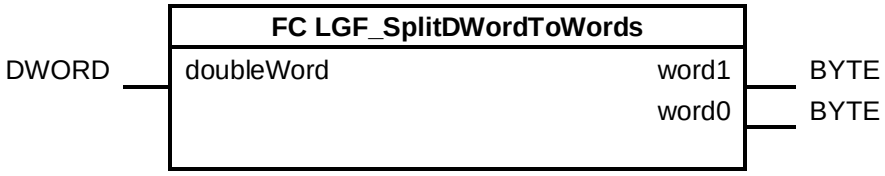
3.6.12 LGF_SplitDWordToWords

Short description

This block converts a DWord variable into 2 Word variables.

Block

Figure 3-83: FC LGF_SplitDWordToWords



Input parameters

Table 3-189: Input parameters

Parameters	Data type	Description
doubleWord	DWORD	Word order to be parsed

Output parameters

Table 3-190: Output parameters

Parameters	Data type	Description
word1 … word0	WORD	Word variables, in which the individual words are stored

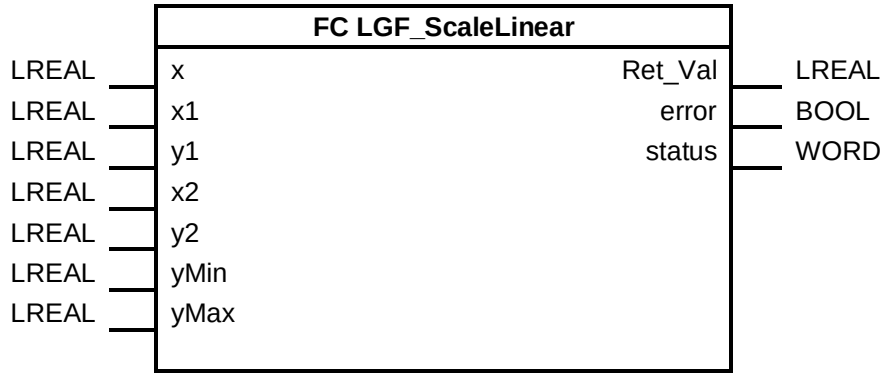
3.6.13 LGF_ScaleLinear

Short description

This function scales an input variable via a linear straight line equation.

Block

Figure 3-84: FC LGF_ScaleLinear



Input parameters

Table 3-191: Input parameters

Parameters	Data type	Description
x	LREAL	Input value to be scaled.
x1	LREAL	Point 1 (P_1) of the linear function.
y1	LREAL	
x2	LREAL	Point 2 (P_2) of the linear function.
y2	LREAL	
yMin	LREAL	Lower limit value of the output.
yMax	LREAL	High limit value of the output.

Output parameters

Table 3-192: Output parameters

Parameters	Data type	Description
Ret_Val	LREAL	Output value, scaled.
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).

Status and error displays

Table 3-193: Status and error codes

Status	Meaning	Remedy
16#0000	No error.	-
16#8200	Lower limit value “yMin” is greater than high limit value “yMax”.	Select low limit below the high limit.
16#6001	Output value limited to yMin	-
16#6002	Output value limited to yMax	-

Principle of operation

The function linearly scales an input variable (e.g. an analog input value) to a specific output variable (e.g. level).

To determine the output variable, the following linear equation is used in the function:

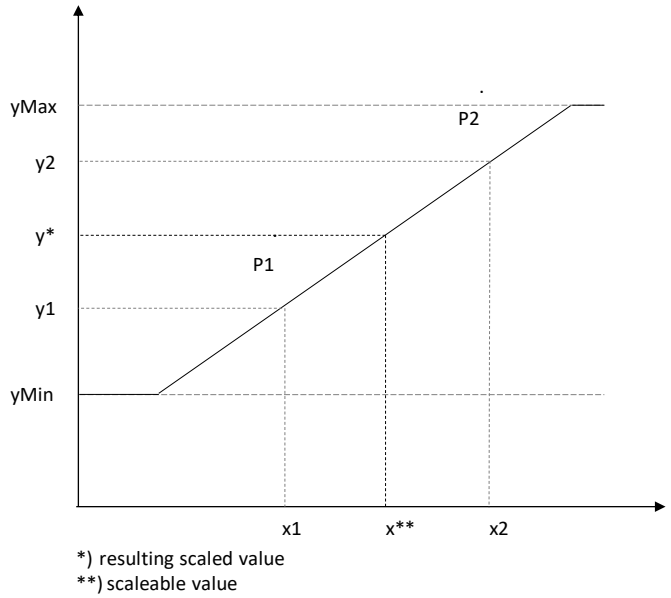
$$y = \frac{y_2 - y_1}{x_2 - x_1} \cdot (x - x_1) + y_1$$

The straight line is described by the two points, P_1 and P_2. You specify the points as a Cartesian coordinate system using x and y coordinates.

Note If the values of the parameters “x1” and “x2” are the same, the value of “y1” is output on output “y”.

By specifying “yMin” and “yMax” you can restrict the calculated value of “y” to a range limited at top and bottom. Thus you avoid override and underride ranges.

Figure 3-85: Scaling



*) resulting scaled value
**) scaleable value

Example

A signal from 4 to 20 mA is applied on an analog input module. This signal is converted to the CPU internal value from 0 to 27648 to measure a level. 0 corresponds to a level of 0.0 m and 27648 to a level of 1.7 m.

The block must then be parameterized as follows:

- x1 = 0; y1 = 0.0 (P1)
- x2 = +27648; y2 = 1.7 (P2)
- yMin = 0.0
- yMax = 1.7

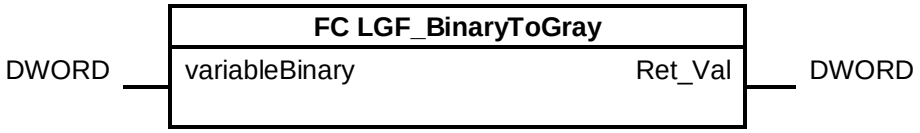
3.6.14 LGF_BinaryToGray

Short description

This block converts a binary coded value into a Gray-coded value.

Block

Figure 3-86: FC LGF_BinaryToGray



Input parameters

Table 3-194: Input parameters

Parameters	Data type	Description
variableBinary	DWORD	Binary coded value

Output parameters

Table 3-195: Output parameters

Parameters	Data type	Description
Ret_Val	DWORD	Gray-coded value

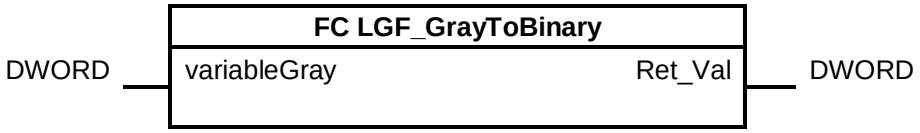
3.6.15 LGF_GrayToBinary

Short description

This block converts a Gray coded value into a binary coded value.

Block

Figure 3-87: FC LGF_GrayToBinary



Input parameters

Table 3-196: Input parameters

Parameters	Data type	Description
variableGray	DWORD	Gray-coded value

Output parameters

Table 3-197: Output parameters

Parameters	Data type	Description
Ret_Val	DWORD	Binary coded value

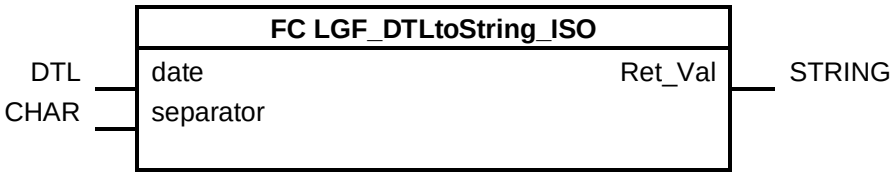
3.6.16 LGF_DTLtoString_ISO

Short description

This function block converts a date of data type DTL into a character string of data type STRING in international format (YYYY MM DD…).

Block

Figure 3-88: FC LGF_DTLtoString_ISO



Input parameters

Table 3-198: Input parameters

Parameters	Data type	Description
date	DTL	Date
separator	CHAR	Separator between the components of the output date.

Output parameters

Table 3-199: Output parameters

Parameters	Data type	Description
Ret_Val	STRING	Output string in accordance with the ISO 8601 format. Example: ‘2019-01-22 14:06:51.524621000’

Principle of operation

The block reads a date of data type DTL and converts the individual components of the date (year, month, day, hour…) into a character string and outputs it in international format. The separator between the components of the date is variable.

International format (ISO 8601)

Figure 3-89: Structure of the character string in accordance with ISO 8601

	Format																												
outString	Y	Y	Y	Y	-	M	M	-	D	D		H	H	:	M	M	:	S	S	.	NS	NS	NS	NS	NS	NS	NS	NS	NS
Position	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29

Separator

At the input parameter “separatorDate”, you specify the separator between the components of the calendar date.

Example: separatorDate = ‘/’
outString = ‘2016**/**03**/**16…’

separatorDate = ‘-’
outString = ‘2016-03-16…’

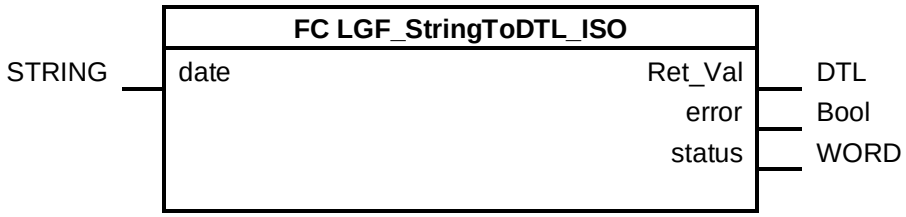
3.6.17 LGF_StringToDTL_ISO

Short description

This function block converts a character string in international format with date components into the data type DTL.

Block

Figure 3-90: FC LGF_StringToDTL_ISO



Input parameters

Table 3-200: Input parameters

Parameters	Data type	Description
date	STRING	Date as a character string according to the format. Example: '2019-01-22 14:06:51.524621000'

Output parameters

Table 3-201: Output parameters

Parameters	Data type	Description
Ret_Val	DTL	Displays the read in date
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).

Status and error displays

Table 3-202: Status/error codes

Status	Meaning	Remedy / notes
16#0000	No error	-
16 #7000	Initial value	-
16#8201	Format: Year	Year specification does not correspond to the format or specification (outside the value range of DTL)
16#8202	Format: Month	Month specification does not correspond to the format or specification (outside the value range of DTL)
16#8203	Format: Tag	Day specification does not correspond to the format or specification (outside the value range of DTL)
16#8204	Format: Hour	Hour specification does not correspond to the format or specification (outside the value range of DTL)
16#8205	Format: Minute	Minute specification does not correspond to the format or specification (outside the value range of DTL)
16#8206	Format: Second	Second specification does not correspond to the format or specification (outside the value range of DTL)
16#8207	Format: Nanosecond	Nanosecond indication does not correspond to the format or (indication outside the value range of DTL)

Principle of operation

The block reads a date as a character string and converts it to the data type DTL. The individual date components in the character string are separated according to the international format. The separator between the components in the character string is irrelevant.

International format (ISO 8601)

Figure 3-91: Structure of the character string in accordance with ISO 8601

	Format																											
outString	Y	Y	Y	Y	-	M	M	-	D	D		H	H	:	M	M	:	S	S	.	NS	NS	NS	NS	NS	NS	NS	NS	NS
Position	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29

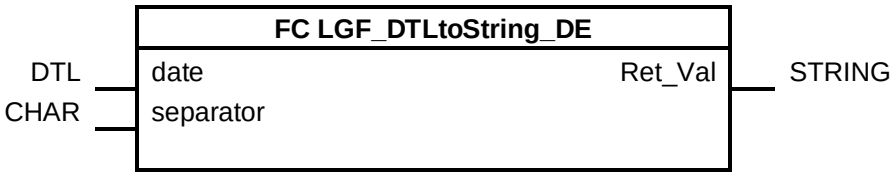
3.6.18 LGF_DTLtoString_DE

Short description

This block converts a date of data type DTL into a character string of data type STRING in the traditional format (DD MM YYYY…).

Block

Figure 3-92: FC LGF_DTLtoString_DE



Input parameters

Table 3-203: Input parameters

Parameters	Data type	Description
date	DTL	Date
separator	CHAR	Separator between the components of the output date.

Output parameters

Table 3-204: Output parameters

Parameters	Data type	Description
Ret_Val	STRING	Output string according to the traditional format. Example: ‘22-01-2019 14:07:57.696417000’

Principle of operation

The block reads a date of data type DTL and converts the individual components of the date (year, month, day, hour…) into a character string and outputs it in traditional format (DE). The separator between the components of the date is variable.

Traditional format (DE)

Figure 3-93: Structure of the character string in a traditional format

	Format																												
outString	D	D	-	M	M	-	Y	Y	Y	Y		H	H	:	M	M	:	S	S	.	NS	NS	NS	NS	NS	NS	NS	NS	NS
Position	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29

Separator

At the input parameter “separatorDate”, you specify the separator between the components of the calendar date.

Example: separatorDate = ‘/’
outString = ‘2016**/**03**/**16…’

separatorDate = ‘-’
outString = ‘2016-03-16…’

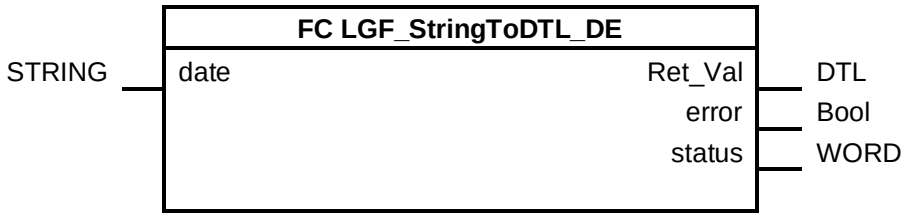
3.6.19 LGF_StringToDTL_DE

Short description

This function block converts a character string in the traditional format (DE) with date components into the data type DTL.

Block

Figure 3-94: FC LGF_StringToDTL_DE



Input parameters

Table 3-205: Input parameters

Parameters	Data type	Description
date	STRING	Date as a character string according to the format. Example: '22-01-2019 14:07:57.696417000'

Output parameters

Table 3-206: Output parameters

Parameters	Data type	Description
Ret_Val	DTL	Displays the read in date
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).

Status and error displays

Table 3-207: Status/error codes

Status	Meaning	Remedy / notes
16#0000	No error	-
16 #7000	Initial value	-
16#8201	Format: Year	Year specification does not correspond to the format or specification (outside the value range of DTL)
16#8202	Format: Month	Month specification does not correspond to the format or specification (outside the value range of DTL)
16#8203	Format: Tag	Day specification does not correspond to the format or specification (outside the value range of DTL)
16#8204	Format: Hour	Hour specification does not correspond to the format or specification (outside the value range of DTL)
16#8205	Format: Minute	Minute specification does not correspond to the format or specification (outside the value range of DTL)
16#8206	Format: Second	Second specification does not correspond to the format or specification (outside the value range of DTL)
16#8207	Format: Nanosecond	Nanosecond indication does not correspond to the format or (indication outside the value range of DTL)

Principle of operation

The block reads a date as a character string and converts it to the data type DTL. The individual date components in the character string are separated according to the traditional format (DE). The separator between the components in the character string is irrelevant.

Traditional format (DE)

Figure 3-95: Structure of the character string in a traditional format

	Format																											
outString	D	D	-	M	M	-	Y	Y	Y	Y		H	H	:	M	M	:	S	S	.	NS	NS	NS	NS	NS	NS	NS	NS	NS
Position	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29

3.6.20 LGF_TaddrToString

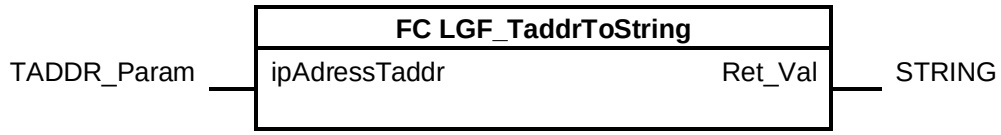
Short description

The system data type “TADDR_Param” contains address information consisting of an IPV4 address and the port number.

The LGF_TaddrToString function converts a TADDR_Param system data type variable to a “String” data type variable.

Block

Figure 3-96: FC LGF_TaddrToString



Input parameters

Table 3-208: Input parameters

Parameters	Data type	Description
ipAddressTaddr	TADDR_Param	IPV4 address

Output parameters

Table 3-209: Output parameters

Parameters	Data type	Description
Ret_Val	STRING	IPV4 address

Principle of operation

The function converts the IPV4 address with or without port number. The system data type “TADDR_Param” is a structured data type. This structure contains the variable “REM_PORT_NR”. If this variable is “0”, no port is written to the parameter “Ret_Val”.

Example

- Ret_val without port number: ‘192.168.11.11’
- Ret_val with port number: ‘192.168.11.11:3294’

3.6.21 LGF_StringToTaddr

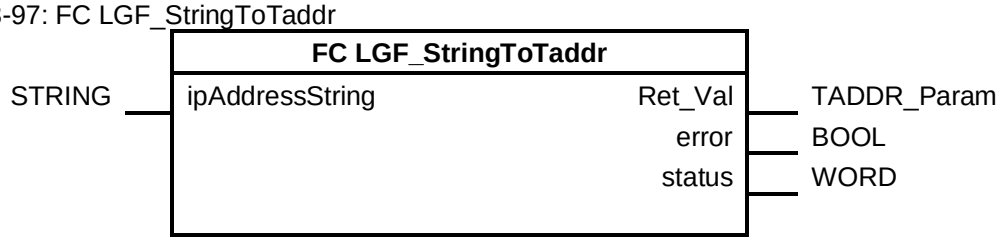
Short description

The system data type “TADDR_Param” contains address information consisting of an IPV4 address and the port number.

The LGF_StringToTaddr function converts a String data type variable to a TADDR_Param system data type variable.

Block

Figure 3-97: FC LGF_StringToTaddr



Input parameters

Table 3-210: Input parameters

Parameters	Data type	Description
ipAddressString	STRING	IPV4 address

Output parameters

Table 3-211: Output parameters

Parameters	Data type	Description
Ret_Val	TADDR_param	IPV4 address
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).

Status and error displays

Table 3-212: Status/error codes

Status	Meaning	Remedy / notes
16#0000	No error	-
16#8110	Too many characters in the xth octet of the IP address	
16#8120	No number/character in the xth octet of the IP address	
16#8130	Number in octet exceeds the maximum possible range of 255.	
16#8150	Too many characters when converting the port string.	
16#8151	No characters in the port string. Port string is empty.	
16#8151	Port number exceeds the maximum possible range of 65535.	

Principle of operation

The function converts the IPV4 address with or without port number from data type “STRING” to “TADDR_param”. The string must be in the following form:

[0..255].[0..255].[0..255].[0..255] without port number

or

[0..255].[0..255].[0..255].[0..255]:[0..65535] with port number

Example

- The standard string format for an IPV4 address without port number: ‘192.168.11.11’
- The standard string format for an IPV4 address with port number: ‘192.168.11.11:3294’

Note If you do not specify a port number in the ipAddressString parameter, the Ret_Val.REM_PORT_NR parameter returns 0.

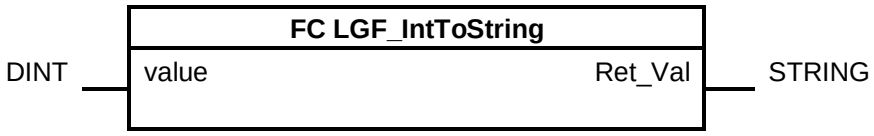
3.6.22 LGF_IntToString

Short description

This block converts a variable of the system data type “DInt” into a variable of the data type “String”.

Block

Figure 3-98: FC LGF_IntToString



Input parameters

Table 3-213: Input parameters

Parameters	Data type	Description
value	DINT	Integer value

Output parameters

Table 3-214: Output parameters

Parameters	Data type	Description
Ret_Val	STRING	Converted value as string. Example: ‘+16927’

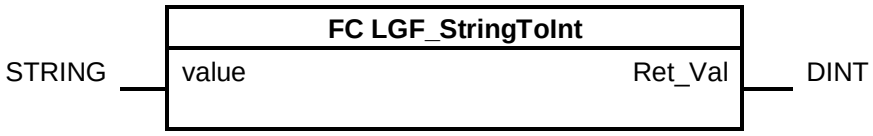
3.6.23 LGF_StringToInt

Short description

This block converts a variable of data type “String” into a variable of system data type “DInt”.

Block

Figure 3-99: FC LGF_StringToInt



Input parameters

Table 3-215: Input parameters

Parameters	Data type	Description
value	STRING	Value to be converted as string. Example: ‘+16927’

Output parameters

Table 3-216: Output parameters

Parameters	Data type	Description
Ret_Val	DINT	Converted integer value

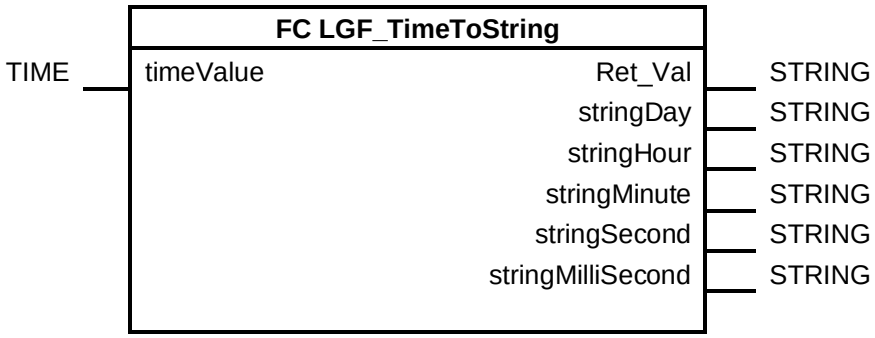
3.6.24 LGF_TimeToString

Short description

This block converts a variable of the system data type “Time” into a variable of the data type “String”.

Block

Figure 3-100: FC LGF_TimeToString



Input parameters

Table 3-217: Input parameters

Parameters	Data type	Description
timeValue	TIME	Time value Example: T#1D_3H_45M_6S

Output parameters

Table 3-218: Output parameters

Parameters	Data type	Description
Ret_Val	STRING	Converted time as string. Example: ‘1D3H45M6S0MS’
stringDay	STRING	Converted day as string. Example: ‘1’
stringHour	STRING	Converted hour as string. Example: ‘3’
stringMinute	STRING	Converted minute as string. Example: ‘45’
stringSecond	STRING	Converted second as string. Example: ‘6’
stringMilliSecond	STRING	Converted millisecond as string. Example: ‘0’

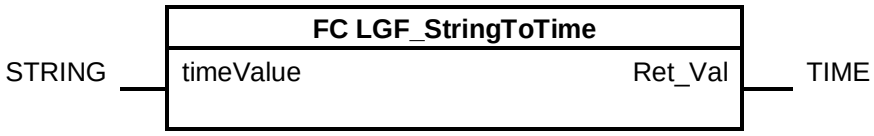
3.6.25 LGF_StringToTime

Short description

This block converts a variable of the data type “String” into a variable of the system data type “Time”.

