SIEMENS

Library Description • 03/2017

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

STEP 7 (TIA Portal) V14

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

Warranty and Liability

Note

The Application Examples are not binding and do not claim to be complete regarding the circuits shown, equipping and any eventuality. The Application Examples do not represent customer-specific solutions. They are only intended to provide support for typical applications. You are responsible for ensuring that the described products are used correctly. These Application Examples do not relieve you of the responsibility to use safe practices in application, installation, operation and maintenance. When using these Application Examples, you recognize that we cannot be made liable for any damage/claims beyond the liability clause described. We reserve the right to make changes to these Application Examples at any time without prior notice.
If there are any deviations between the recommendations provided in these Application Examples and other Siemens publications − e.g. Catalogs − the contents of the other documents have priority.

We do not accept any liability for the information contained in this document.
Any claims against us – based on whatever legal reason – resulting from the use of the examples, information, programs, engineering and performance data etc., described in this Application Example shall be excluded. Such an exclusion shall not apply in the case of mandatory liability, e.g. under the German Product Liability Act ("Produkthaftungsgesetz"), in case of intent, gross negligence, or injury of life, body or health, guarantee for the quality of a product, fraudulent concealment of a deficiency or breach of a condition which goes to the root of the contract ("wesentliche Vertragspflichten"). The damages for a breach of a substantial contractual obligation are, however, limited to the foreseeable damage, typical for the type of contract, except in the event of intent or gross negligence or injury to life, body or health. The above provisions do not imply a change of the burden of proof to your detriment.
Any form of duplication or distribution of these Application Examples or excerpts hereof is prohibited without the expressed consent of the Siemens AG.

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 only form one element of such a concept.
Customer is responsible to prevent unauthorized access to its plants, systems, machines and networks. Systems, machines and components should only be connected to the enterprise network or the internet if and to the extent necessary and with appropriate security measures (e.g. use of firewalls and network segmentation) in place.
Additionally, Siemens' guidance on appropriate security measures should be taken into account. For more information about industrial security, please visit http://www.siemens.com/industrialsecurity.

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends to apply product updates as soon as available and to always use the latest product versions. Use of product versions that are no longer supported, and failure to apply latest updates may increase customer's exposure to cyber threats.
To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed under http://www.siemens.com/industrialsecurity.

Table of Contents

1 Library Overview

1.1 General

TIA Portal features an extensive number of “ready-to-use” instructions (mathematical functions, times, timers, etc.). Furthermore, there are additional useful basic functions.

These functions are provided in the form of a library and can be used freely. The finished functions are freely adjustable and can thus be used universally.

The library described here is version-numbered and is continually expanded. Information on the versioning can be found in chapter 5.1 Library versioning .

1.2 Hardware and software requirements

Requirements for this library

To be able 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:

- S7-1200 and S7-1200 F product family
- S7-1500 and S7-1500 F product family
- Simulation with S7-PLCSIM (from V14 and higher)

Software

- STEP 7 (TIA Portal) Basic or Professional from V14 or higher

Note

It is generally possible to open a library with STEP 7 Basic, even if it contains STEP 7 Professional elements (e.g. S7-1500 controller). In this case, you will be informed by a message upon opening the library.

All elements (types and master copies) can be used if they are supported by the installed hardware in the TIA portal.

In case you attempt to copy elements from the library with STEP 7 Basic which are not supported (e.g. S7-1500 controller), an error message is displayed.

1.3 Library resources

The following section gives you an overview of the size of the blocks of the library in the main memory.

Assignment of memory space to the individual blocks

Table 1-1: Memory usage (CPU 1212 DC/DC/DC V4.2, CPU 1511-1 PN V2.0)

Block	CPU 1212 allocation (in byte)		CPU 1511 allocation (in byte)	
	Load memory	Work memory	Load memory	Work memory
Bit logic operations				
FB LGF_PulseRelay V1.0.2	7295	201	7367	302
Date and timer operations				
FB LGF_Astro V1.1.4	48132	3525	48226	3604
FB LGF_SetTime V1.0.2	26835	2231	27040	2301
FB LGF_TimerSwitch V1.1.1	35911	4161	35861	4272
FC LGF_CalendarDayWeekV1.0.0	21867	1751	22071	1826
Counter operations				
FC LGF_CountFalInDWord V1.0.1	14890	1124	14896	1188
FC LGF_CountRisInDWord V1.0.1	14759	1124	14766	1188
Comperator operations				
FC LGF_CompareVariant V1.0.2	10955	620	10966	684
FC LGF_CompareReal V1.0.0	6281	118	6302	182
Math operations				
FC LGF_AverageAndDeviation V1.0.2	29602	3194	29614	3258
FB LGF_FloatingAverage V1.1.0	16590	748	16506	830
FC LGF_MatrixAddition V2.0.0	11514	489	11556	553
FC LGF_MatrixInverse V2.0.0	15936	1017	15976	1081
FC LGF_MatrixMultiplication V2.0.0	12172	546	12213	610
FC LGF_MatrixSubtraction V2.0.0	11545	489	11590	553
FC LGF_MatrixTranspose V2.0.0	9971	383	10010	447
FB LGF_MinMaxHistory V1.0.1	6150	114	6144	178
FC LGF_RandomBasic V1.0.0	6844	191	6868	255
FC LGF_RandomInt V1.0.1	9104	241	9115	305
FB LGF_RandomReal V1.0.2	9528	281	9550	345
FC LGF_SearchMinMax V1.0.1	36145	4642	36195	4706
FC LGF_XRoot V1.0.1	4688	49	4693	113
FC LGF_HighLowLimit V1.0.0	8978	262	9015	334
FC LGF_Integration V1.0.0	11058	323	11073	393
Data handling				
FB LGF_FIFO V1.0.2	20681	1688	20773	1784
FB LGF_ShellSortInt V1.1.1	18458	1458	18526	1540
FB LGF_ShellSortUint V1.1.1	18563	1458	18637	1540

Block	CPU 1212 allocation (in byte)		CPU 1511 allocation (in byte)	
	Load memory	Work memory	Load memory	Work memory
FB LGF_ShellSortReal V1.1.1	18484	1458	18530	1540
Converter operations				
FC LGF_BinaryToGray V1.0.2	4327	36	4335	100
FC LGF_GrayToBinary V1.0.2	11803	872	11809	936
FC LGF_BitsToWord V1.0.0	6052	186	6079	250
FC LGF_WordToBits V1.0.0	6226	175	6250	239
FC LGF_DTLtoString V1.0.1	16403	876	16395	935
FC LGF_StringToDTL V1.0.1	19911	1116	19886	1139
FC LGF_TemperatureConvert V1.0.1	6644	242	6624	306
FC LGF_ScaleLinear V1.0.0	29123	4443	29144	4507
FC LGF_StringToTaddr V1.0.0	21707	883	21715	941
FC LGF_TaddrToString V1.0.0	10780	388	10825	447
Signal generators				
FB LGF_Frequency V1.1.2	10878	345	10871	412
FB LGF_Impulse V1.2.0	7730	131	7719	200
FB LGF_SawTooth V1.0.1	10137	249	10179	320
Technology operations				
FB LGF_LimRateOfChangeBasic V1.0.1	11985	357	12022	430
FB LGF_LimRateOfChangeAdvanced V1.0.1	24401	1474	24704	1706
FB LGF_Ramp V1.0.0	27084	1440	27144	1517
FB LGF_NonLin V1.0.0	12901	564	13014	640

2 Working with the Library

All blocks in the “LGF” library are freely usable in connection with S7-1200 and S7-1500 controllers.

Most blocks are stored as type in the library. The blocks are therefore version-numbered and can fully exploit the advantages.

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

Note Information on the general use of libraries can be found in the S7-1200/1500 program guide under the chapter “libraries”.

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

Note All blocks in the LGF have been created according to the programming style guide.

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

Further information on libraries in the TIA portal:

- How do you open libraries in STEP 7 (TIA Portal)?
  https://support.industry.siemens.com/cs/ww/en/view/37364723
- automation in less than 10 minutes TIA portal: Time Savers – Global libraries
  https://support.industry.siemens.com/cs/ww/en/view/78529894
- Which elements of STEP 7 (TIA Portal) can you store in a library as Type or as Master Copy?
  https://support.industry.siemens.com/cs/ww/en/view/109476862
- How can you automatically open a global library upon starting up TIA portal from V13 or higher, and how can you use it, for example, as a company library?
  https://support.industry.siemens.com/cs/ww/en/view/100451450

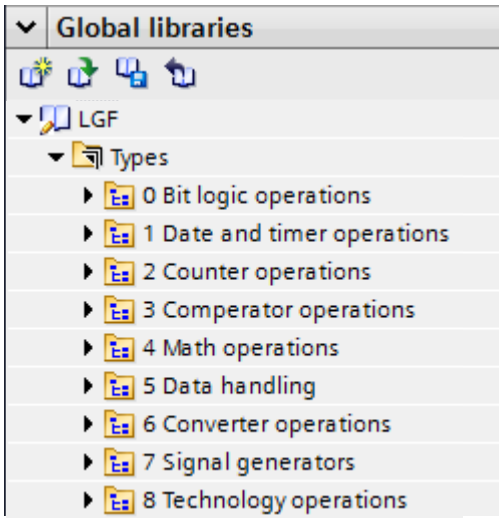
3 Explanation of the blocks

The following chapters describe all blocks of the library “Library General Functions”. The chapters follow the same structure as the library itself.

All blocks are divided into areas of application or categories:

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

Figure 3-1: Global library (LGF)



3.0 Bit logic operations

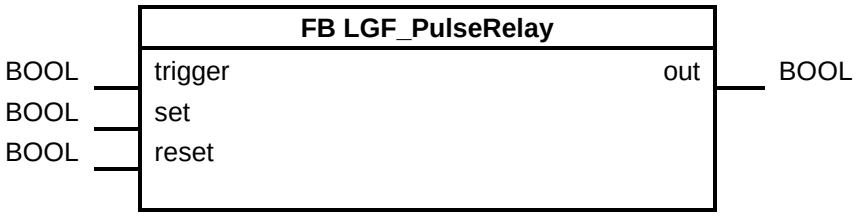
3.0.1 FB LGF_PulseRelay

Short description

This block equals a surge relay or a toggle flip-flop including a set and reset input.

Block

Figure 3-2: FB LGF_PulseRelay



Input parameters

Table 3-1: Input parameter

Parameter	Data type	Description
Trigger	BOOL	Every rising edge changes the boolean value of the “out” output.
Set	BOOL	Every rising edge sets the boolean value of the “out” output to “TRUE”.
Reset	BOOL	Every rising edge sets the boolean value of the “out” output to “FALSE”.

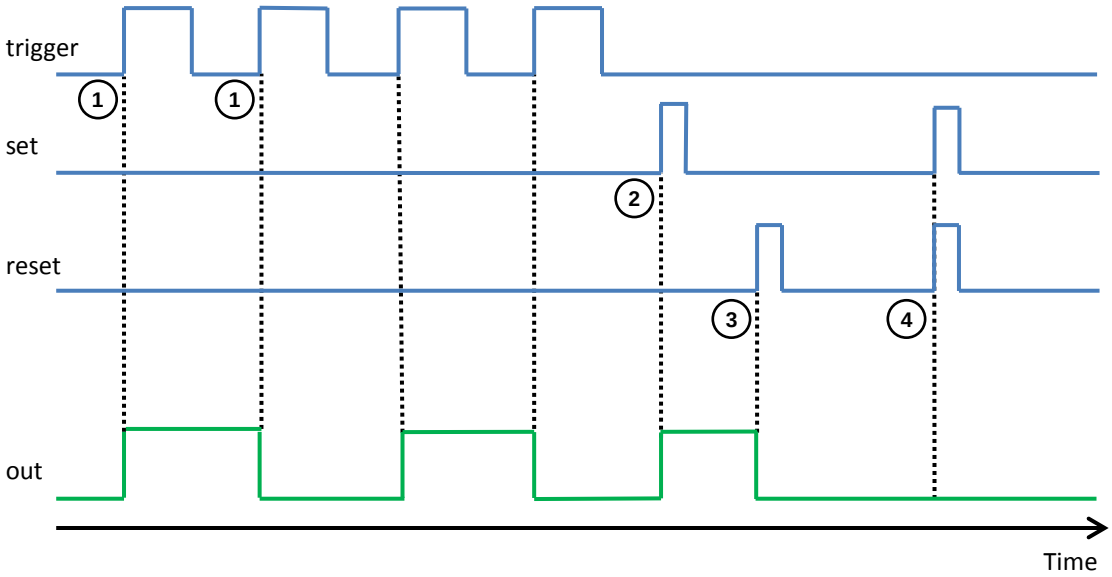
Output parameters

Table 3-2: Output parameter

Parameter	Data type	Description
Out	BOOL	Signal output

Mode of Operation

Figure 3-3: LGF_PulseRelay signal diagram



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

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

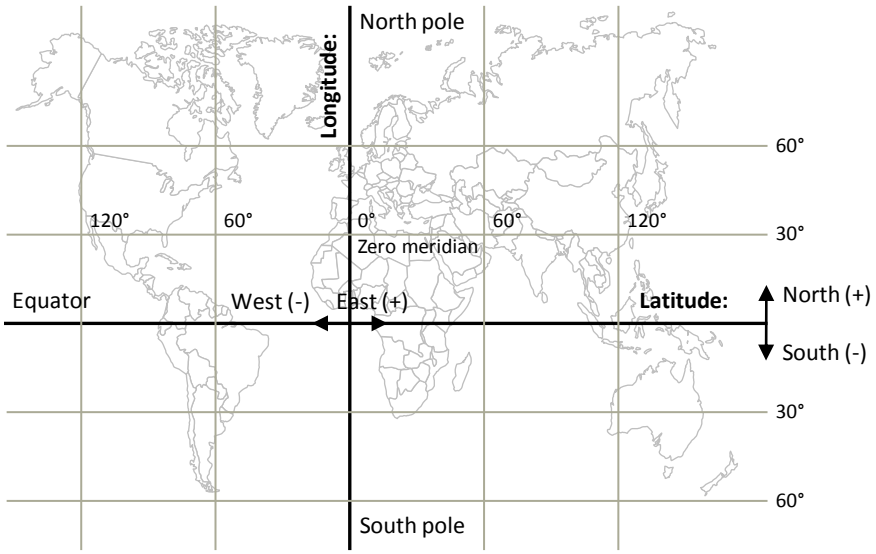
3.1 Date and timer operations

3.1.1 FB LGF_Astro

Short description

This block calculates the sunrise and sunset times for a particular location on earth. The exact position is transmitted to the block in the form of geographic coordinates (longitude and latitude).

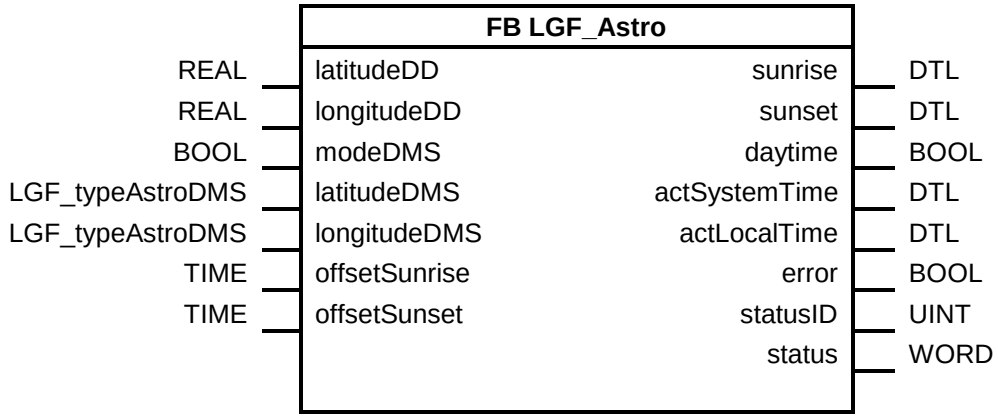
Figure 3-4: Earth with longitude and latitude



Information on time synchronization can be found in the following entry:
https://support.industry.siemens.com/cs/ww/en/view/86535497

Block

Figure 3-5: FB LGF_Astro



Input parameters

Table 3-3: Input parameter

Parameter	Data type	Description
latitudeDD	REAL	Latitude in degrees with decimal values (unit: degrees decimal), Permissible value range [-90.00000..+90.00000]° This is a common coordinate format in GPX files (GPS).
longitudeDD	REAL	Longitude in degrees with decimal values (unit: degrees decimal)[1] Permissible value range [-180.0000..+180.0000]°
modeDMS	BOOL	1. Transfer format of position in “degrees decimal” via the formal parameters “latitudeDD” and “longitudeDD” 1. Transfer format of position in direction, degree, minute and seconds via the formal parameters “latitudeDMS” and “longitudeDMS”
latitudeDMS	LGF_typeAstroDMS	Latitude in compass direction; degree; minutes and seconds in the PLC data type “LGF_typeAstroDMS“. Permissible parameter values [N,S]; [0..90]; [0..59]; [0..59] Permissible parameter values (sum of parameter values) [N, S, n, s]; [0..90]° This is a common coordinate format in map navigation.
longitudeDMS	LGF_typeAstroDMS	Longitude in direction; degree; minutes and seconds in the PLC data type “LGF_typeAstroDMS“. Permissible parameter values [E, W]; [0..180]; [0..59]; [0..59] Permissible parameter values (sum of parameter values) [E, W, e, w]; [0..180]° The international standard letter designating east is “E” (East).
offsetSunrise	TIME	Offset of power-on time for “daytime”
offsetSunset	TIME	Offset of power-off time for “daytime”

Output parameters

Table 3-4: Output parameter

Parameter	Data type	Description
sunrise	DTL	Sunrise at specified location considering the “offsetSunrise“
sunset	DTL	Sunset at specified location considering the “offsetSunset“
daytime	BOOL	If the controller’s local time is between “sunrise” and “sunset” then “daytime” returns the value “TRUE”.
actSystemTime	DTL	Current system time (UTC)
actLocalTime	DTL	Current local time
error	BOOL	0: no errors 1: Block error, “statusID” returns error source, “status” returns error code.
statusID	UINT	“statusID” returns the ID of the block reporting the status. See table below.
status	WORD	“status” returns the status/error code (see table below).

Status and error displays

Table 3-5: Status/error codes

statusID	status	Meaning	Remedy/notes
1	16#7000	Initial value	-
1	16#0000	No errors	-
1	16#8200	Incorrect direction information at “latitudeDMS.dir“ input	Only the following characters are allowed: N, n, S, s, W, w, E, e
1	16#8201	Incorrect values at “latitudeDMS“	Check the values at • “latitudeDMS.deg“ • “latitudeDMS.min“ • “latitudeDMS.sec“
1	16#8202	Incorrect direction information at “longitudeDMS.dir“	Only the following characters are allowed: N, n, S, s, W, w, E, e
1	16#8203	Incorrect values at “longitudeDMS“	Check the values at • “longitudeDMS.deg“ • “longitudeDMS.min“ • “longitudeDMS.sec“
1	16#8204	Incorrect value at “latitudeDD” input.	Check the actual value at the input.
1	16#8205	Incorrect value at “longitudeDD” input.	Check the actual value at the input.
2	-	Error/status of subordinate block “RD_SYS_T“.	-
3	-	Error/status of subordinate block “RD_LOC_T“.	-

Note If “statusID” is > 1, all values of the “status” output came directly from called up instructions (see table on output parameters). In this case, get the information on the respective instructions from the TIA Portal Online Help.

