UNI-PRO

DEVELOPMENT ENVIRONMENT FOR PROGRAMMABLE CONTROLLERS

STANDARD LIBRARIES MANUAL

CODE 114UPROSLE42

Important notice

This Instruction Manual should be read carefully before use, and all warnings should be observed; the Manual should then be kept for future reference.

Summary

1 GENERAL POINTS

This document lists the libraries provided with the UNI-PRO Development Environment program. The libraries in question may be differentiated into two main groups:

- Standard libraries
- System libraries

The standard libraries contain all the relevant details for the realisation of a generic project. They make available functions for performing calculations, conversions and a range of standard functions that may be useful in a general-purpose application.
The system libraries contain functions pertaining to the project and the hardware used. Effectively, they have the necessary features for controlling network communication (if in place), managing analogue/digital input/outputs, the controller memory for managing any logs and all the specifics relating to the UNI-PRO environment.

To aid understanding and clarify the methods of use, each library is provided with an example project. Please refer to the UNI-PRO folder “ *\...\Samples\Libraries\* ” to view this information.

For more detailed explanations of the software properties of the libraries, please refer to the development environment software manual.

For the particulars of the hardware please refer to the documentation relating to the instruments used.

2 SYSTEM LIBRARIES

2.1 Analogue Inputs

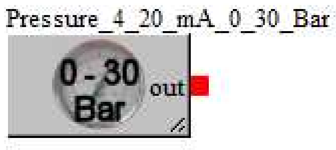
Humidity_4_20_mA

Outputs	*Type*	*Limits*	*Description*
Value	CJ_ANALOG	Value: 0..100 Error: 0..2	Returns the value of the relative humidity

Description
The *Humidity_4_20_mA* library uses a 4-20 mA analogue input and returns a linear current-humidity conversion as output. An output of between 0-100 (percentage points), proportional to the current measured by the sensor, is provided. The *Error* field is defined as: 0. No error 1. Sensor short circuit 2. Damaged or missing sensor

Notes
The library is template type in nature. The output value has an accuracy of one percentage unit.

Pressure_4_20_mA_0_30_Bar



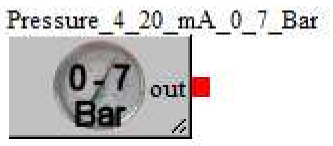
Outputs	*Type*	*Limits*	*Description*
Value	CJ_ANALOG	Value: 0..30 Error: 0..2	Returns the value of the relative pressure

Description

The *Pressure_4_20_mA_0_30_Bar* library uses a 4-20 mA analogue input and returns a linear current-pressure conversion as output. An output of 0-30 Bar, proportional to the current measured by the sensor, is provided. The *Error* field is defined as: 0. No error 1. Sensor short circuit 2. Damaged or missing sensor

Notes
The library is template type in nature. The output value has an accuracy of one tenth of a Bar.

Pressure_4_20_mA_0_7_Bar

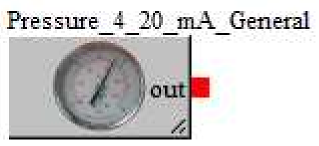


Outputs	***Type***	***Limits***	***Description***
Value	CJ_ANALOG	Value: 0..7 Error: 0..2	Returns the value of the relative pressure

Description
The *Pressure_4_20_mA_0_7_Bar* library uses a 4-20 mA analogue input and returns a linear current-pressure conversion as output. An output of 0-7 Bar, proportional to the current measured by the sensor, is provided. The *Error* field is defined as: 0. No error 1. Sensor short circuit 2. Damaged or missing sensor

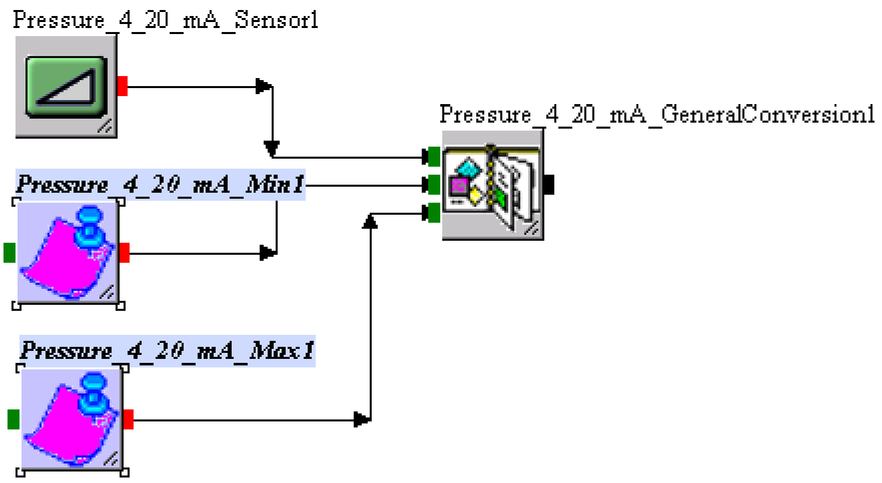
Notes
The library is template type in nature. The output value has an accuracy of one tenth of a Bar.