Block

Figure 3-101: FC LGF_StringToTime



Input parameters

Table 3-219: Input parameters

Parameters	Data type	Description
timeValue	STRING	Time to be converted as string. Example: ‘1D3H45M6S0MS’

Output parameters

Table 3-220: Output parameters

Parameters	Data type	Description
Ret_Val	TIME	Converted time value Example: T#1D_3H_45M_6S

3.6.26 LGF_UnixTimeToDTL

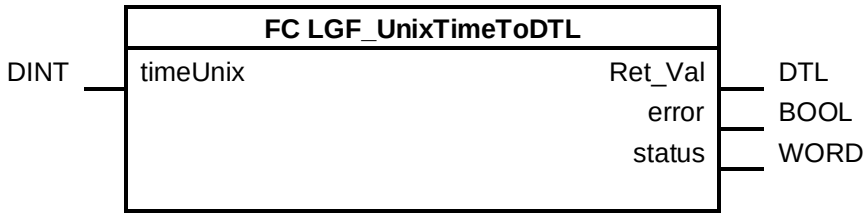
Short description

This block converts a Unix time of data type DInt to a date and time of data type DTL. The timestamp is calculated in UTC. This means that the time zone is not taken into account.

Only times after 01/01/1990 are permitted.

Block

Figure 3-102: FC LGF_UnixTimeToDTL



Input parameters

Table 3-221: Input parameters

Parameters	Data type	Description
timeUnix	DINT	Unix time

Output parameters

Table 3-222: Output parameters

Parameters	Data type	Description
Ret_Val	DTL	Converted time (date and time) If error = TRUE, Ret_Val = 0
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).

Status and error displays

Table 3-223: Status and error codes

Status	Meaning	Remedy
16#0000	No error.	-
16#8000	Unix time (timeUnix) is before 01/01/1990	-
16#6001	Unix time (timeUnix) is exactly at the lower limit of 01.01.1990.	-

3.6.27 LGF_DTLToUnixTime

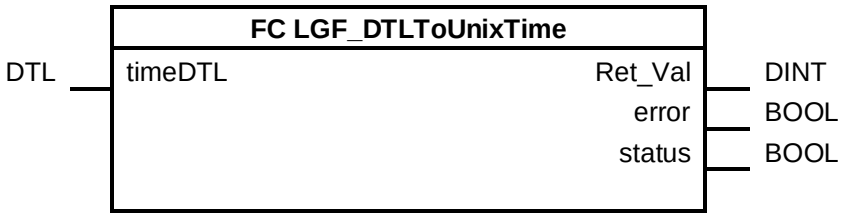
Short description

This block converts the date and time of data type DTL to a Unix time of data type DInt. The timestamp is calculated in UTC. This means that the time zone is not taken into account.

Only times after 01/01/1990 are permitted.

Block

Figure 3-103: FC LGF_DTLToUnixTime



Input parameters

Table 3-224: Input parameters

Parameters	Data type	Description
timeDTL	DTL	Date and time

Output parameters

Table 3-225: Output parameters

Parameters	Data type	Description
Ret_Val	DINT	Converted Unix time
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).

Status and error displays

Table 3-226: Status and error codes

Status	Meaning	Remedy
16#0000	No error.	-
16#8000	Unix time (timeUnix) is before 01/01/1990	-

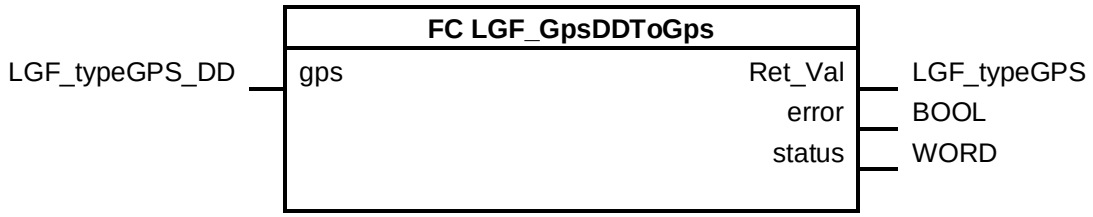
3.6.28 LGF_GpsDDToGps

Short description

This function block converts GPS data from decimal degrees to GPS data in direction, degrees, minutes, and seconds.

Block

Figure 3-104: FC LGF_GpsDDToGps



Input parameters

Table 3-227: Input parameters

Parameters	Data type	Description
gps	LGF_typeGPS_DD	GPS data to be converted (decimal degrees), e.g. 52.520817 13.40945

Output parameters

Table 3-228: Output parameters

Parameters	Data type	Description
Ret_Val	LGF_typeGPS	Converted GPS data (direction, degrees, minutes, and seconds), e.g. N52° 31′ 14.941″ E13° 24′ 34.020″
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).

Status and error displays

Table 3-229: Status/error codes

status	Meaning	Remedy / notes
16#0000	No error	-
16#8201	Latitude Value error	-
16#8203	Longitude Value error	-

LGF_typeGPS_DD data type

Table 3-230: LGF_typeGPS_DD data type

Parameters	Data type	Description
latitude	REAL	Degrees latitude with decimal places (Unit: degree decimal), North = positive; South = negative valid value range [-90.00000..90.00000]
longitude	REAL	Longitude in degrees with decimal places (Unit: degree decimal), East = positive; West = negative valid range [-180.0000..180.0000]

LGF_typeGPS data type

Table 3-231: LGF_typeGPS data type

Parameters	Data type	Description
latitude	LGF_typeGPS_DMS	Latitude in degrees, minutes, and seconds
longitude	LGF_typeGPS_DMS	Longitude in degrees, minutes, and seconds

LGF_typeGPS_DMS data type

Table 3-232: LGF_typeGPS_DMS data type

Parameters	Data type	Description
dir	CHAR	Direction [N, S, E, W, n, s, e, w]
deg	UINT	Grad LAT [-89..+89]; LON [-179..+179]
min	UINT	Minutes [0..+59]
sec	UINT	Seconds [0..+59]

3.6.29 LGF_GpsToGpsDD

Short description

This function block converts GPS data, in the format direction, degrees, minutes, and seconds, into GPS data in decimal degrees.

Block

Figure 3-105: FC LGF_GpsToGpsDD

Input parameters

Table 3-233: Input parameters

Parameters	Data type	Description
gps	LGF_typeGPS	GPS data to be converted (direction, degrees, minutes, and seconds), e.g. N52° 31′ 14.941″ E13° 24′ 34.020″

Output parameters

Table 3-234: Output parameters

Parameters	Data type	Description
Ret_Val	LGF_typeGPS_DD	Converted GPS data (decimal degrees), e.g. 52.520817 13.40945
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).

Status and error displays

Table 3-235: Status/error codes

status	Meaning	Remedy / notes
16#0000	No error	-
16#8200	Latitude Direction error	-
16#8201	Latitude Value error	-
16#8202	Longitude Value error	-
16#8203	Longitude Value error	-

LGF_typeGPS data type

Table 3-236: LGF_typeGPS data type

Parameters	Data type	Description
latitude	LGF_typeGPS_DMS	Latitude in degrees, minutes, and seconds
longitude	LGF_typeGPS_DMS	Longitude in degrees, minutes, and seconds

LGF_typeGPS_DMS data type

Table 3-237: LGF_typeGPS_DMS data type

Parameters	Data type	Description
dir	CHAR	Direction [N, S, E, W, n, s, e, w]
deg	UINT	Grad LAT [-89..+89]; LON [-179..+179]
min	UINT	Minutes [0..+59]
sec	UINT	Seconds [0..+59]

LGF_typeGPS_DD data type

Table 3-238: LGF_typeGPS_DD data type

Parameters	Data type	Description
latitude	REAL	Degrees latitude with decimal places (Unit: degree decimal), North = positive; South = negative valid value range [-90.00000..90.00000]
longitude	REAL	Longitude in degrees with decimal places (Unit: degree decimal), East = positive; West = negative valid range [-180.0000..180.0000]

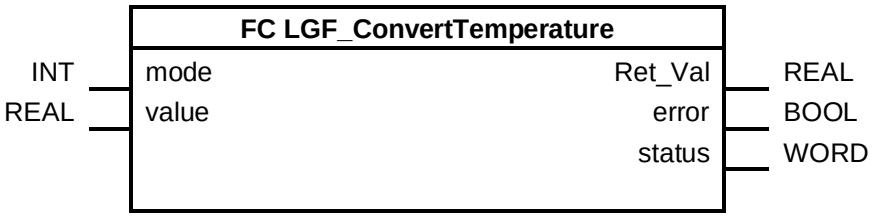
3.6.30 LGF_ConvertTemperature

Short description

This block converts °Celsius to °Fahrenheit or Kelvin, as well as Kelvin to °Fahrenheit or Rankine, and vice versa.

Block

Figure 3-106: FC LGF_ConvertTemperature



Input parameters

Table 3-239: Input parameters

Parameters	Data type	Description
mode	INT	Mode 1. °Celsius to °Fahrenheit 2. °Fahrenheit to °Celsius 3. °Celsius to Kelvin 4. Kelvin to °Celsius 5. °Fahrenheit to Kelvin 6. Kelvin to °Fahrenheit 7. Rankine in Kelvin 8. Kelvin in Rankine
value	REAL	Temperature to be converted

Output parameters

Table 3-240: Output parameters

Parameters	Data type	Description
Ret_Val	REAL	Converted temperature
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).

Status and error displays

Table 3-241: Status/error codes

Status	Meaning	Remedy / notes
16#0000	No error	-
16#8200	Incorrect mode on input “mode”.	See description of the input parameters

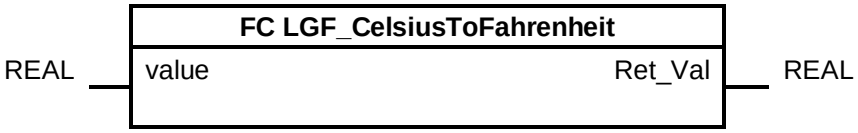
3.6.31 LGF_CelsiusToFahrenheit

Short description

This block converts °Celsius into °Fahrenheit.

Block

Figure 3-107: FC LGF_CelsiusToFahrenheit



Input parameters

Table 3-242: Input parameters

Parameters	Data type	Description
value	REAL	Temperature to be converted to °Celsius

Output parameters

Table 3-243: Output parameters

Parameters	Data type	Description
Ret_Val	REAL	Converted temperature in °Fahrenheit

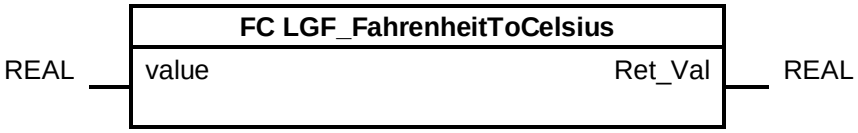
3.6.32 LGF_FahrenheitToCelsius

Short description

This block converts °Fahrenheit into °Celsius.

Block

Figure 3-108: FC LGF_FahrenheitToCelsius



Input parameters

Table 3-244: Input parameters

Parameters	Data type	Description
value	REAL	Temperature to be converted to °Fahrenheit

Output parameters

Table 3-245: Output parameters

Parameters	Data type	Description
Ret_Val	REAL	Converted temperature in °Celsius

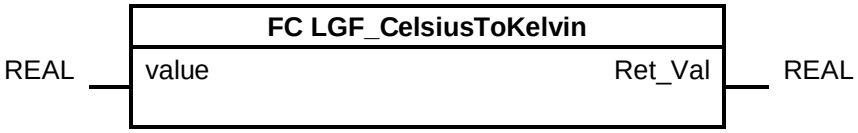
3.6.33 LGF_CelsiusToKelvin

Short description

This block converts °Celsius into Kelvin.

Block

Figure 3-109: FC LGF_CelsiusToKelvin



Input parameters

Table 3-246: Input parameters

Parameters	Data type	Description
value	REAL	Temperature to be converted to °Celsius

Output parameters

Table 3-247: Output parameters

Parameters	Data type	Description
Ret_Val	REAL	Converted temperature in Kelvin

3.6.34 LGF_KelvinToCelsius

Short description

This block converts Kelvin into °Celsius.

Block

Figure 3-110: FC LGF_KelvinToCelsius

Input parameters

Table 3-248: Input parameters

Parameters	Data type	Description
value	REAL	Temperature to be converted to Kelvin

Output parameters

Table 3-249: Output parameters

Parameters	Data type	Description
Ret_Val	REAL	Converted temperature in °Celsius

3.6.35 LGF_KelvinToFahrenheit

Short description

This block converts Kelvin into °Fahrenheit.

Block

Figure 3-111: FC LGF_KelvinToFahrenheit

Input parameters

Table 3-250: Input parameters

Parameters	Data type	Description
value	REAL	Temperature to be converted to Kelvin

Output parameters

Table 3-251: Output parameters

Parameters	Data type	Description
Ret_Val	REAL	Converted temperature in °Fahrenheit

3.6.36 LGF_FahrenheitToKelvin

Short description

This block converts °Fahrenheit into Kelvin.

Block

Figure 3-112: FC LGF_FahrenheitToKelvin

Input parameters

Table 3-252: Input parameters

Parameters	Data type	Description
value	REAL	Temperature to be converted to °Fahrenheit

Output parameters

Table 3-253: Output parameters

Parameters	Data type	Description
Ret_Val	REAL	Converted temperature in Kelvin

3.6.37 LGF_KelvinToRankine

Short description

This block converts Kelvin into Rankine.

Block

Figure 3-113: FC LGF_KelvinToRankine

Input parameters

Table 3-254: Input parameters

Parameters	Data type	Description
value	REAL	Temperature to be converted to Kelvin

Output parameters

Table 3-255: Output parameters

Parameters	Data type	Description
Ret_Val	REAL	Converted temperature in Rankine

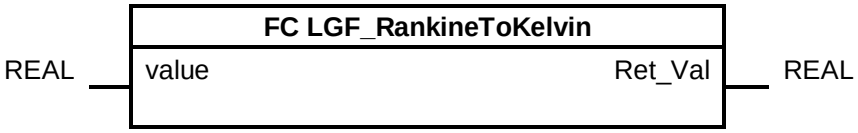
3.6.38 LGF_RankineToKelvin

Short description

This block converts Rankine into Kelvin.

Block

Figure 3-114: FC LGF_RankineToKelvin



Input parameters

Table 3-256: Input parameters

Parameters	Data type	Description
value	REAL	Converted temperature in Rankine

Output parameters

Table 3-257: Output parameters

Parameters	Data type	Description
Ret_Val	REAL	Converted temperature in Kelvin

3.7 Signal generators

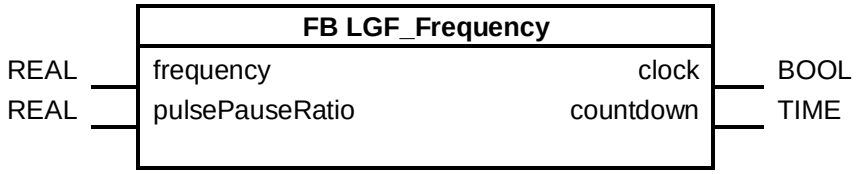
3.7.1 LGF_Frequency

Short description

The block generates a signal that changes between the values “0” and “1” depending on a defined frequency and a pulse pause ratio.

Block

Figure 3-115: FB LGF_Frequency



Input parameters

Table 3-258: Input parameters

Parameters	Data type	Description
frequency	REAL	Clock frequency in Hz
pulsePauseRatio	REAL	Pulse pause ratio (standard: 1.0 corresponds to 1:1)

Output parameters

Table 3-259: Output parameters

Parameters	Data type	Description
clock	BOOL	Output changes with defined frequency
countdown	TIME	Remaining time of the current “clock” state

Principle of operation

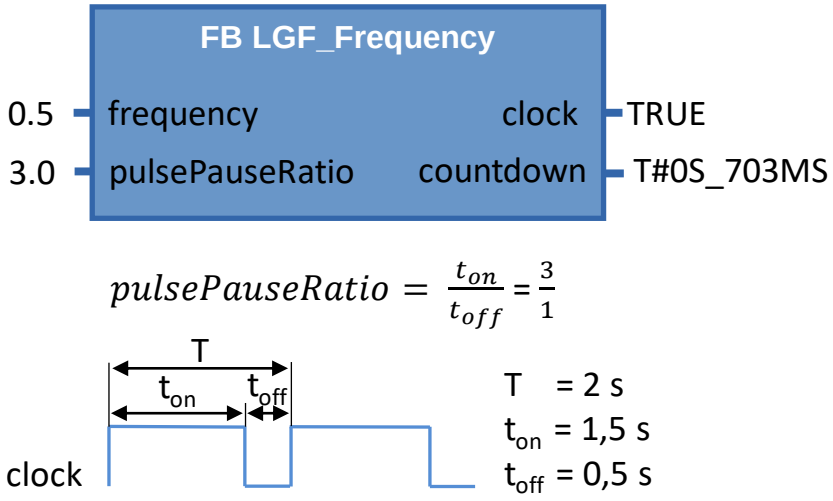
The “clock” output is a boolean value that toggles at the desired frequency. The “pulsePauseRatio” input is used to set the pulse pause ratio.

The output “countdown” outputs the remaining time of the current state of “clock”.

If the desired frequency or pulse pause ratio is less than or equal to “0.0”, the output “clock” = FALSE and “countdown” = “0 s”.

Example

Figure 3-116: FB LGF_Frequency



Note The “clock” of the FB LGF_Frequency depends on the cycle time of the OB Main. To increase the accuracy, the FB can also be called in a cyclic interrupt OB with a low time interval.

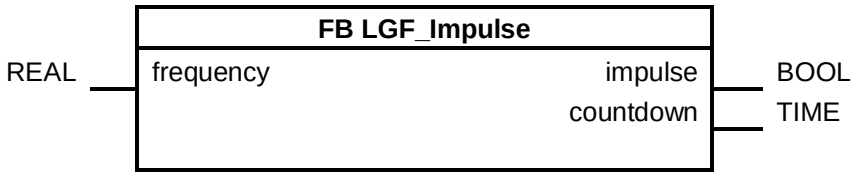
3.7.2 LGF_Impulse

Short description

This block generates pulses at a given frequency. The pulse is always present for one (control) cycle.

Block

Figure 3-117: FB LGF_Impulse



Input parameters

Table 3-260: Input parameters

Parameters	Data type	Description
frequency	REAL	Clock frequency in Hz

Output parameters

Table 3-261: Output parameters

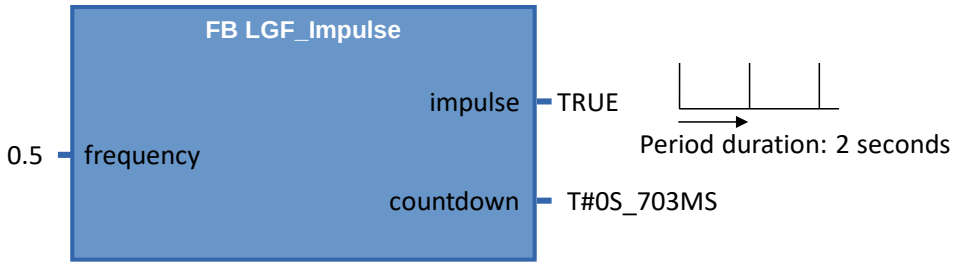
Parameters	Data type	Description
impulse	BOOL	Signal with pulses
countdown	TIME	Time until next pulse

Principle of operation

The block generates pulses at the output “impulse” with the frequency “frequency”. The block always begins with a pulse and sets the next pulse after the period that has elapsed.

Example

Figure 3-118: Example



Note

New as of V1.2.0
The LGF_Impulse block (from V1.2.0) no longer calls the LGF_Frequency block.

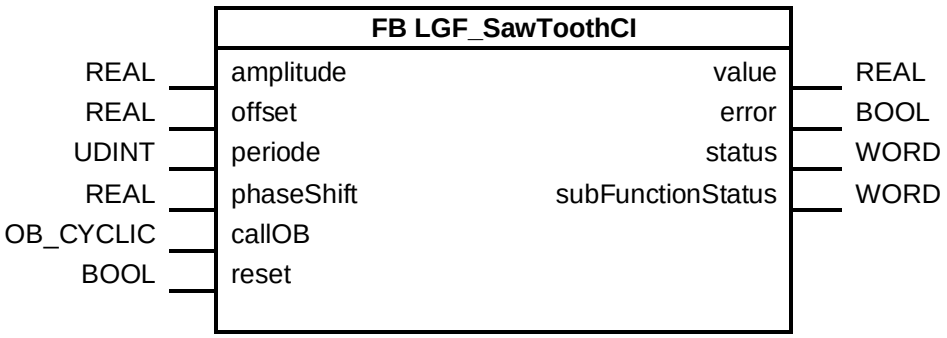
3.7.3 LGF_SawToothCI

Short description

This block generates a sawtooth-shaped signal profile. For this it uses the time interval of the calling **C**yclic **I**nterrupt OB.

Block

Figure 3-119: FB LGF_SawToothCI



Input parameters

Table 3-262: Input parameters

Parameters	Data type	Description
amplitude	REAL	Amplitude of the signal profile.
offset	REAL	Offset of the signal profile in the Y-direction.
period	UDINT	Period duration of the signal profile in [ms]
phaseShift	REAL	Phase offset in [ms]
callOB	OB_CYCLIC	Calling wake-alarm interrupt OB (cyclic interrupt OB)
reset	BOOL	Reset of the signal profile.

Note Changes in the input parameters will be effective immediately.

Output parameters

Table 3-263: Output parameters

Parameters	Data type	Description
value	REAL	Current value of the sawtooth signal.
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).
subFunctionStatus	WORD	Status or return value of the called FCs and system blocks.

Status and error displays

Table 3-264: Status/error codes

status	Meaning	Remedy / notes
16#0000	No error	-
16#8600	OB on input “callOB” is not configured / present.	Interconnect the constant name of a configured cyclic interrupt OB at the input “callOB”.
16#8601	Error in “QRY_CINT” command.	Check the error code in “subFunctionStatus”

Note The status of called commands is output in “subFunctionStatus”. In this case, the output value in “status” indicates which command caused the error. In this case, refer to the TIA Portal Online Help section for information on the respective commands.

Principle of operation

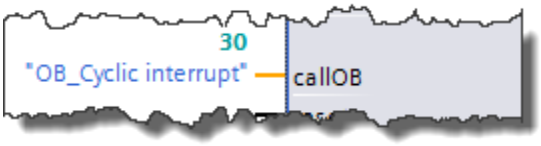
The block calculates the values for a sawtooth-shaped signal profile, which is output to the output parameter “value”.

The “amplitude”, the “offset” in the Y-direction, the “period”, and the “phase shift” can be set at the input parameters.

The input parameter “reset” resets the signal profile. At the “value” output parameter, the value “0” is output as long as “reset” is set to “TRUE”.

The block must be called in a cyclic interrupt OB. The time interval of the calling cyclic interrupt OB is determined in the FB with the command “QRY_CINT”. For this, the constant name of the calling cyclic interrupt OB must be interconnected at the input parameter “callOB”.

Figure 3-120: Interconnecting the cyclic interrupt OB



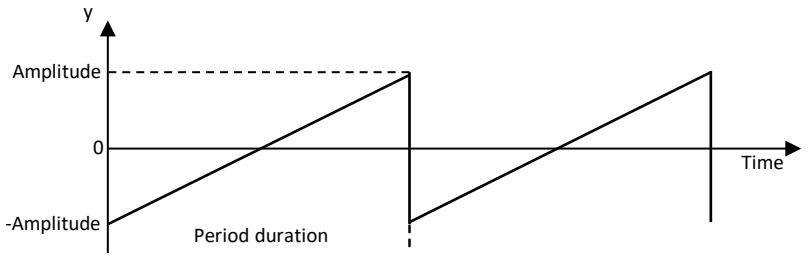
The number of calculated values of the signal profile per period duration is calculated as follows:

$$Quantity\ Values = \frac{Period\ duration}{Time\ interval\ Cyclic\ interrupt\ OB}$$

Note To obtain a continuous signal profile of the curve, the time interval of the cyclic interrupt OB should not be selected too large depending on the period duration.

The Figure below shows the signal profile of the calculated values.