Mode of Operation

If processes are to be run automatically, depending on the day and night change, an astronomical clock function is required. Examples for this would be switching exterior lighting on and off of opening and closing roller shutters.

If these processes are to be executed time-delayed, that is, at a specified time before or after sunrise or sunset, an offset is required for each of them as well.

Note To execute the function exactly, it needs to be ensured that the 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 sunrise and sunset times. The offset times are added on to the sunrise and sunset and returned at the “sunrise” and “sunset” outputs. If the system time of the SIMATIC controller is in-between these values, the “daytime” output is set to the value “TRUE”.

Note

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

- at correspondingly high offset values
- at a location beyond the polar circle

The coordinates can be entered in the “DMS” format (with PLC data type “LGF_typeAstroDMS”), or in “Degree.Decimal”.

Which format is active can be set with the “modeDMS” formal parameter (see Table 3-3).

The entry of the coordinate values is checked for valid values. In case of invalid values, a corresponding error code is returned at “status” (see Table 3-5)

If there is an invalid coordinate value at a formal parameter and if this formal parameter was activated via “modeDMS”, the “sunrise” and “sunset” outputs are set to the value DTL#1970-01-01-00:00:00.

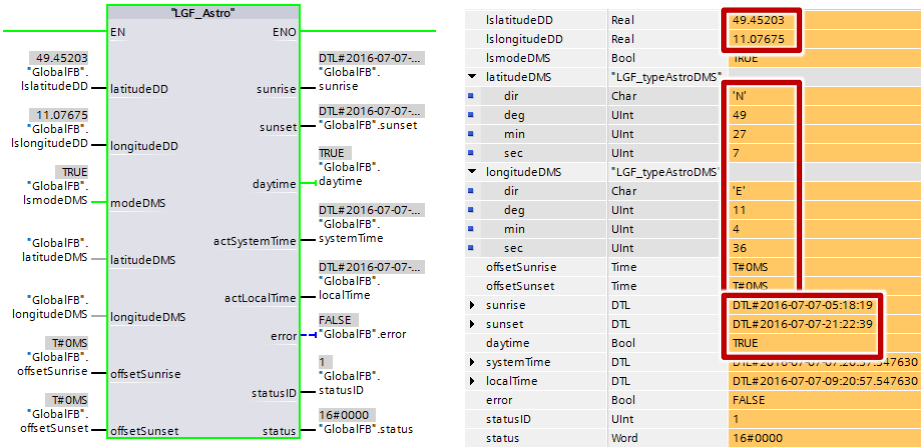
Example

The following example describes the block’s mode of operation.

Table 3-6: Geographic coordinates for Nuremberg-Moorenbrunn, date and system time

longitude:	+ 11.07675°	or	E 11° 4' 36"
Latitude:	+ 49.45203°	or	N 49° 27' 7"
Date:	07.07.2016	Local time:	09:20 AM

Figure 3-6: FB LGF_Astro, online monitoring of the block with the parameters as well as the actual parameters via the monitoring table



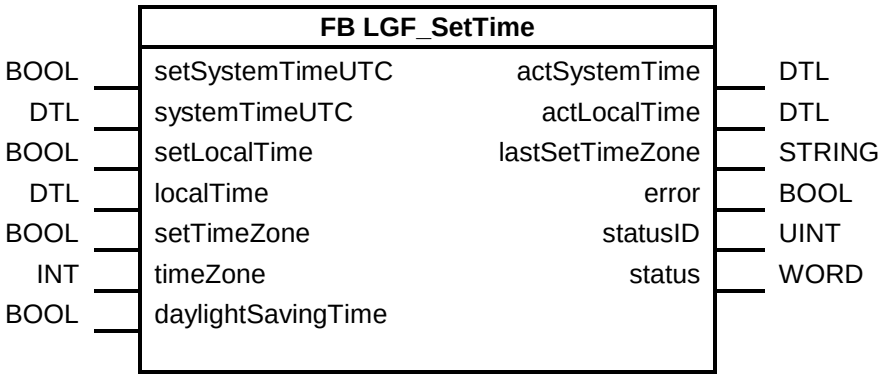
3.1.2 FB LGF_SetTime

Short description

This block summarizes the functions system time, local time and setting time zone.

Block

Figure 3-7: FB LGF_SetTime



Input parameters

Table 3-7: Input parameter

Parameter	Data type	Description
setSystemTimeUTC	BOOL	Rising edge sets the specified system time at the “systemTimeUTC” input
systemTimeUTC	DTL	Specified system time, corresponds to UTC (Coordinated Universal Time)
setLocalTime	BOOL	Rising edge accepts the specified local time at the “localTime” input
localTime	DTL	Defined local time
setTimeZone	BOOL	Rising edge accepts • the value at the “timeZone” input • the value at the “daylightSavingTime“ input
timeZone	INT	Defined local time (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 AM - until last Sunday in October at 3:00 AM FALSE: no daylight saving time changeover

Output parameters

Table 3-8: Output parameter

Parameter	Data type	Description
systemTime	DTL	Current system time (UTC)
localTime	DTL	Current local time
lastSetTimeZone	STRING	Time zone that has last been set by the block
error	BOOL	0: no errors 1: Block error, “statusID” returns error source, “status” returns error code.
statusID	UINT	“statusID” returns the ID of the block reporting the status. See table below.
status	WORD	“status” returns the status/error code (see table below).

Status and error displays

Table 3-9: Status/error codes

statusID	status	Meaning	Remedy/notes
1	16#7000	Initial value	-
1	16#0000	No errors	-
1	16#8200	No valid time zone has been returned at the “timeZone” input.	Use only allowed values (see mode of operation).
2	-	Error/status of subordinate block “SET_TIMEZONE“	-

Note If “statusID” is > 1, all values of the “status” output came directly from called up instructions (see table on output parameters). In this case, get the information on the respective instructions from the TIA Portal Online Help.

Mode of Operation

This block summarizes the functions system time, local time and setting time zone.

The following time zones are possible at 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

Input "timeZone"	Time zone
-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) Baghdad, Kuwait, Riyadh
330	(UTC +03:30) Iran
400	(UTC +04:00) Moscow, St. Petersburg, Volgograd, Abu Dhabi, Muscat
430	(UTC +04:30) Afghanistan
500	(UTC +05:00) Islamabad, Karachi, Tashkent
530	(UTC +05:30) India, Sri Lanka
545	(UTC +05:45) India, Sri Lanka
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 Corea
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) Kiribati

Note

Daylight saving time/winter time

The parameters (time difference, beginning of daylight saving time, beginning of winter time) can be adjusted in the static "statTimeZone" tag.

Standard values of the static "statTimeZone" tag:

- Time difference: 60 min
- Beginning of daylight saving time: Last Sunday in March, 02:00 am
- Beginning of winter time: Last Sunday in October, 03:00 am

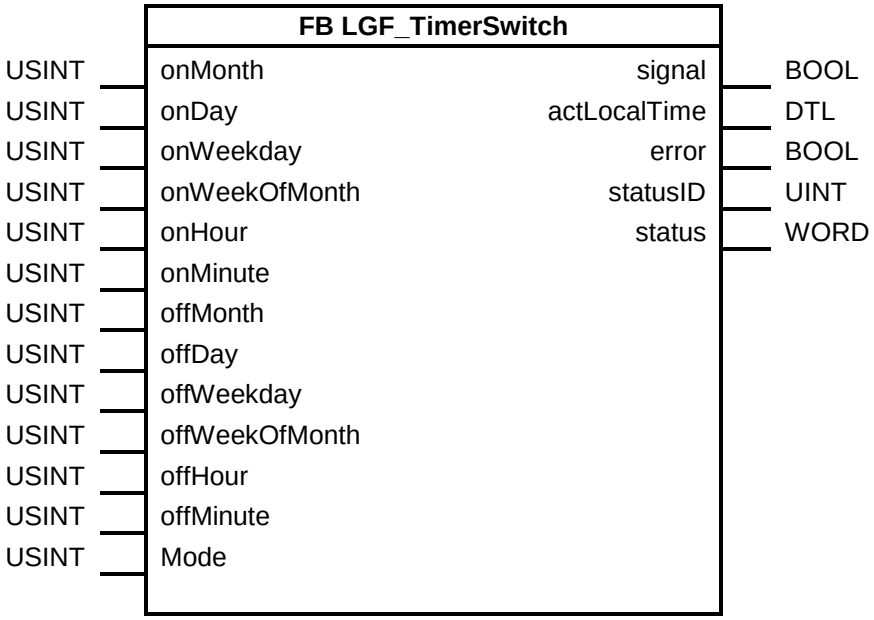
3.1.3 FB LGF_TimerSwitch

Short description

This block is a timer. It is possible to set daily, weekly, monthly and annual scheduler points, as well as scheduler points for weekdays or weekend days.

Block

Figure 3-8: FB LGF_TimerSwitch



Input parameters

Table 3-10: Input parameter

Parameter	Data type	Description
onMonth	USINT	Month, in which the signal is to be set.
onDay	USINT	Day, on which the signal is to be set.
onWeekday	USINT	Weekday, on which the signal is to be set. (Sunday = 1)
onHour	USINT	Hour, at which the signal is to be set.
onMinute	USINT	Minute, at which the signal is to be set.
offMonth	USINT	Month, in which the signal is to be reset.
offDay	USINT	Day, on which the signal is to be reset.
offWeekday	USINT	Weekday, on which the signal is to be reset. (Sunday = 1)
offHour	USINT	Hour, at which the signal is to be reset.
offMinute	USINT	Minute, at which the signal is to be reset.
mode	USINT	Specification of module (see operating notes).

Output parameters

Table 3-11: Output parameter

Parameter	Data type	Description
signal	BOOL	Output signal
actLocalTime	DTL	Current local time
error	BOOL	0: no errors 1: Block error, “statusID” returns error source, “status” returns error code.
statusID	UINT	“statusID” returns the ID of the block reporting the status. See table below.
status	WORD	“status” returns the status/error code (see table below).

Status and error displays

Table 3-12: Status/error codes

statusID	status	Meaning	Remedy/notes
1	16#7000	Initial value	-
1	16#0000	No errors	-
1	16#8200	No valid actual value was returned at the “mode” input.	Allowed values “1”, “2”, “3”, “4”, “5”, “6”
2	-	Error/status of subordinate block “RD_LOC_T“	-

Note If “statusID” is > 1, all values of the “status” output came directly from called up instructions (see table on output parameters). In this case, get the information on the respective instructions from the TIA Portal Online Help.

Mode of Operation

The block provides different kinds of timers which are specified in the “mode” parameter:

- Daily timer (mode = 1)
- Weekly timer (mode = 2)
- Monthly timer (mode = 3)
- Annual timer (mode = 4)
- On weekdays, from monday till friday (mode = 5)
- On weekend, saturday and sunday (mode = 6)

Depending on the mode, the following formal parameters need to be activated:

Table 3-13: Required formal parameters for the respective mode

Mode	Required formal parameters
Daily timer (mode = 1)	• onHour / offHour • onMinute / offMinute
Weekly timer (mode = 2)	• onWeekday / offWeekday • onHour / offHour • onMinute / offMinute
Monthly timer (mode = 3)	• onDay / offDay • onHour / offHour • onMinute / offMinute
Annual timer (mode = 4)	• onMonth / offMonth • onDay / offDay • onHour / offHour • onMinute / offMinute
On weekdays (mode = 5)	• onHour / offHour • onMinute / offMinute
Weekend (mode = 6)	• onHour / offHour • onMinute / offMinute

If the set start time corresponds to the controller's actual local time, the "signal" output is set to "TRUE". If the set switch-off time corresponds to the controller's actual local time, the "signal" output is reset again.

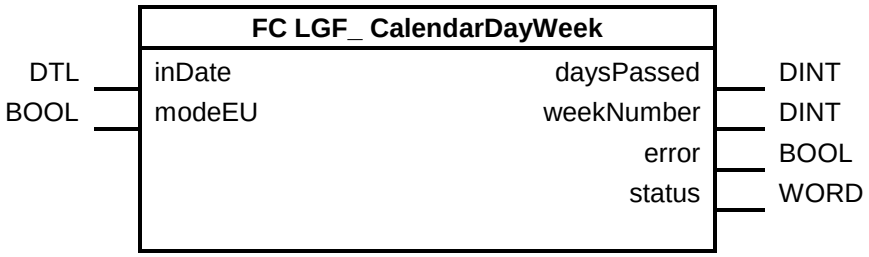
3.1.4 FC LGF_CalendarDayWeek

Short description

Using the specified date, this function calculates the calender week and the number of days that have passed since the beginning of the year.

Block

Figure 3-9: FC LGF_ CalendarDayWeek



Input parameters

Table 3-14: Input parameter

Parameter	Data type	Description
inDate	DTL	Date for calculating the calendar week and the days since January 1.
modeEU	BOOL	Selecting the calculation mode for calendar weeks. 0: for the calculation in the US and in many other countries 1: for the calculation in european countries according to the ISO 8601

Output parameters

Table 3-15: Output parameter

Parameter	Data type	Description
daysPassed	DINT	Days passed since January 1.
weekNumber	DINT	Number of calendar week.
error	BOOL	Error display. 0: no error. 1: Block error, “status” returns error code.
status	WORD	“status” returns the status/error code (see table below).

Status and error displays

Table 3-16: Status/error codes

Status	Meaning	Remedy
16#0000	No errors.	-
16#8201	Date at “inDate” input below the lower limit.	Select “inDate” greater than 1971-01-01.
16#8202	Date at “inDate” input above the upper limit.	Select “inDate” lower than 2261-12-31.

Mode of Operation

There are two calculating modes for determining the calendar week for the specified date. Via the “modeEU“ input, you can select the calculatin mode:

- modeEU = 0: For the calculation in the US and in many other countries
- modeEU = 1: for the calculation in european countries according to the ISO 8601

For the calculation in european countries according to the ISO 8601

- Calendar weeks have 7 days, start on a Monday and are counted continuously over the year.
- The first calendar week of a year is the one containing the first Thursday.
- Each year has either 52 or 53 calendar weeks.
- A year has 53 weeks if the following properties apply:
  - A common year starts and ends on a Thursday.
  - A leap year either starts on a Wednesday and ends on a Thursday or it starts on a Thursday and ends on a Friday.
- The 29, 30 and 31 December may already belong to the first calendar week of the following year.
- The 1, 2 and 3 January may still belong to the last calendar week of the previous year.

For the calculation in the US and in many other countries

- Calendar weeks have 7 days, start on a Sunday and are counted continuously over the year.
- The first calendar week of a year is the one containing the 1 January.
- Each year has either 52 or 53 calendar weeks.
- A year has 53 weeks if the following properties apply:
  - A common year starts on a Saturday and ends on a Saturday.
  - A leap year either starts on a Saturday and ends on a Sunday or it starts on a Friday and ends on a Saturday.
- The days after the last Saturday in December may already belong to the first calendar week of the following year.

3.2 Counter operations

3.2.1 FC LGF_CountFallnDWord

Short description

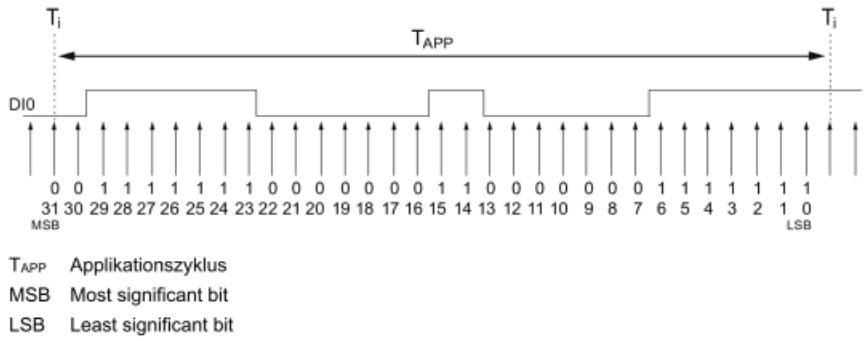
This block analyses a tag of the DWORD type and returns, how often a 1-0 sequence (falling edge) occurs in the tag.

Application Example

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

With the oversampling function, the technology module captures the status of the respective digital input per application cycle (e.g. OB61) at 32 times and in even chronological intervals. These 32 states are collectively returned at the check-back interface in the form of 32 bit values.

Figure 3-10: Oversampling example of DI0 at the DIDQ 16x24V TM Timer.



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

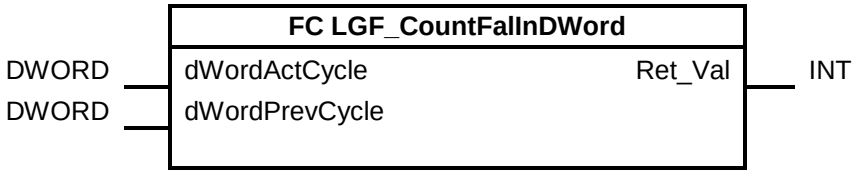
In this case, the LGF_CountFallnDWord block is used to count the number of falling edges.

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

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

Block

Figure 3-11: FC LGF_CountFallnDWord



Input parameters

Table 3-17: Input parameter

Parameter	Data type	Description
dWordActCycle	DWORD	Double word, in which the falling edges are counted
dWordPrevCycle	DWORD	Double word from the previous cycle

Output parameters

Table 3-18: Output parameter

Parameter	Data type	Description
Ret_Val	INT	Number of falling edges in the double word

Mode of Operation

The block counts the number of falling edges (1-0 transitions) in a tag of the DWORD data type from left to right. In this, the “Ret_Val” output returns the number of falling edges.

For falling edges to be detected at tag bounds, the “dWordPrevCycle” input needs to be interconnected with the tag of the previous cycle.

Example

The following example describes the block’s mode of operation. In this case, it is assumed that a signal of unknown length is continually scanned per cycle in the form of double words (DWORD).

Within this signal, 1-0 sequences (falling edges) are to be continually counted and returned. For falling edges to be detected at tag bounds as well in this example, the “dWordPrevCycle” input needs to be interconnected with the tag of the previous cycle.

Table 3-19: Example

DWORD previous cycle (dWordPrevCycle)	DWORD actual cycle (dWordActCycle)
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 FC LGF_CountRisInDWord

Short description

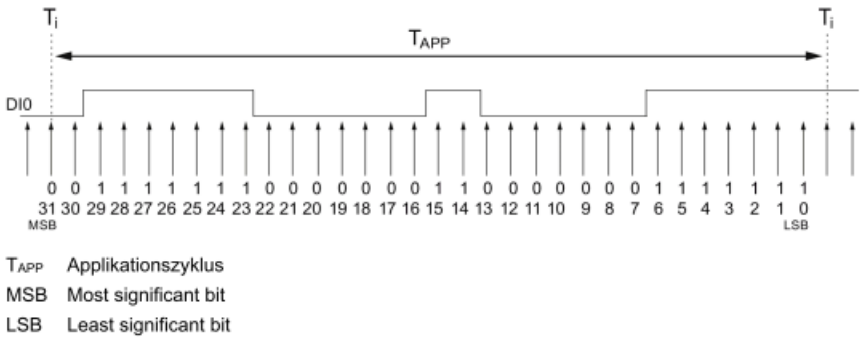
This block analyzes a tag of the DWORD type and returns, how often a 0-1 sequence (rising edge) occurs in the tag.