Pressure_4_20_mA_General



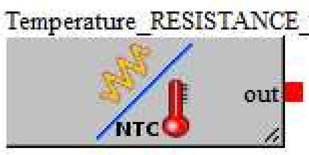
Outputs	*Type*	*Limits*	*Description*
Value	CJ_ANALOG	Value: min..max Error: 0..2	Returns the value of the relative pressure

Description
The *Pressure_4_20_mA_General* library uses a 4-20 mA analogue input and returns a linear current-pressure conversion as output. The output range is established by parameters within the library. To establish the desired range simply alter the *value* property of the two *Pressure_4_20_mA_Min1* and *Pressure_4_20_mA_Max1* parameters.



Notes
The library is template type in nature. The output value has an accuracy of one tenth of a Bar. To alter the decimal accuracy of the values, simply adjust the *precision* property of the parameters in the library.

Temperature_RESISTANCE_NTC



Output	*Type*	*Limits*	*Description*
out	CJ_ANALOG	-	"TEMPERATURE" probe

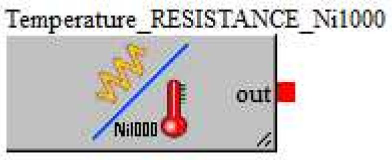
Description

The library allows transforming a “RESISTANCE” probe into a “TEMPERATURE” one.
On the inside, the standard EVCO NTC probe is traced, according to the following chart (-19.0°C / +42.0°C).

°C	OHM
-19.00	64400
-13.00	48500
-7.00	37000
-1.00	28400
5.00	22000
11.00	17200
18.00	13000
25.00	10000
32.00	7700
42.00	5400
49.00	4300

Notes
The library is a template library.

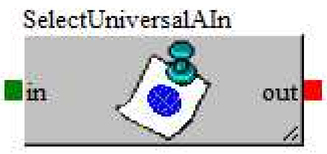
Temperature_RESISTANCE_Ni1000



Output	***Type***	***Limits***	***Description***
out	CJ_ANALOG	-	Converted probe “TEMPERATURE” Ni1000 DIN 43760

Description
The library allows to transforming a “RESISTANCE” probe into a “TEMPERATURE” one of type Ni1000 DIN 43760. It contains an ANALOG INPUT opportunally configurated, so dragging this library in the project will be automatically inserted an ANALOG INPUT that will be joined in Join Tool of Uni-Pro software.

SelectUniversalAIn



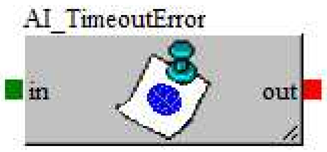
Input	*Type*	*Limits*	*Description*
in	CJ_BYTE	in: 1..10	Establishes the type of the nth analogue input

Output	*Type*	*Limits*	*Description*
out	CJ_BYTE	out: 1..10	Returns the type of analogue input set

Description
The *SelectUniversalAIn* library allows forcing the type of analogue input sensor *n* in the controller. The range of permitted input values for this library is 1-6, and each value identifies a particular sensor type. The association is the following: ➢ 1 : SENSOR_PTC. PTC sensor ➢ 2 : SENSOR_NTC. NTC sensor ➢ 3 : SENSOR_0_20mA. 0-20 mA current sensor ➢ 4 : SENSOR_4_20mA. 4-20 mA current sensor ➢ 5 : SENSOR_0_5V. 0-5 V voltage sensor ➢ 6 : SENSOR_0_10V. 0-10 V voltage sensor ➢ 7 : SENSOR_PT1000. PT1000 sensor ➢ 8 : NTC 10K2. NTC 10K2 probe ➢ 9 : NTC 10K3. NTC 10K3 probe ➢ 10 : RESISTANCE. value of the resi stance connected to the input (Ohm / 10) To set the default value, adjust the *value* property of the *Select_UniversalAIn* parameter in the library.