Figure 3-121: Signal profile at “offset” = 0



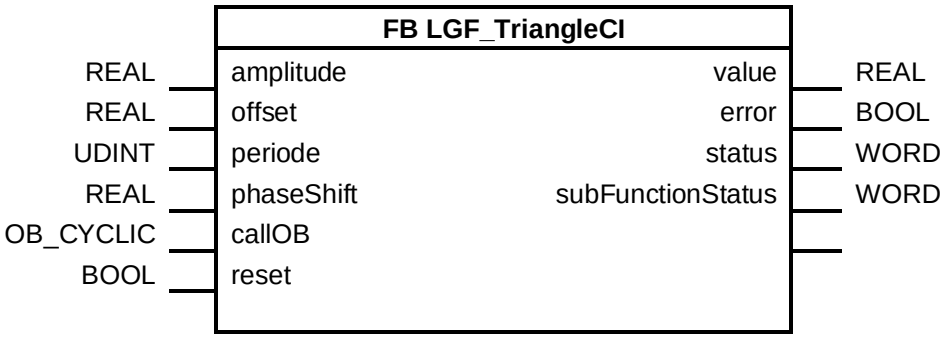
3.7.4 LGF_TriangleCI

Short description

This block generates a triangular signal profile. For this it uses the time interval of the calling **C**yclic **I**nterrupt OB.

Block

Figure 3-122: FB LGF_TriangleCI



Input parameters

Table 3-265: Input parameters

Parameters	Data type	Description
amplitude	REAL	Amplitude of the signal profile.
offset	REAL	Offset of the signal profile in the Y-direction.
period	UDINT	Period duration of the signal profile in [ms]
phaseShift	REAL	Phase offset in [ms]
callOB	OB_CYCLIC	Calling wake-alarm interrupt OB (cyclic interrupt OB)
reset	BOOL	Reset of the signal profile.

Note Changes in the input parameters will be effective immediately.

Output parameters

Table 3-266: Output parameters

Parameters	Data type	Description
value	REAL	Current value of the triangle signal.
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).
subFunctionStatus	WORD	Status or return value of the called FCs and system blocks.

Status and error displays

Table 3-267: Status/error codes

status	Meaning	Remedy / notes
16#0000	No error	-
16#8600	OB on input “callOB” is not configured / present.	Interconnect the constant name of a configured cyclic interrupt OB at the input “callOB”.
16#8601	Error in “QRY_CINT” command.	Check the error code in “subFunctionStatus”

Note The status of called commands is output in “subFunctionStatus”. In this case, the output value in “status” indicates which command caused the error. In this case, refer to the TIA Portal Online Help section for information on the respective commands.

Principle of operation

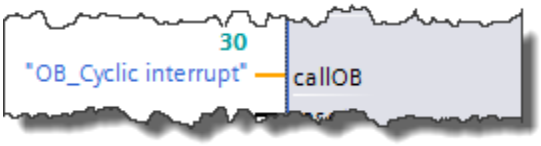
The block calculates the values for a triangular signal profile, which are output to the output parameter “value”.

The “amplitude”, the “offset” in the Y-direction, the “period”, and the “phase shift” can be set at the input parameters.

The input parameter “reset” resets the signal profile. At the “value” output parameter, the value “0” is output as long as “reset” is set to “TRUE”.

The block must be called in a cyclic interrupt OB. The time interval of the calling cyclic interrupt OB is determined in the FB with the command “QRY_CINT”. For this, the constant name of the calling cyclic interrupt OB must be interconnected at the input parameter “callOB”.

Figure 3-123: Interconnecting the cyclic interrupt OB



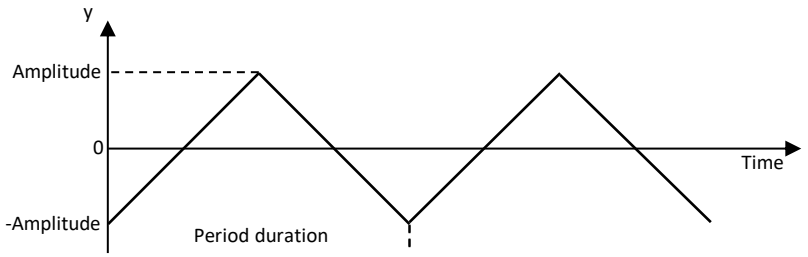
The number of calculated values of the signal profile per period duration is calculated as follows:

$$Quantity\ Values = \frac{Period\ duration}{Time\ interval\ Cyclic\ interrupt\ OB}$$

Note To obtain a continuous signal profile of the curve, the time interval of the cyclic interrupt OB should not be selected too large depending on the period duration.

The Figure below shows the signal profile of the calculated values.

Figure 3-124: Signal profile at “offset” = 0



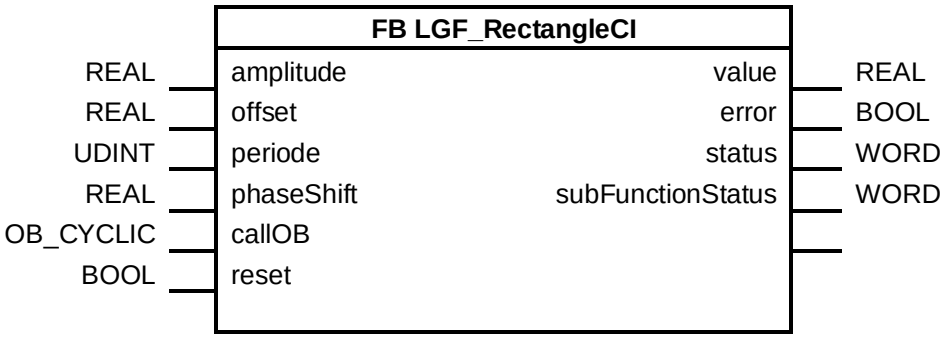
3.7.5 LGF_RectangleCI

Short description

This block generates a rectangular signal profile. For this it uses the time interval of the calling **C**yclic **I**nterrupt OB.

Block

Figure 3-125: FB LGF_RectangleCI



Input parameters

Table 3-268: Input parameters

Parameters	Data type	Description
amplitude	REAL	Amplitude of the signal profile.
offset	REAL	Offset of the signal profile in the Y-direction.
period	UDINT	Period duration of the signal profile in [ms]
phaseShift	REAL	Phase offset in [ms]
callOB	OB_CYCLIC	Calling wake-alarm interrupt OB (cyclic interrupt OB)
reset	BOOL	Reset of the signal profile.

Note Changes in the input parameters will be effective immediately.

Output parameters

Table 3-269: Output parameters

Parameters	Data type	Description
value	REAL	Current value of the rectangle signal.
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).
subFunctionStatus	WORD	Status or return value of the called FCs and system blocks.

Status and error displays

Table 3-270: Status/error codes

status	Meaning	Remedy / notes
16#0000	No error	-
16#8600	OB on input “callOB” is not configured / present.	Interconnect the constant name of a configured cyclic interrupt OB at the input “callOB”.
16#8601	Error in “QRY_CINT” command.	Check the error code in “subFunctionStatus”

Note The status of called commands is output in “subFunctionStatus”. In this case, the output value in “status” indicates which command caused the error. In this case, refer to the TIA Portal Online Help section for information on the respective commands.

Principle of operation

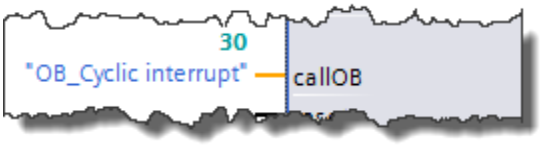
The block calculates the values for a rectangular signal profile, which are output to the output parameter “value”.

The “amplitude”, the “offset” in the Y-direction, the “period”, and the “phase shift” can be set at the input parameters.

The input parameter “reset” resets the signal profile. At the “value” output parameter, the value “0” is output as long as “reset” is set to “TRUE”.

The block must be called in a cyclic interrupt OB. The time interval of the calling cyclic interrupt OB is determined in the FB with the command “QRY_CINT”. For this, the constant name of the calling cyclic interrupt OB must be interconnected at the input parameter “callOB”.

Figure 3-126: Interconnecting the cyclic interrupt OB



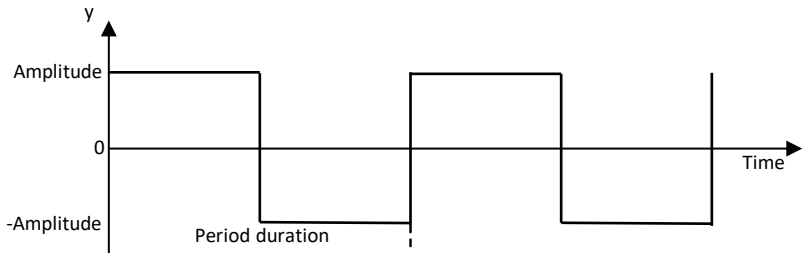
The number of calculated values of the signal profile per period duration is calculated as follows:

$$Quantity\ Values = \frac{Period\ duration}{Time\ interval\ Cyclic\ interrupt\ OB}$$

Note To obtain a continuous signal profile of the curve, the time interval of the cyclic interrupt OB should not be selected too large depending on the period duration.

The Figure below shows the signal profile of the calculated values.

Figure 3-127: Signal profile at “offset” = 0



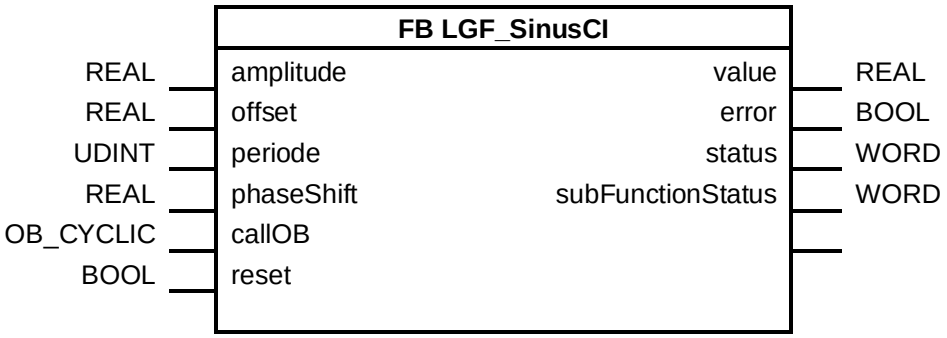
3.7.6 LGF_SinusCI

Short description

This block generates a sinusoidal signal profile. For this it uses the time interval of the calling **C**yclic **I**nterrupt OB.

Block

Figure 3-128: FB LGF_SinusCI



Input parameters

Table 3-271: Input parameters

Parameters	Data type	Description
amplitude	REAL	Amplitude of the signal profile.
offset	REAL	Offset of the signal profile in the Y-direction.
period	UDINT	Period duration of the signal profile in [ms]
phaseShift	REAL	Phase offset in [ms]
callOB	OB_CYCLIC	Calling wake-alarm interrupt OB (cyclic interrupt OB)
reset	BOOL	Reset of the signal profile.

Note Changes in the input parameters will be effective immediately.

Output parameters

Table 3-272: Output parameters

Parameters	Data type	Description
value	REAL	Current value of the triangle signal.
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).
subFunctionStatus	WORD	Status or return value of the called FCs and system blocks.

Status and error displays

Table 3-273: Status/error codes

status	Meaning	Remedy / notes
16#0000	No error	-
16#8600	OB on input “callOB” is not configured / present.	Interconnect the constant name of a configured cyclic interrupt OB at the input “callOB”.
16#8601	Error in “QRY_CINT” command.	Check the error code in “subFunctionStatus”

Note The status of called commands is output in “subFunctionStatus”. In this case, the output value in “status” indicates which command caused the error. In this case, refer to the TIA Portal Online Help section for information on the respective commands.

Principle of operation

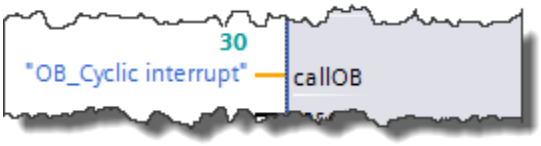
The block calculates the values for a sinusoidal signal profile, which are output to the output parameter “value”.

The “amplitude”, the “offset” in the Y-direction, the “period”, and the “phase shift” can be set at the input parameters.

The input parameter “reset” resets the signal profile. At the “value” output parameter, the value “0” is output as long as “reset” is set to “TRUE”.

The block must be called in a cyclic interrupt OB. The time interval of the calling cyclic interrupt OB is determined in the FB with the command “QRY_CINT”. For this, the constant name of the calling cyclic interrupt OB must be interconnected at the input parameter “callOB”.

Figure 3-129: Interconnecting the cyclic interrupt OB



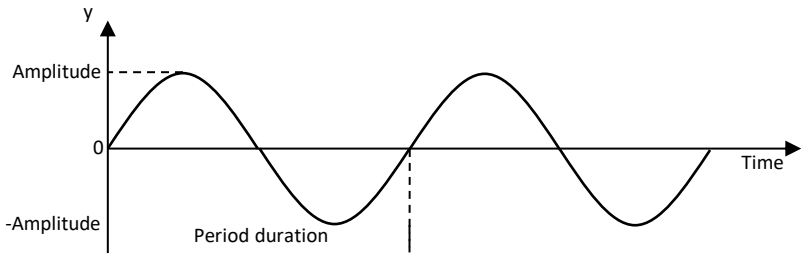
The number of calculated values of the signal profile per period duration is calculated as follows:

$$Quantity\ Values = \frac{Period\ duration}{Time\ interval\ Cyclic\ interrupt\ OB}$$

Note To obtain a continuous signal profile of the curve, the time interval of the cyclic interrupt OB should not be selected too large depending on the period duration.

The Figure below shows the signal profile of the calculated values.

Figure 3-130: Signal profile at “offset” = 0



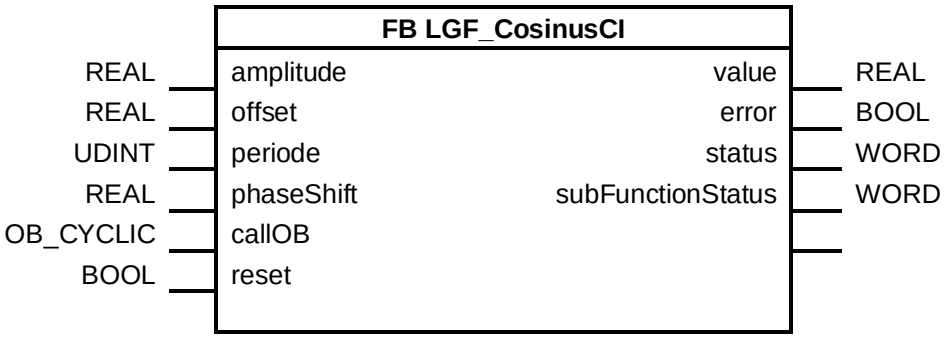
3.7.7 LGF_CosinusCI

Short description

This block generates a cosinusoidal signal profile. For this it uses the time interval of the calling **C**yclic **I**nterrupt OB.

Block

Figure 3-131: FB LGF_CosinusCI



Input parameters

Table 3-274: Input parameters

Parameters	Data type	Description
amplitude	REAL	Amplitude of the signal profile.
offset	REAL	Offset of the signal profile in the Y-direction.
period	UDINT	Period duration of the signal profile in [ms]
phaseShift	REAL	Phase offset in [ms]
callOB	OB_CYCLIC	Calling wake-alarm interrupt OB (cyclic interrupt OB)
reset	BOOL	Reset of the signal profile.

Note Changes in the input parameters will be effective immediately.

Output parameters

Table 3-275: Output parameters

Parameters	Data type	Description
value	REAL	Current value of the triangle signal.
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).
subFunctionStatus	WORD	Status or return value of the called FCs and system blocks.

Status and error displays

Table 3-276: Status/error codes

status	Meaning	Remedy / notes
16#0000	No error	-
16#8600	OB on input “callOB” is not configured / present.	Interconnect the constant name of a configured cyclic interrupt OB at the input “callOB”.
16#8601	Error in “QRY_CINT” command.	Check the error code in “subFunctionStatus”

Note The status of called commands is output in “subFunctionStatus”. In this case, the output value in “status” indicates which command caused the error. In this case, refer to the TIA Portal Online Help section for information on the respective commands.

Principle of operation

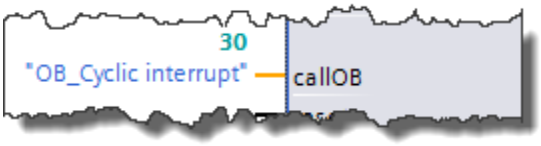
The block calculates the values for a cosine-shaped signal profile, which are output to the output parameter “value”.

The “amplitude”, the “offset” in the Y-direction, the “period”, and the “phase shift” can be set at the input parameters.

The input parameter “reset” resets the signal profile. At the “value” output parameter, the value “0” is output as long as “reset” is set to “TRUE”.

The block must be called in a cyclic interrupt OB. The time interval of the calling cyclic interrupt OB is determined in the FB with the command “QRY_CINT”. For this, the constant name of the calling cyclic interrupt OB must be interconnected at the input parameter “callOB”.

Figure 3-132: Interconnecting the cyclic interrupt OB



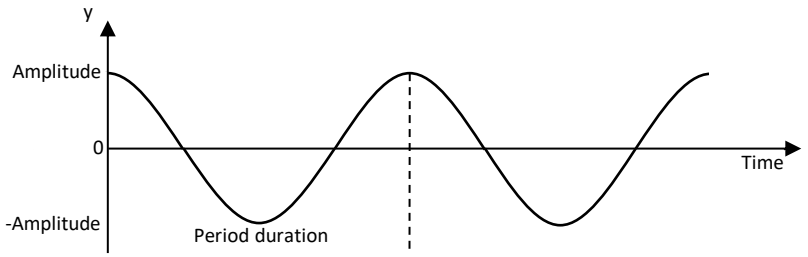
The number of calculated values of the signal profile per period duration is calculated as follows:

$$Quantity\ Values = \frac{Period\ duration}{Time\ interval\ Cyclic\ interrupt\ OB}$$

Note To obtain a continuous signal profile of the curve, the time interval of the cyclic interrupt OB should not be selected too large depending on the period duration.

The Figure below shows the signal profile of the calculated values.

Figure 3-133: Signal profile at “offset” = 0



3.8 Technology Operations

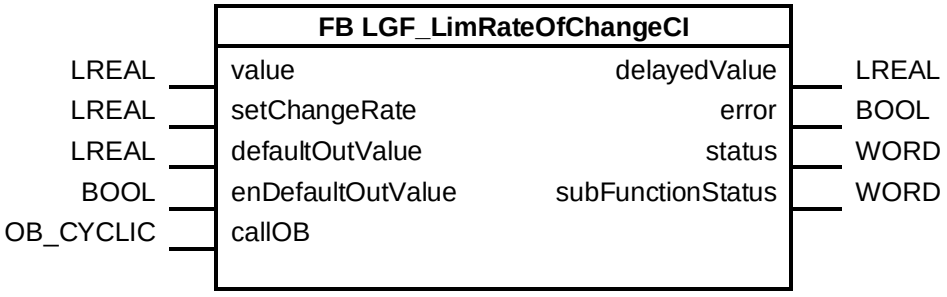
3.8.1 LGF_LimRateOfChangeCI

Short description

This block limits the rate of change of an input variable. A jump function becomes a ramp function.

Block

Figure 3-134: FB LGF_LimRateOfChangeCI



Input parameters

Table 3-277: Input parameters

Parameters	Data type	Description
value	LREAL	Input variable (jump function)
setChangeRate	LREAL	Rate of change of ramp function (1/second)
defaultOutValue	LREAL	Value for pre-assignment of the output variable
enDefaultOutValue	BOOL	Pre-assign output value
callOB	OB_CYCLIC	Calling cyclic interrupt OB

Output parameters

Table 3-278: Output parameters

Parameters	Data type	Description
delayedValue	LREAL	Output variable
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).
subFunctionStatus	WORD	Status or return value of the called FCs and system blocks.

Status and error displays

Table 3-279: Status/error codes

status	Meaning	Remedy / notes
16#0000	No error	-
16#8200	Negative rate of change.	The parameter for the change rate must not be negative.
16#8600	Error in “QRY_CINT” command.	Check the error code in “subFunctionStatus”
16#8601	OB on input “callOB” is not configured / present.	Interconnect the constant name of a configured cyclic interrupt OB at the input “callOB”.

Note The status of called commands is output in “subFunctionStatus”. In this case, the output value in “status” indicates which command caused the error. In this case, refer to the TIA Portal Online Help section for information on the respective commands.

Principle of operation

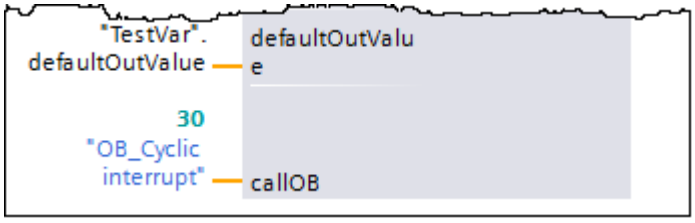
The ramp is a limit line and refers to a rate of change per second; if, for example, “setChangeRate = 10.0” is parameterized at a sampling time of 1s/100ms/10ms for every block call, then if “value > delayedValue”, 10.0/1.0/0.1 is added to “delayedValue” until “value” is reached.

The limitation of the rate of change applies to both positive and negative values for the rise and fall.

The output “delayedValue” can be preset or initialized.

The time interval of the calling cyclic interrupt OB is determined by interconnecting the calling cyclic interrupt OB at the input parameter “callOB”.

Figure 3-135: Interconnecting the cyclic interrupt OB

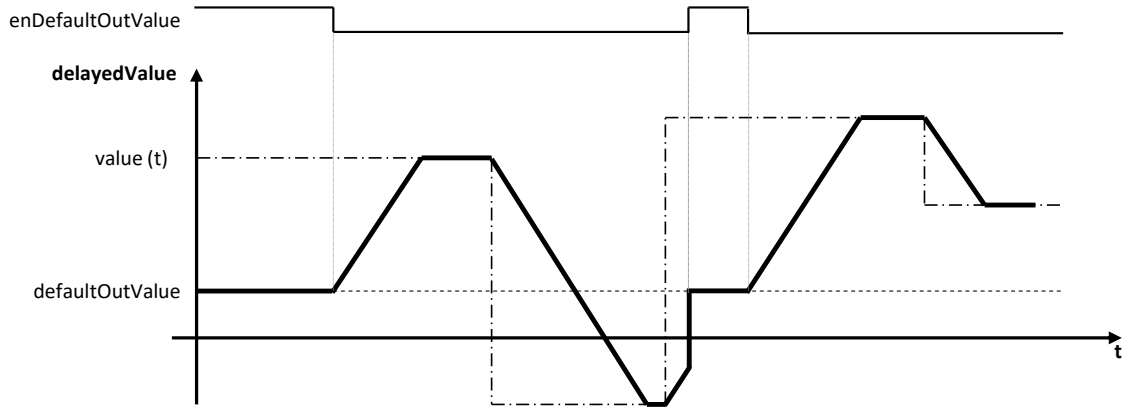


Pre-assigning an output

If “enDefaultOutValue = TRUE” is set, the value at “defaultOutValue” is output. When changing from TRUE to FALSE, the output “delayedValue” is ramped from “defaultOutValue” to “value”. When changing from FALSE to TRUE, the output “delayedValue” immediately jumps to “defaultOutValue”.