Application example:

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

With the oversampling function, the technology module captures the status of the respective digital input per application cycle (e.g. OB61) at 32 times and in even chronological intervals. These 32 states are collectively returned at the check-back interface in the form of 32 bit values.

Figure 3-12: Oversampling example of DI0 at the DIDQ 16x24V TM Timer.



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

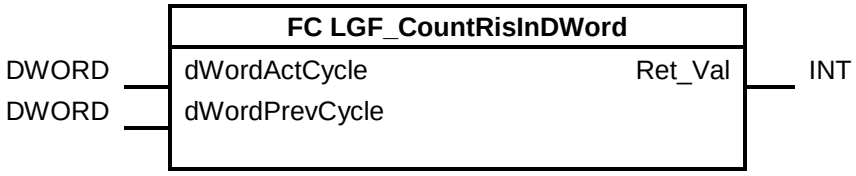
In this case, the LGF_CountRisInDWord block is used to count the number of rising edges.

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

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

Block

Figure 3-13: FC LGF_CountRisInDWord



Input parameters

Table 3-20: Input parameter

Parameter	Data type	Description
dWordActCycle	DWORD	Double word, in which the rising edges are counted
dWordPrevCycle	DWORD	Double word from the previous cycle

Output parameters

Table 3-21: Output parameter

Parameter	Data type	Description
Ret_Val	INT	Number of rising edges in the double word

Mode of Operation

The block counts the number of rising edges (0-1 transitions) in a tag of the DWORD data type from left to right. In this, the “Ret_Val” output returns the number of rising edges.

For rising edges to be detected at tag bounds as well, the “dWordPrevCycle” input needs to be interconnected with the tag of the previous cycle.

Example

The following example describes the block’s mode of operation. In this case, it is assumed that a signal of unknown length is continually scanned per cycle in the form of double words (DWORD).

Within this signal, 0-1 sequences (rising edges) are to be continually counted and returned. For rising edges to be detected at tag bounds as well in this example, the “dWordPrevCycle” input needs to be interconnected with the double word of the previous cycle.

Table 3-22: Example

DWORD previous cycle (dWordPrevCycle)	DWORD actual cycle (dWordActCycle)
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 Comperator operations

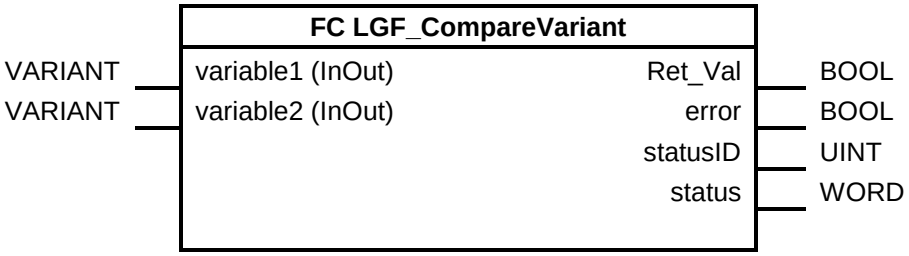
3.3.1 FC LGF_CompareVariant

Short description

This block compares two structured actual parameters (array, PLC data type) and returns, whether they correspond to the same type and have the same values.

Block

Figure 3-14: FC LGF_CompareVariant



Input/Output parameters (InOut)

Table 3-23: Input/Output parameters (InOut)

Parameter	Data type	Description
variable1	VARIANT	Comparison tag with arbitrary data type
variable2	VARIANT	Comparison tag with arbitrary data type

Output parameters

Table 3-24: Output parameter

Parameter	Data type	Description
Ret_Val	BOOL	0: Values of comparison tags or PLC data types differ. 1: Values of comparison tags are the same and PLC data types are identical.
Error	BOOL	0: no errors 1: Block error, “statusID” returns error source, “status” returns error code.
statusID	UINT	“statusID” returns the ID of the block reporting the status. See table below.
Status	WORD	“status” returns the status/error code (see table below).

Status and error displays

Table 3-25: Status/error codes

statusID	Status	Meaning	Remedy/notes
1	16#7000	Initial value	-
1	16#0000	No errors	-

Mode of Operation

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

Note

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

- Tags of the “Struct” data type cannot be compared.
- With strings, differences may occur in the range between actual length and maximum length.
- A disparity can also be displayed with “same” tags, if the structure contains REAL numbers.
- Tags of the “ARRAY of BOOL“ type cannot be checked for equality with the function, because the used “CountOfElements” instruction also counts the filling elements (e.g. with an ARRAY[0..1] of BOOL, 8 is output).

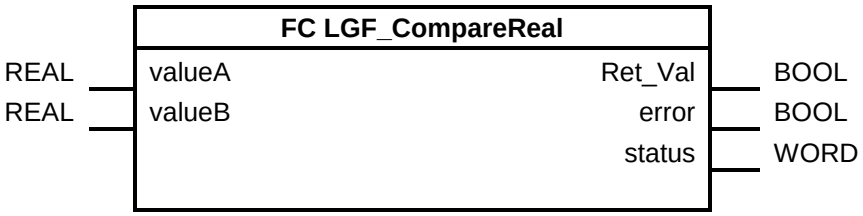
3.3.2 FC LGF_CompareReal

Short description

Via an approximation formula, this function checks the equality of two REAL numbers.

Block

Figure 3-15: FC LGF_CompareReal



Input parameters

Table 3-26: Input parameter

Parameter	Data type	Description
valueA	REAL	Number to be compared.
valueB	REAL	Number to be compared.

Output parameters

Table 3-27: Output parameter

Parameter	Data type	Description
Ret_Val	BOOL	0: not equal 1: equal
Error	BOOL	Error display. 0: no error. 1: Block error, “status” returns error code.
Status	WORD	“status” returns the status/error code (see table below).

Status and error displays

Table 3-28: Status/error codes

Status	Meaning	Remedy
16#0000	No errors.	-
16#8601	|valueA|+|valueB| exceeds the maximum value range of a REAL number.	|valueA|+|valueB| must be $+3.402823 * 10^{38}$ smaller.

Mode of Operation

Via an approximation formula, two REAL numbers, “valueA” and “valueB” are checked for equality. In order to compare the two numbers regardless of their size, their ration is examined.

Approximation formula

$$\frac{|\text{valueA} - \text{valueB}|}{|\text{valueA}| + |\text{valueB}|} \leq \text{EPSILON}$$

Two REAL numbers are classified as equal if the ratio of the difference to the sum of both numbers is smaller or equal to EPSILON.

The value of EPSILON describes the accuracy with which two numbers are classified as equal. In the function, EPSILON is defined as a constant with a value of $1.0 \cdot 10^{-6}$.

To avoid a division by zero, the approximation formula is changed in the function:

$$|\text{valueA} - \text{valueB}| \leq \text{EPSILON} * (|\text{valueA}| + |\text{valueB}|)$$

The following table shows two examples to clarify the function principle.

Table 3-29

	Example 1	**Example 2**
$\|\text{valueA}\| + \|\text{valueB}\|$	5	5.000.000
EPSILON	$1.0 \cdot 10^{-6}$	
Maximum difference of "valueA" and "valueB", in which two numbers are still classified a equal.	0.000005	5

Note If you require a different accuracy for the examination of the ratio between the numbers for the application, you need to adjust the constant EPSILON in the function to your needs.

3.4 Math operations

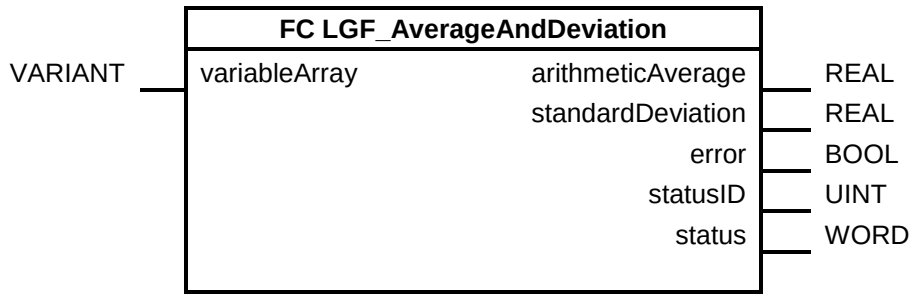
3.4.1 FC LGF_AverageAndDeviation

Short description

This block determines the arithmetic average and the standard deviation from a series of numbers.

Block

Figure 3-16: FC LGF_AverageAndDeviation



Input parameters

Table 3-30: Input parameter

Parameter	Data type	Description
variableArray	VARIANT	Series of numbers which which to calculate

Output parameters

Table 3-31: Output parameter

Parameter	Data type	Description
arithmeticAverage	REAL	Arithmetic mean
standardDeviation	REAL	Standard deviation
Error	BOOL	0: no errors 1: Block error, “statusID” returns error source, “status” returns error code.
statusID	UINT	“statusID” returns the ID of the block reporting the status. See table below.
Status	WORD	“status” returns the status/error code (see table below).

Status and error displays

Table 3-32: Status/error codes

statusID	status	Meaning	Remedy/notes
1	16#7000	Initial value	-
1	16#0000	No errors	-
1	16#8200	At the “variableArray” input, the actual parameter is not an array.	-
1	16#8201	The data type of the array elements is not supported (see mode of operation).	-
2	-	Error/status of subordinate block “MOVE_BLK_VARIANT“.	-

Note If “statusID” is > 1, all values of the “status” output came directly from called up instructions (see table on output parameters). In this case, get the information on the respective instructions from the TIA Portal Online Help.

Mode of Operation

Via the “variableArray” input, an array of any size is connected. After a data type query in the block, the arithmetic average and the standard deviation are calculated from the values and returned.

Note An array with too many elements may lead to the scan cycle monitoring time being exceeded.

Note Note: Only the data types Int, UInt, DInt, UDInt, USInt, SInt and Real are supported.

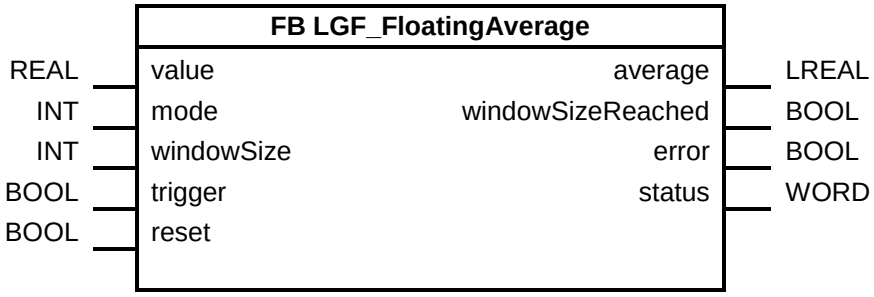
3.4.2 FB LGF_FloatingAverage

Short description

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

Block

Figure 3-17: LGF_FloatingAverage



Input parameters

Table 3-33: Input parameter

Parameter	Data type	Description
Value	REAL	Values from which the moving average is to be determined.
Mode	INT	Selection of mode (see mode of operation); Default value: mode=1.
windowsSize	INT	Window size for the moving averaging in the range of 1..100 The default value is 100.
Trigger	BOOL	Trigger tag, impulse
Reset	BOOL	The block is reset and the calculation starts over.

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

Further information can be found in the chapter “Overview of data type conversion” in the TIA Portal online help or under:

https://support.industry.siemens.com/cs/ww/en/view/109011420/58427923211

Output parameters

Table 3-34: Output parameter

Parameter	Data type	Description
Average	LREAL	Moving average
windowSizeReached	BOOL	0: maximum window size not yet reached 1: maximum window size reached
Error	BOOL	0: no errors 1: Block errors
Status	WORD	Status/ error code (see table below)

Status and error displays

Table 3-35: Status/error codes

status	Meaning	Remedy/notes
16#0000	No errors	-
16#8201	No mode selected	Select a mode (1 or 2).
16#8202	Incorrect window size	Set a value between 1 and 100.

Mode of Operation

The block calculates the (moving) average on the basis of the set window size. The window size specifies the maximum number of the values last read in. After the maximum number of values has been read in, the “windowSizeReached” output will be set and each newly read in value replaces the respective oldest value (FIFO principle).

To read in the values, two modes are available which are determined via the “mode” parameter:

- mode = 1: Read in with each impulse at the “trigger” input
- mode = 2: Cyclic read in

The block is activated as soon as a mode has been selected. It is possible to select between the modes during operation.

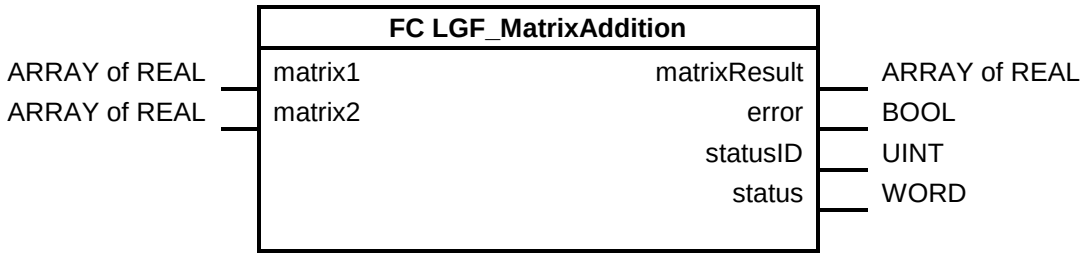
3.4.3 FC LGF_MatrixAddition

Short description

This block adds up two matrices of the same size of the “ARRAY of REAL” data type.

Block

Figure 3-18: FC LGF_MatrixAddition



Input parameters

Table 3-36: Input parameter

Parameter	Data type	Description
matrix1	ARRAY of REAL	First summand (Matrix)
matrix2	ARRAY of REAL	Second summand (Matrix)

Output parameters

Table 3-37: Output parameter

Parameter	Data type	Description
matrixResult	ARRAY of REAL	Sum (Matrix)
Error	BOOL	0: no errors 1: Block error, “statusID” returns error source, “status” returns error code.
statusID	UINT	“statusID” returns the ID of the block reporting the status. See table below.
Status	WORD	“status” returns the status/error code (see table below).

Status and error displays

Table 3-38: Status/error codes

statusID	status	Meaning	Remedy/notes
1	16#0000	No errors	-
1	16#8200	Lower bounds of arrays are different.	All arrays need to have the same lower bound, e.g.: Array[**0**..2, **0**..2] of Real
1	16#8201	Upper bounds of arrays are different.	All arrays need to have the same upper bound, e.g.: Array[0..**2**, 0..**2**] of Real

Mode of Operation

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

Note Please note that all input and output matrices need to have the same lower and upper bound and therefore the same number of columns and rows.

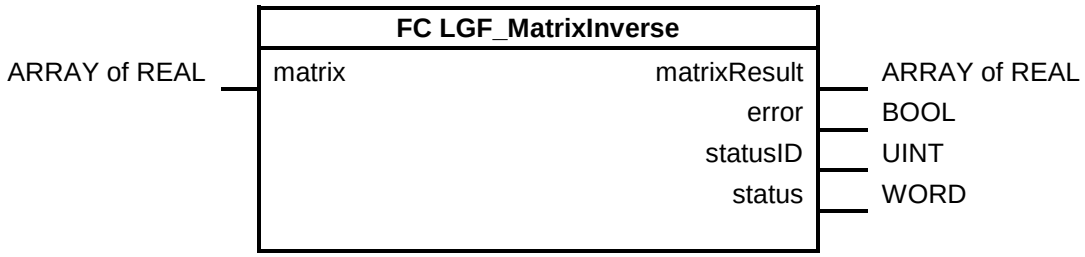
3.4.4 FC LGF_MatrixInverse

Short description

This block inverts a square matrice of the “ARRAY of REAL” data type.

Block

Figure 3-19: FC LGF_MatrixInverse



Input parameters

Table 3-39: Input parameter

Parameter	Data type	Description
Matrix	ARRAY of REAL	Input matrix

Output parameters

Table 3-40: Output parameter

Parameter	Data type	Description
matrixResult	ARRAY of REAL	Inverse matrix
Error	BOOL	0: no errors 1: Block error, “statusID” returns error source, “status” returns error code.
statusID	UINT	“statusID” returns the ID of the block reporting the status. See table below.
Status	WORD	“status” returns the status/error code (see table below).

Status and error displays

Table 3-41: Status/error codes

statusID	status	Meaning	Remedy/notes
1	16#0000	No errors	-
1	16#8200	Lower bounds of arrays are different.	All arrays need to have the same lower bound, e.g.: Array[**0**..2, **0**..2] of Real
1	16#8201	Upper bounds of arrays are different.	All arrays need to have the same upper bound, e.g.: Array[0..**2**, 0..**2**] of Real
1	16#8202	Input matrix is not quadratic.	The number of lines needs to be equal to the number of columns.
1	16#8203	Use of algorithm for input matrix not possible.	First element ($a_{1,1}$) of input matrix must not be zero.

Mode of Operation

The block inverts a square matrix of any size with the Shipley-Coleman procedure.

Note

Note that the input matrix needs to be quadratic. This means that the number of lines needs to be equal to the number of columns.

The output matrix needs to be dimensioned as large and have the same array bounds as the input matrix.

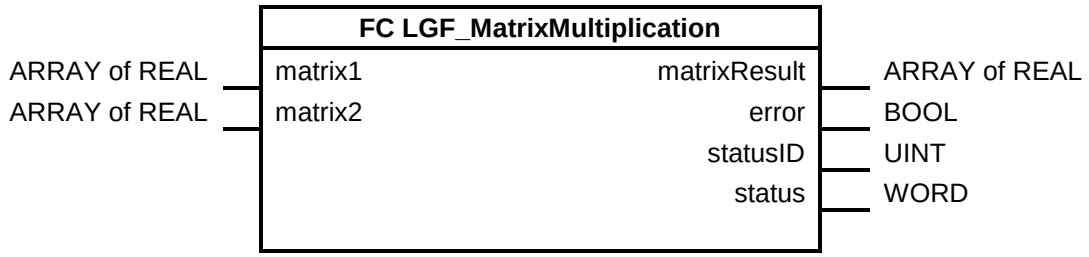
3.4.5 FC LGF_MatrixMultiplication

Short description

This block multiplies two matrices of the “Array of REAL” data type.

Block

Figure 3-20: FC LGF_MatrixMultiplication



Input parameters

Table 3-42: Input parameter

Parameter	Data type	Description
matrix1	ARRAY of REAL	First factor: Matrix to be multiplied
matrix2	ARRAY of REAL	Second factor: Matrix to be multiplied

Note

New as of V2.0.0
The input parameters “rows1”, “columns1”, “rows2” and “columns2” are no longer needed for this block version, as the variable bounds of array[*,*] are queried via the “LOWER_BOUND” and “UPPER_BOUND” instructions.

Output parameters

Table 3-43: Output parameter

Parameter	Data type	Description
matrixResult	ARRAY of REAL	Product: The resulting matrix
Error	BOOL	0: no errors 1: Block error, “statusID” returns error source, “status” returns error code.
statusID	UINT	“statusID” returns the ID of the block reporting the status. See table below.
Status	WORD	“status” returns the status/error code (see table below).