Notes
The library is template type in nature. Please refer to the hardware documentation to establish the controller's maximum number of universal inputs. In some cases, it may be that, besides this parameter, it is necessary to configure the analogue input by means of specific jumpers. Following each alteration, it is necessary to switch the power off and then on for the modifications to take effect. It is essential that these libraries be used once only per project.

AI_TimeoutError



Input	*Type*	*Limits*	*Description*
in	CJ_BYTE	1..255	Delay time detection errors probe

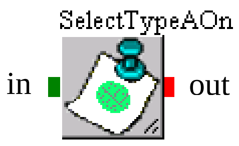
Output	*Type*	*Limits*	*Description*
out	CJ_BYTE	1..255	Delay time detection errors probe

Description
The library *AI_TimeoutError* allows to set a delay time for the detection of errors probe. To set the default value, adjust the *value* property of the *P_ErrorProbeTimeout* parameter present in the library.

Note
The library is template type in nature. It is essential that these libraries be used once only per project.

2.2 Analogue Outputs

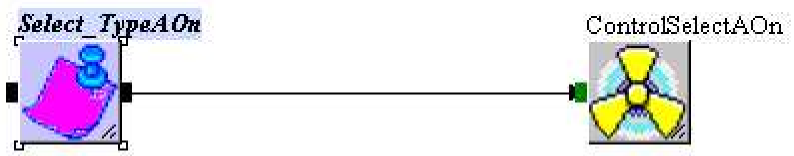
SelectTypeAOn



Input	*Type*	*Limits*	*Description*
In	CJ_BYTE	in: 0..6	Establishes the analogue output type

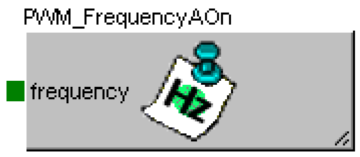
Output	*Type*	*Range*	*Description*
Out	CJ_BYTE	out: 0..6	Returns the type of analogue output set

Description
The *SelectTypeAOn* library allows setting the type of the nth analogue output. The range of permitted values for this library is 0-5, and each value identifies a particular output type. The number association is as follows: ➢ 0: FAN. Phase cut-off analogue output ➢ 1: 0-20mA. 0-20 mA current analogue output ➢ 2: 4-20mA. 4-20 mA current analogue output ➢ 3: 0-10V. 0-10 V voltage analogue output ➢ 4: PWM. Pwm analogue output ➢ 5: BELIMO. Analogue output to drive Belimo/MPBus modules ➢ 6: EEV_U. Analoque output to drive an unipolar valve To set the default value, adjust the *value* property of the *Select_TypeAOn* parameter present in the library.



Notes
The library is template type in nature. Please refer to the hardware documentation to establish the controller's maximum number of outputs. In some cases, it may be possible that, besides this parameter, it is necessary to configure the analogue output by means of specific jumpers. Following each alteration, it is necessary to switch the power off and then on for the modifications to take effect. It is essential that these libraries be used once only per project.

PWM_FrequencyAOn



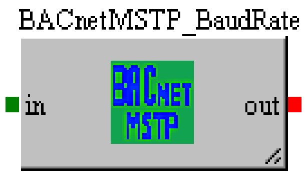
Input	*Type*	*Limits*	*Description*
frequency	CJ_WORD	in: 10..2000	It sets the cycle frequency (Hertz) of the analog output configured as PWM

Description
The library *PWM_FrequencyAOn* allows to set the cycle frequency of the analog output when this last is configured as PWM. The range of allowed values for this library is 10Hz.... 2,000 Hz; possible values outside this range will be rejected.

Notes
Look at the hardware documentation to learn the maximum number of outputs supported by the controller. It is necessary to use these libraries once a project.

2.3 BACnet MSTP

BACnetMSTP_BaudRate



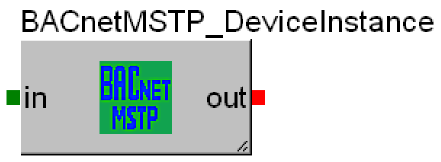
Input	*Type*	*Limits*	*Description*
in	CJ_BYTE	in: 0..3	Sets the baud rate

Output	*Type*	*Limits*	*Description*
out	CJ_BYTE	out: 0..3	Returns the baud rate

Description
The library allows setting the baud rate for communication over BACnet MSTP network. The following baud rates are supported: ➢ 0 : 9.6 Kbit ➢ 1 : 19.2 Kbit ➢ 2 : 38.4 Kbit ➢ 3 : 76.8 Kbit To set the default value, adjust the *value* property of the *P_BACnetMSTP_BaudRate* parameter in the library.