Functional processes

Figure 3-136: Ramp function sequence



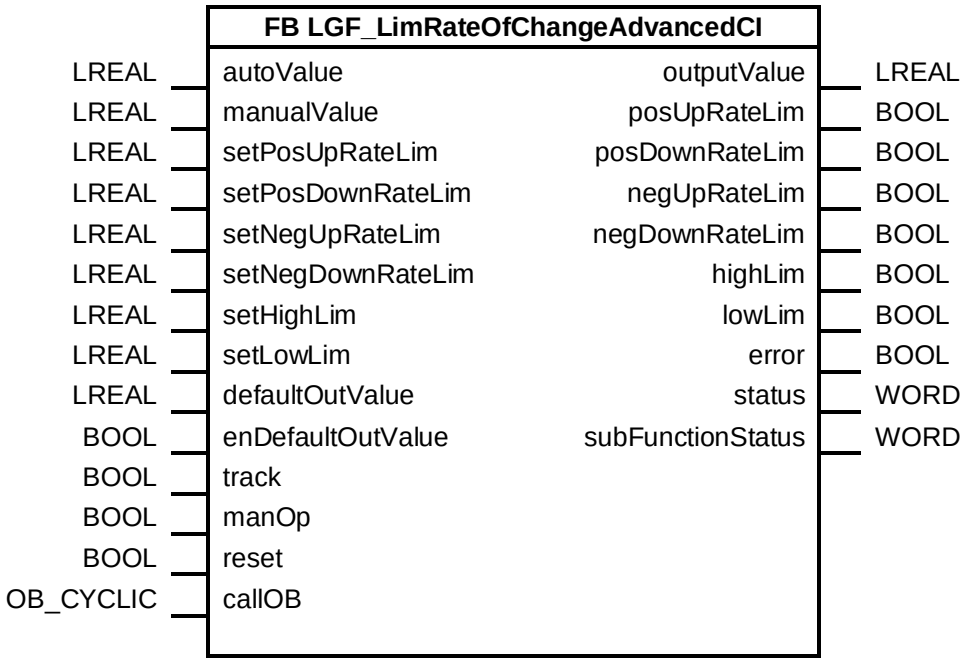
3.8.2 LGF_LimRateOfChangeAdvancedCI

Short description

The block LGF_LimRateOfChangeAdvanced limits the rate of change of an input variable. Jump functions become ramp functions. In addition, the block has various operating modes.

Block

Figure 3-137: FB LGF_LimRateOfChangeAdvancedCI



Input parameters

Table 3-280: Input parameters

Parameters	Data type	Description
autoValue	LREAL	Process variable (jump function)
manualValue	LREAL	Value in manual mode
setPosUpRateLim	LREAL	Rate of change per second for the rising ramp in the positive value range
setPosDownRateLim	LREAL	Rate of change per second for the falling ramp in the positive value range
setNegUpRateLim	LREAL	Rate of change per second for the rising ramp in the negative value range
setNegDownRateLim	LREAL	Rate of change per second for the falling ramp in the negative value range
setHighLim	LREAL	High limit
setLowLim	LREAL	Low limit
defaultOutValue	LREAL	Value for pre-assignment of the output variable
enDefaultOutValue	BOOL	Preset output value (outputValue = defaultOutValue)
track	BOOL	Input variable switching/tracking (outputValue = autoValue)
manOp	BOOL	Manual mode (outputValue = manualValue)

Parameters	Data type	Description
reset	BOOL	Restart
callOB	OB_CYCLIC	Calling cyclic interrupt OB

Output parameters

Table 3-281: Output parameters

Parameters	Data type	Description
outputValue	LREAL	Output variable
posUpRateLim	BOOL	Rise limitation in positive range tripped
posDownRateLim	BOOL	Fall limitation in positive range tripped
negUpRateLim	BOOL	Rise limitation in negative range tripped
negDownRateLim	BOOL	Fall limitation in negative range tripped
highLim	BOOL	High limit tripped
lowLim	BOOL	Low limit tripped
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).
subFunctionStatus	WORD	Status or return value of the called FCs and system blocks.

Status and error displays

Table 3-282: Status/error codes

status	Meaning	Remedy / notes
16#0000	No error	-
16#8200	Error: “setHighLim” < “setLowLim”	The high limit “setHighLim” must be greater than the low limit “setLowLim”.
16#8202	Negative rate of change.	The parameter for the change rate must not be negative.
16#8600	Error in “QRY_CINT” command.	Check the error code in “subFunctionStatus”
16#8601	OB on input “callOB” is not configured / present.	Interconnect the constant name of a configured cyclic interrupt OB at the input “callOB”.

Note The status of called commands is output in “subFunctionStatus”. In this case, the output value in “status” indicates which command caused the error. In this case, refer to the TIA Portal Online Help section for information on the respective commands.

Principle of operation

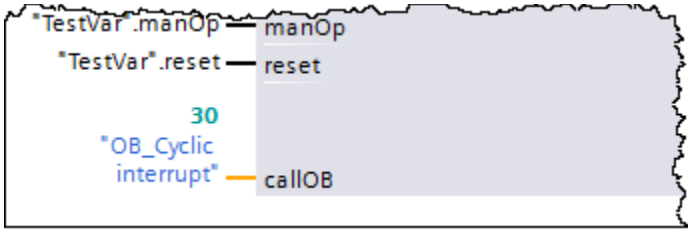
For the positive/negative value range, two rates of change in each case for the ramp (rising and falling values) can be parameterized. The following operating modes can be selected via control inputs:

- Restart
- Pre-assigning an output
- Normal operation (automatic)
- Switch through controlled variable (manual)
- Tracking

The output variable can be limited through two parameterizable limits. An active limitation of the rate of change of a ramp, as well as an active limitation of the output variable are reported via outputs.

The time interval of the calling cyclic interrupt OB is determined by interconnecting the calling cyclic interrupt OB at the input parameter “callOB”.

Figure 3-138: Interconnecting the cyclic interrupt OB



Restart

At restart “reset = TRUE”, the output “outputValue” is reset to 0.0. If “enDefaultOutValue = TRUE” is set, “defaultOutValue” is output. All signal outputs are set to FALSE.

Pre-assigning an output

If “enDefaultOutValue = TRUE” is set, the value at “defaultOutValue” is output. When changing from TRUE to FALSE, “outputValue” is ramped from “defaultOutValue” to “autoValue”. When changing from FALSE to TRUE, the output “outputValue” immediately jumps to “defaultOutValue”.

Normal operation

The ramps are straight lines of limitation and are based on a rate of change per second; if, for example, the parameter “setPosUpRateLim = 10.0” is assigned, then at a sampling time of 1 s/100 ms/10 ms, 10.0/1.0/0.1 will be added to “outputValue” at each block call, if “autoValue > outputValue”, until “autoValue” is reached.

The limitation of the rate of change can be parameterized in both positive and negative ranges for the increase and decrease.

Table 3-283: Marking of the ramps

Parameters	Ramp
setPosUpRateLim	outputValue > 0 and |outputValue| rising
setPosDownRateLim	outputValue > 0 and |outputValue| falling
setNegUpRateLim	outputValue < 0 and |outputValue| rising
setNegDownRateLim	outputValue < 0 and |outputValue| falling

If the ramps are not parameterized (“setPosUpRateLim”, “setPosDownRateLim”, “setNegUpRateLim”, and “setNegDownRateLim” equal 0.0), the output remains at 0.0 and normal operation is disabled.

Tracking

If the input “track = TRUE” is set, the input variable “autoValue” is interconnected directly to the output variable “outputValue”. Thus jumps of the input variable will also be output.

Switch through controlled variable

If “manOp = TRUE” is set, the controlled variable “manualValue” is interconnected directly to the output variable “outputValue”.

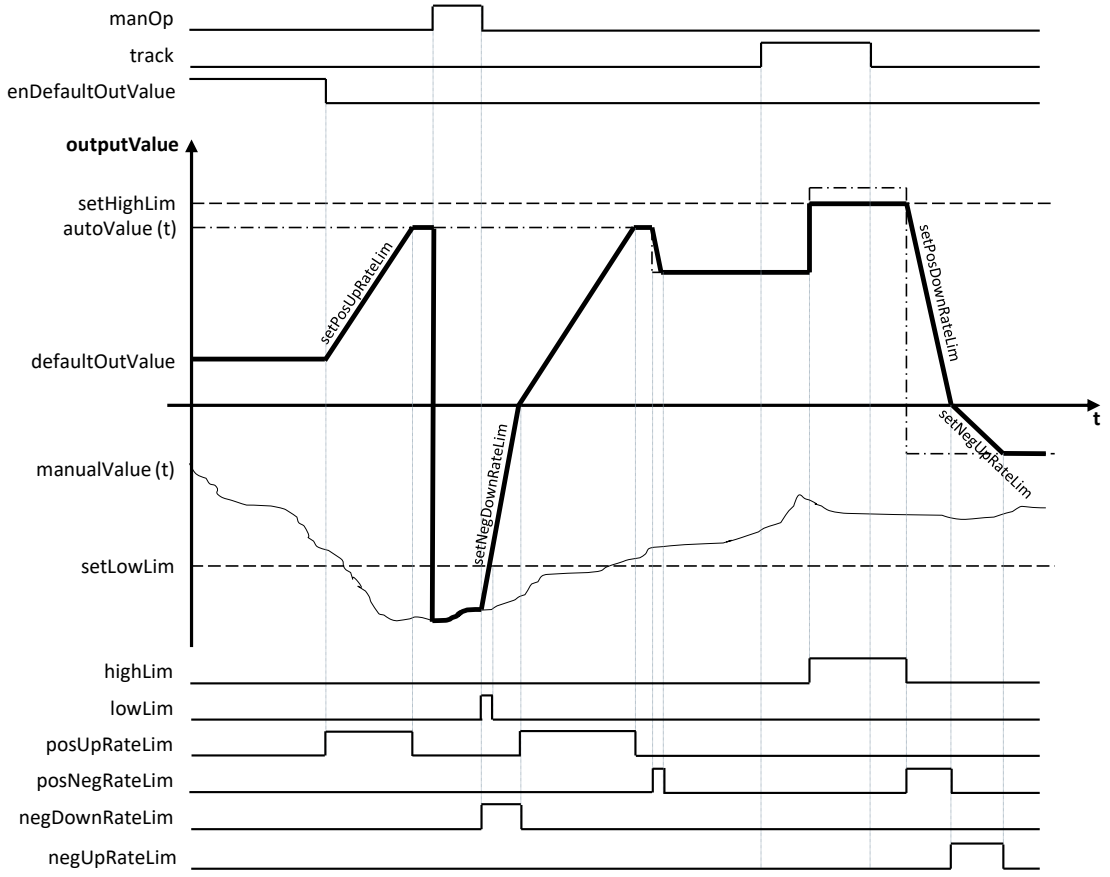
In this operating mode, the parameterization of the ramps or the high/low limitation of the output variable, and the pre-assignment of the output, are ineffective.

When changing from TRUE to FALSE, the output “outputValue” is ramped again after “autoValue”.

As soon as the value range between the low and high limits is reached, the high and low limits are reactivated.

Functional processes

Figure 3-139: Ramp function sequence, operating modes



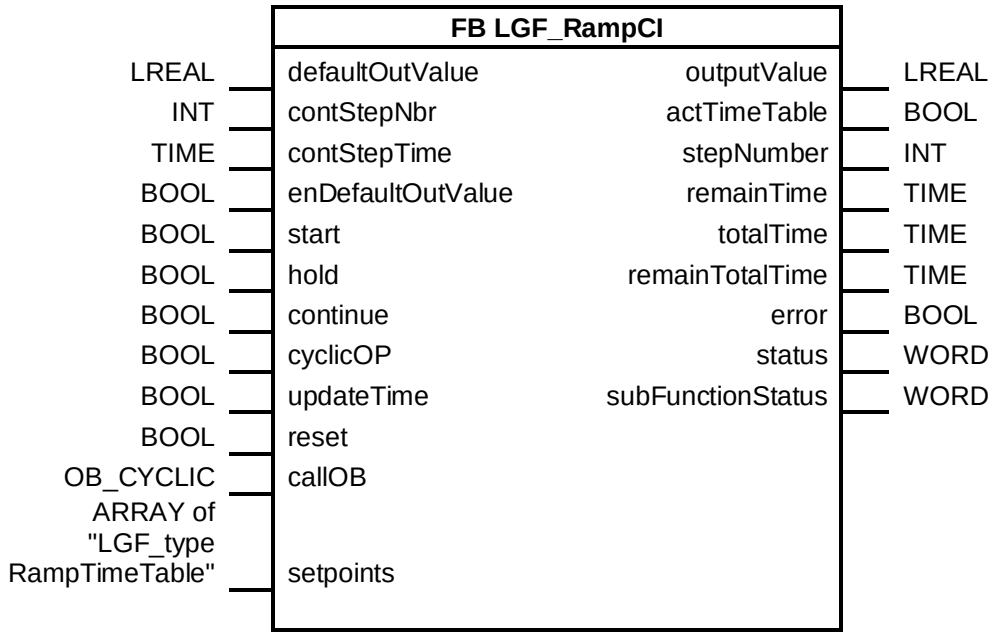
3.8.3 LGF_RampCI

Short description

This block generates a speed curve based on an interpolation point table. Linear interpolation occurs between the points within the prescribed time.

Block

Table 3-284: FB LGF_RampCI



Input parameters

Table 3-285: Input parameters

Parameters	Data type	Description
defaultOutValue	LREAL	Value for pre-assignment of the output variable
contStepNbr	INT	Number of the next interpolation point for continuing
contStepTime	TIME	Remaining time to continue to the interpolation point “contStepNbr”
enDefaultOutValue	BOOL	Preassigning “defaultOutValue” to output variable
start	BOOL	Run down the interpolation point table
hold	BOOL	Hold current value on the output
continue	BOOL	Continuing
cyclicOP	BOOL	Repeat interpolation point table cyclically
updateTime	BOOL	Update time values
reset	BOOL	Restart
callOB	OB_CYCLIC	Calling cyclic interrupt OB
setpoints	ARRY of “LGF_typeRampTimeTable”	Interpolation point table. You can find information on the data type “LGF_typeRampTimeTable” under the item “Global data” .

Output parameters

Table 3-286: Output parameters

Parameters	Data type	Description
outputValue	LREAL	Output variable
actTimeTable	BOOL	The interpolation point table will be edited.
stepNumber	INT	current interpolation point number (interpolation point that is approached)
remainTime	TIME	Remaining time until reaching the next interpolation point
totalTime	TIME	Total time
remainTotalTime	TIME	Total remaining time
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).
subFunctionStatus	WORD	Status or return value of the called FCs and system blocks.

Status and error displays

Table 3-287: Status/error codes

status	Meaning	Remedy / notes
16#0000	No error	-
16 #7000	Initial value	Restart has been executed.
16#7001	First call	Rising edge “start”.
16#7002	Subsequent call	Input “cyclicOP” set.
16#8200	OB on input “callOB” is not configured / present.	Interconnect the constant name of a configured cyclic interrupt OB at the input “callOB”.
16#8201	Low array limit <> 0	The array with the interpolation points must start with the index 0.
16#8400	Error in “QRY_CINT” command.	Check the error code in “subFunctionStatus”

Note The status of called commands is output in “subFunctionStatus”. In this case, the output value in “status” indicates which command caused the error. In this case, refer to the TIA Portal Online Help section for information on the respective commands.

Global data

Together with the block, you automatically receive the PLC data type “LGF_typeRampTimeTable”, which is composed of the parameters “outVal” for the value of a base point and “time” for the time, until the next base point is reached. The declaration takes place in a one-dimensional array of the data type “LGF_typeRampTimeTable” beginning with the index 0. The array is created in a global data block and then passed to the module “LGF_RampCI”.

Figure 3-140: Example of the declaration of the interpolation points

	setpoints	Array[0..9] of "typeTimeTable"	
	setpoints[0]	"typeTimeTable"	
	outVal	Real	1.0
	time	Time	t#5s
	setpoints[1]	"typeTimeTable"	
	outVal	Real	5.0
	time	Time	t#3s
	setpoints[2]	"typeTimeTable"	
	setpoints[3]	"typeTimeTable"	

The parameter “time” of the last interpolation point must be parameterized with 0s, since there is no longer any successor interpolation point.

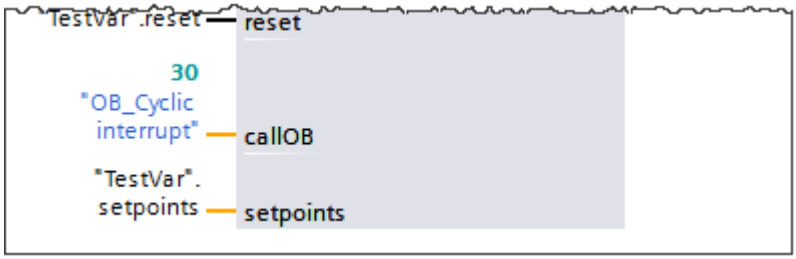
Principle of operation

With this block, speed curves can be executed on the basis of parameterized interpolation points; in each call cycle values are output according to a schedule, and interpolation takes place between the interpolation points.

In each cycle the currently approached interpolation point number “stepNumber”, the actual remaining time “remainTime” until reaching the interpolation point, the total time “totalTime,” and the total remaining time until reaching the end of the speed curve “remainTotalTime”, are output. In addition, the output “actTimeTable” is set if the projected speed curve is currently being output.

The time interval of the calling cyclic interrupt OB is determined by interconnecting the calling cyclic interrupt OB at the input parameter “callOB”.

Figure 3-141: Interconnecting the cyclic interrupt OB



The following operating modes can be selected via control inputs:

- Restart
- Pre-assigning an output
- Output a speed curve
- Stop processing
- Specify processing step and processing time
- Switch-on cyclic operation
- Update total time and remaining time

Overview of the operating modes

Table 3-288: Overview of the operating modes

Operating mode	enDefaultOutValue	start	hold	continue	cyclicOP	updateTime	reset	Output/action
Restart							TRUE ↑	Block is initialized.
Pre-assigning an output	TRUE	TRUE					FALSE	defaultOutValue
Output a speed curve	FALSE	TRUE ↑	FALSE		FALSE		FALSE	outputValue (t); end value is held after processing
Stop speed curve	FALSE	TRUE	TRUE	FALSE			FALSE	current value of outputValue (t) is held
Specify processing step and processing time	FALSE	TRUE	TRUE	TRUE ↑			FALSE	outputValue (old)
			FALSE					Continue with parameterized interpolation point
Switch-on cyclic operation	FALSE	TRUE	FALSE		TRUE		FALSE	outputValue (t); after end of automatic restart
Update total time and remaining time						TRUE ↑	FALSE	Total time and remaining time are updated.

Restart

The output “outValue” is reset to 0.0 with a rising edge at the input “reset”. With “enDefaultOutValue” = TRUE, “defaultOutValue” is output at output. The total time and total remaining time are updated and output.

Pre-assigning an output

If the speed curve should begin with a certain output value, then “enDefaultOutValue” must be TRUE. In this case the value “defaultOutValue” is present on the output of the timer. The internal processing of the speed curve continues during this time. If “enDefaultOutValue” changes to FALSE again, interpolation is performed to the currently active calibration point.

Output a speed curve

With a rising edge at the input “start”, the speed curve is output—as long as “start” is TRUE or until the speed curve is terminated by reaching the last interpolation point. Through a subsequent rising edge the speed curve is output again. In addition, the total time is updated at each switch-on.

Switch-on cyclic operation

If, in addition to the input “start”, the input “cyclicOP” is also set to TRUE, the speed curve automatically returns to the start point after outputting the last interpolation point value and starts a new cycle.

There is no interpolation between the last interpolation point value and the starting point. The following must apply for a smooth transition: last interpolation point value = start point.

Stop speed curve

With “hold” = TRUE the value of the output variable (including time processing) is frozen. When resetting “hold” = FALSE, the program continues at the point of interruption or at a parameterized point (see “Defining the processing step and processing time”). The processing time of the speed curve is extended by the holding time “T1*”. (see Figure 3-142).

Specify processing step and processing time

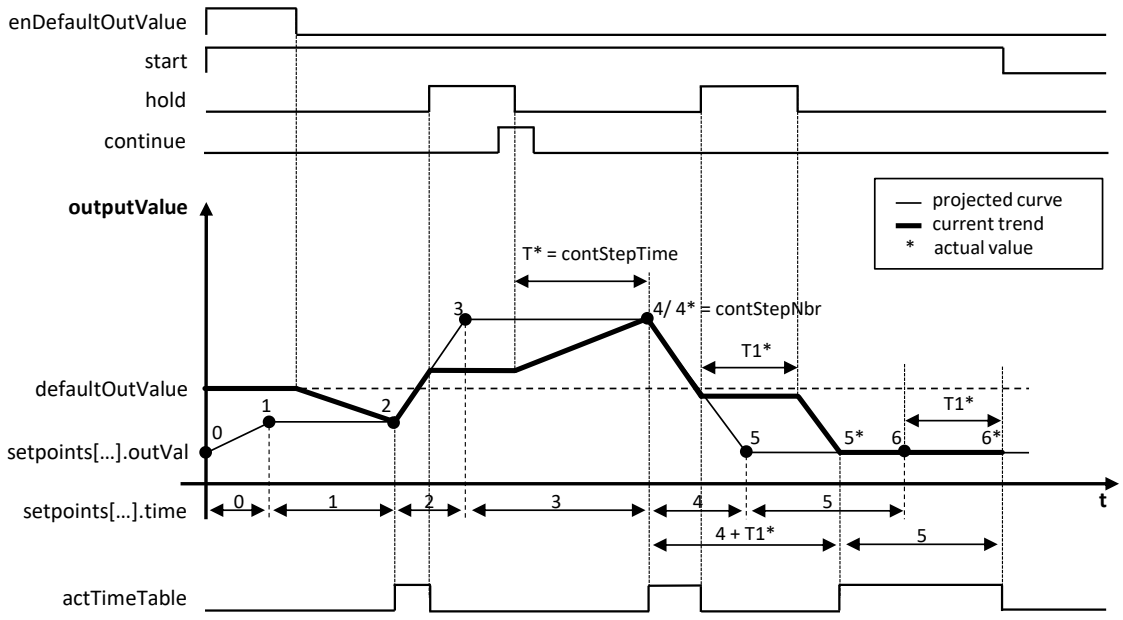
If the input parameter “continue” is set to TRUE for continuation while the speed curve is stopped (“hold” = TRUE), then after the input “hold” has been reset the interpolation point number “contStepNbr” (target interpolation point) will be approached within the time “contStepTime” (interpolation). The total remaining time will be recalculated.

Updating total time and total remaining time

If values of the interpolation points are changed, the total time and the total remaining time of the speed curve can change. Since calculation of “totalTime” and “remainTotalTime” can significantly increase the processing time of the function block at many interpolation points, the calculation is only executed once with a rising edge on the “updateTime” input.

Functional processes

Figure 3-142: Functional processes



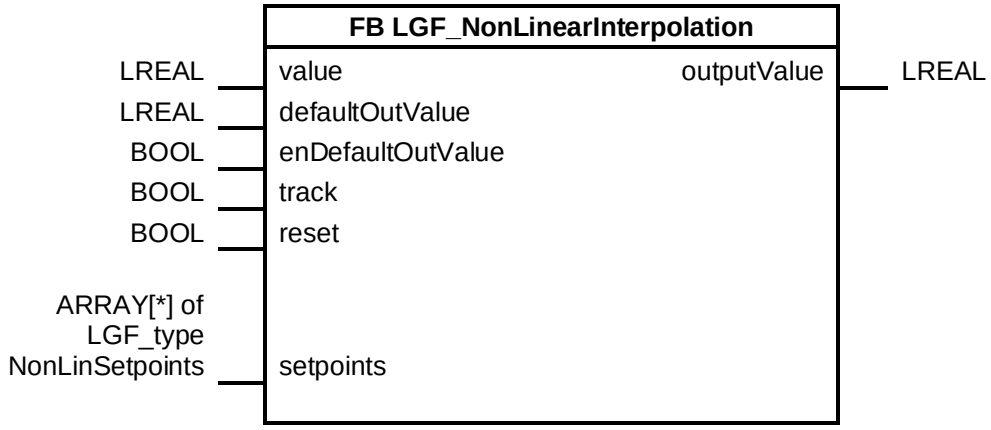
3.8.4 LGF_NonLinearInterpolation

Short description

This block implements a characteristic curve. The characteristic curve is defined via an interpolation point table with linear interpolation between the interpolation points. A prescribed input value generates an output value in each cycle based on the characteristic curve from the interpolation point table.