Status and error displays

Table 3-44: Status/error codes

statusID	status	Meaning	Remedy/notes
1	16#0000	No errors	-
1	16#8200	The number of columns of the first matrix does not correspond to the number of rows of the second matrix.	The array bounds also need to be identical.
1	16#8201	The size of the output matrix (m x n) results from the number of rows (m) of “matrix1” and the number of columns (n) from “matrix2”.	The array bounds also need to be identical.

Mode of Operation

The block multiplies two matrices of varying sizes. The individual elements of the two incoming matrices are read, added up and then returned in the “matrixResults” matrix.

Note

Note that the numbers of columns of the first matrix needs to be equal to the number of rows of the second matrix.

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

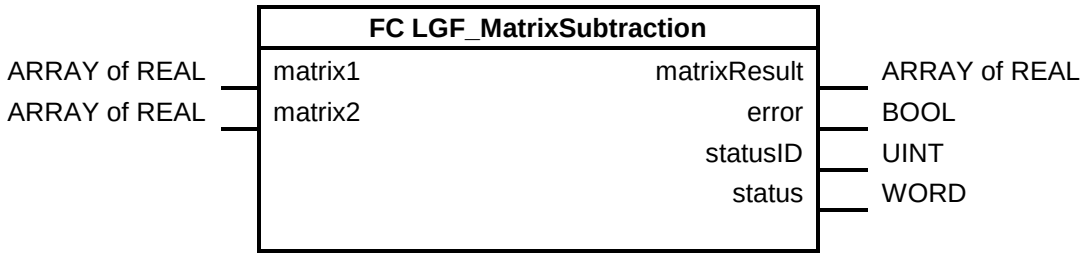
3.4.6 FC LGF_MatrixSubtraction

Short description

This block subtracts a matrix of the “ARRAY of REALR” data type from another.

Block

Figure 3-21: FC LGF_MatrixSubtraction



Input parameters

Table 3-45: Input parameter

Parameter	Data type	Description
matrix1	ARRAY of REAL	Minuend: From this matrix, “matrix2” is subtracted.
matrix2	ARRAY of REAL	Subtrahend: This matrix is subtracted from “matrix1”.

Output parameters

Table 3-46: Output parameter

Parameter	Data type	Description
matrixResult	ARRAY of REAL	Difference: The resulting matrix
Error	BOOL	0: no errors 1: Block error, “statusID” returns error source, “status” returns error code.
statusID	UINT	“statusID” returns the ID of the block reporting the status. See table below.
Status	WORD	“status” returns the status/error code (see table below).

Status and error displays

Table 3-47: Status/error codes

statusID	status	Meaning	Remedy/notes
1	16#0000	No errors	-
1	16#8200	Lower bounds of arrays are different.	All arrays need to have the same lower bound, e.g.: Array[**0**..2, **0**..2] of Real
1	16#8201	Upper bounds of arrays are different.	All arrays need to have the same upper bound, e.g.: Array[0..**2**, 0..**2**] of Real

Mode of Operation

The block subtracts two matrices of varying sizes. The individual fields of the two matrices are read, subtracted and then returned in the “matrixResults” matrix.

Note Please note that all input and output matrices need to have the same number of columns and rows.

3.4.7 FC LGF_MatrixTranspose

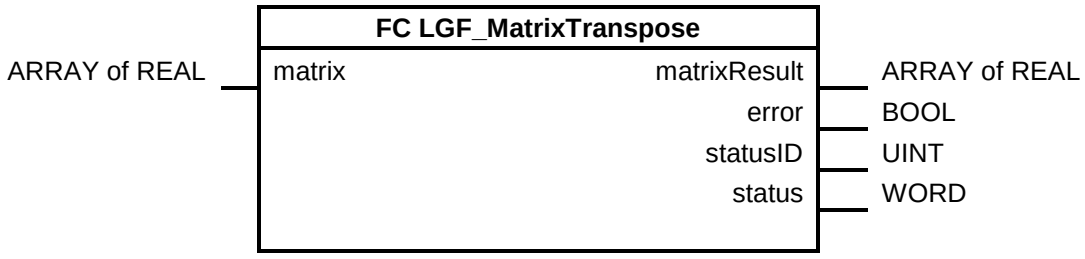
Short description

This block transposes a matrix of the ARRAY of REAL data type.

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

Block

Figure 3-22: FC LGF_MatrixTranspose



Input parameters

Table 3-48: Input parameter

Parameter	Data type	Description
Matrix	ARRAY of REAL	matrix to be transposed.

Note **New as of V2.0.0**
The input parameters “rows” and “columns” are no longer needed for this block version, as the variable bounds of array[*,*] are queried via the “LOWER_BOUND” and “UPPER_BOUND” instructions.

Output parameters

Table 3-49: Output parameter

Parameter	Data type	Description
matrixResult	ARRAY of REAL	resulting matrix
Error	BOOL	0: no errors 1: Block error, “statusID” returns error source, “status” returns error code.
statusID	UINT	“statusID” returns the ID of the block reporting the status. See table below.
Status	WORD	“status” returns the status/error code (see table below).

Status and error displays

Table 3-50: Status/error codes

statusID	status	Meaning	Remedy/notes
1	16#0000	No errors	-
1	16#8200	Lower bounds of arrays are different.	All arrays need to have the same lower bound, e.g.: Array[**0**..2, **0**..2] of Real

statusID	status	Meaning	Remedy/notes
1	16#8201	Upper bounds of arrays are different.	All arrays need to have the same upper bound, e.g.: Array[0..**2**, 0..**2**] of Real

3.4.8 FB LGF_MinMaxHistory

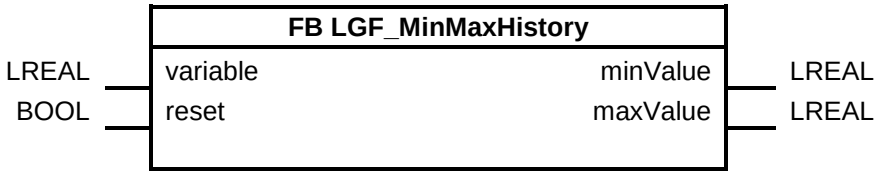
Short description

This block reads in a tag's value each time it is called up and returns the maximum and minimum value that has been read in since the first call up.

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

Block

Figure 3-23: FB LGF_MinMaxHistory



Input parameters

Table 3-51: Input parameter

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

Output parameters

Table 3-52: Output parameter

Parameter	Data type	Description
minValue	LREAL	Minimum value since the first call up or since the activation of the "reset" input.
maxValue	LREAL	Maximum value since the first call up or since the activation of the "reset" input.

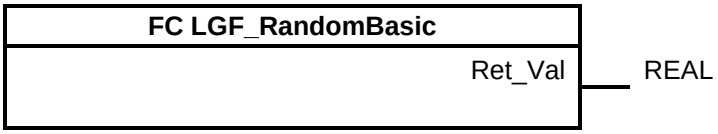
3.4.9 FC LGF_RandomBasic

Short description

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

Block

Figure 3-24: FC LGF_RandomBasic



Output parameters

Table 3-53: Output parameter

Parameter	Data type	Description
Ret_Val	REAL	Random number

Mode of Operation

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

Background information

The random value is generated from the nano seconds of the current system time of the CPU. In this, the byte order of this value is inverted and then converted to a standard floating-point number.

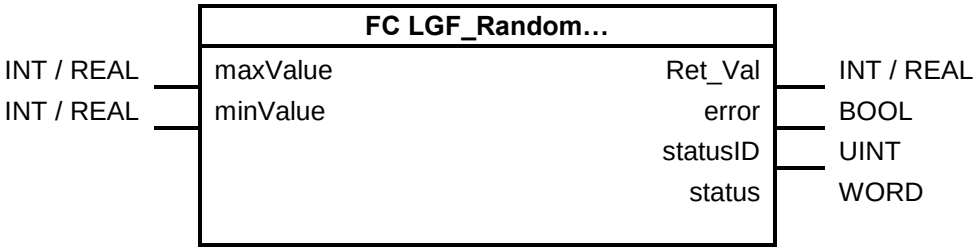
3.4.10 FC LGF_RandomINT / LGF_RandomReal

Short description

This block generates a “random” value between a defined maximum and minimum value per call up. The random number has the INT /REAL data type.

Block

Figure 3-25: FC LGF_RandomINT / FC LGF_RandomReal



Input parameters

Table 3-54: Input parameter

Parameter	Data type	Description
maxValue	INT / REAL	Defines the random number’s upper limit value.
minValue	INT / REAL	Defines the random number’s lower limit value.

Output parameters

Table 3-55: Output parameter

Parameter	Data type	Description
Ret_Val	INT / REAL	Random number
Error	BOOL	0: no errors 1: Block error, “statusID” returns error source, “status” returns error code.
statusID	UINT	“statusID” returns the ID of the block reporting the status. See table below.
Status	WORD	“status” returns the status/error code (see table below).

Status and error displays

Table 3-56: Status/error codes

statusID	status	Meaning	Remedy/notes
1	16#7000	Initial value	-
1	16#0000	No errors	-
1	16#8200	“minValue“ is greater than “maxValue“.	-
2	-	Error/status of subordinate block “RD_SYS_T“.	-

Note If “statusID” is > 1, all values of the “status” output came directly from called up instructions (see table on output parameters). In this case, get the information on the respective instructions from the TIA Portal Online Help.

Mode of Operation

This block generates random values that are between the specified “minValue” value and “maxValue” value. This random value is returned via “Ret-Val”.

Background information

The random value is generated from the nano seconds of the current system time of the CPU. In this, the byte order of this value is inverted and then converted to a standard floating-point number.

Note If no maximum and minimum value (= 0) is determined, the block returns random values from the entire range of values of INT / REAL.

3.4.11 FC LGF_SearchMinMax

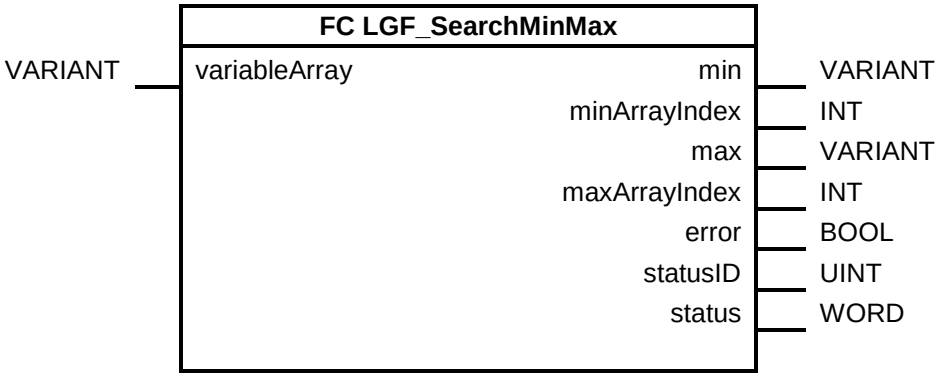
Short description

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

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

Block

Figure 3-26: FC LGF_SearchMinMax



Input parameters

Table 3-57: Input parameter

Parameter	Data type	Description
variableArray	VARIANT	Array in whose fields the maximum and minimum is searched for.

Output parameters

Table 3-58: Output parameter

Parameter	Data type	Description
minValue	VARIANT	Smallest value found.
minArrayIndex	INT	Start index of the array plus minArrayIndex results in the array index of the smallest value. The index starts with 0.
maxValue	VARIANT	Largest value found.
maxArrayIndex	INT	Start index of the array plus maxArrayIndex results in the array index of the largest value. The index starts with 0.
Error	BOOL	0: no errors 1: Block error, “statusID” returns error source, “status” returns error code.
statusID	UINT	“statusID” returns the ID of the block reporting the status. See table below.
Status	WORD	“status” returns the status/error code (see table below).

Status and error displays

Table 3-59: Status/error codes

statusID	status	Meaning	Remedy/notes
1	16#7000	Initial value	-
1	16#0000	No errors	-
1	16#8200	At the “variableArray” input, the actual parameter is not an array.	-
1	16#8201	The data type of the elements of the array is not supported.	Only the data types Int, UInt, DInt, UDInt, USInt, SInt and Real are supported.
1	16#8202	The elements of the array do not have the same data type as the “minValue” and “maxValue” outputs.	-
2	-	Error/status of subordinate block “MOVE_BLK_VARIANT“.	-

Note If “statusID” is > 1, all values of the “status” output came directly from called up instructions (see table on output parameters). In this case, get the information on the respective instructions from the TIA Portal Online Help.

Mode of Operation

Via the “variableArray” input, an array of any size is connected. After a data type query in the block, the elements are one after another copied into a tag of the respective type and compared. The smallest and the largest value are returned as well as their respective index in the array.

Note With multiple identical minimum and maximum values, the index of the first minimum or maximum value is returned.

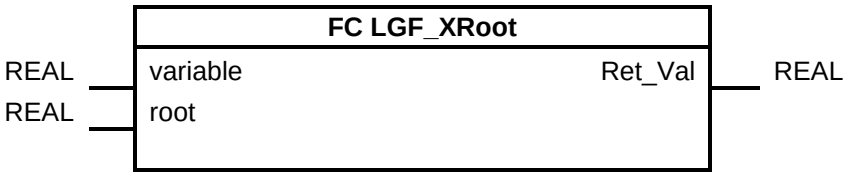
3.4.12 FC LGF_XRoot

Short description

This block calculates the xth root of a numerical tag.

Block

Figure 3-27: FC LGF_Random



Input parameters

Table 3-60: Input parameter

Parameter	Data type	Description
Variable	REAL	Tag, of which the root is to be calculated.
Root	REAL	Root (e.g. 3 as third root)

Output parameters

Table 3-61: Output parameter

Parameter	Data type	Description
Ret_Val	REAL	Output of result

Mode of Operation

The block calculates the nth root of a number. To perform this function, the following formula is extended.

$$\text{number} = e^{\log_e(\text{number})}$$

This results in:

$$\text{Ret\_Val} = \sqrt[\text{root}]{\text{number}} = \text{number}^{\frac{1}{\text{root}}} = (e^{\log_e(\text{number})})^{\frac{1}{\text{root}}} = e^{\ln(\text{number}) * \frac{1}{\text{root}}}$$

In STEP 7 (TIA Portal) the function equals “EXP“ $e^{(\dots)}$ and the function “LN“ ln(…).

This results in the following formula:

$$\text{Ret\_Val} = \text{EXP}((1/\text{root}) * \text{LN}(\text{number}))$$

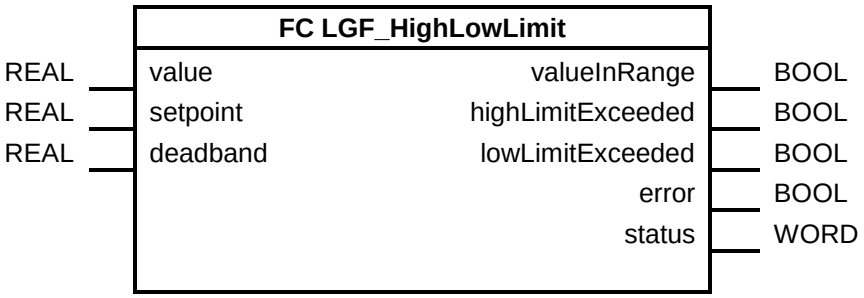
3.4.13 FC LGF_HighLowLimit

Short description

The functions checks whether a value is within a defined value range. The value range is defined via a setpoint value and a deadband around this setpoint value. The function calculates the lower and higher limit of the value range.

Block

Figure 3-28: FC LGF_HighLowLimit



Input parameters

Table 3-62: Input parameter

Parameter	Data type	Description
Value	INT / REAL	Value that is to be checked on whether it is within the defined value range
Setpoint	INT / REAL	Setpoint value
Deadband	INT / REAL	Deadband

Output parameters

Table 3-63: Output parameter

Parameter	Data type	Description
valueInRange	BOOL	“TRUE”, if “value” is within the value range (Deadband around setpoint value).
valueOverHighLimit	BOOL	“TRUE”, if “value” is higher than the higher limit value (“setpoint” + 0,5 * “deadband”).
valueUnderLowLimit	BOOL	“TRUE”, if “value” is lower than the lower limit value (“setpoint” - 0,5 * “deadband”).
Error	BOOL	“FALSE”: no errors "TRUE”: Block errors
Status	WORD	“status” returns the status/error code (see table below)

Status and error displays

Table 3-64: Status/error codes

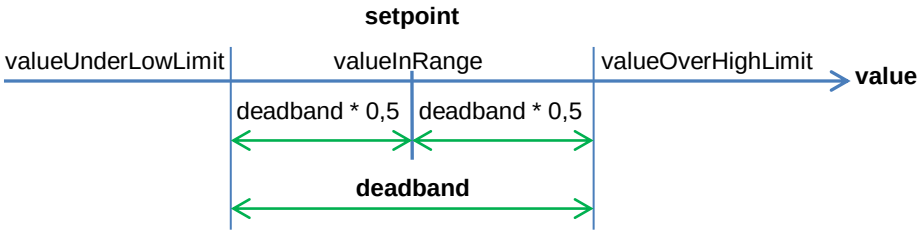
Status	Meaning	Remedy/notes
16#0000	No errors	-
16#8101	False value range “highValue”	-

Status	Meaning	Remedy/notes
16#8102	False value range “lowValue”	-
16#8200	“lowValue” is greater than “highValue”.	-

Mode of Operation

The “setpoint” and “deadband” tags define a value range. The function checks whether the “value” value is below, within or above the value range. The outputs “valueUnderLowLimit”, “valueInRange” or “valueOverHighLimit” indicate where the “value” value is.

Figure 3-29: Function principle



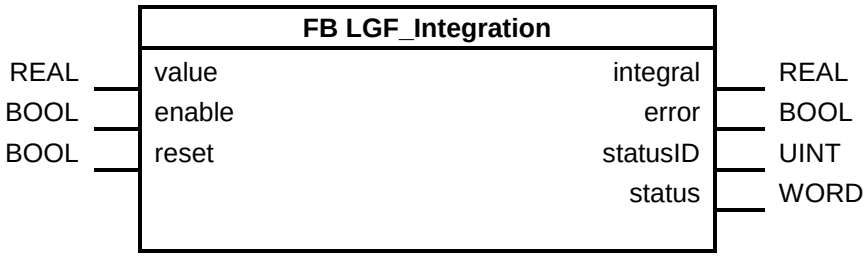
3.4.14 FB LGF_Integration

Short description

This block calculates the approximate area under a function curve. The function curve is delivered as analog value (REAL) that varies over time. At the output, the integral value is returned.

Block

Figure 3-30: FB LGF_Integration



Input parameters

Table 3-65: Input parameter

Parameter	Data type	Description
Value	REAL	Analog value of continous function curve
Enable	BOOL	Activation of integral calculation If this input receives the value “FALSE”, the integral calculation is stopped and the “integral” output returns the value last calculated.
Reset	BOOL	Sets the “integral” output to “0.0”.

Output parameters

Table 3-66: Output parameter

Parameter	Data type	Description
Integral	LREAL	Integrated value
Error	BOOL	0: no errors 1: Block error, “statusID” returns error source, “status” returns error code.
statusID	UINT	“statusID” returns the ID of the block reporting the status. See table below.
Status	WORD	“status” returns the status/error code (see table below).