Notes
The library is template type in nature. It is essential that this library be used only once per project.

BACnetMSTP_DeviceInstance



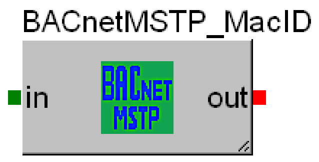
Input	***Type***	***Limits***	***Description***
in	CJ_LONG	in: 0.. 4194303	Sets the device identifier

Output	***Type***	***Limits***	***Description***
out	CJ_LONG	out: 0.. 4194303	Returns the device identifier

Description
The library allows setting the univocal identifier of the device in the BACnet network. To set the default value, adjust the *value* property of the *P_BACnetMSTP_DeviceInstance* parameter in the library.

Notes
The library is template type in nature. It is essential that this library be used only once per project.

BACnetMSTP_MacID



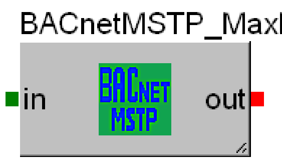
Input	***Type***	***Limits***	***Description***
in	CJ_BYTE	in: 1..127	Sets the MacID address

Output	***Type***	***Limits***	***Description***
out	CJ_BYTE	Out: 1..127	Returns the MacID address

Description
The library allows setting the physical address of the RS485 device in the BACnet MSTP network. To set the default value, adjust the *value* property of the *P_BACnetMSTP_MacID* parameter in the library.

Notes
The library is template type in nature. It is essential that this library be used only once per project.

BACnetMSTP_MaxMaster



Input	***Type***	***Limits***	***Description***
in	CJ_BYTE	in: 1..127	Sets the maximum address to which passes the token in the network.

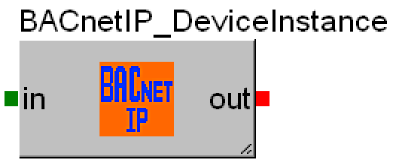
Output	***Type***	***Limits***	***Description***
out	CJ_BYTE	Out: 1..127	Returns the maximum address to which passes the token in the network.

Description
The library allows setting the maximum address to which passes the token in the BACnet communication, searching a new device in the network. To set the default value, adjust the *value* property of the *P_BACnetMSTP_MaxMaster* parameter in the library.

Notes
The library is template type in nature. It is essential that this library be used only once per project.

2.4 BACnet IP

BACnetIP_DeviceInstance



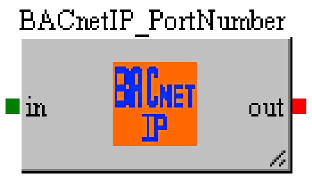
Input	*Type*	*Limits*	*Description*
in	CJ_LONG	in: 0.. 4194303	Sets the device identifier

Output	*Type*	*Limits*	*Description*
out	CJ_LONG	out: 0.. 4194303	Returns the device identifier

Description
The library allows setting the univocal identifier of the device in the BACnet network. To set the default value, adjust the *value* property of the *P_BACnetIP_DeviceInstance* parameter in the library.

Notes
The library is template type in nature. It is essential that this library be used only once per project.

BACnetIP_PortNumber



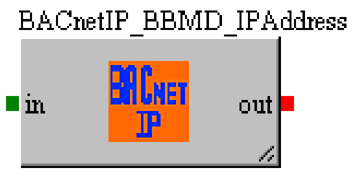
Input	*Type*	*Limits*	*Description*
in	CJ_WORD	47808 .. 47823 0xBAC0..0xBACF	Sets the BACnet IP port number

Output	*Type*	*Limits*	*Description*
out	CJ_WORD	47808 .. 47823 0xBAC0..0xBACF	Returns the BACnet IP port number

Description
The library allows setting the Port Number property used by the controller in the BACnet IP network. To set the default value, adjust the *value* property of the *P_BACnetIP_PortNumber* parameter in the library.

Notes
The library is template type in nature. It is essential that this library be used only once per project.