Block

Figure 3-143: FB LGF_NonLinearInterpolation



Input parameters

Table 3-289: Input parameters

Parameters	Data type	Description
value	REAL	Input value for calculating the output value over the defined characteristic curve.
defaultOutValue	REAL	Standard output value without using the characteristic curve
enDefaultOutValue	BOOL	Activation of the standard output value The standard output value is output as long as this input is set. (outputValue = defaultOutValue)
track	BOOL	The value of the output “outputValue” follows the value of the input “value” without using the characteristic curve as long as this input is set. (outputValue = value)
reset	BOOL	If the interpolation point table is changed in running operation, the input “reset” must be activated afterwards. Otherwise, the block cannot guarantee correct execution. (outputValue = 0.0)

Input/output parameters (InOut)

Table 3-290: Input/output parameters (InOut)

Parameters	Data type	Description
setpoints	ARRAY[*] of LGF_typeNonLinSetpoints	Support point table for defining the characteristic curve (polynomial).

Output parameters

Table 3-291: Output parameters

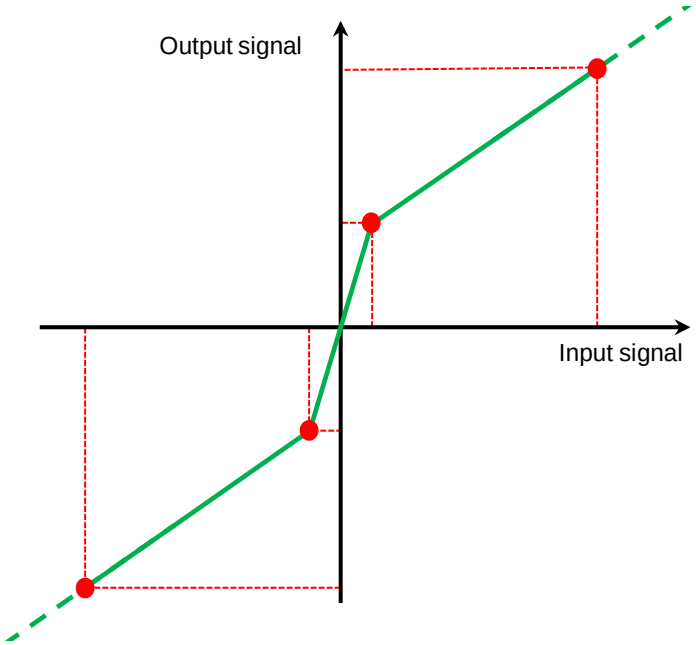
Parameters	Data type	Description
outputValue	REAL	The output value that has been calculated from the input value over the defined characteristic curve.

Principle of operation

The value of the output “outputValue” based on the following priority:

1. As long as the input “enDefaultOutValue” is set, the value defined via the parameter “defaultOutValue” will be output as output value.
2. As long as the input “reset” is set, the block is reset and the output value is 0.0.
3. As long as the “track” input is set, the output value will be output directly as input value, without consideration of the characteristic curve.
4. Based on the input value, a characteristic curve value is calculated via the linearly interpolated, interpolation point table and output as an output value.
   - If the input value is between two interpolation points within the interpolation point table, the output value is calculated as the intersection with the connecting line between the preceding and following interpolation points (see Figure 3-144).
   - If the input value is before the first interpolation point (lowest value defined in the interpolation point table), the output value will be calculated as the intersection of the line formed by the first two interpolation points of the interpolation point table.
   - If the input value is after the last interpolation point (highest value defined in the interpolation point table), the output value will be calculated as the intersection of the line formed by the last two interpolation points of the interpolation point table.

Figure 3-144: Sample path of the output signal



NOTICE	To keep the computing time of the block as short as possible, there is no check of the parameterization or the data of the interpolation point table. When entering the interpolation points in the interpolation point table, the following particularities must be taken into account. If these particularities are not taken into account the block can malfunction. • At least two interpolation points must be entered in the interpolation point table. • The interpolation points in the interpolation point table must be entered in the Table in ascending order of the input values.

Interpolation point table

The interpolation point table is implemented through a variable of the data type Array. The type of the array corresponds to the PLC data type “LGF_typeNonLinSetpoints”.

You can create the interpolation point table in any global data block. The size of the array depends on the number of interpolation points.

Example

Figure 3-145: Sample data block

	Name	Data type	Start value
1	Static		
2	nonLinSetpoints	Array[0..4] of "LGF_typeNonLinSetpoints"	
3	nonLinSetpoints[0]	"LGF_typeNonLinSetpoints"	
4	inputValue	LReal	-2000.0
5	outputValue	LReal	-2200.0
6	nonLinSetpoints[1]	"LGF_typeNonLinSetpoints"	
7	inputValue	LReal	-200.0
8	outputValue	LReal	-400.0
9	nonLinSetpoints[2]	"LGF_typeNonLinSetpoints"	
10	inputValue	LReal	0.0
11	outputValue	LReal	0.0
12	nonLinSetpoints[3]	"LGF_typeNonLinSetpoints"	
13	inputValue	LReal	200.0
14	outputValue	LReal	400.0
15	nonLinSetpoints[4]	"LGF_typeNonLinSetpoints"	
16	inputValue	LReal	2000.0
17	outputValue	LReal	2200.0

3.8.5 Rules of simulated controlled systems

In the entry “Closed-Loop Control of Simulated Controlled Systems” you will find the block library “LSim” for simulation of controlled systems for the controller families SIMATIC S7-1200 and SIMATIC S7-1500.

https://support.industry.siemens.com/cs/ww/en/view/79047707

3.9 Measurement data processing

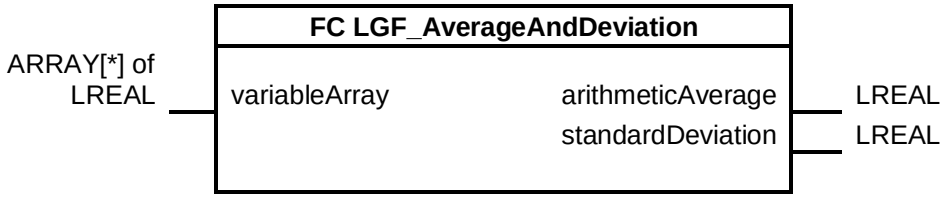
3.9.1 LGF_AverageAndDeviation

Short description

This block calculates the arithmetic mean and the standard deviation from a series of numbers.

Block

Figure 3-146: FC LGF_AverageAndDeviation



Input parameters

Table 3-292: Input parameters

Parameters	Data type	Description
variableArray	ARRAY[*] of LREAL	Sequence of numbers to calculate with

Output parameters

Table 3-293: Output parameters

Parameters	Data type	Description
arithmeticAverage	LREAL	Arithmetic average value
standardDeviation	LREAL	Standard deviation

Principle of operation

An array of any size is connected via the “variableArray” input. After reading-out the array boundaries, the arithmetic mean value and the standard deviation will be calculated from the values and both will be output.

Note An array with too many elements can cause the cycle monitoring time to be exceeded.

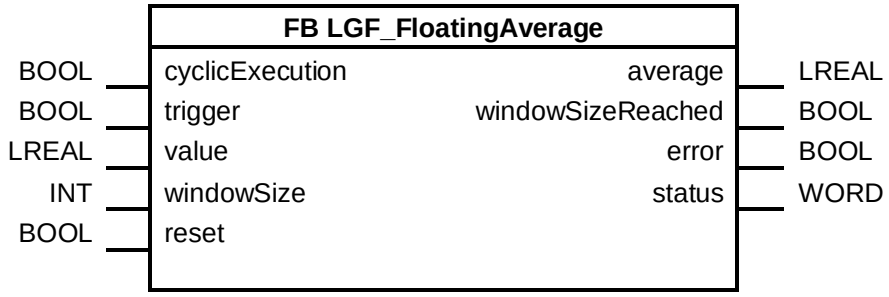
3.9.2 LGF_FloatingAverage

Short description

This block calculates a moving arithmetic mean value from REAL values. This method can be used to smooth data series. The values can be read in cyclically or triggered.

Block

Figure 3-147: LGF_FloatingAverage



Input parameters

Table 3-294: Input parameters

Parameters	Data type	Description
cyclicExecution	BOOL	Cyclic program
trigger	BOOL	Read in with every pulse at input “trigger”
value	LREAL	Values from which the moving average is to be determined.
windowsSize	INT	Window length for sliding averaging in the range from 1..100 The standard value is 100 dB.
reset	BOOL	The block is reset and the calculation starts again.

Note

The block “LGF_FloatingAverage” does not query the data type for the input parameter “value”. For data types other than REAL, either an implicit conversion is performed automatically or an error is generated during compilation.

You can find further information in the Chapter “Overview of Data Type Conversion” in the Online Help section of the TIA Portal or under:

https://support.industry.siemens.com/cs/ww/en/view/109773506/100611494667

Output parameters

Table 3-295: Output parameters

Parameters	Data type	Description
average	LREAL	Moving average
windowSizeReached	BOOL	FALSE: Maximum window width not yet reached TRUE: Maximum window width reached
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).

Status and error displays

Table 3-296: Status/error codes

status	Meaning	Remedy / notes
16#0000	No error	-
16#8200	No correct window width	Set a value between 1 and 100.

Principle of operation

The block calculates the (moving) mean value based on the set window width. The window width indicates the maximum number of values read in last. After the maximum number of values has been read, the output “windowSizeReached” is set and each newly read value replaces the oldest value (FIFO principle).

Two options are available for reading the values. With the input “cyclicExecution”, the values are read and calculated cyclically. With the “trigger” input, the values are read in and calculated with each pulse.

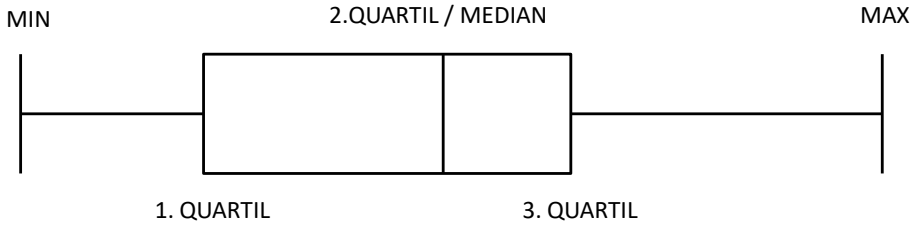
3.9.3 LGF_Boxplot_DInt

Short description

If you want to get an overview of existing data, you can use a Boxplot diagram. A Boxplot shows you in which area the data is located and how it is distributed over this area. A Boxplot consists of the following parameters:

- Minimum (smallest occurring value of the sample)
- Lower or first quartile (below this value are 25% of the sample values)
- Median or second quartile (below this value are 50% of the sample values)
- Upper or third quartile (below this value are 75% of the sample values)
- Maximum (largest occurring value of the sample)

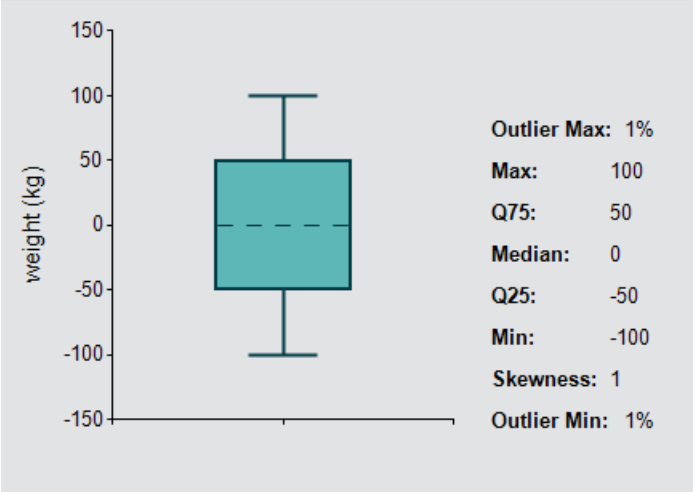
Figure 3-148: Box plot



WinCC-Control

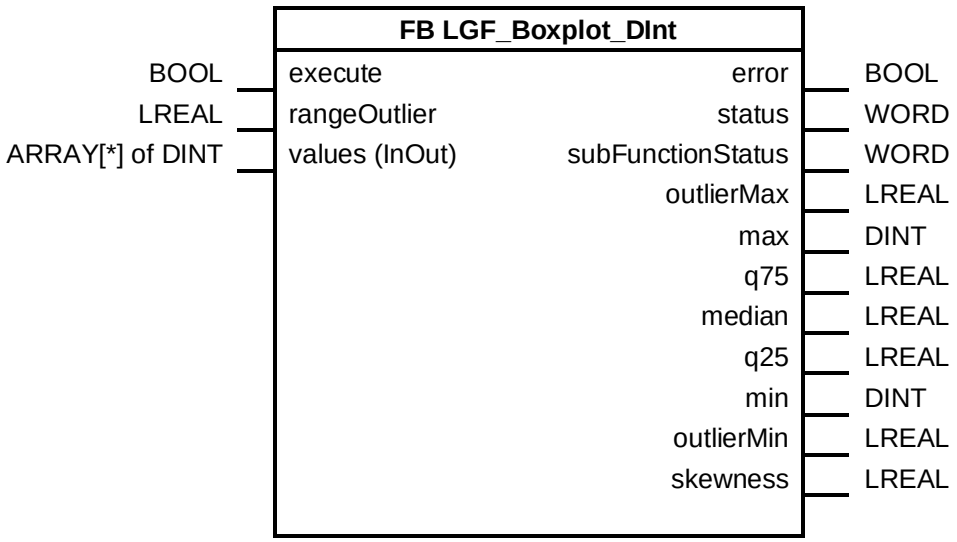
To visualize the Boxplot, the Siemens Industry Online Support offers you a Net-Control, which you can use in conjunction with WinCC Runtime Professional. You can find the download under the entry ID: 81662739.

Figure 3-149: .Net Control “Boxplot”



Block

Figure 3-150: LGF_Boxplot_DInt



Input parameters

Table 3-297: Input parameters

Parameters	Data type	Description
execute	BOOL	Activation of the calculation with each positive edge.
rangeOutlier	LREAL	Outlier detection: • 0: Outlier detection is deactivated • 0-1: Invalid value • >1: Outlier detection is activated.

Input/output parameters (InOut)

Table 3-298: Input/output parameters (InOut)

Parameters	Data type	Description
values	ARRAY[*] of DINT	Array that should be used for calculation.

Output parameters

Table 3-299: Output parameters

Parameters	Data type	Description
outlierMax	LREAL	Upper outliers in %.
max	DINT	Maximum Value, not an outlier.
q75	LREAL	3rd quartile or Q75 of the data series.
median	LREAL	2nd quartile or Median of the data series.
q25	LREAL	1st quartile or Q25 of the data series.
min	DINT	Minimum Value, not an outlier.
outlierMin	LREAL	Lower outliers in %.
skewness	LREAL	Skewness of the data series.
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.

Parameters	Data type	Description
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).
subFunctionStatus	WORD	Status or return value of the called FCs and system blocks.

Status and error displays

Table 3-300: Status/error codes

status	Meaning	Remedy / notes
16#0000	No error	-
16 #7000	Block is not being edited	-
16#7001	First FB call.	-
16#8200	Negative array boundary not allowed	Check the array at the input.
16#8600	Error in command “LGF_ShellSort_DInt”.	Check the error code in “subFunctionStatus”. Information concerning this block is provided in Chapter 3.5.3.
16#9101	The parameter “rangeOutlier” type is invalid	Enter a valid “rangeOutlier” value for the parameter: • 0: Outlier detection is deactivated • >1 Valid value.

Principle of operation

The block sorts the data series and then calculates the so-called “five-point summary”:

Table 3-301: Five-point summary

Characteristic value of the five-point summary	Output parameter of the block
Minimum (smallest occurring value of the sample)	min
Lower or first quartile (below this value are 25% of the sample values)	q25
Median or second quartile (below this value are 50% of the sample values)	median
Upper or third quartile (below this value are 75% of the sample values)	q75
Maximum (largest occurring value of the sample)	max

If outlier detection is activated, the block first calculates the limits. From these limit values, the values are recognized as outliers:

$$Bound_{lower} = q25 - rangeOutlier * (q75 - q25)$$

$$Bound_{upper} = q75 + rangeOutlier * (q75 - q25)$$

The block then calculates new values for the parameters “max” and “min”, which lie within the outlier limits. The outliers are counted and output as a percentage.

To make it easier to judge how the data is distributed, the block also calculates the skew. The skewness lies between the values -1 and 1 with the following meaning:

- -1: extremely left skewed distribution
- 0: symmetrical distribution
- 1: extreme right-skew distribution

The elements of the passed array are sorted in ascending order by the block. The "LGF_Shellsort_DInt" block is used for sorting. (see 3.5.3).

The parameters are calculated as follows:

Table 3-302: Boxplot formulas

Parameters	Formula
q25 (1st quartile)	$q_{25} = x_{(k)}$ with $k = \frac{\left\{\left\lceil \frac{1}{2}(n+1)\right\rceil+1\right\}}{2} = \frac{n+3}{4}$
median / q50 (2nd quartile)	$q_{50} = x_{([n+1]/2)}$
q75 (3rd quartile)	$q_{75} = x_{(n+1-k)}$ with $n+1-k = \frac{3n+1}{4}$ n := number of samples (size of array) If the result of the element to be determined (from which the quartiles can be derived) is not an integer, the quartile is calculated from the linear fraction between the two adjacent samples.
skewness	$\text{skewness} = \frac{(q_{25} + q_{75}) - 2 * \text{q}_{50}}{\text{q}_{75} - \text{q}_{25}}$ **Note:** This is just an approximation.

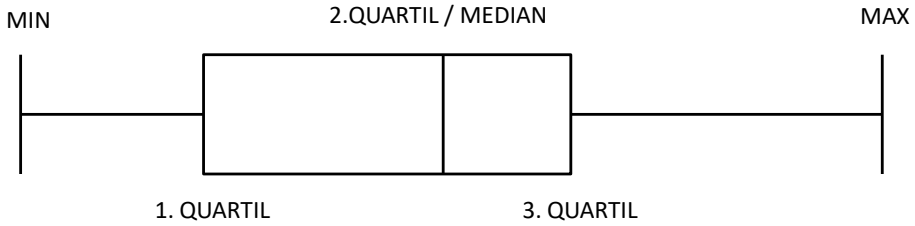
3.9.4 LGF_Boxplot_UDInt

Short description

If you want to get an overview of existing data, you can use a Boxplot diagram. A Boxplot shows you in which area the data is located and how it is distributed over this area. A Boxplot consists of the following parameters:

- Minimum (smallest occurring value of the sample)
- Lower or first quartile (below this value are 25% of the sample values)
- Median or second quartile (below this value are 50% of the sample values)
- Upper or third quartile (below this value are 75% of the sample values)
- Maximum (largest occurring value of the sample)

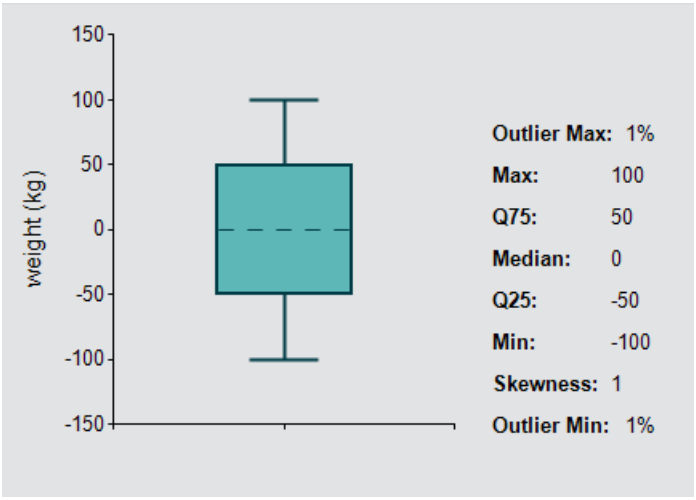
Figure 3-151: Box plot



WinCC-Control

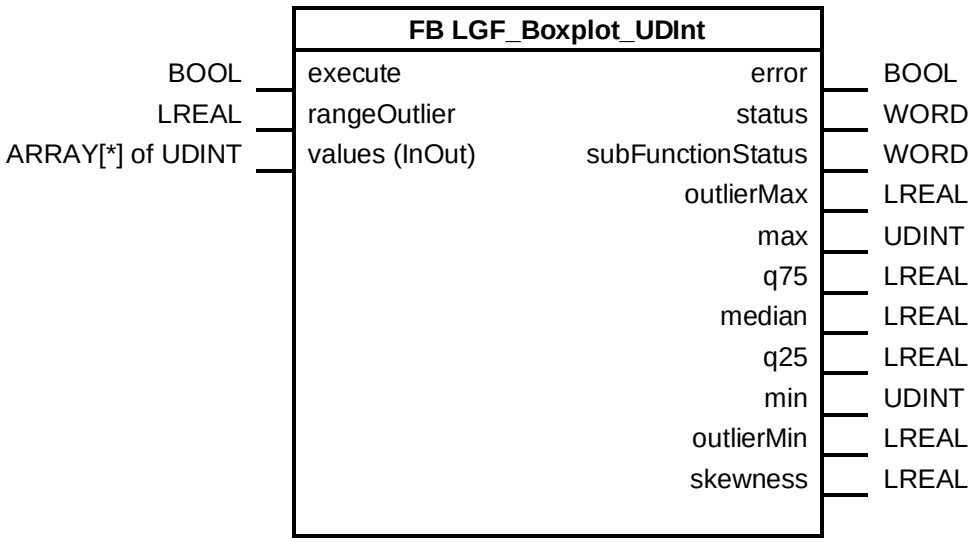
To visualize the Boxplot, the Siemens Industry Online Support offers you a Net-Control, which you can use in conjunction with WinCC Runtime Professional. You can find the download under the entry ID: 81662739.

Figure 3-152: .Net Control “Boxplot”



Block

Figure 3-153: LGF_Boxplot_UDInt



Input parameters

Table 3-303: Input parameters

Parameters	Data type	Description
execute	BOOL	Activation of the calculation with each positive edge.
rangeOutlier	LREAL	Outlier detection: • 0: Outlier detection is deactivated • 0-1: Invalid value • >1: Outlier detection is activated.

Input/output parameters (InOut)

Table 3-304: Input/output parameters (InOut)

Parameters	Data type	Description
values	ARRAY[*] of UDINT	Array that should be used for calculation.

Output parameters

Table 3-305: Output parameters

Parameters	Data type	Description
outlierMax	LREAL	Upper outliers in %.
max	UDINT	Maximum Value, not an outlier.
q75	LREAL	3rd quartile or Q75 of the data series.
median	LREAL	2nd quartile or Median of the data series.
q25	LREAL	1st quartile or Q25 of the data series.
min	UDINT	Minimum Value, not an outlier.
outlierMin	LREAL	Lower outliers in %.
skewness	LREAL	Skewness of the data series.
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.

Parameters	Data type	Description
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).
subFunctionStatus	WORD	Status or return value of the called FCs and system blocks.