Status and error displays

Table 3-67: Status/error codes

statusID	Status	Meaning	Remedy/notes
0	16#0000	No errors	-
2	-	Error/status of subordinate block “RD_SYS_T“.	-

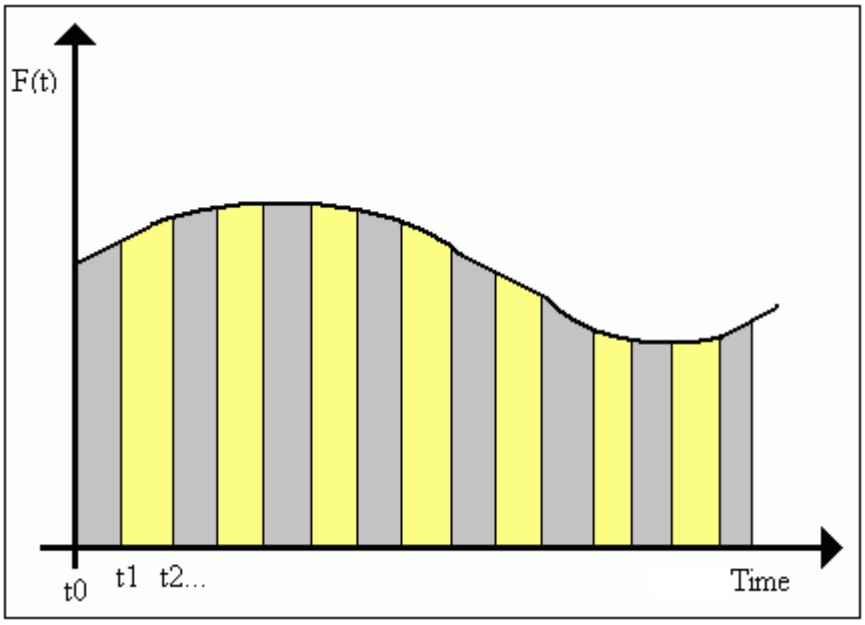
Note If “statusID” is > 1, all values of the “status” output came directly from called up instructions (see table on output parameters). In this case, get the information on the respective instructions from the TIA Portal Online Help.

Mode of Operation

The integral calculation contains the summation of those trapezoidal areas that span between the last two function values at the “value” input and the time. The elapsed time is calculated via the CPU’s system time. Tis trapezoidal area is identical to the result from the average value of both process values and the time interval.

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

Figure 3-31: Function principle



To launch the integral calculation for the input value at the “value” parameter, you need to

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

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

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

3.5 Data handling

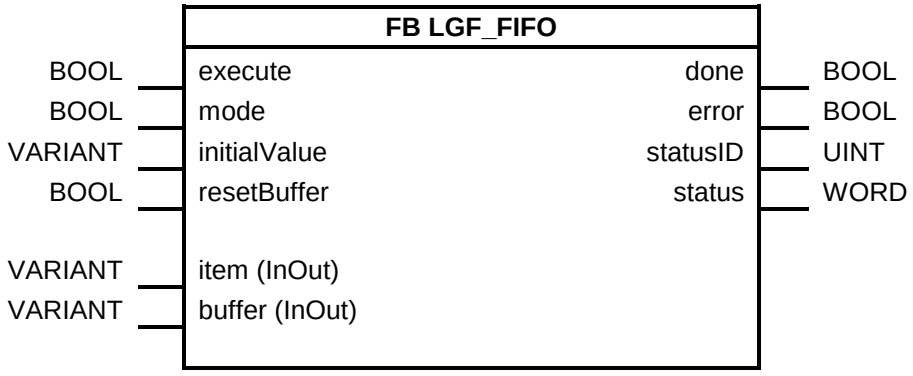
3.5.1 FB LGF_FIFO

Short description

This block stores incoming jobs/data and returns the oldest job that has not yet been processed.

Block

Figure 3-32: FB LGF_FIFO



Input parameters

Table 3-68: Input parameter

Parameter	Data type	Description
execute	BOOL	Requirement of a pass.
mode	BOOL	Selecting the mode. TRUE: Writing of “item” value into “buffer” FALSE: Reading the value from “buffer” and output at “item”
initialValue	VARIANT	Value for initialization of buffer (usually: 0)
resetBuffer	BOOL	Clearing out and initializing the buffer.

Input/Output parameters (InOut)

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

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

Output parameters

Table 3-70: Output parameter

Parameter	Data type	Description
done	BOOL	1: Pass completed.
error	BOOL	0: no errors 1: Block error, “statusID” returns error source, “status” returns error code.
statusID	UINT	“statusID” returns the ID of the block reporting the status. See table below.
status	WORD	“status” returns the status/error code (see table below).

Status and error displays

Table 3-71: Status/error codes

statusID	status	Meaning	Remedy/notes
1	16#7000	Initial value	-
1	16#0000	No errors	-
1	16#8001	The buffer is empty.	-
1	16#8002	The buffer is full.	-
1	16#8200	No array is applied to the “buffer” input.	-
1	16#8201	The data type of the InOut parameter “item” does not correspond to the data type of the array elements at the “buffer” input.	-
1	16#8202	The data type of the input “initialValue” does not correspond to the data type of the InOut parameter “item”.	-
1	16#8601	The tag “nextEmptyItemIndex“ is not within the array limits.	-
1	16#8602	The tag “firstItemIndex“ is not within the array limits.	-
2	-	Error/status of subordinate block “MOVE_BLK_VARIANT“.	-

Note If “statusID” is > 1, all values of the “status” output came directly from called up instructions (see table on output parameters). In this case, get the information on the respective instructions from the TIA Portal Online Help.

Mode of Operation

For the block to be started up, a positive edge needs to be detected at the “execute” input.

To call up the next read-task to be processed, the value “FALSE” must be activated at the “mode” input. If this is the case, the next task/data to be processed is returned at the “item” InOut parameter and this field is replaced in the buffer with the value at the “initialValue” parameter.

To save a new write task in the buffer, the “TRUE” value needs to be activated at the “mode” input. If this is the case, the value at the “item” InOut parameter or the task in the buffer is saved at the next free location.

If during a pass, the “TRUE” value is activated at the “resetBuffer” input, all fields in the buffer are reset to the value specified at the “InitialValue” input. After this, the buffer can once more be filled with tasks/data.

3.5.2 FB LGF_ShellSortInt / LGF_ShellSortUInt / LGF_ShellSortReal

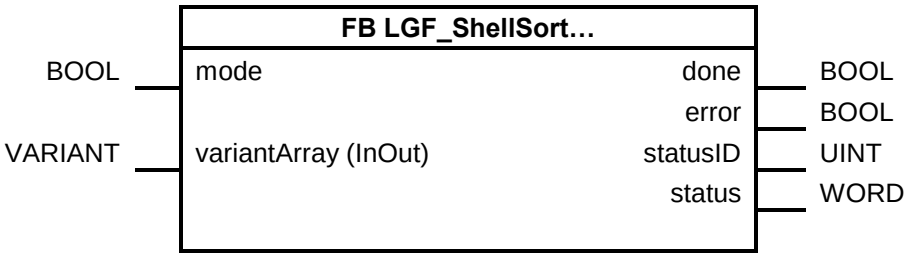
Short description

This block sorts an array with any number of elements (1000 max.) in ascending or descending order. The following data types are supported:

- Array of “Int” type: LGF_ShellSortInt
- Array of “UInt” type: LGF_ShellSortUInt
- Array of “Real” type: LGF_ShellSortReal

Block

Figure 3-33: FB LGF_ShellSort…



Input parameters

Table 3-72: Input parameter

Parameter	Data type	Description
mode	BOOL	0: sort in ascending order (default) 1: sort in descending order

Input/Output parameters (InOut)

Table 3-73: Input/Output parameters (InOut)

Parameter	Data type	Description
variantArray	VARIANT	Array that is to be sorted.

Output parameters

Table 3-74: Output parameter

Parameter	Data type	Description
done	BOOL	1: Sorting completed.
error	BOOL	0: no errors 1: Block error, “statusID” returns error source, “status” returns error code.
statusID	UINT	“statusID” returns the ID of the block reporting the status. See table below.
status	WORD	“status” returns the status/error code (see table below).

Status and error displays

Table 3-75: Status/error codes

statusID	status	Meaning	Remedy/notes
1	16#7000	Initial value	-
1	16#0000	No errors	-
1	16#8200	At the “variantArray” input, the actual parameter only has one element.	At the “variantArray” input, connect an array with at least two elements.
1	16#8201	At the “variableArray” input, the actual parameter is not an array.	
1	16#8202	At the “variantArray” input, the actual parameter does not have the suitable data type.	At the “variantArray” input, connect an array with the suitable data type: LGF_ShellSortInt: Array of type Int LGF_ShellSortUInt: Array of type UInt LGF_ShellSortReal: Array of type Real
1	16#8203	At the “variantArray” input, the actual parameter has too many elements.	By default, arrays with up to 1000 elements can be sorted.
2	-	Error/status of subordinate block “MOVE_BLK_VARIANT“ during reading the array.	-
3	-	Error/status of subordinate block “MOVE_BLK_VARIANT“ during writing the array.	-

Note If “statusID” is > 1, all values of the “status” output came directly from called up instructions (see table on output parameters). In this case, get the information on the respective instructions from the TIA Portal Online Help.

Mode of Operation

The block sorts according to the shell sorting method. Note that the block’s execution time strongly depends on how many elements the array to be sorted has. The following overview shows a few of the block’s measured values which are subject to the number of array elements.

Table 3-76: Execution times of the “LGF_ShellSort…” block

Number of array elements	S7-1212C DC/DC/DC	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 distributed over several SPS cycles. Therefore, the execution time directly affects the SPS cycle time. Keep this behavior in mind with regards to your project and the controller used and adjust the controller’s monitoring time, if necessary.

3.6 Converter operations

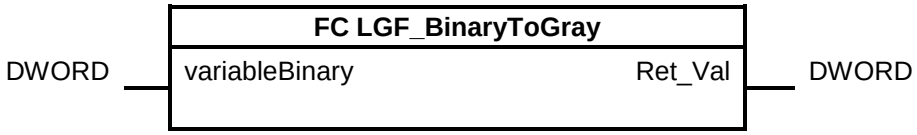
3.6.1 FC LGF_BinaryToGray

Short description

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

Block

Figure 3-34: FC LGF_BinaryToGray



Input parameters

Table 3-77: Input parameter

Parameter	Data type	Description
variableBinary	DWORD	Binary coded value

Output parameters

Table 3-78: Output parameter

Parameter	Data type	Description
Ret_Val	DWORD	Gray coded value

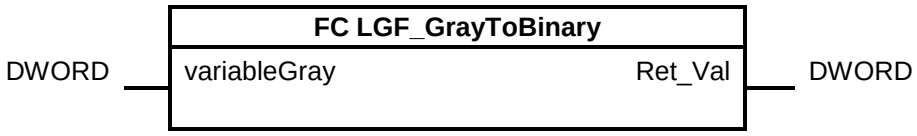
3.6.2 FC LGF_GrayToBinary

Short description

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

Block

Figure 3-35: FC LGF_GrayToBinary



Input parameters

Table 3-79: Input parameter

Parameter	Data type	Description
variableGray	DWORD	Gray coded value

Output parameters

Table 3-80: Output parameter

Parameter	Data type	Description
Ret_Val	DWORD	Binary coded value

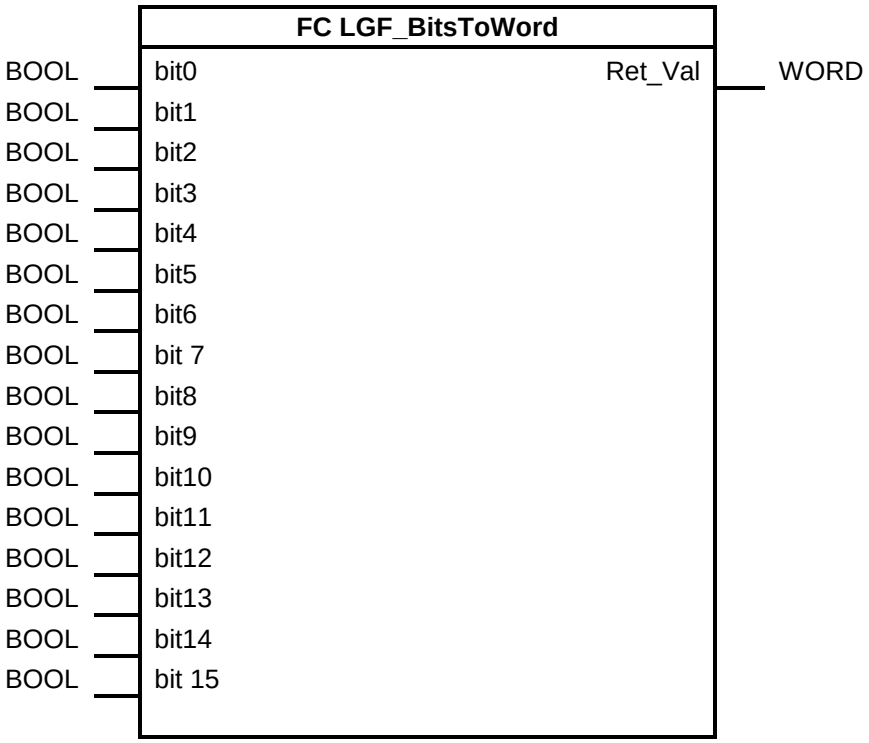
3.6.3 FC LGF_BitsToWord

Short description

This block converts 16 BOOL variables into a WORD variable.

Block

Figure 3-36: FC LGF_BitsToWord



Input parameters

Table 3-81: Input parameter

Parameter	Data type	Description
bit0 … bit 15	BOOL	Bit variables

Output parameters

Table 3-82: Output parameter

Parameter	Data type	Description
Ret_Val	WORD	Word variable

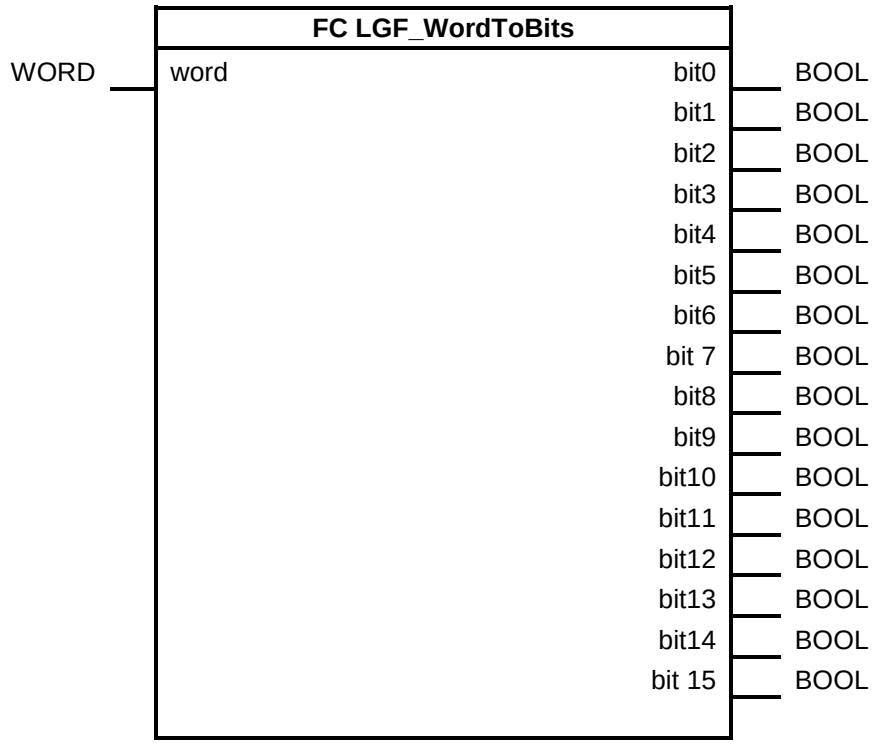
3.6.4 FC LGF_WordToBits

Short description

This block converts a WORD variable into 16 BOOL variables.

Block

Figure 3-37: FC LGF_WordToBits



Input parameters

Table 3-83: Input parameter

Parameter	Data type	Description
Word	WORD	WORD variable

Output parameters

Table 3-84: Output parameter

Parameter	Data type	Description
bit0 … bit 15	BOOL	Bit variables

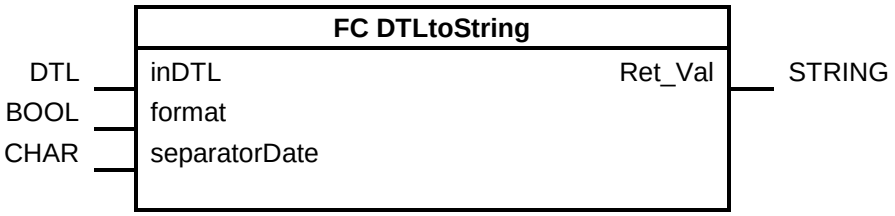
3.6.5 FC LGF_DTLtoString

Short description

This block converts a date of the DTL data type into a character string of the STRING data type.

Block

Figure 3-38: FC LGF_DTLtoString



Input parameters

Table 3-85: Input parameter

Parameter	Data type	Description
inDTL	DTL	Date
format	BOOL	Format selection of the displayed character string: 0: international (YYYY MM DD ...) 1: traditional (DD MM YYYY ...)
separatorDate	CHAR	Separator between the components of the displayed date.

Output parameters

Table 3-86: Output parameter

Parameter	Data type	Description
Ret_Val	STRING	Displayed character string

Mode of Operation

The block reads a date of the DTL data type, converts the date's individual components (year, month, day, hour...) into a character string and outputs it. The individual components are placed at the correct position in the character string according to the format selection. The separator between the date's components is variable.

Format selection

Via the "format" input parameter, you can select between the international (ISO 8601) and the traditional (DE) date format for the returned character string.

1. **International format (ISO 8601)**
   If the "format" input parameter has not been set (format = FALSE), the date is returned as character string in the international format.
   Example: 2016-03-16 13:34:12.123456789

   The following figure shows the position of the individual characters in the character string.

Figure 3-39: Position of the individual characters

	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

2. **Traditional format (DE)**
   If the “format” input parameter has been set (format = TRUE), the date is returned as character string in the traditional format.
   Example: 2016-16-03 13:34:12.123456789

   The following figure shows the position of the individual characters in the character string.

Figure 3-40: Position of the individual characters

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

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

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

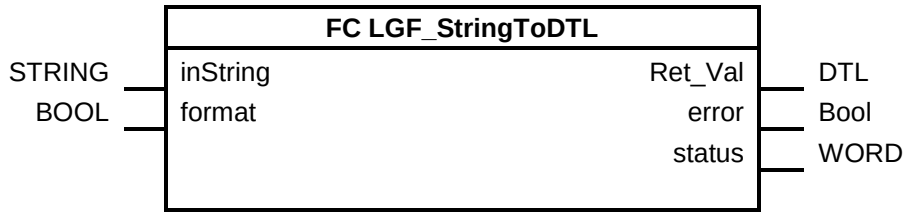
3.6.6 FC LGF_StringToDTL

Short description

This block converts a character string of the String format with date components into the DTL data type.

Block

Figure 3-41: FC LGF_StringToDTL



Input parameters

Table 3-87: Input parameter

Parameter	Data type	Description
inString	STRING	Date as character string
format	BOOL	Format selection of the read in character string: 0: international (YYYY MM DD ...) 1: traditional (DD MM YYYY ...)