BACnetIP_BBMD_IPAddress



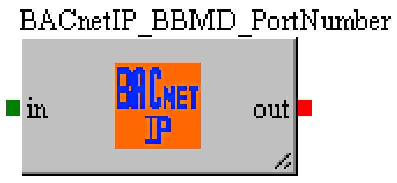
Input	***Type***	***Limits***	***Description***
in	CJ_IP_ADDRESS	0.0.0.0 255.255.255.255	Sets the IP address of the BBMD

Output	***Type***	***Limits***	***Description***
out	CJ_IP_ADDRESS	0.0.0.0 255.255.255.255	Returns the IP address of the BBMD

Description
The library allows setting the IP address of the BACnet Broadcast Management Device (BBMD) in the BACnet network. To set the default value, adjust the *value* property of the *P_BACnetIP_BBMDAddress* parameter in the library.

Notes
The library is template type in nature. It is essential that this library be used only once per project.

BACnetIP_BBMD_PortNumber



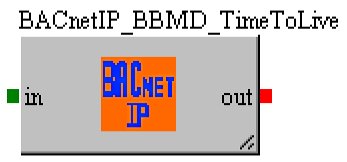
Input	*Type*	*Limits*	*Description*
in	CJ_WORD	47808 .. 47823 0xBAC0..0xBACF	Sets the port number of the BBMD

Output	*Type*	*Limits*	*Description*
out	CJ_WORD	47808 .. 47823 0xBAC0..0xBACF	Returns the port number of the BBMD

Description
The library allows setting property BBMD Port Number to which the device can communicate broadcast messages in the BACnet IP network. To set the default value, adjust the *value* property of the *P_ P_BACnetIP_BBMDPortNumber* parameter in the library.

Notes
The library is template type in nature. It is essential that this library be used only once per project.

BACnetIP_BBMD_TimeToLive



Input	*Type*	*Limits*	*Description*
in	CJ_WORD	15 .. 65535	Sets the timeout in seconds of the BBMD service

Output	*Type*	*Limits*	*Description*
out	CJ_WORD	15 .. 65535	Returns the timeout in seconds of the BBMD service

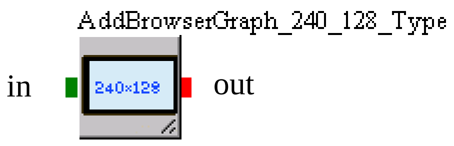
Description

The library allows setting the property BBMD Time To Live, the time in seconds during which the BBMD forwards the broadcast messages. To set the default value, adjust the *value* property of the *P_BACnetIP_BBMDTimeToLive* parameter in the library.

Notes
The library is template type in nature. It is essential that this library be used only once per project.

2.5 Browser

AddBrowserGraph_240_128_Type

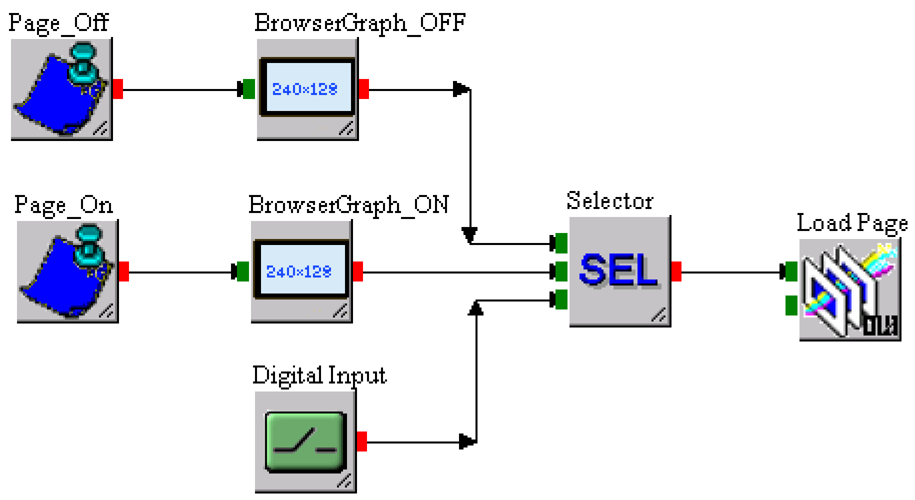


Input	***Type***	***Limits***	***Description***
In	CJ_WORD	1..240	Indicates the ID of the page which it is desired to process

Output	***Type***	***Limits***	***Description***
Out	CJ_WORD	0..65535	Returns the information necessary for processing

Description
Through this library, it is possible to add information on the type of browser necessary for processing the page with the identifying ID. Indeed, within a project, there may be pages with the same ID, created for different browsers. By means of the *AddBrowserGraph_240_128_Type* library, it is possible to select the 240 * 128 pixel graphic page and use it in the application.