Status and error displays

Table 3-306: Status/error codes

status	Meaning	Remedy / notes
16#0000	No error	-
16 #7000	Block is not being edited	-
16#7001	First FB call.	-
16#8200	Negative array boundary not allowed	Check the array at the input.
16#8600	Error in command “LGF_ShellSort_UDInt”.	Check the error code in “subFunctionStatus”. Information on this block can be found in Chapter 3.5.4.
16#9101	The parameter “rangeOutlier” type is invalid	Enter a valid “rangeOutlier” value for the parameter: • 0: Outlier detection is deactivated • >1 Valid value.

Principle of operation

The block sorts the data series and then calculates the so-called “five-point summary”:

Table 3-307: Five-point summary

Characteristic value of the five-point summary	Output parameter of the block
Minimum (smallest occurring value of the sample)	min
Lower or first quartile (below this value are 25% of the sample values)	q25
Median or second quartile (below this value are 50% of the sample values)	median
Upper or third quartile (below this value are 75% of the sample values)	q75
Maximum (largest occurring value of the sample)	max

If outlier detection is activated, the block first calculates the limits. From these limit values, the values are recognized as outliers:

$$Bound_{lower} = q25 - rangeOutlier * (q75 - q25)$$

$$Bound_{upper} = q75 + rangeOutlier * (q75 - q25)$$

The block then calculates new values for the parameters “max” and “min”, which lie within the outlier limits. The outliers are counted and output as a percentage.

To make it easier to judge how the data is distributed, the block also calculates the skew. The skewness lies between the values -1 and 1 with the following meaning:

- -1: extremely left skewed distribution
- 0: symmetrical distribution
- 1: extreme right-skew distribution

The elements of the passed array are sorted in ascending order by the block. The “LGF_Shellsort_UDInt” block is used for sorting. (see 3.5.4).

The parameters are calculated as follows:

Table 3-308: Boxplot formulas

Parameters	Formula
q25 (1st quartile)	$q_{25} = x_{(k)}$ with $k = \frac{\left\{\left[\frac{1}{2}(n+1)\right]+1\right\}}{2} = \frac{n+3}{4}$
median / q50 (2nd quartile)	$q_{50} = x_{([n+1]/2)}$
q75 (3rd quartile)	$q_{75} = x_{(n+1-k)}$ with $n+1-k = \frac{3n+1}{4}$ n := number of samples (size of array) If the result of the element to be determined (from which the quartiles can be derived) is not an integer, the quartile is calculated from the linear fraction between the two adjacent samples.
skewness	$\text{skewness} = \frac{(q_{25} + q_{75}) - 2 * q_{50}}{q_{75} - q_{25}}$ **Note:** This is just an approximation.

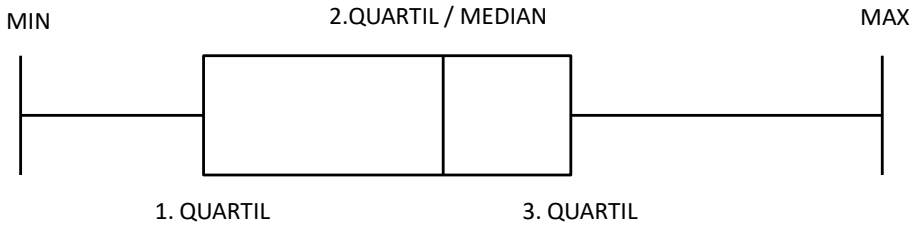
3.9.5 LGF_Boxplot_LReal

Short description

If you want to get an overview of existing data, you can use a Boxplot diagram. A Boxplot shows you in which area the data is located and how it is distributed over this area. A Boxplot consists of the following parameters:

- Minimum (smallest occurring value of the sample)
- Lower or first quartile (below this value are 25% of the sample values)
- Median or second quartile (below this value are 50% of the sample values)
- Upper or third quartile (below this value are 75% of the sample values)
- Maximum (largest occurring value of the sample)

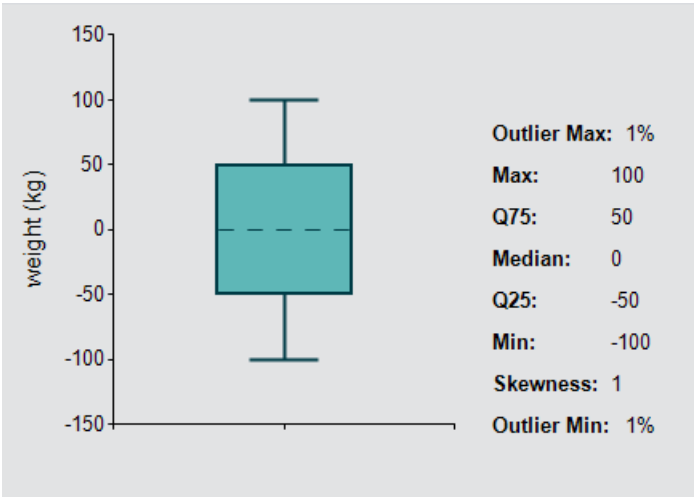
Figure 3-154: Box plot



WinCC-Control

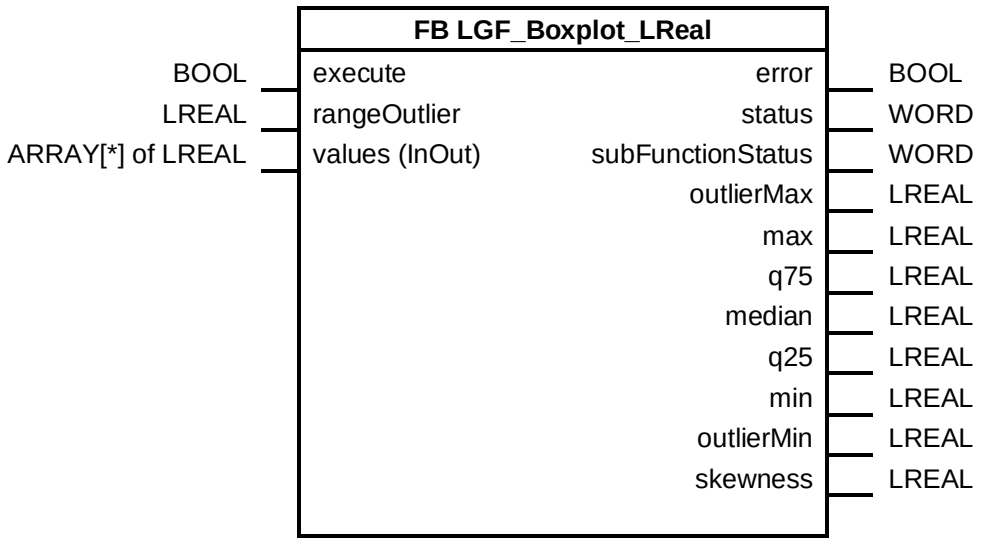
To visualize the Boxplot, the Siemens Industry Online Support offers you a Net-Control, which you can use in conjunction with WinCC Runtime Professional. You can find the download under the entry ID: 81662739.

Figure 3-155: .Net Control “Boxplot”



Block

Figure 3-156: LGF_Boxplot_LReal



Input parameters

Table 3-309: Input parameters

Parameters	Data type	Description
execute	BOOL	Activation of the calculation with each positive edge.
rangeOutlier	LREAL	Outlier detection: • 0: Outlier detection is deactivated • 0-1: Invalid value • >1: Outlier detection is activated.

Input/output parameters (InOut)

Table 3-310: Input/output parameters (InOut)

Parameters	Data type	Description
values	ARRAY[*] of LREAL	Array that should be used for calculation.

Output parameters

Table 3-311: Output parameters

Parameters	Data type	Description
outlierMax	LREAL	Upper outliers in %.
max	LREAL	Maximum Value, not an outlier.
q75	LREAL	3rd quartile or Q75 of the data series.
median	LREAL	2nd quartile or Median of the data series.
q25	LREAL	1st quartile or Q25 of the data series.
min	LREAL	Minimum Value, not an outlier.
outlierMin	LREAL	Lower outliers in %.
skewness	LREAL	Skewness of the data series.
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.

Parameters	Data type	Description
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).
subFunctionStatus	WORD	Status or return value of the called FCs and system blocks.

Status and error displays

Table 3-312: Status/error codes

status	Meaning	Remedy / notes
16#0000	No error	-
16 #7000	Block is not being edited	-
16#7001	First FB call.	-
16#8200	Negative array boundary not allowed	Check the array at the input.
16#8600	Error in command “LGF_ShellSort_LReal”.	Check the error code in “subFunctionStatus”. Information concerning this block is provided in Chapter 3.5.5.
16#9101	The parameter “rangeOutlier” type is invalid	Enter a valid “rangeOutlier” value for the parameter: • 0: Outlier detection is deactivated • >1 Valid value.

Principle of operation

The block sorts the data series and then calculates the so-called “five-point summary”:

Table 3-313: Five-point summary

Characteristic value of the five-point summary	Output parameter of the block
Minimum (smallest occurring value of the sample)	min
Lower or first quartile (below this value are 25% of the sample values)	q25
Median or second quartile (below this value are 50% of the sample values)	median
Upper or third quartile (below this value are 75% of the sample values)	q75
Maximum (largest occurring value of the sample)	max

If outlier detection is activated, the block first calculates the limits. From these limit values, the values are recognized as outliers:

$$Bound_{lower} = q25 - rangeOutlier * (q75 - q25)$$

$$Bound_{upper} = q75 + rangeOutlier * (q75 - q25)$$

The block then calculates new values for the parameters “max” and “min”, which lie within the outlier limits. The outliers are counted and output as a percentage.

To make it easier to judge how the data is distributed, the block also calculates the skew. The skewness lies between the values -1 and 1 with the following meaning:

- -1: extremely left skewed distribution
- 0: symmetrical distribution
- 1: extreme right-skew distribution

The elements of the passed array are sorted in ascending order by the block. The “LGF_Shellsort_LReal” block is used for sorting. (see 3.5.5).

The parameters are calculated as follows:

Table 3-314: Boxplot formulas

Parameters	Formula
q25 (1st quartile)	$q_{25} = x_{(k)}$ with $k = \frac{\left\{\left[\frac{1}{2}(n+1)\right]+1\right\}}{2} = \frac{n+3}{4}$
median / q50 (2nd quartile)	$q_{50} = x_{([n+1]/2)}$
q75 (3rd quartile)	$q_{75} = x_{(n+1-k)}$ with $n+1-k = \frac{3n+1}{4}$ n := number of samples (size of array) If the result of the element to be determined (from which the quartiles can be derived) is not an integer, the quartile is calculated from the linear fraction between the two adjacent samples.
skewness	$\text{skewness} = \frac{(q_{25} + q_{75}) - 2 * \text{q}_{50}}{\text{q}_{75} - \text{q}_{25}}$ **Note:** This is just an approximation.

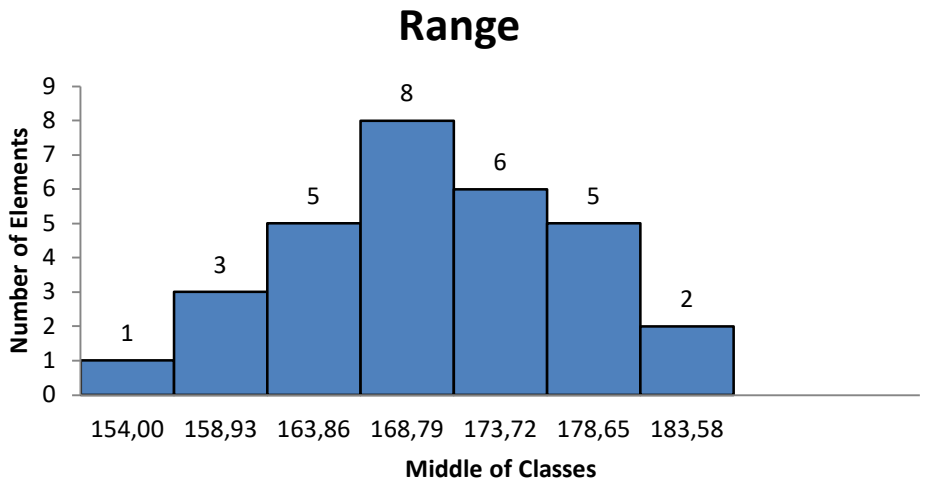
3.9.6 LGF_Histogram_DInt

Short description

The histogram shows the frequency distribution of a sample by class. A class describes a value interval in which the individual frequencies are added together. After specifying the number of classes, the class width and the respective class center are calculated. The number of classes is limited to 15.

The distribution is represented as a rectangle around the class mean with the class width and the cumulated frequency as height.

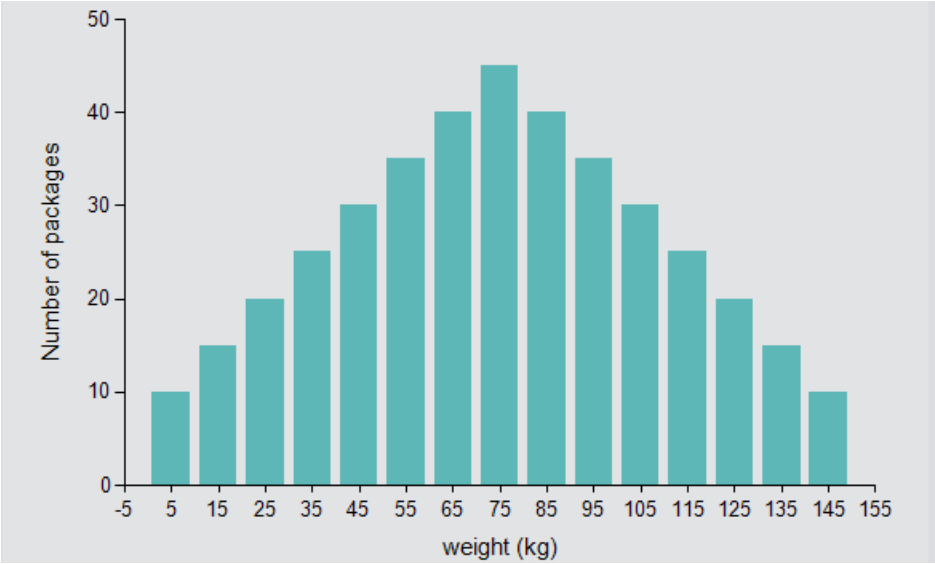
Figure 3-157: Distribution



WinCC-Control

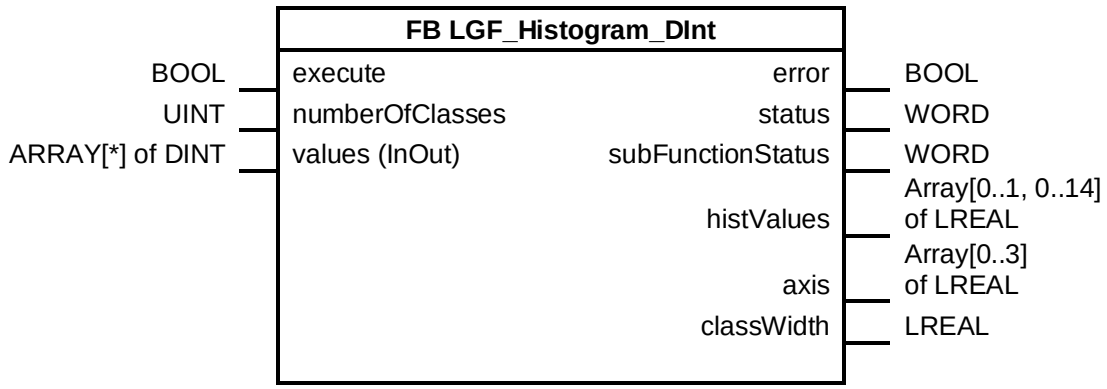
To visualize the histogram, Siemens Industry Online Support offers you a .Net Control that you can use in conjunction with WinCC Runtime Professional. You can find the download under the entry ID: 81662739.

Figure 3-158: .Net-Control



Block

Figure 3-159: LGF_Histogram_DInt



Input parameters

Table 3-315: Input parameters

Parameters	Data type	Description
execute	BOOL	Activation of the calculation with each positive edge.
numberOfClasses	UINT	Number of desired classes.

Input/output parameters (InOut)

Table 3-316: Input/output parameters (InOut)

Parameters	Data type	Description
values	ARRAY[*] of DINT	Array that should be used for calculation.

Output parameters

Table 3-317: Output parameters

Parameters	Data type	Description
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).
subFunctionStatus	WORD	Status or return value of the called FCs and system blocks.
histValues	ARRAY [0..1, 0..14] of LREAL	Outputs the calculated values in a two-dimensional array. • HistValues[0,0..14] displays the relative frequency of the individual classes. • HistValues[1,0..14] displays the class centers. • If fewer than 15 classes are desired, the array elements that are not required are output with 0.
axis	Array[0..3] of LREAL	Specifies the axis values: • Lower X axis value • Upper X axis value • Lower Y axis value • Upper Y axis value
classWidth	LREAL	Returns the calculated class width.

Status and error displays

Table 3-318: Status/error codes

status	Meaning	Remedy / notes
16#0000	No error	-
16 #7000	Block is not being edited	-
16#7001	First FB call.	-
16#8600	Error in command “LGF_ShellSort_DInt”.	Check the error code in “subFunctionStatus”. Information concerning this block is provided in Chapter 3.5.3.
16#9101	Incorrect number of classes	Give the parameter “NumberOfClasses” a valid value (1 to 15).

Principle of operation

The block sorts the passed data and calculates the general class width using the passed class count and data range. The block then counts the values that lie within a class. In order to draw a histogram, the block also calculates the necessary X and Y coordinates.

The elements of the passed array “values” are sorted in ascending order by the block. The block “LGF_ShellSort_DInt” is used for sorting (see 3.5.3).

The number of classes can be specified using the following rule of thumb:

$$Number\ of\ Classes = \sqrt{number\ of\ elements}$$

e.g. 100 values -> $Number\ of\ Classes = \sqrt{100} = 10$

Formulas

The block uses the following formula to calculate the class width:

$$classWidthh = \frac{max\ - min}{Number\ of\ classes}$$

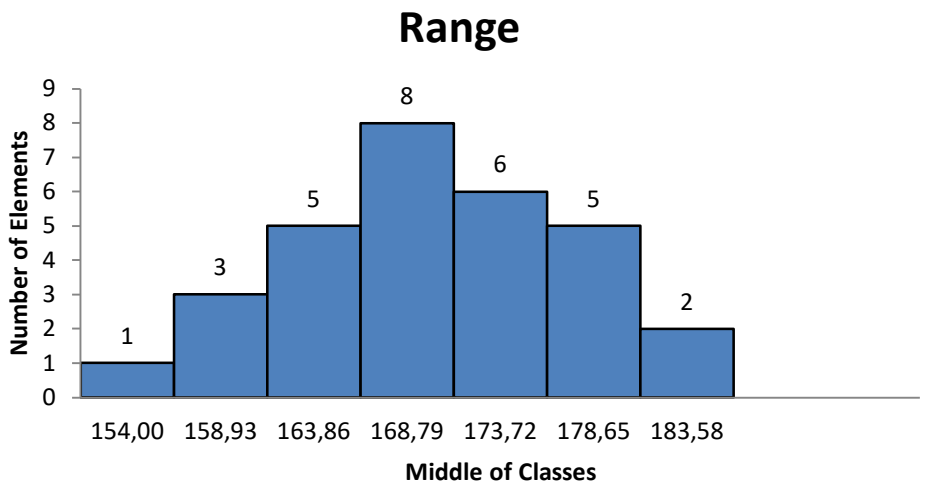
3.9.7 LGF_Histogram_UDInt

Short description

The histogram shows the frequency distribution of a sample by class. A class describes a value interval in which the individual frequencies are added together. After specifying the number of classes, the class width and the respective class center are calculated. The number of classes is limited to 15.

The distribution is represented as a rectangle around the class mean with the class width and the cumulated frequency as height.

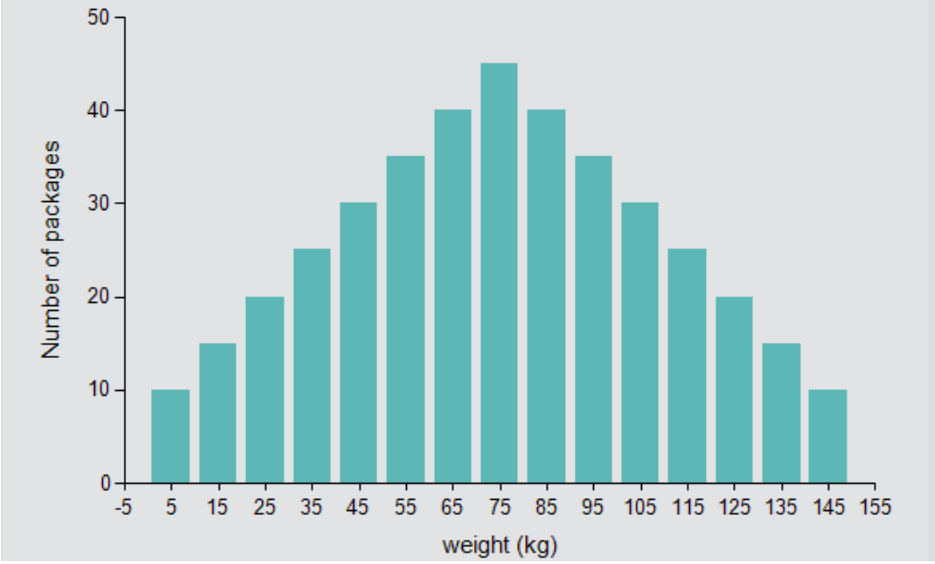
Figure 3-160: Distribution



WinCC-Control

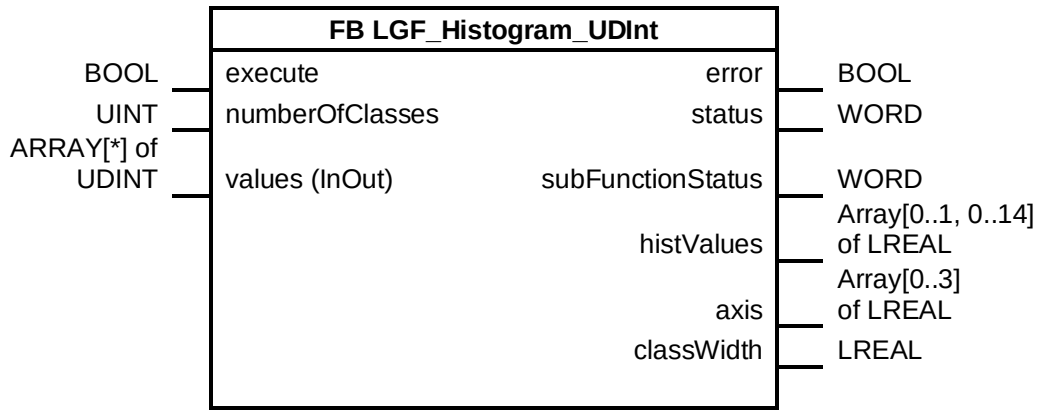
To visualize the histogram, Siemens Industry Online Support offers you a .Net Control that you can use in conjunction with WinCC Runtime Professional. You can find the download under the entry ID: 81662739.

Figure 3-161: .Net-Control



Block

Figure 3-162: LGF_Histogram_UDInt



Input parameters

Table 3-319: Input parameters

Parameters	Data type	Description
execute	BOOL	Activation of the calculation with each positive edge.
numberOfClasses	UINT	Number of desired classes.

Input/output parameters (InOut)

Table 3-320: Input/output parameters (InOut)

Parameters	Data type	Description
values	ARRAY[*] of UDINT	Array that should be used for calculation.