Output parameters

Table 3-88: Output parameter

Parameter	Data type	Description
Ret_Val	DTL	Returns the read in date
error	BOOL	0: no error 1: Block error, “status” returns error code
status	WORD	“status” returns the status/error code (see table below)

Status and error displays

Table 3-89: Status/Error codes

Status	Meaning	Remedy/notes
16#0000	No errors	-
16#7000	Initial value	-
16#8201	Format: Year	Year specification does not correspond to format or specification (outside the value range of DTL)
16#8202	Format: Month	Month specification does not correspond to format or specification (outside the value range of DTL)
16#8203	Format: Day	Day specification does not correspond to format or specification (outside the value range of DTL)
16#8204	Format: Hour	Hour specification does not correspond to format or

Status	Meaning	Remedy/notes
		specification (outside the value range of DTL)
16#8205	Format: Minute	Minute specification does not correspond to format or specification (outside the value range of DTL)
16#8206	Format: Second	Second specification does not correspond to format or specification (outside the value range of DTL)
16#8207	Format: Nanosecond	Nanosecond specification does not correspond to format or specification (outside the value range of DTL)

Mode of Operation

This block reads in a date as character string and converts it into the DTL data type. The individual date components in the character string are separated according to the format selection (positioning of data components in the character string). In this, the separator between the components in the character string is irrelevant.

Format selection

Via the “format” input parameter, you select, whether the read in character string is specified in the international (ISO 8601) or traditional (DE) data format.

1. **International format (ISO 8601)**
   If the “format” input parameter has not been set (format = FALSE), the date in the character string is read in in the international format.
   Example: inString = `2016-03-16 13:34:12.001`
   format = FALSE
   outDTL = DTL#2016-03-16-13:34:12.001000

   Figure 2-10 shows the position of the individual characters in the character string.

Figure 3-42: Position of the individual characters

	Format																												
inString	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

2. **Traditional format (DE)**
   If the “format” input parameter has been set (format = TRUE), the date in the character string is read in in the traditional format.
   Example: inString = ‘16/03/2016 13:34:12.1‘
   format = TRUE
   outDTL = DTL#2016-03-16-13:34:12.100000

   The following figure shows the position of the individual characters in the character string.

Figure 3-43: Position of the individual characters

	Format																												
inString	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

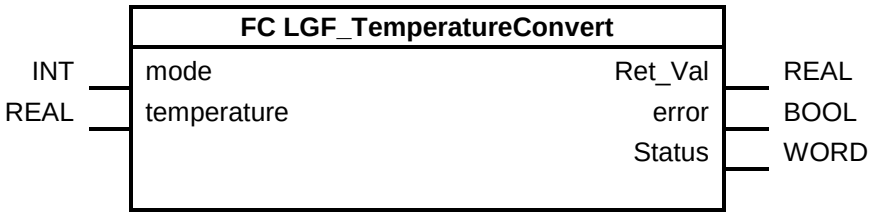
3.6.7 FC LGF_TemperatureConvert

Short description

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

Block

Figure 3-44: FC LGF_TemperatureConvert



Input parameters

Table 3-90: Input parameter

Parameter	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
temperature	REAL	Temperature to be converted

Output parameters

Table 3-91: Output parameter

Parameter	Data type	Description
Ret_Val	REAL	Converted temperature
error	BOOL	0: no errors 1: Block errors
status	WORD	“status” returns the status/error code (see table below).

Status and error displays

Table 3-92: Status/error codes

Status	Meaning	Remedy/notes
16#7000	Initial value	-
16#0000	No errors	-
16#8200	Incorrect mode at “mode” input.	See description of input parameters

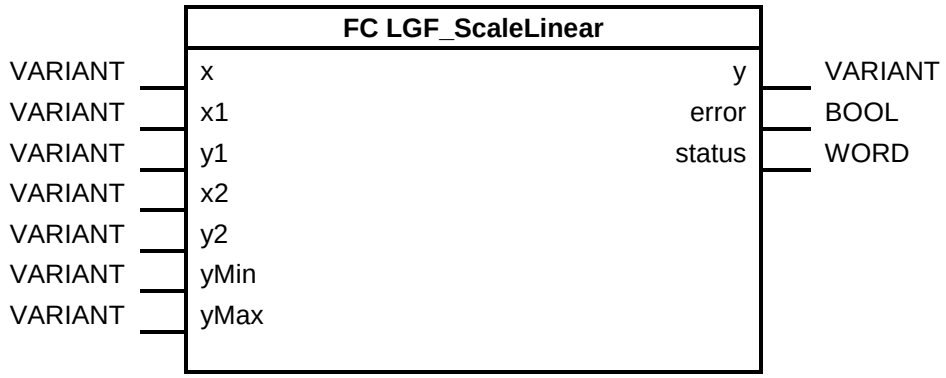
3.6.8 FC LGF_ScaleLinear

Short description

This function scales an input value via a linear function.

Block

Figure 3-45: FC LGF_ScaleLinear



Input parameters

Table 3-93: Input parameter

Parameter	Data type	Description
x	VARIANT	Input value that is to be scaled.
x1	VARIANT	Point 1 (P_1) of the linear function.
y1	VARIANT	
x2	VARIANT	Point 2 (P_2) of the linear function.
y2	VARIANT	
yMin	VARIANT	Lower limit value of the output.
yMax	VARIANT	Higher limit value of the output.

Output parameters

Table 3-94: Output parameter

Parameter	Data type	Description
y	VARIANT	Output value, scaled.
error	BOOL	Error display. 0: no error. 1: Block error, “status” returns error code.
status	WORD	“status” returns the status/error code (see table below).

Status and error displays

Table 3-95: Status and error codes

Status	Meaning	Remedy
16#0000	No errors.	-
16#8200	Actual parameter at the “x” input is of the wrong data type.	The “x” input must only be configured with the INT or REAL data type.
16#8201	Actual parameter at the “y” output is of the	The “y” output must only be configured with

Status	Meaning	Remedy
	wrong data type.	the INT or REAL data type.
16#8202	Actual parameters at the “x”, “x1” and “x2” inputs are of different data types.	All x parameters must either be INT or REAL.
16#8203	Actual parameters at the “y1”, “y2”, “yMin”, “yMax” inputs and at the “y” output are of different data types.	All y parameters must either be INT or REAL.
16#8204	Lower limit value “yMin” is greater than higher limit value “yMax”.	Select the lower limit value below the higher limit value.

Mode of Operation

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

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

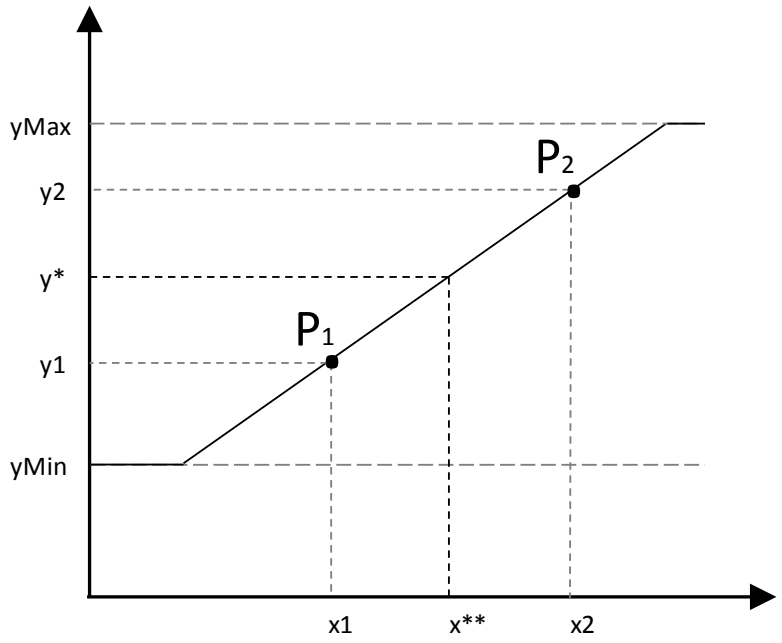
$$y = \frac{y_2 - y_1}{x_2 - x_1} \cdot (x - x_1) + y_1$$

The straight line is described by two points, P_1 and P_2. By way of a Cartesian coordinate system, the points are specified via x and y coordinates.

Note If the values of the “x1” and “x2” parameters are identical, the value of “y1” is returned at the “y” output.

By specifying “yMin” and “yMax”, you can set up a lower and upper limit for the calculated “y” value. You therefore avoid overdrive and underdrive ranges.

Figure 3-46: Scaling



You can configure the formal parameters with the following data types:

Table 3-96: Data types

Formal parameter / Scaling	x, x1, x2	y, y1, y2, yMin, yMax
	Data type	
INT → INT	INT	INT
INT → REAL	INT	REAL
REAL → INT	REAL	INT
REAL → REAL	REAL	REAL

Example

A signal of 4 to 20mA is applied to an analog input module. This signal is converted to the CPU-internal value of to 27648, in order to measure a filling level. In this, 0 corresponds to a filling level of 0.0m and 27648 a filling level of 1.7m.

You then need to configure the block as follows:

- x1 = 0; y1 = 0,0 (P1)
- x2 = +27648; y2 = 1,7 (P2)
- yMin = 0.0
- yMax = 1.7

3.6.9 FC LGF_StringToTaddr

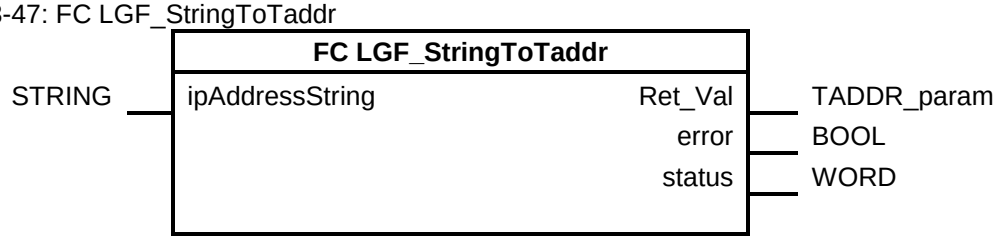
Short description

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

The LGF_StringToTaddr function converts a variable of the “String” data type into a variable of the “TADDR_Param” system data type.

Block

Figure 3-47: FC LGF_StringToTaddr



Input parameters

Table 3-97: Input parameter

Parameter	Data type	Description
ipAddressString	STRING	IPV4 address

Output parameters

Table 3-98: Output parameter

Parameter	Data type	Description
Ret_Val	TADDR_param	IPV4 address
error	BOOL	0: no errors 1: Block errors
status	WORD	“status” returns the status/error code (see table below).

Status and error displays

Table 3-99: Status/error codes

Status	Meaning	Remedy/notes
16#0000	No errors	-
16#8101	False value in first Range of the IP address	Check the IP address at the input
16#8102	False value in second Range of the the IP address	
16#8103	False value in third Range of the IP address	
16#8104	False value in fourth Range of the the IP address	
16#8105	False value in port range of the IP address	

Mode of Operation

The function converts the IPV4 address with or without port number from the “STRING” data type to “TADDR_param”. The string must correspond to 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 enter a port number in the “ipAddressString” parameter, the “Ret_Val.REM_PORT_NR” parameter outputs the value “0”.

3.6.10 FC LGF_TaddrToString

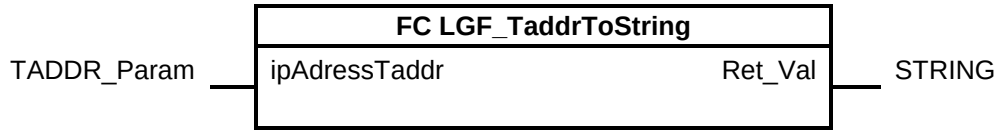
Short description

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

The LGF_TaddrToString function converts a variable of the “TADDR_param” system data type into a variable of the “String” data type.

Block

Figure 3-48: FC LGF_TaddrToString



Input parameters

Table 3-100: Input parameter

Parameter	Data type	Description
ipAdressTaddr	TADDR_Param	IPV4 address

Output parameters

Table 3-101: Output parameter

Parameter	Data type	Description
Ret_Val	STRING	IPV4 address

Mode of Operation

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

Example

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

3.7 Signal generators

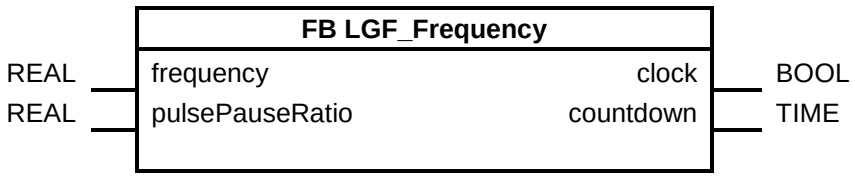
3.7.1 FB LGF_Frequency

Short description

The block generates a signal which, depending on a defined frequency and a pulse-to-pause ratio, switches between the values “0” and “1”.

Block

Figure 3-49: FB LGF_Frequency



Input parameters

Table 3-102: Input parameter

Parameter	Data type	Description
frequency	REAL	Clock frequency in Hz
pulsePauseRatio	REAL	Pulse-pause ratio (Default: 1.0 equals 1:1)

Output parameters

Table 3-103: Output parameter

Parameter	Data type	Description
clock	BOOL	Output switches with defined frequency
countdown	TIME	Remaining time of current “clock” status

Mode of Operation

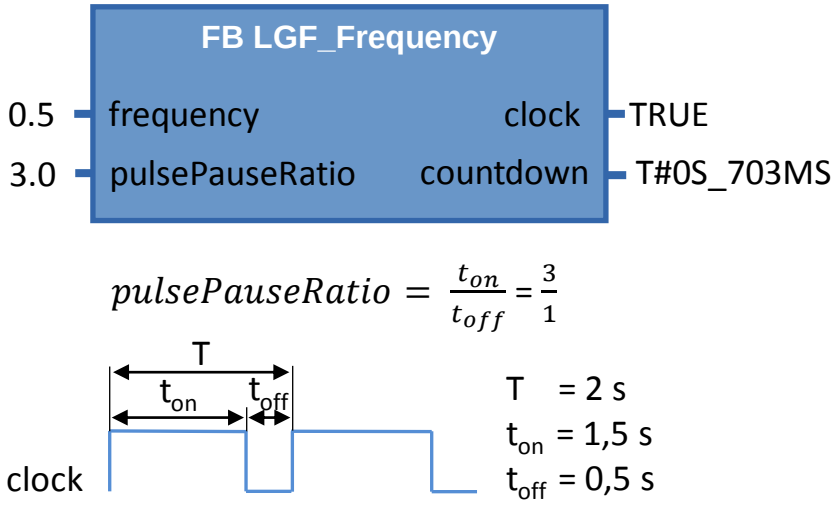
The “clock” output is a Boolean value that toggles in the desired frequency. Via the “pulsePauseRatio“ input, the pulse-pause ratio is set.

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

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

Example

Figure 3-50: FB LGF_Frequency



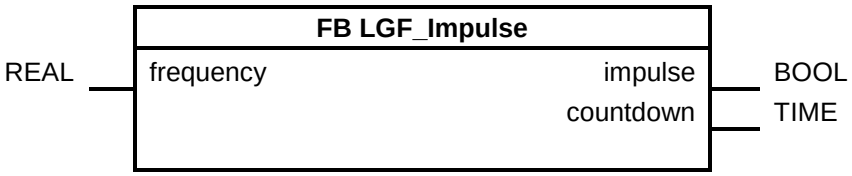
3.7.2 FB LGF_Impulse

Short description

This block generates impulses in a predefined frequency. The impulse is always available for a (controller) cycle.

Block

Figure 3-51: FB LGF_Impulse



Input parameters

Table 3-104: Input parameter

Parameter	Data type	Description
frequency	REAL	Clock frequency in Hz

Output parameters

Table 3-105: Output parameter

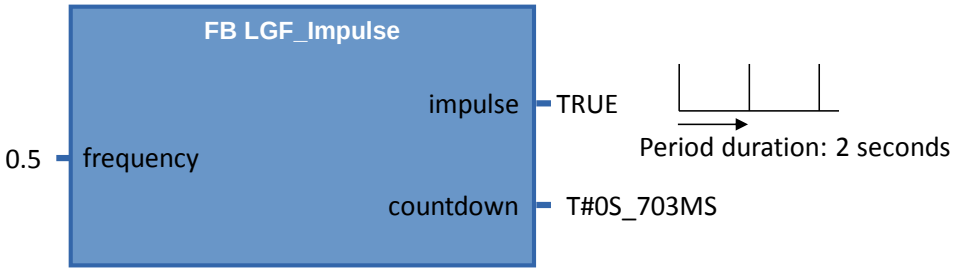
Parameter	Data type	Description
impulse	BOOL	Signal with impulses
countdown	TIME	Time until next impulse

Mode of Operation

The block generates impulses at the “impulse” output with the frequency “frequency”. The block always begins with an impuls and sets the following impulse after the period time has elapsed.

Example

Figure 3-52: Example



Note

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

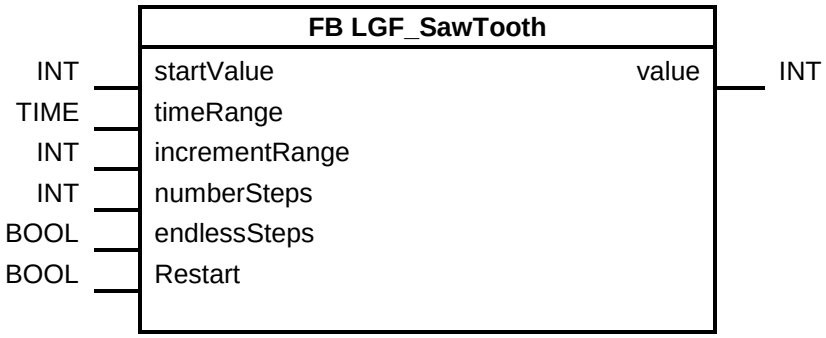
3.7.3 FB LGF_SawTooth

Short description

This block generates a sawtooth-shaped signal path. Every sawtooth consists of a defined number of steps (increments).

Block

Figure 3-53: FB LGF_SawTooth



Input parameters

Table 3-106: Input parameter

Parameter	Data type	Description
startValue	INT	Start value at the the signal starts.
timeRange	TIME	Time, after which the “value” output parameter is incremented
incrementRange	INT	Size of jump from one increment to the next.
numberSteps	INT	Number of increments per sawtooth. (In case of an endless sawtooth-signal, this specification is not required).
endlessSteps	BOOL	Specification, whether an endless sawtooth-signal shall be generated.
restart	BOOL	Sawtooth-signal restarts at start value “start value”.

Note Note that changes to the input parameters only become effective upon “restart”.

Output parameters

Table 3-107: Output parameter

Parameter	Data type	Description
value	INT	Current value of the sawtooth-signal.

Mode of Operation

The block calculates the values for a sawtooth-shaped signal path which are output at the “value” output parameter. The signal starts with the start value “startValue”

and is added up with the “increment” value every time the “timeRange” time interval expires. The value can also be negative.

If the “endlessSteps” tag is set to “FALSE”, the number of summation steps is counted. If it exceeds the “numberSteps” value, then the output parameter “value” is reset to the start value. A new sawtooth starts.