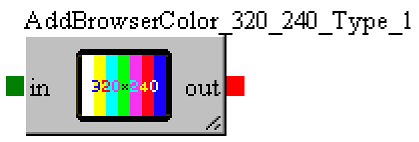
Notes
An example of the use of this library is presented below. Adjusting the Digital Input selects the page to be viewed. *Page_On* contains the ID of the on page, while *Page_Off* contains a different ID for the off page. If Digital Input = 1 select *Page_On*. If Digital Input = 0 select *Page_Off*. Upon receiving the trigger, the *LoadPage* command sends the command to load the page selected by the *Selector*.



Please refer to the example in the UNI-PRO *Samples* folder.

The subsequently explained libraries follow the same principle of use.

AddBrowserColor_320_240_Type

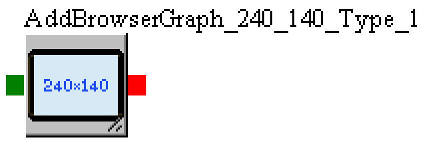


Input	*Type*	*Limits*	*Description*
in	CJ_WORD	1..240	Indicates the ID of the page which it is desired to process

Output	*Type*	*Limits*	*Description*
out	CJ_WORD	0..65535	Returns the information necessary for processing

Description
Through this library, it is possible to add information on the type of browser necessary for processing the page with the identifying ID. Indeed, within a project, there may be pages with the same ID, created for different browsers. By means of the *AddBrowserColor_320_240_Type* library, it is possible to select the 320 * 240 pixel graphic touch color page and use it in the application.

AddBrowserGraph_240_140_Type

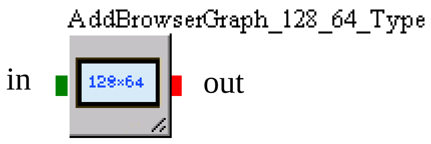


Input	*Type*	*Limits*	*Description*
in	CJ_WORD	1..240	Indicates the ID of the page which it is desired to process

Output	*Type*	*Limits*	*Description*
out	CJ_WORD	0..65535	Returns the information necessary for processing

Description
Through this library, it is possible to add information on the type of browser necessary for processing the page with the identifying ID. Indeed, within a project, there may be pages with the same ID, created for different browsers. By means of the *AddBrowserGraph_240_140_Type* library, it is possible to select the 240 * 140 pixel graphic page and use it in the application.

AddBrowserGraph_128_64_Type

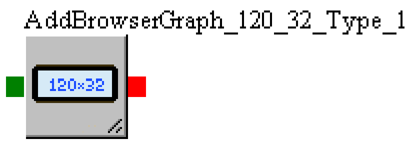


Input	*Type*	*Limits*	*Description*
in	CJ_WORD	1..240	Indicates the ID of the page which it is desired to process

Output	*Type*	*Limits*	*Description*
out	CJ_WORD	0..65535	Returns the information necessary for processing

Description
Through this library, it is possible to add information on the type of browser necessary for processing the page with the identifying ID. Indeed, within a project, there may be pages with the same ID, created for different browsers. By means of the *AddBrowserGraph_128_64_Type* library, it is possible to select the 128 * 64 pixel graphic page and use it in the application.

AddBrowserGraph_120_32_Type

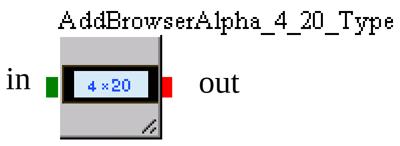


Input	*Type*	*Limits*	*Description*
in	CJ_WORD	1..240	Indicates the ID of the page which it is desired to process

Output	*Type*	*Limits*	*Description*
out	CJ_WORD	0..65535	Returns the information necessary for processing

Description
Through this library, it is possible to add information on the type of browser necessary for processing the page with the identifying ID. Indeed, within a project, there may be pages with the same ID, created for different browsers. By means of the *AddBrowserGraph_120_32_Type* library, it is possible to select the 120 * 32 pixel graphic page and use it in the application.