Output parameters

Table 3-321: Output parameters

Parameters	Data type	Description
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).
subFunctionStatus	WORD	Status or return value of the called FCs and system blocks.
histValues	ARRAY [0..1, 0..14] of LREAL	Outputs the calculated values in a two-dimensional array. • HistValues[0,0..14] displays the relative frequency of the individual classes. • HistValues[1,0..14] displays the class centers. • If fewer than 15 classes are desired, the array elements that are not required are output with 0.
axis	Array[0..3] of LREAL	Specifies the axis values: • Lower X axis value • Upper X axis value • Lower Y axis value • Upper Y axis value
classWidth	LREAL	Returns the calculated class width.

Status and error displays

Table 3-322: Status/error codes

status	Meaning	Remedy / notes
16#0000	No error	-
16 #7000	Block is not being edited	-
16#7001	First FB call.	-
16#8600	Error in command “LGF_ShellSort_UDInt”.	Check the error code in “subFunctionStatus”. Information on this block can be found in Chapter 3.5.4.
16#9101	Incorrect number of classes	Give the parameter “NumberOfClasses” a valid value (1 to 15).

Principle of operation

The block sorts the passed data and calculates the general class width using the passed class count and data range. The block then counts the values that lie within a class. In order to draw a histogram, the block also calculates the necessary X and Y coordinates.

The elements of the passed array “values” are sorted in ascending order by the block. The block “LGF_ShellSort_UDInt” is used for sorting (see 3.5.4).

The number of classes can be specified using the following rule of thumb:

$$Number\ of\ Classes = \sqrt{number\ of\ elements}$$

e.g. 100 values -> $Number\ of\ Classes = \sqrt{100} = 10$

Formulas

The block uses the following formula to calculate the class width:

$$classWidthh = \frac{max\ - min}{Number\ of\ classes}$$

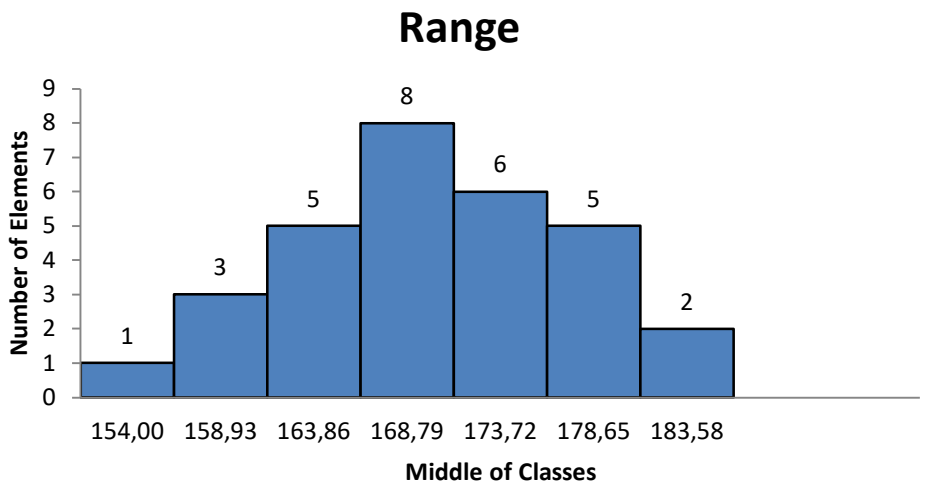
3.9.8 LGF_Histogram_LReal

Short description

The histogram shows the frequency distribution of a sample by class. A class describes a value interval in which the individual frequencies are added together. After specifying the number of classes, the class width and the respective class center are calculated. The number of classes is limited to 15.

The distribution is represented as a rectangle around the class mean with the class width and the cumulated frequency as height.

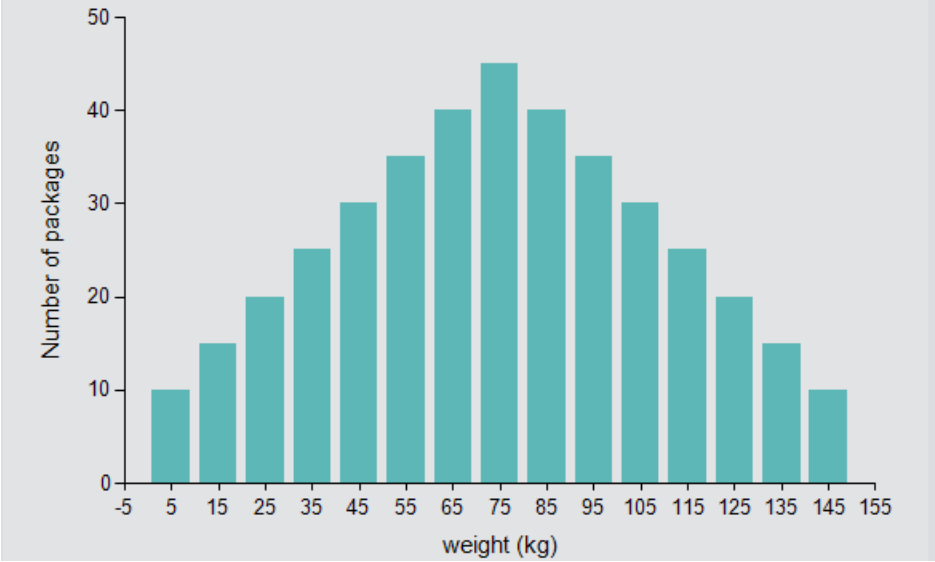
Figure 3-163: Distribution



WinCC-Control

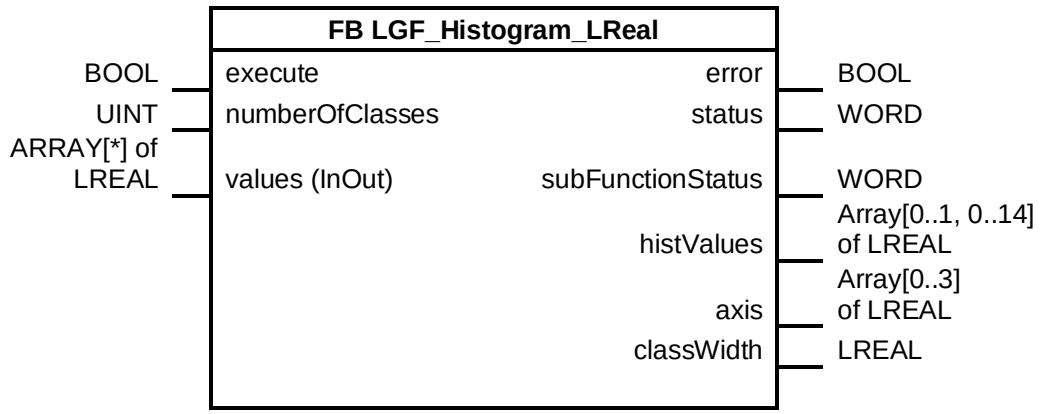
To visualize the histogram, Siemens Industry Online Support offers you a .Net Control that you can use in conjunction with WinCC Runtime Professional. You can find the download under the entry ID: 81662739.

Figure 3-164: .Net-Control



Block

Figure 3-165: LGF_Histogram_LReal



Input parameters

Table 3-323: Input parameters

Parameters	Data type	Description
execute	BOOL	Activation of the calculation with each positive edge.
numberOfClasses	UINT	Number of desired classes.

Input/output parameters (InOut)

Table 3-324: Input/output parameters (InOut)

Parameters	Data type	Description
values	ARRAY[*] of LREAL	Array that should be used for calculation.

Output parameters

Table 3-325: Output parameters

Parameters	Data type	Description
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).
subFunctionStatus	WORD	Status or return value of the called FCs and system blocks.
histValues	ARRAY [0..1, 0..14] of LREAL	Outputs the calculated values in a two-dimensional array. • HistValues[0,0..14] displays the relative frequency of the individual classes. • HistValues[1,0..14] displays the class centers. • If fewer than 15 classes are desired, the array elements that are not required are output with 0.
axis	Array[0..3] of LREAL	Specifies the axis values: • Lower X axis value • Upper X axis value • Lower Y axis value • Upper Y axis value
classWidth	LREAL	Returns the calculated class width.

Status and error displays

Table 3-326: Status/error codes

status	Meaning	Remedy / notes
16#0000	No error	-
16 #7000	Block is not being edited	-
16#7001	First FB call.	-
16#8600	Error in command “LGF_ShellSort_LReal”.	Check the error code in “subFunctionStatus”. Information concerning this block is provided in Chapter 3.5.5.
16#9101	Incorrect number of classes	Give the parameter “NumberOfClasses” a valid value (1 to 15).

Principle of operation

The block sorts the passed data and calculates the general class width using the passed class count and data range. The block then counts the values that lie within a class. In order to draw a histogram, the block also calculates the necessary X and Y coordinates.

The elements of the passed array “values” are sorted in ascending order by the block. The block “LGF_ShellSort_LReal” is used for sorting (see 3.5.5).

The number of classes can be specified using the following rule of thumb:

$$Number\ of\ Classes = \sqrt{number\ of\ elements}$$

e.g. 100 values -> $Number\ of\ Classes = \sqrt{100} = 10$

Formulas

The block uses the following formula to calculate the class width:

$$classWidthh = \frac{max\ - min}{Number\ of\ classes}$$

3.9.9 LGF_SimpleSmoothingFB/LGF_SimpleSmoothingFC

Short description

The simplest form of smoothing a sequence of measured values is to calculate the linear mean value by three points.

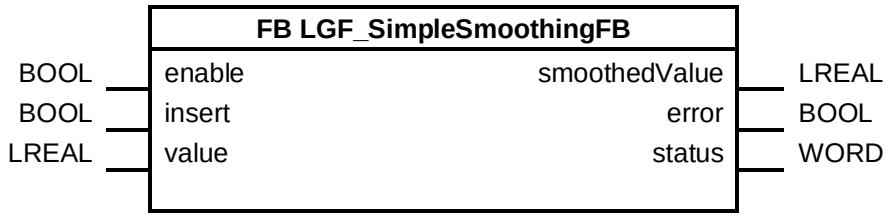
The block is implemented as a function and as a function block.

Table 3-327: Comparison FC-FB

Function (FC)	Function block (FB)
The function calculates the linear mean value acyclically. The function reads an array that is smoothed. N-2 smoothed measured values can be calculated from N measured values. Therefore, the output array in the index (0) and index (N) contains the value 0.	The function block calculates the linear mean value cyclically. The function block reads-in a value with each positive edge on the “insert” input. As soon as three values have been read in, the block calculates a smoothed value and outputs it.

Function block (FB)

Figure 3-166: LGF_SimpleSmoothingFB



Input parameters

Table 3-328: Input parameters

Parameters	Data type	Description
enable	BOOL	Activates the block. As long as enable is “TRUE”, the block can accept values on the parameter “value”.
insert	BOOL	Accepts the value at the input “value” at positive edge and outputs a “smoothedValue” if three values have been read in.
value	LReal	Value that is to be included in the smoothing.

Output parameters

Table 3-329: Output parameters

Parameters	Data type	Description
smoothedValue	LREAL	The smoothed values.
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).

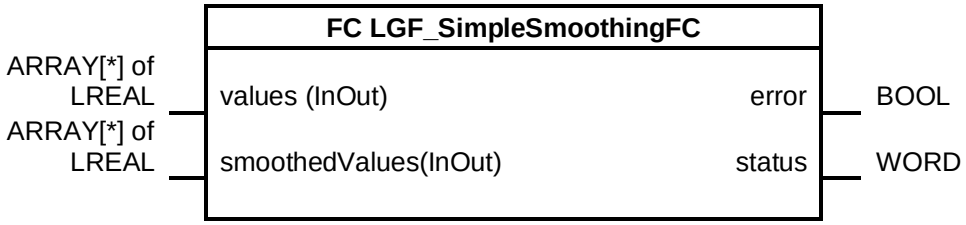
Status and error displays

Table 3-330: Status/error codes

status	Meaning	Remedy / notes
16#0000	No error	Processing was completed successfully
16 #7000	Block is not being edited	The block waits for activation through the parameter “enable”.
16#7001	First FB call.	-
16#7002	Processing is active.	Subsequent call of the FB
16#7010	Too few values	The block requires three values to calculate a smoothed value. Transfer additional values with a positive edge on the “insert” input.

Function (FC)

Figure 3-167: LGF_SimpleSmoothingFC



Input/output parameters (InOut)

Table 3-331: Input/output parameters (InOut)

Parameters	Data type	Description
values	ARRAY[*] of LREAL	Values that are to be included in the smoothing.
smoothedValues	ARRAY[*] of LREAL	The smoothed values.

Output parameters

Table 3-332: Output parameters

Parameters	Data type	Description
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).

Status and error displays

Table 3-333: Status/error codes

status	Meaning	Remedy / notes
16#0000	No error	Processing was completed successfully
16#8400	Too few values	The block requires three values to calculate a smoothed value. Increase the size of the array at the input parameter “values”. Adapt the array on the output parameter “smoothedValues” to the new size.
16#8401	Unequal array sizes	The arrays “values” and “smoothedValues” must have the same size.

Principle of operation

The block calculates the smoothed values using the following formula:

$$\overline{y(n)} = \frac{y(n-1) + y(n) + y(n+1)}{3}$$

The calculated value is output or the calculated values are output at output “smoothedValue”.

Based on this formula, the FC cannot calculate values for the elements 0 and N.

3.9.10 LGF_SmoothByPolynomFB/LGF_SmoothByPolynomFC

Short description

For smoothing, a 3rd degree polynomial is placed through five value points. The error squares of the distances between polynomial and real value are minimized. The smoothed values can be determined from the polynomial parameters obtained in this way.

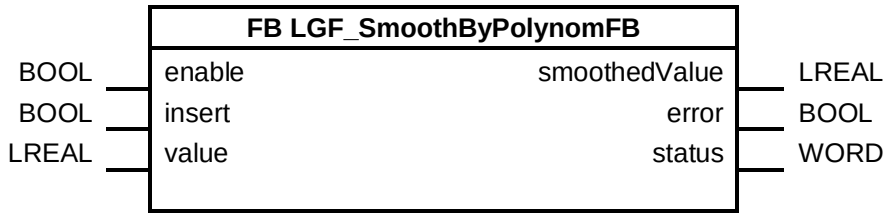
The block is implemented as a function and as a function block.

Table 3-334: Comparison FC-FB

Function (FC)	Function block (FB)
The function calculates the smoothed values acyclically. The function reads an array that is smoothed. N-4 smoothed measured values can be calculated from N measured values. The output array contains the value 0 in the index (0,1,N-1,N). However, replacement values can be calculated.	The function block calculates the smoothed values cyclically. The function block reads-in a value with each positive edge on the “insert” input. As soon as five values have been read in, the block calculates a smoothed value and outputs it.

Function block (FB)

Figure 3-168: LGF_SmoothByPolynomFB



Input parameters

Table 3-335: Input parameters

Parameters	Data type	Description
enable	BOOL	Activates the block. As long as enable is “TRUE”, the block can accept values on the parameter “value”.
insert	BOOL	Accepts the value at input “value” and outputs a “smoothedValue” if five values have been read in.
value	LREAL	Value that is to be included in the smoothing.

Output parameters

Table 3-336: Output parameters

Parameters	Data type	Description
smoothedValue	LREAL	Smoothed value.
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).

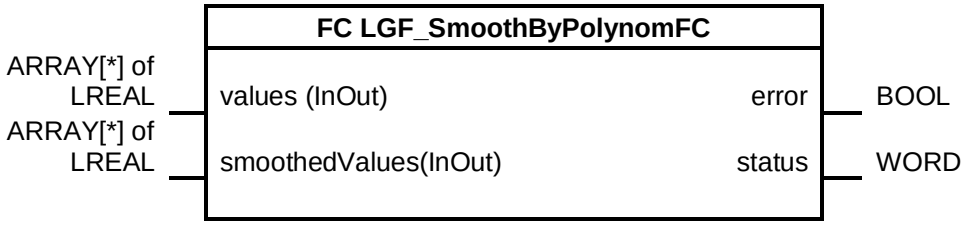
Status and error displays

Table 3-337: Status/error codes

status	Meaning	Remedy / notes
16#0000	No error	Processing was completed successfully
16 #7000	Block is not being edited	The block waits for activation through the parameter “enable”.
16#7001	First FB call.	-
16#7002	Processing is active.	Subsequent call of the FB
16#7010	Too few values	The block requires five values to calculate a smoothed value. Transfer additional values with a positive edge on the “insert” input.

Function (FC)

Figure 3-169: LGF_SmoothByPolynomFC



Input/output parameters (InOut)

Table 3-338: Input/output parameters (InOut)

Parameters	Data type	Description
values	ARRAY[*] of LREAL	Values that are to be included in the smoothing.
smoothedValues	ARRAY[*] of LREAL	The smoothed values.

Output parameters

Table 3-339: Output parameters

Parameters	Data type	Description
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).

Status and error displays

Table 3-340: Status/error codes

status	Meaning	Remedy / notes
16#0000	No error	Processing was completed successfully
16#8400	Unequal array sizes	The arrays “values” and “smoothedValues” must have the same size.
16#8401	Too few values	The block requires five values to calculate a smoothed value. Increase the size of the array at the input parameter “values”. Adapt the array on the output parameter “smoothedValues” to the new size.

Principle of operation

The 3rd degree compensation polynomial is calculated as follows:

$$\overline{y(n)} = \frac{1}{35} * (-3 * y(n-2) + 12 * y(n-1) + 17 * y(n) + 12 * y(n+1) - 3 * y(n+2))$$

N-4 smoothed measured values can thus be calculated from the N measured values. The output array contains the value 0 in the index (0.1, N-1, N).

These “missing” values are calculated with the following formalisms:

$$\overline{\overline{y(n-2)}} = \frac{1}{70} * \left(69 * y(n-2) + 4 * y(n-1) - 6 * y(n) + 4 * y(n+1) - y(n+2)\right)$$

$$\overline{\overline{y(n-1)}} = \frac{2}{70} * (2 * y(n-2) + 27 * y(n-1) + 12 * y(n) - 8 * y(n+1) + 2 * y(n+2))$$

$$\overline{\overline{y(n+1)}} = \frac{2}{70} * (2 * y(n-2) - 8 * y(n-1) + 12 * y(n) + 27 * y(n+1) + 2 * y(n+2))$$

$$\overline{\overline{y(n+2)}} = \frac{1}{70} * (-y(n-2) + 4 * y(n-1) - 6 * y(n) + 4 * y(n+1) + 69 * y(n+2))$$

3.9.11 LGF_DifferenceQuotientFB/LGF_DifferenceQuotientFC

Short description

This block numerically differentiates a signal sampled equidistantly in time. For example, the velocity can be calculated from a measured locus curve, or the acceleration can be calculated from the measured velocity. In order to minimize the effects of a scattering measurement signal, this algorithm uses a compensating polynomial.

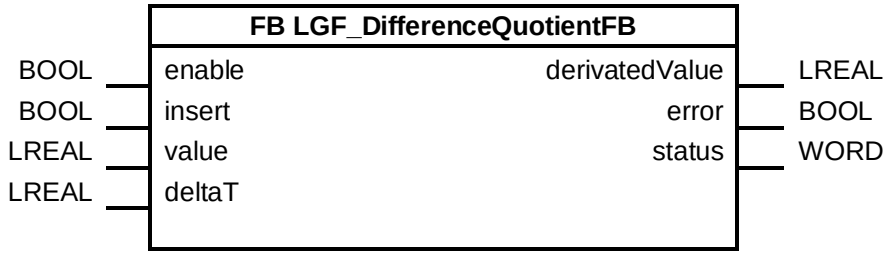
The block is implemented as a function and as a function block.

Table 3-341: Comparison FC-FB

Function (FC)	Function block (FB)
The function calculates the differentiated values acyclically. The function reads an array that is differentiated. N-4 smoothed measured values can be calculated from N measured values. The output array contains the value 0 in the index (0,1,N-1,N). However, replacement values can be calculated.	The function block calculates the differentiated values cyclically. The function block reads-in a value with each positive edge on the “insert” input. As soon as five values have been read in, the block calculates a differentiated value and outputs it.

Function block (FB)

Figure 3-170: LGF_DifferenceQuotientFB



Input parameters

Table 3-342: Input parameters

Parameters	Data type	Description
enable	BOOL	Activates the block. As long as enable is “TRUE”, the block can accept values on the parameter “value”.
insert	BOOL	Accepts the value at the “value” input and outputs a “derivatedValue” if five values have been read in.
value	LREAL	Value that must be included in the differentiation.
deltaT	LREAL	equidistant distance between two measured values. (e.g. 1s)

Output parameters

Table 3-343: Output parameters

Parameters	Data type	Description
derivatedValue	LREAL	Differentiated value.
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).

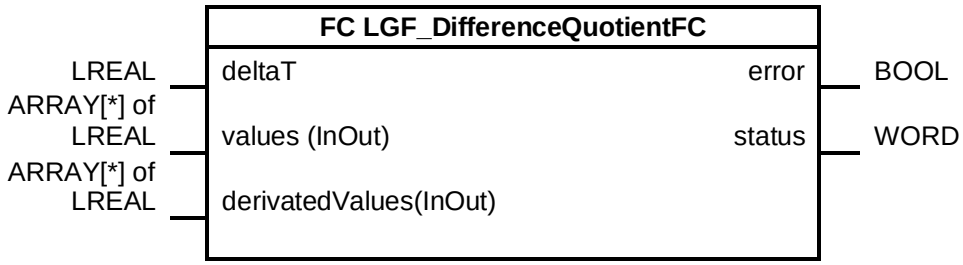
Status and error displays

Table 3-344: Status/error codes

status	Meaning	Remedy / notes
16#0000	No error	Processing was completed successfully
16 #7000	Block is not being edited	The block waits for activation through the parameter “enable”.
16#7001	First FB call.	-
16#7002	Processing is active.	Subsequent call of the FB
16#7010	Too few values	The block requires five values to calculate a differentiated value. Transfer additional values with a positive edge on the “insert” input.
10#8200	Error: “deltaT” = 0	“deltaT” can’t be 0.

Function (FC)

Figure 3-171: LGF_DifferenceQuotientFC



Input parameters

Table 3-345: Input parameters

Parameters	Data type	Description
deltaT	LREAL	Equidistant distance between two measured values. (e.g. 1s)

Input/output parameters (InOut)

Table 3-346: Input/output parameters (InOut)

Parameters	Data type	Description
values	ARRAY[*] of LREAL	Values that will be included in the differentiation.
derivatedValues	ARRAY[*] of LREAL	The differentiated value range.

Output parameters

Table 3-347: Output parameters

Parameters	Data type	Description
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).

Status and error displays

Table 3-348: Status/error codes

status	Meaning	Remedy / notes
16#0000	No error	Processing was completed successfully
10#8200	Error: “deltaT” = 0	“deltaT” can’t be 0.
16#8400	Unequal array sizes	The arrays “values” and “derivatedValues” must have the same size.
16#8401	Too few values	The block requires five values to calculate a differentiated value. Increase the size of the array at the input parameter “values”.

Principle of operation

To calculate the difference quotient of a scattering signal, a third degree compensation polynomial is first placed through the measured values. This polynomial is then differentiated. With this method, even a distorted input signal can be sensibly differentiated.

The difference quotient is calculated with the following formula:

$$\dot{\bar{y}}(n) = \frac{1}{12deltaT}(y(n-2) - 8y(n-1) + 8y(n+1) - y(n+2))$$

deltaT: equidistant distance between two measured values (e.g. 1s).