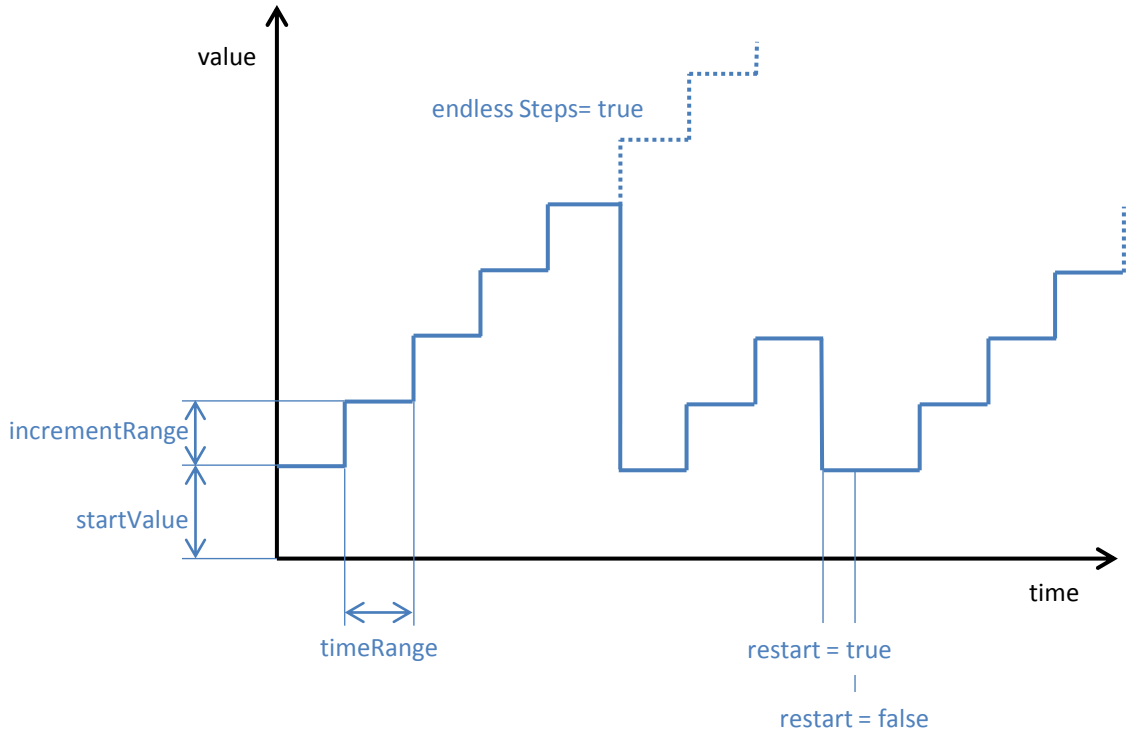
If the “endlessSteps” tag is set to “TRUE”, then - once beginning at “startValue” - the “increment” value is continuously added up. If the maximum positive INT value range (32767) of the “value” output parameter is exceeded, “value” switches to the maximum negative INT value range (-32768) and is continuously added up.

Note

The duration of a sawtooth with “endlessSteps” on “FALSE” is calculated as follows:

Duration = „timeRange * („numberSteps“ + 1)

Figure 3-54: Signal path of “value” output



3.8 Technology operations

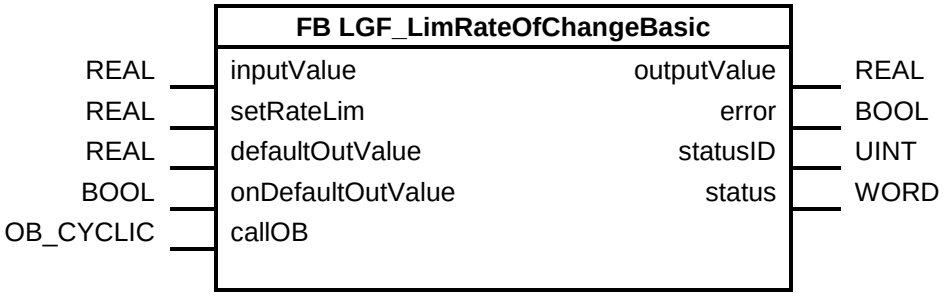
3.8.1 FB LGF_LimRateOfChangeBasic

Short description

This block limits the rate of change of an input value. A jump function turns into a ramp function.

Block

Figure 3-55: FB LGF_LimRateOfChangeBasic



Input parameters

Table 3-108: Input parameter

Parameter	Data type	Description
inputValue	REAL	Input value (jump function)
setRateLim	REAL	Rate of change of ramp function (1/second).
onDefaultOutValue	BOOL	Preset output value
dafaultOutValue	REAL	Value for preset of output value
callOB	OB_CYCLIC	Calling cyclic interrupt OB.

Output parameters

Table 3-109: Output parameter

Parameter	Data type	Description
outputValue	REAL	output quantity
error	BOOL	0: no errors 1: Block error, “statusID” returns error source, “status” returns error code.
statusID	UINT	“statusID” returns the ID of the block reporting the status. See table below.
status	WORD	“status” returns the status/error code (see table below)

Status and error displays

Table 3-110: Status/error codes

statusID	status	Meaning	Remedy/notes
1	16#0000	No errors	-
1	16#7000	Initial value	“onDefaultOutValue“ active
1	16#8200	OB at “callOB” input is not configured/present.	At the “callOB” input, connect a configured/present cyclic interrupt OB.
1	16#8201	Negative rate of change.	The parameter for the rate of change must not be negative.
2	-	Error/status of subordinate function “QRY_CINT” while querying the cyclic interrupt parameters.	Possible cause: OB at “callOB“ input is of the wrong type.

Note If “statusID” is > 1, all values of the “status” output came directly from called up instructions (see table on output parameters). In this case, get the information on the respective instructions from the TIA Portal Online Help.

Mode of Operation

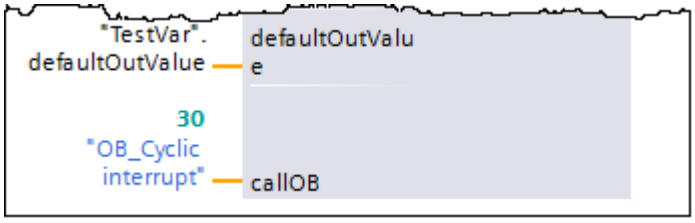
The ramp is a limitation line and refers to a change of rate per second; if “setRateLim = 10.0“ is parameterized, for example, then at a sampling time of 1s/100ms/10ms and if “inputValue > outputValue“ is set, 10.0/1.0/0.1 are added to “outputValue“ during each block call, until “inputValue” has been reached.

The limitation of the change of rate is valid for the increase and the decrease in both, the positive and the negative value range.

The “outputValue” output can be preset or initialized.

The time interval of the calling cycle interrupt OB is detected by connecting the calling cycle interrupt OB at the “callOB” input parameter.

Figure 3-56: Connect cyclic interrupt OB

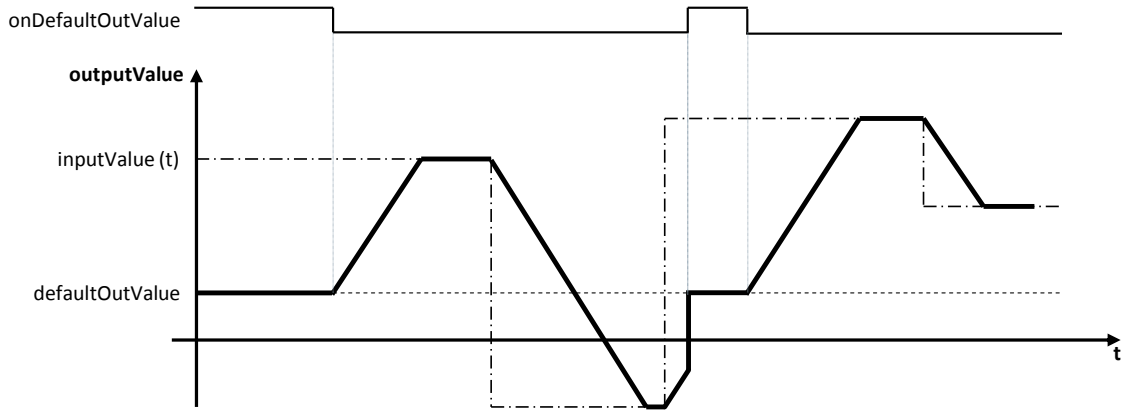


Presetting the output

If “onDefaultOutValue = TRUE“ is set, then “defaultOutValue” is the active output; upon switching from TRUE to FALSE, “outputValue” is run as a ramp from “outputValue” to “inputValue”. Upon switching from FALSE to TRUE, the “outputValue” output immediately switches to “defaultOutValue”.

Function characteristics

Figure 3-57: Ramp function growth



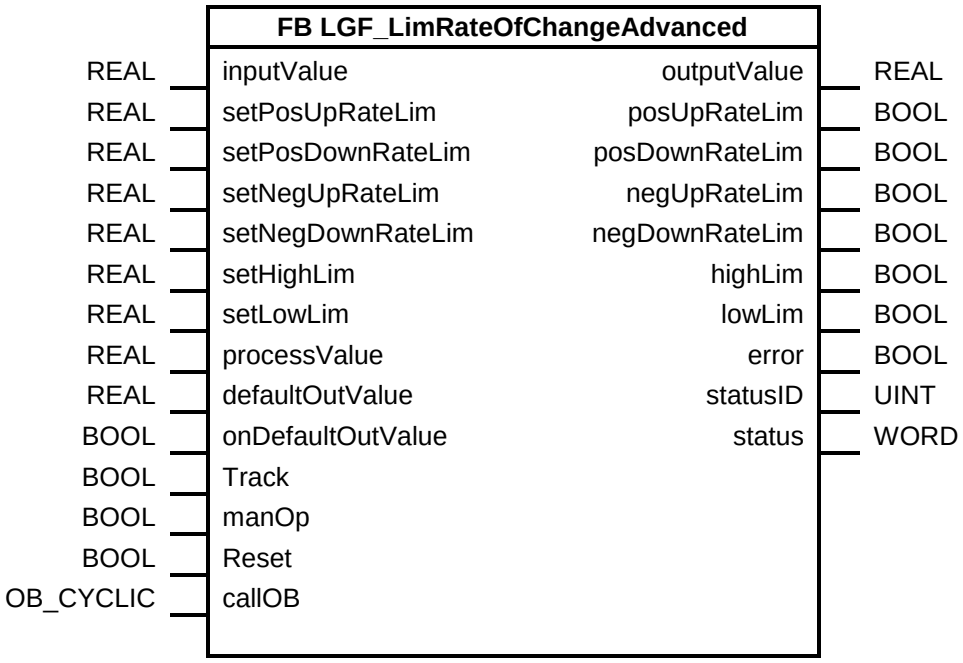
3.8.2 FB LGF_LimRateOfChangeAdvanced

Short description

The LGF_LimRateOfChangeAdvanced block limits the rate of change of an input value. Jumpf functions turn to ramp functions. Additionally, the block has several operating modes.

Block

Figure 3-58: FB LGF_LimRateOfChangeAdvanced



Input parameters

Table 3-111: Input parameter

Parameter	Data type	Description
inputValue	REAL	Input value (jump function)
setPosUpRateLim	REAL	Rate of change per second for up ramp in positive value range
setPosDownRateLim	REAL	Rate of change per second for down ramp in positive value range
setNegUpRateLim	REAL	Rate of change per second for up ramp in negative value range
setNegDownRateLim	REAL	Rate of change per second for down ramp in negative value range
setHighLim	REAL	Upper limit
setLowLim	REAL	Lower limit
processValue	REAL	Process value
defaultOutValue	REAL	Value for preset of output value
onDefaultOutValue	BOOL	Preset output value
track	BOOL	Switching input value (tracking)

Parameter	Data type	Description
manOp	BOOL	Switching process value
reset	BOOL	Restart
callOB	OB_CYCLIC	Calling cyclic interrupt OB.

Output parameters

Table 3-112: Output parameter

Parameter	Data type	Description
outputValue	REAL	output quantity
posUpRateLim	BOOL	Up rate limitation in positive area switched on
posDownRateLim	BOOL	Down rate limitation in positive area switched on
negUpRateLim	BOOL	Up rate limitation in negative area switched on
negDownRateLim	BOOL	Down rate limitation in negative area switched on
highLim	BOOL	Upper limit switched on
lowLim	BOOL	Lower limit switched on
error	BOOL	0: no errors 1: Block error, “statusID” returns error source, “status” returns error code.
statusID	UINT	“statusID” returns the ID of the block reporting the status. See table below
status	WORD	“status” returns the status/error code (see table below)

Status and error displays

Table 3-113: Status/error codes

statusID	status	Meaning	Remedy/notes
1	16#0000	No errors	-
1	16#8200	OB at “callOB” input is not configured/present.	At the “callOB” input, connect a configured/present cyclic interrupt OB.
1	16#8201	“setHighLim“ < “setLowLim“	Upper limit “setHighLim” needs to be greater than the lower limit “setLowLim”.
1	16#8202	Negative rate of change.	The parameters for the rate of change must only be >= 0.0.
2	-	Error/status of subordinate function “QRY_CINT” while querying the cyclic interrupt parameters.	Possible cause: OB at “callOB“ input is of the wrong type.

Note If “statusID” is > 1, all values of the “status” output came directly from called up instructions (see table on output parameters). In this case, get the information on the respective instructions from the TIA Portal Online Help.

Mode of Operation

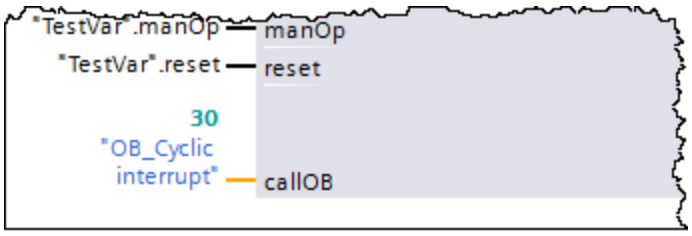
For the positive/negative value range, two rates of change are parameterizable for the ramp (rising and falling values). Via controller inputs, the following operating modes can be selected:

- Restart
- Presetting the output
- Normal operation (automatic)
- Tracking
- Switching process value (manual)

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

The time interval of the calling cycle interrupt OB is detected by connecting the calling cycle interrupt OB at the “callOB” input parameter.

Figure 3-59: Connect cyclic interrupt OB



Restart

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

Presetting the output

If “onDefaultOutValue = TRUE“ is set, then “defaultOutValue” is the active output; upon switching from TRUE to FALSE, “outputValue” is run as a ramp from “outputValue” to “inputValue”. Upon switching from FALSE to TRUE, the “outputValue” output immediately switches to “defaultOutValue”.

Normal operation

The ramps are a limitation lines and refer to a change of rate per second; if “setPosUpRateLim = 10.0“ is parameterized, for example, then at a sampling time of 1s/100ms/10ms and if “inputValue > outputValue“ is set, 10.0/1.0/0.1 are added to “outputValue“ during each block call, until “inputValue” has been reached.

The limitation of the change of rate is parameterizable for the rise and the fall in both, the positive and the negative value range.

Table 3-114: Marking of ramps

Parameter	Ramp
setPosUpRateLim	outputValue > 0 and |outputValue| ascending
setPosDownRateLim	outputValue > 0 and |outputValue| descending
setNegUpRateLim	outputValue < 0 and |outputValue| ascending
setNegDownRateLim	outputValue < 0 and |outputValue| descending

If the ramps are not parameterized (”setPosUpRateLim“, “setPosDownRateLim“, “setNegUpRateLim“ and “setNegDownRateLim“ are 0.0), then the output remains at a value of 0.0 and normal operation is not functioning.

Tracking

If the input “track = TRUE“ is set, the input value “inputValue” is switched directly to the output value “outputValue”. Thus, jumps of the input value are also output.

Switching process value

If “manOp = TRUE“ is set, the process value “processValue” is switched directly to the output value “outputValue”.

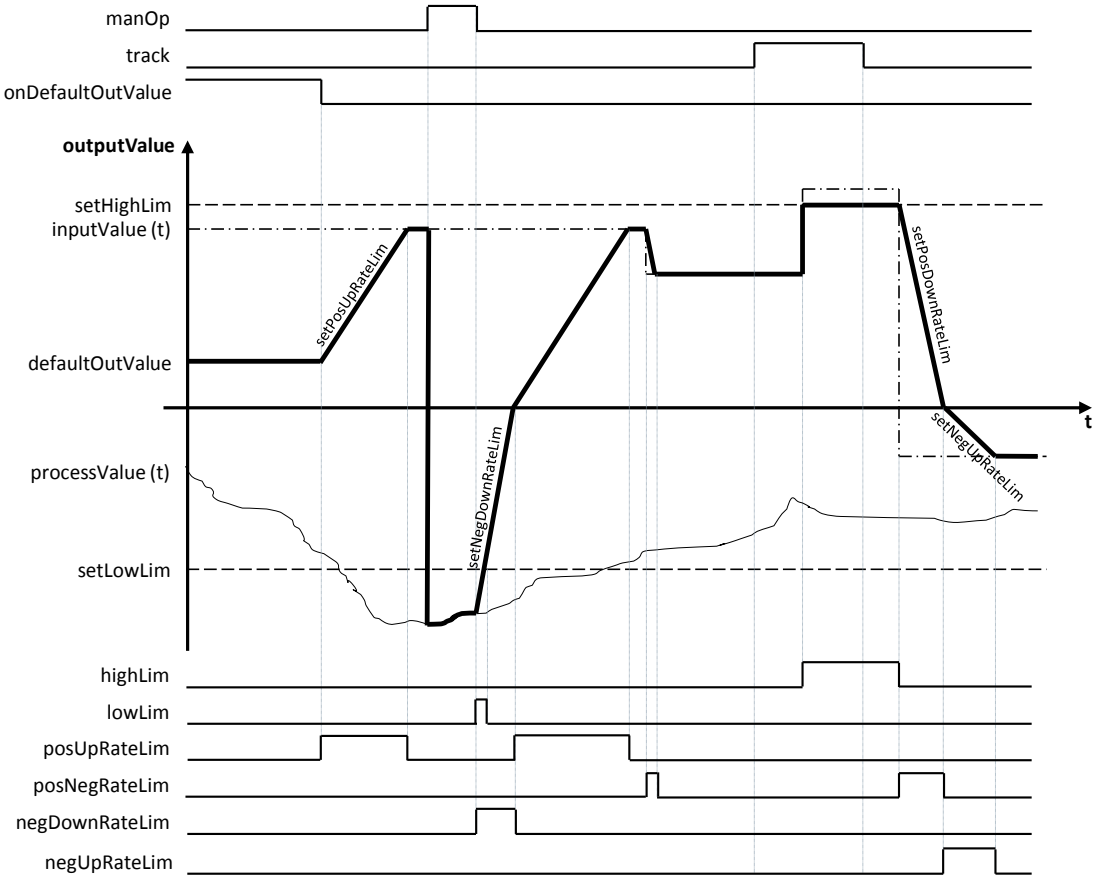
In this operating mode, the parameterization of the ramps or the upper/lower limits of the output value and the preset of the output are ineffective.

Upon switching from TRUE to FALSE, the output “outputValue” is once more run as a ramp to “inputValue”.