The function (FC) can calculate N-4 differentiated and smoothed measured values from N measured values. The output array would be assigned with 0 in the index (0,1,N-1,N). However, the following formalisms can be used to calculate substitute values:

$$\dot{\bar{y}}(n-2) = \frac{1}{84deltaT}(-125y(n-2) + 136y(n-1) + 48y(n) - 88y(n+1) + 29y(n+2))$$

$$\dot{\bar{y}}(n-1) = \frac{1}{84deltaT}(-38y(n-2) - 2y(n-1) + 24y(n) + 26y(n+1) - 10y(n+2))$$

$$\dot{\bar{y}}(n+1) = \frac{1}{84\mathrm{d}deltaT}(10y(n-2) - 26y(n-1) - 24y(n) + 2y(n+1) + 38y(n+2))$$

$$\dot{\bar{y}}(n+2) = \frac{1}{84deltaT}(-29y(n-2) + 88y(n-1) - 48y(n) - 136y(n+1) + 125y(n+2))$$

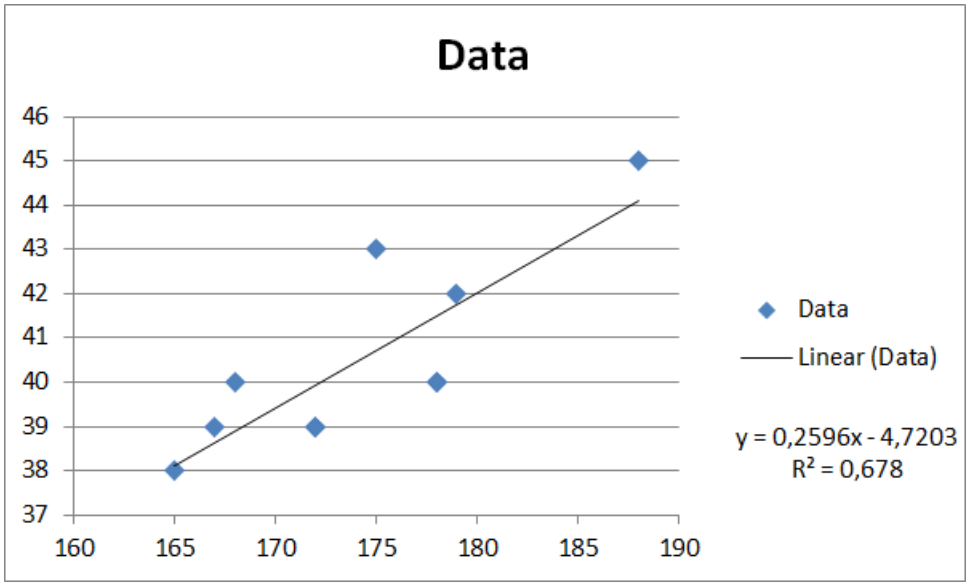
deltaT: equidistant distance between two measured values (e.g. 1s).

3.9.12 LGF_RegressionLine

Short description

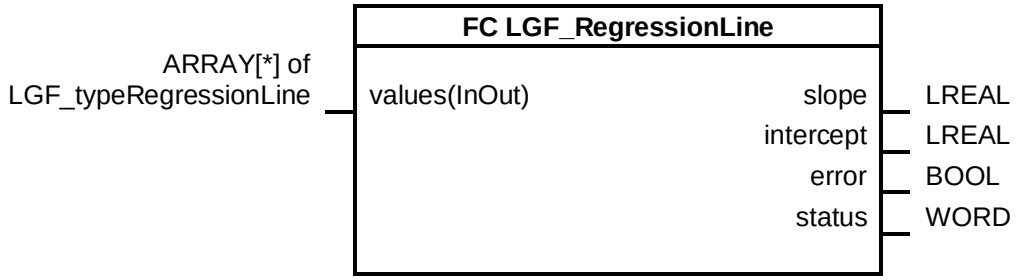
The simplest case of a regression is the regression line. This means that the assumed relationship between the input and output signal is a linear straight line.

Figure 3-172: Regression line



Block

Figure 3-173: LGF_RegressionLine



Input/output parameters (InOut)

Table 3-349: Input/output parameters (InOut)

Parameters	Data type	Description
values	ARRAY[*] of LGF_typeRegressionLine	The data points are transferred with their X and Y values. The data type “LGF_typeRegressionLine” has the following structure: • x (Real) • y (Real)

Output parameters

Table 3-350: Output parameters

Parameters	Data type	Description
slope	LREAL	Gradient of straight line
intercept	LREAL	The intersection with the Y axis
error	BOOL	FALSE: No error TRUE: An error occurred during the execution of the FB.
status	WORD	16#0000-16#7FFF: Status of the FB, 16#8000-16#FFFF: Error identification (see following Table).

Status and error displays

Table 3-351: Status/error codes

status	Meaning	Remedy / notes
16#0000	No error	Processing was completed successfully
16#8200	Too few values	The block requires at least two pairs of values to calculate a regression line. Increase the size of the array at the input parameter “values” in the second dimension.

Principle of operation

The block calculates the regression line with the following line equation:

$f(x) = a + b * x$

b: Gradient of straight line

a: Intersection with y-axis

The gradient b is calculated using the following equation:

$$b = \frac{n * \sum_1^N (x(n) * y(n)) - (\sum_1^N x(n) * \sum_1^N y(n))}{n * \sum_1^N x^2(n) - (\sum_1^N x(n) * \sum_1^N x(n))}$$

The intersection with the Y axis is calculated using the following equation:

$$a = \frac{\sum_1^N y(n)}{N} - b * \frac{\sum_1^N x(n)}{N}$$

3.10 Legacy

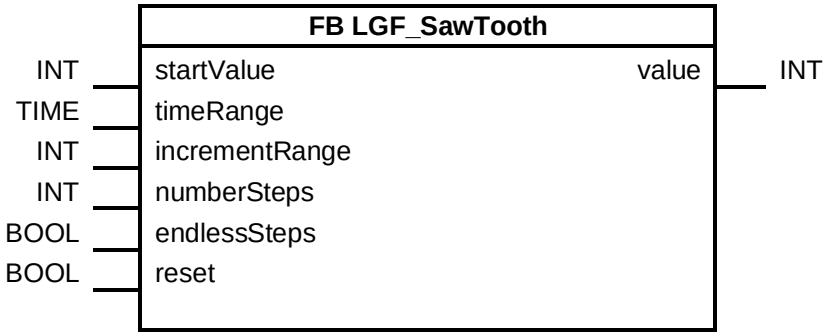
3.10.1 LGF_SawTooth

Short description

This block generates a sawtooth-shaped signal profile. Each sawtooth consists of a defined number of steps (increments).

Block

Figure 3-174: FB LGF_SawTooth



Input parameters

Table 3-352: Input parameters

Parameters	Data type	Description
startValue	INT	Start value at which the signal begins.
timeRange	TIME	Time after which the output parameter “value” is incremented
incrementRange	INT	Size of the jump from one increment to the next.
numberSteps	INT	Number of increments per sawtooth. (In the case of an endless sawtooth signal, this information is not necessary).
endlessSteps	BOOL	Specifies whether an endless sawtooth signal will be generated.
reset	BOOL	Sawtooth starts again at the start value, “startValue”.

Note Please note that changes at the input parameters only become effective with “reset”.

Output parameters

Table 3-353: Output parameters

Parameters	Data type	Description
value	INT	Current value of the sawtooth signal.

Principle of operation

The block calculates the values for a sawtooth-shaped signal profile, which is output to the output parameter “value”. The signal begins with the start value “startValue” and is added with the value “increment” after each elapse of the time interval “timeRange”. The value can also be negative.

If the variable “endlessSteps” is set to “FALSE”, the number of add operations is counted. If this exceeds the value “numberSteps”, the output parameter “value” is set back to the start value. A new sawtooth begins.

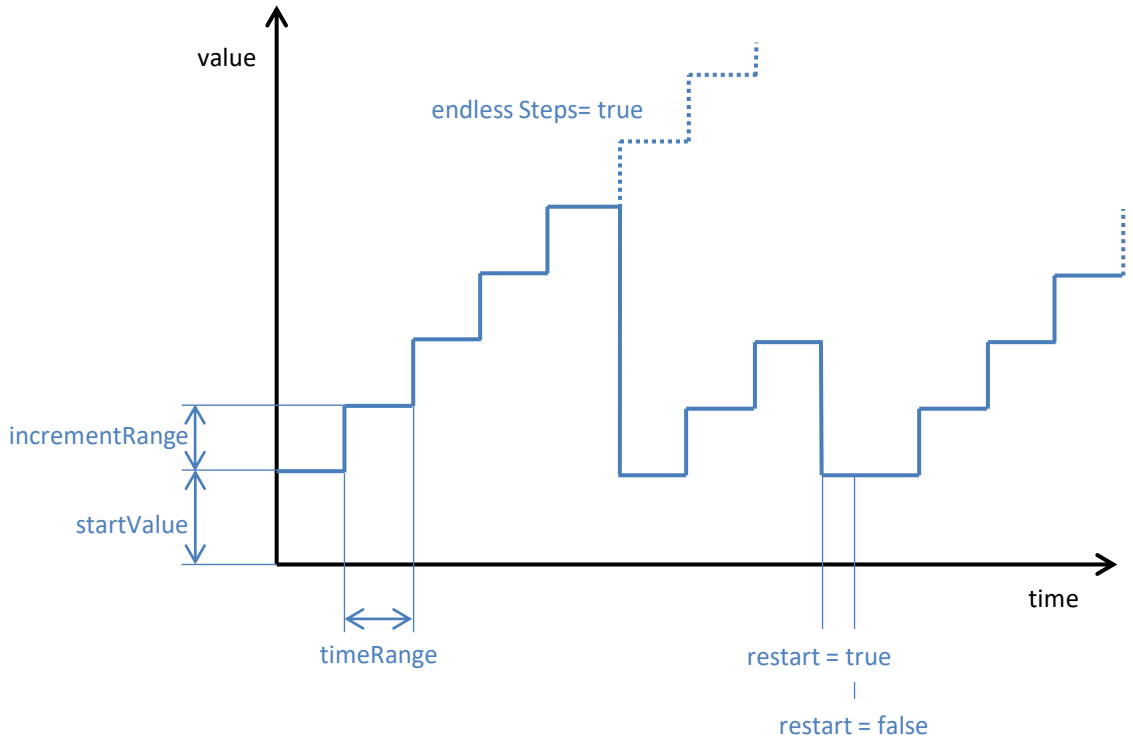
If the variable “endlessSteps” is set to “TRUE”, the value “increment” is added without interruption, starting once at “startValue”. If the maximum positive INT value range (32767) of the output parameter “value” is exceeded, “value” changes to the maximum negative INT value range (-32768) and will continue to be added up.

Note

The duration of a sawtooth at “endlessSteps” on “FALSE” is calculated as follows:

Duration = “timeRange * (“numberSteps” + 1)

Figure 3-175: Signal profile of the output “value”



4 Appendix

4.1 Service and support

Industry Online Support

Do you have any questions or need assistance?

Siemens Industry Online Support offers round the clock access to our entire service and support know-how and portfolio.

The Industry Online Support is the central address for information about our products, solutions and services.

Product information, manuals, downloads, FAQs, application examples and videos – all information is accessible with just a few mouse clicks:
https://support.industry.siemens.com

Technical Support

The Technical Support of Siemens Industry provides you fast and competent support regarding all technical queries with numerous tailor-made offers – ranging from basic support to individual support contracts. Please send queries to Technical Support via Web form:
www.siemens.com/industry/supportrequest

SITRAIN – Training for Industry

We support you with our globally available training courses for industry with practical experience, innovative learning methods and a concept that’s tailored to the customer’s specific needs.

For more information on our offered trainings and courses, as well as their locations and dates, refer to our web page:
www.siemens.com/sitrain

Service offer

Our range of services includes the following:

- Plant data services
- Spare parts services
- Repair services
- On-site and maintenance services
- Retrofitting and modernization services
- Service programs and contracts

You can find detailed information on our range of services in the service catalog web page:
support.industry.siemens.com/cs/sc

Industry Online Support app

You will receive optimum support wherever you are with the "Siemens Industry Online Support" app. The app is available for Apple iOS, Android and Windows Phone:
support.industry.siemens.com/cs/ww/en/sc/2067

4.2 Links and literature

Table 4-1

No.	Subject
\1\	Siemens Industry Online Support https://support.industry.siemens.com
\2\	Link to the article page of the Application Example https://support.industry.siemens.com/cs/ww/en/view/109479728
\3\	Programming Guide and Programming Style Guide https://support.industry.siemens.com/cs/ww/en/view/81318674
\4\	Library with PLC data types (LPD) for STEP 7 (TIA Portal) and SIMATIC S7-1200 / S7-1500 https://support.industry.siemens.com/cs/ww/en/view/109482396
\5\	Guideline on Library Handling in Tia Portal https://support.industry.siemens.com/cs/ww/en/view/109747503
\6\	Libraries in the TIA Portal https://support.industry.siemens.com/cs/ww/en/view/109738702

4.3 Change documentation

4.3.1 Versioning of the library

The library and library elements are maintained in accordance with the Table below:

Table 4-2: Definition of the version

P	1.	2.	3
	Non-compatible change	Compatible change	Error correction
	• Reduction of interfaces • Changing the interfaces • Incompatible extension of functionality	• Extension of the interfaces • Compatible extension of functionality	• Bug fix • Upgrade to new TIA Portal version

Versioning example

Table 4-3: Example for changing the version

Library	FB1	FB2	FC1	FC2	Comment
1.0.0	1.0.0	1.0.0	1.0.0	-	released
1.0.1	1.0.1	1.0.0	1.0.0	-	Troubleshooting of FB1
1.0.2	1.0.1	1.0.1	1.0.0	-	Optimization of FB2
1.1.0	1.1.0	1.0.1	1.0.0	-	Extension to FB1
1.2.0	1.2.0	1.0.1	1.0.0	-	Extension to FB1
2.0.0	2.0.0	1.0.1	2.0.0	-	new functionality on FB1 and FC1
2.0.1	2.0.0	1.0.2	2.0.0	-	Troubleshooting FB2
3.0.0	2.0.0	1.0.2	2.0.0	1.0.0	New function FC2
3.0.1	2.0.1	1.0.3	2.0.1	1.0.1	Upgrade to new TIA Portal version
3.0.2	2.0.2	1.0.4	2.0.2	1.0.0	New functions, bug fixes

4.3.2 Change log

Table 4-4 : Change log

Version	Date	Change
V1.0.0	09/2015	First version
V1.0.1	10/2015	V1.0.1 • T_ADD command was replaced by “+”.
V1.0.2	10/2015	LGF_BinaryToGray V1.0.1 • Name changed LGF_GrayToBinary V1.0.1 • Name changed
V1.0.3	11/2015	LGF_CompareVariant V1.0.1 • Error correction
V1.0.4	11/2015	LGF_SawTooth V1.0.1 • Error correction

Version	Date	Change
V1.0.5	11/2015	LGF_Astro V1.0.2 • Error correction LGF_AverageAndDeviation V1.0.1 • Error correction LGF_TimerSwitch V1.0.1 • Error correction LGF_FIFO V1.0.1 • Error correction
V2.0	07/2016	New: Chapter on “Library Resources” FB LGF_PulseRelay V1.0.0 FB LGF_SetTime V1.0.0 FB LGF_FloatingAverage V1.0.0 FC LGF_DTLtoString V1.0.0 FC LGF_StringToDTL V1.0.0 FB LGF_LimRateOfChangeBasic V1.0.0 FB LGF_LimRateOfChangeAdvanced V1.0.0 Revised: LGF_Astro V1.1.1 • Output systemTime and localTime added FB LGF_TimerSwitch V1.1.0 • Two new modes: Weekday, weekend FB LGF_ShallSort… V1.1.0 • New mode: Descending Sort FB LGF_Frequency V1.1.0 • New function: Pulse-pause ratio adjustable FB LGF_Pulses V1.1.0 • Calls new LGF_Frequency V1.1.0.
V2.0.1	01/2017	Revised: LGF_Astro V1.1.2 • Error correction for sunrise and sunset calculation.
V2.0.2	01/2017	Revised: All blocks: Upgrade TIA V14
V3.0.0	03/2017	New: FC LGF_CalendarDayWeekV1.0.0 FC LGF_CompareReal V1.0.0 FC LGF_RandomBasic V1.0.0 FC LGF_HighLowLimit V1.0.0 FC LGF_BitsToWord V1.0.0 FC LGF_WordToBits V1.0.0 FC LGF_ScaleLinear V1.0.0 FC LGF_StringToTaddr V1.0.0 FC LGF_TaddrToString V1.0.0 FB LGF_Ramp V1.0.0 FB LGF_NonLin V1.0.0 Revised: Supplement in chapter 4.3.1 Versioning of the library FB LGF_PulseRelay V1.0.2 • Commentary correction

Version	Date	Change
		FB LGF_Astro V1.1.4 • Code optimization • FB LGF_SetTime V1.0.2 • Correction: FB number: automatic FB LGF_FloatingAverage V1.1.0 • Code optimization • New input parameter “windowSize” FC LGF_MatrixAddition V2.0.0 • Code optimization • Input parameters changed to ARRAY* FC LGF_MatrixInverse V2.0.0 • Code optimization • Input parameters changed to ARRAY* FC LGF_MatrixMultiplication V2.0.0 • Code optimization • Input parameters changed to ARRAY* FC LGF_MatrixSubtraction V2.0.0 • Code optimization • Input parameters changed to ARRAY* FC LGF_MatrixTranspose V2.0.0 • Code optimization • Input parameters changed to ARRAY* FB LGF_Pulses V1.2.0 • Code optimization: no longer calling of LGF_Frequency •
V3.0.1	05/2017	Revised: FB LGF_Ramp V1.0.1 • Commentary correction
V4.0.0	09/2018	New: FC LGF_GermanHoliday V1.0.0 FC LGF_Factorial V1.0.0 FC LGF_Distance V1.0.0 FB LGF_LIFO V1.0.0 FC LGF_CRC8 V1.0.0 FC LGF_CRC8For1Byte V1.0.0 FC LGF_CRC16 V1.0.0 FC LGF_CRC32 V1.0.0 FC LGF_IntToString V1.0.0 FC LGF_TimeToString V1.0.0 FB LGF_SawToothCI V1.0.0 FB LGF_TriangleCI V1.0.0 FB LGF_RectangleCI V1.0.0 FB LGF_SinusCI V1.0.0 FB LGF_CosinusCI V1.0.0 Revised: Addition of Chapter 2.2 Simulability with SIMATIC S7-PLCSIM Advanced FB LGF_Astro V1.1.5 • Code optimization FB LGF_TimerSwitch V1.1.2 • Code optimization

Version	Date	Change
V4.0.1	-	Only for TIA Portal V15
V4.0.2	10/2018	FB LGF_TimerSwitch V1.1.3 Connection to type restored.
V4.0.3	02/2019	All blocks: Upgrade TIA V15.1 Upd1 User-defined documentation (user help) for all blocks. New: FB LGF_Boxplot FB LGF_Histogram FB LGF_SimpleSmoothingFB FC LGF_SimpleSmoothingFC FB LGF_SmoothByPolynomFB FC LGF_SmoothByPolynomFC FB LGF_DifferenceQuotiontFB FC LGF_DifferenceQuotiontFC FB LGF_RegressionLine FC LGF_UnixTimeToDTL FC LGF_DTLToUnixTime FB LGF_CountFalInDWordFB (FC LGF_CountFalInDWord → 99_Legacy) FB LGF_CountRisInDWordFB (FC LGF_CountRisInDWord → 99_Legacy) Revised: FC LGF_XRoot V1.0.4 • Code optimization FC LGF_Distance V1.1.0 • Code optimization FB LGF_FIFO V2.0.0 • “done” output deleted FB LGF_LIFO V2.0.0 • “done” output deleted FB LGF_ShellSortInt V2.0.0 • “done” output deleted FB LGF_ShellSortUint V2.0.0 • “done” output deleted FB LGF_ShellSortReal V2.0.0 • “done” output deleted FC LGF_ScaleLinear V2.0.0 • Data type changed (Variant → LREAL) FC LGF_AverageAndDeviation V2.0.0 • Data type changed (Variant → ARRAY[*] of LREAL) FB LGF_SetTime V1.0.5 • Rising edge at input REQ of the SET_TIMEOUT command
V4.0.4	06/2019	Bugfixing FB LGF_DifferenceQuotientFB V1.0.1 FB LGF_PulseRelay V1.0.5
V4.0.5	09/2019	FB LGF_SetTime V1.0.6

Version	Date	Change
		• Doc. revised and new time zones added
V4.0.6	11/2019	Bugfixing • FB LGF_SmoothByPolynomFB
V5.0.0	04/2020	All blocks: Upgrade TIA V16 **Renamed**: FB LGF_Astro → FB LGF_AstroClock LGF_CountFalInDWordFB -> LGF_CountFalInDWord LGF_CountRisInDWordFB -> LGF_CountRisInDWord LGF_GermanHoliday -> LGF_IsGermanHoliday LGF_CompareReal -> LGF_CompareLReal LGF_Compare -> LGF_MatrixCompare LGF_MinMaxHistory -> LGF_StoreMinMax LGF_RandomBasic -> LGF_Random_Real LGF_RandomReal -> LGF_RandomRange_Real LGF_XRoot -> LGF_NthRoot LGF_HighLowLimit -> LGF_IsValueInRange LGF_Factorial -> LGF_GetFactorial LGF_Distance -> LGF_CalcDistance_2D LGF_CRC8 -> LGF_CalcCRC8 LGF_CRC8For1Byte -> LGF_CalcCRC8 For1Byte LGF_CRC16 -> LGF_CalcCRC16 LGF_CRC32 -> LGF_CalcCRC32 LGF_BitsToWord -> LGF_MergeBitsToWord LGF_WordToBits -> LGF_SplitWordToBits LGF_DTLtoString -> LGF_DTLtoString_ISO LGF_StringToDTL -> LGF_StringToDTL_ISO LGF_TemperatureConvert -> LGF_ConvertTemperature LGF_LimRateOfChangeBasic -> LGF_LimRateOfChangeCI LGF_LimRateOfChangeAdvanced -> LGF_LimRateOfChangeAdvancedCI LGF_Ramp -> LGF_RampCI LGF_NonLin -> LGF_NonLinearInterpolation LGF_Boxplot -> LGF_Boxplot_DInt LGF_Histogram -> LGF_Histogram_DInt **New**: LGF_BitSet LGF_BitSetTo LGF_BitReset LGF_BitTest LGF_BitToggle LGF_BitCount LGF_CalendarDayWeek_ISO LGF_CalendarDayWeek_US LGF_CompareLRealByPrecision LGF_MatrixScalarMultiplication LGF_Random_DInt LGF_Random_UDInt LGF_RandomRange_DInt LGF_RandomRange_UDInt

Version	Date	Change
		LGF_SearchMinMax_LReal LGF_SearchMinMax_Dint LGF_SearchMinMax_UDInt LGF_IsValueInTolerance LGF_IsValueInLimits LGF_CalcDistance_3D LGF_CalcCRC8Advanced LGF_CalcCRC16Advanced LGF_CalcCRC32Advanced LGF_MergeBitsToByte LGF_MergeBitsToDword LGF_MergeBytesToWord LGF_MergeBytesToDWord LGF_MergeWordsToDWord LGF_SplitByteToBits LGF_SplitDWordToBits LGF_SplitWordToBytes LGF_SplitDWordToBytes LGF_SplitDWordsToWords LGF_DTLtoString_DE LGF_StringToDTL_DE LGF_StringToInt LGF_StringToTime LGF_GpsDDToGps LGF_GpsToGpsDD LGF_CelsiusToFahrenheit LGF_FahrenheitToCelsius LGF_CelsiusToKelvin LGF_KelvinToCelsius LGF_KelvinToFahrenheit LGF_FahrenheitToKelvin LGF_KelvinToRankine LGF_RankineToKelvin LGF_Boxplot_UDInt LGF_Boxplot_LReal LGF_Histogram_UDInt LGF_Histogram_LReal Legacy LGF_SawTooth → 99 Legacy