As soon as the value section between the lower and the upper limitation has been reached, the upper and lower limitation is reactivated.

Function characteristics

Figure 3-60: Ramp function, operating modes



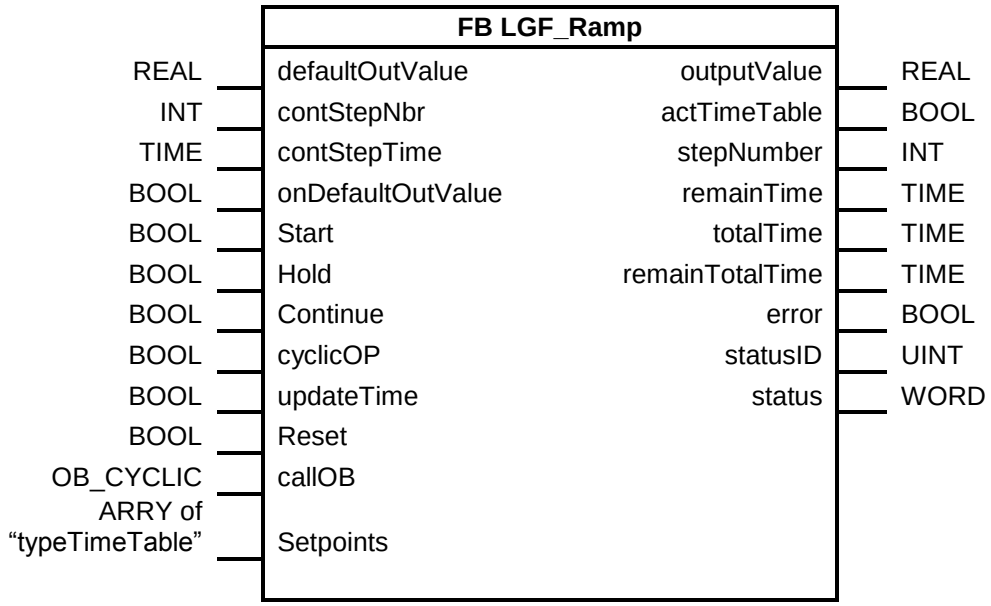
3.8.3 FB LGF_Ramp

Short description

This block generates a speed curve on the basis of a support point table. Between both points, there is a linear interpolation within the specified time.

Block

Table 3-115: FB LGF_Ramp



Input parameters

Table 3-116: Input parameter

Parameter	Data type	Description
defaultOutValue	REAL	Value for preset of output value
contStepNbr	INT	Number of the next support point to continue
contStepTime	TIME	Remaining time to continue until the support point "contStepNbr"
onDefaultOutValue	BOOL	Preset output quantity with "defaultOutValue"
start	BOOL	Track support point table
hold	BOOL	Hold current value at output
continue	BOOL	Continue
cyclicOP	BOOL	Repeat support table cyclically
updateTime	BOOL	Update time
reset	BOOL	Restart
callOB	OB_CYCLIC	Calling cyclic interrupt OB.
setpoints	ARRY of "typeTimeTable"	Support point table. Information on the "typeTimeTable" data type can be found under "Global data".

Output parameters

Table 3-117: Output parameter

Parameter	Data type	Description
outputValue	REAL	output quantity
actTimeTable	BOOL	Support point table is currently being processed.
stepNumber	INT	Current support point number (support point that is switched)
remainTime	TIME	Remaining time until the next support point is reached
totalTime	TIME	Total time
remainTotalTime	TIME	Total remaining time
error	BOOL	0: no errors 1: Block error, “statusID” returns error source, “status” returns error code.
statusID	UINT	“statusID” returns the ID of the block reporting the status. See table below.
status	WORD	“status” returns the status/error code (see table below).

Status and error displays

Table 3-118: Status/error codes

statusID	status	Meaning	Remedy/notes
1	16#0000	No errors	Job completed.
1	16#7000	Initial value	Restart has been performed.
1	16#7001	first call	Rising edge “start”.
1	16#7002	Subsequent call	“cyclicOP” input set.
1	16#8200	OB at “callOB” input is not configured/present.	At the “callOB” input, connect a configured/present cyclic interrupt OB.
1	16#8201	Lower array limit <> 0	The array with the support points needs to begin with the index 0.
2	-	Error/status of subordinate function “QRY_CINT” while querying the cyclic interrupt parameters.	Possible cause: OB at “callOB“ input is of the wrong type.

Note If “statusID” is > 1, all values of the “status” output came directly from called up instructions (see table on output parameters). In this case, get the information on the respective instructions from the TIA Portal Online Help.

Global data

Together with the block, you automatically receive the PLC data type “typeTimeTable” which comprises of the parameters “outVal” for the value of a support point and “time” for the time until the next support point is reached. The declaration is done in a one-dimensional array of the “typeTimeTable” data type, beginning with the index 0. The array is created in a global data block and is then transferred to the “LGF_Ramp” block.

Figure 3-61: Example for the declaration of the support 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 “time” parameter of the last support point needs to be configured with 0s, since there is no longer any subsequent support point.

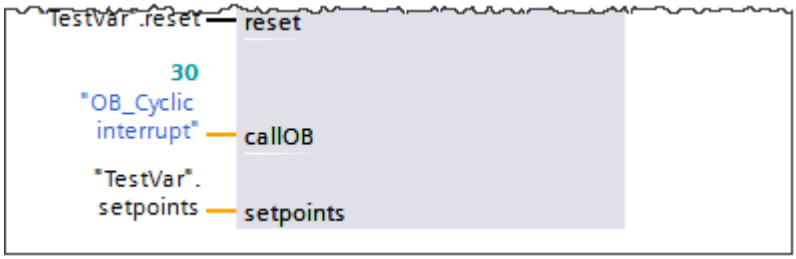
Mode of Operation

With each block, speed curves can be performed on the basis of configured support points; in each call cycle, values are output according to a time schedule and there is an interpolation between the support points.

In each cycle, the currently switched support point number “stepNumber”, the current remaining time “remainTime” until the support point has been reached, the total time “totalTime” and the total remaining time “remainTotalTime” until the speed curve end has been reached are output. Also, the “actTimeTable” output is set in the moment in which the configured speed curve is output.

The time interval of the calling cycle interrupt OB is detected by connecting the calling cycle interrupt OB at the “callOB” input parameter.

Figure 3-62: Connect cyclic interrupt OB



Via control inputs, the following operating modes can be selected:

- Restart
- Presetting the output
- Output speed curve
- Stop processing
- Specify processing step and time
- Switch to cyclic operation
- Update total time and remaining time

Overview of operating modes

Table 3-119: Overview of operating modes

Operating mode	onDefault OutValue	Start	hold	continue	cyclicOP	updateTime	reset	Output/action
Restart							TRUE ↑	Block is initialized.
Preset the output	TRUE	TRUE					FALSE	defaultOutValue
Output speed curve	FALSE	TRUE ↑	FALSE		FALSE		FALSE	outputValue(t); Final value is held after processing
Stop speed curve	FALSE	TRUE	TRUE	FALSE			FALSE	current value of outputValue(t) is held
Specify processing step and time	FALSE	TRUE	TRUE	TRUE ↑			FALSE	outputValue (alt)
			FALSE					Continue with configured support point
Switch to cyclic operation	FALSE	TRUE	FALSE		TRUE		FALSE	outputValue(t); automatic restart after end
Update total time and remaining time						TRUE ↑	FALSE	Total time and remaining time are updated.

Restart

A positive edge at the “reset” input resets the “outValve” output to 0.0. With “onDefaultOutValue” = TRUE, “defaultOutValue” is output at the output. The total time and total remaining time is updated and output at the output.

Preset the output

If the speed curve shall begin with a specific output value, then “onDefaultOutValue” needs to be = TRUE. In this case, the value “defaultOutValue” is applied to the output of the timer. All the while, the internal processing of the speed curve continues. If “onDefaultOutValue” switches to FALSE gain, it will be interpolated to the current active support point.

Output speed curve

With a rising edge at the “start” input, the speed curve will be output - as long as “start” is TRUE or until the speed curve has been finished by reaching the last support point. With another rising edge, the speed curve is output again. Additionally, the total time is updated at each start-up.

Switch to cyclic operation

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

There is no interpolation between the last support point value and the starting point. For a bumpless transition, the following needs to apply: last support point value = starting point.

Stop speed curve

With “hold” = TRUE, the value of the output quantity (including the processing time) is frozen. If “hold” is reset to = FALSE, it will be continued at the point of interruption or at a configured point (see “Specify processing step and time”). The processing time of the speed curve is extended by the hold time “T1*” (see **Fehler! Verweisquelle konnte nicht gefunden werden.**).

Specify processing step and time

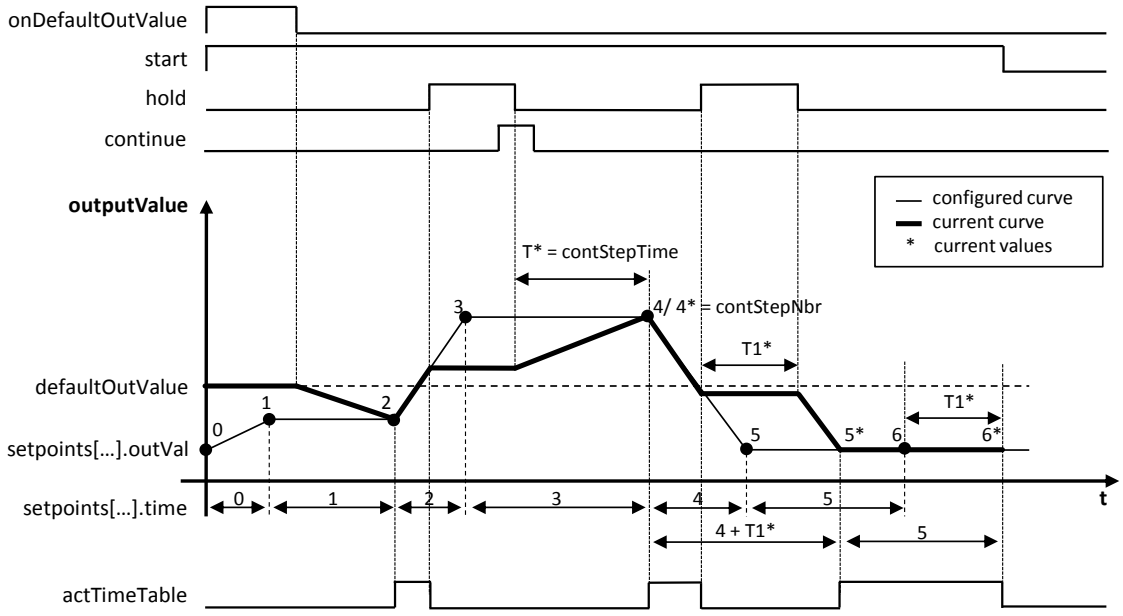
If during the interruption of the speed curve (“hold” = TRUE), the “continue” input parameter for the continuation is set to TRUE, then, after resetting the “hold” input, the support point number “contStepTime” (destination support point) is switched within the “contStepTime” time period (interpolation). The total remaining time will be recalculated.

Update total time and total remaining time

If support point values are changed, the total time and the total remaining time of the speed curve may change. Since for many support points, a calculation of “totalTime” and “remainTotalTime” may significantly increase the processing time of the function block, it will only be performed once, with a rising edge at the “updateTime” input.

Function characteristics

Figure 3-63: Functional sequences



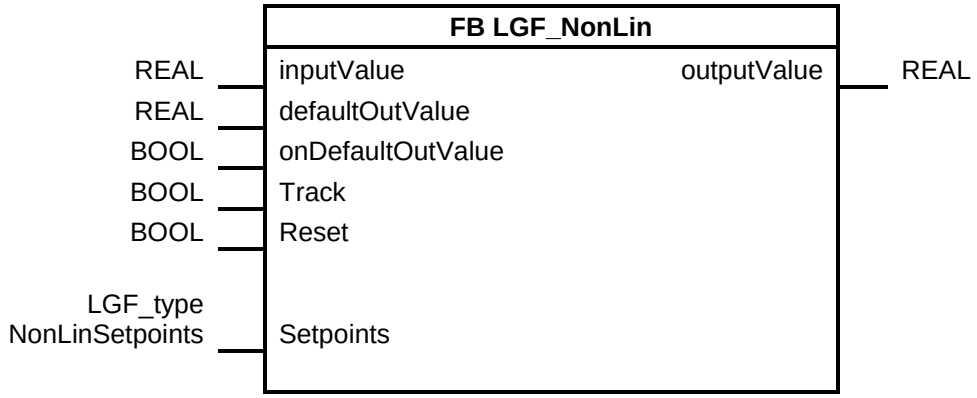
3.8.4 FB LGF_NonLin

Short description

This block realizes a characteristic. The characteristic is specified via a support point table with a linear interpolation between the support points. In every cycle, a specified input value generates an output value, using the characteristic from the support point table.

Block

Figure 3-64: FB LGF_NonLin



Input parameters

Table 3-120: Input parameter

Parameter	Data type	Description
inputValue	REAL	Input value for the calculation of the output value via the defined characteristic.
defaultOutValue	REAL	Default output value without the use of the characteristic
onDefaultOutValue	BOOL	Enabling the default output value The default output value is output for as long as this input is set.
track	BOOL	The value of the “outputValue” output follows the value of the “inputValue” value without the use of the characteristic, as long as this input is set.
reset	BOOL	If during operation, the support point table is changed, the “reset” input needs to be activated subsequently. Otherwise, the block may not function properly.

Input/Output parameters (InOut)

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

Parameter	Data type	Description
setpoints	LGF_typeNonLinSetpoints	Support point table for the definition of the characteristic

Output parameters

Table 3-122: Output parameter

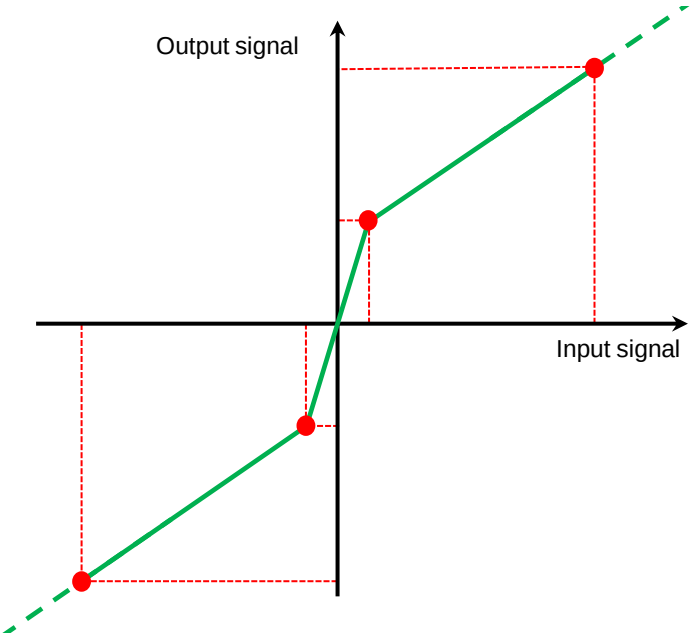
Parameter	Data type	Description
outputValue	REAL	The output value that has been calculated via the defined characteristic on the basis of the input value.

Mode of Operation

The value of the “outputValue” output is output on the basis of the following priorities:

1. As long as the “onDefaultOutValue” input is set, the value defined via the “defaultOutValue” parameter will be output as output value.
2. As long as the “reset” input is set, the block will be reset and the value of 0.0 will be output as output value.
3. As long as the “track” input is set, the input value is directly output as output value, without taking into account the characteristic.
4. On the basis of the input vaue, a characteristic value is calculated via the linearly interpolated support point table and output as output value.
   - If the input value is located between two support points within the support point table, the output value is calculated as intersection with the connection line between the previous and the next support point (see Figure 3-65).
   - If the input value is located before the first support point (lowest value that has been defined in the support point table), the output value is calculated as intersection of the line that is formed by the first two support points of the support point table.
   - If the input value is located after the last support point (highest value that has been defined in the support point table), the output value is calculated as intersection of the line that is formed by the last two support points of the support point table.

Figure 3-65: Exemplary course of the output signal



NOTICE	To keep the calculating time of the block as low as possible, no check of the configuration or the support point table data will be performed. When the support points are entered in the support point table, the following characteristics need to be observed. Otherwise, mafunction of the block may occur. • At least two support points need to be entered in the support point table. • The support points in the support point table need to be entered in ascending order of the input values in the table.

Interpolation point table

The support point table is realized by a tag of the Array data type. The type of the Array corresponds to the “LGF_typeNonLinSetpoint” PLC data type.

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

Example

Figure 3-66: Exemplary data block

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

4 Links & Literature

Table 4-1: Links & literature

	Topic
\1\	Siemens Industry Online Support http://support.automation.siemens.com
\2\	Download page of the entry 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\	Libraries with PLC data types (LPD) for STEP 7 (TIA Portal) and S7-1200 / S7-1500 https://support.industry.siemens.com/cs/ww/en/view/109482396

5 History

5.1 Library versioning

The library and library elements are maintained on basis of the following table:

Table 5-1: Definition of version

V	1.	2.	3
	Non-compatible changes	Compatible changes	Error correction
	• Reduction of interfaces • Modification of interfaces • Incompatible extension of functionality	• Extension of interfaces • Compatible extension of functionality	• Bugfix • Upgrade to new TIA Portal version

Versioning example

Table 5-2: 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	-	Error handling 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	-	Expansion at FB1
1.2.0	1.2.0	1.0.1	1.0.0	-	Expansion at FB1
2.0.0	2.0.0	1.0.1	2.0.0	-	New functionality at FB1 and FC1
2.0.1	2.0.0	1.0.2	2.0.0	-	Error handling of 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
…	…	…	…	…	…

5.2 Change log

Table 5-3: Change log

Version	Date	Modifications
V1.0.0	09/2015	First version
V1.0.1	10/2015	LGF_Astro V1.0.1 • T_ADD instruction 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
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 1.3 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 • systemTime and localTime outputs added FB LGF_TimerSwitch V1.1.0 • Two new modes: Weekday, weekend FB LGF_ShallSort… V1.1.0 • New mode: Sort in descending order FB LGF_Frequency V1.1.0 • New function: Pulse-pause ratio adjustable FB LGF_Impulse 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

Version	Date	Modifications
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: Supplementation in chapter 5.1 Library versioning FB LGF_PulseRelay V1.0.2 • Commentary correction FB LGF_Astro V1.1.4 • Code tuning FB LGF_SetTime V1.0.2 • Correction: FB Number: automatic FB LGF_FloatingAverage V1.1.0 • Code tuning • New input parameter “windowSize” FC LGF_MatrixAddition V2.0.0 • Code tuning • Input parameters changed to ARRAY* FC LGF_MatrixInverse V2.0.0 • Code tuning • Input parameters changed to ARRAY* FC LGF_MatrixMultiplication V2.0.0 • Code tuning • Input parameters changed to ARRAY* FC LGF_MatrixSubtraction V2.0.0 • Code tuning • Input parameters changed to ARRAY* FC LGF_MatrixTranspose V2.0.0 • Code tuning • Input parameters changed to ARRAY* FB LGF_Impulse V1.2.0 • Code tuning: No longer call of LGF_Frequency