AddBrowserAlpha_4_20_Type



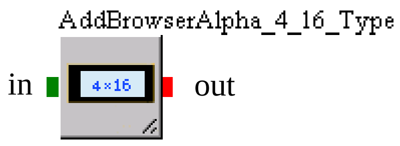
Input	*Type*	*Limits*	*Description*
in	CJ_WORD	1..240	Indicates the ID of the page which it is desired to process

Output	*Type*	*Limits*	*Description*
out	CJ_WORD	0..65535	Returns the information necessary for processing

Description

Through this library, it is possible to add information on the type of browser necessary for processing the page with the identifying ID. Indeed, within a project, there may be pages with the same ID, created for different browsers. By means of the *AddBrowserAlpha_4_20_Type* library, it is possible to select the 4 * 20 character alphanumeric page and use it in the application.

AddBrowserAlpha_4_16_Type

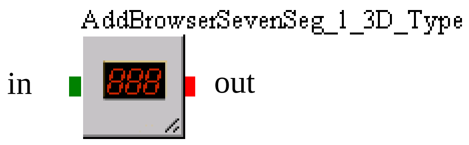


Input	*Type*	*Limits*	*Description*
in	CJ_WORD	1..240	Indicates the ID of the page which it is desired to process

Output	*Type*	*Limits*	*Description*
out	CJ_WORD	0..65535	Returns the information necessary for processing

Description
Through this library, it is possible to add information on the type of browser necessary for processing the page with the identifying ID. Indeed, within a project, there may be pages with the same ID, created for different browsers. By means of the *AddBrowserAlpha_4_16_Type* library, it is possible to select the 4 * 16 character alphanumeric page and use it in the application.

AddBrowserSevenSeg_1_3D_Type

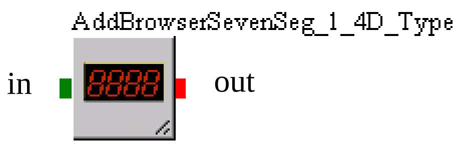


Input	*Type*	*Limits*	*Description*
in	CJ_WORD	1..240	Indicates the ID of the page which it is desired to process

Output	*Type*	*Limits*	*Description*
out	CJ_WORD	0..65535	Returns the information necessary for processing

Description
Through this library, it is possible to add information on the type of browser necessary for processing the page with the identifying ID. Indeed, within a project, there may be pages with the same ID, created for different browsers. By means of the *AddBrowserSevenSegAlpha_1_3D_Type* library, it is possible to select the seven segment three digit alphanumeric page and use it in the application.

AddBrowserSevenSeg_1_4D_Type



Input	***Type***	***Limits***	***Description***
in	CJ_WORD	1..240	Indicates the ID of the page which it is desired to process

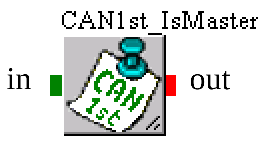
Output	***Type***	***Limits***	***Description***
out	CJ_WORD	0..65535	Returns the information necessary for processing

Description
Through this library, it is possible to add information on the type of browser necessary for processing the page with the identifying ID. Indeed, within a project, there may be pages with the same ID, created for different browsers. By means of the *AddBrowserSevenSegAlpha_1_4D_Type* library, it is possible to select the seven segment four digit alphanumeric page and use it in the application.

2.6 1st CAN & 2nd CAN

These libraries allow the configuration and management of the CAN communication protocol used by the hardware instruments. The UNI-PRO software foresees the use of up to two CAN channels and hence makes two libraries available, *1stCAN* and *2nd CAN*, referring to the first and second communications channels respectively.
It should be stated that not all hardware can potentially use the two CAN communication channels. Please refer to the specific controller documentation for further information on this topic.
This paragraph will explain the functionality of the libraries in relation to the first channel, the characteristics of the second communication channel are the same.

CAN1st_IsMaster



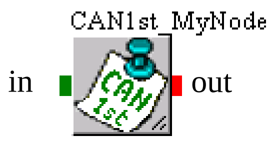
Input	*Type*	*Limits*	*Description*
in	CJ_BIT	0-1	Sets the node as network master

Output	*Type*	*Limits*	*Description*
out	CJ_BIT	0-1	Returns the master status

Description
The *CAN1st_IsMaster* library is used to establish whether the node where the project will be mounted is the network master In order to decide the default value, adjust the *value* property of the *CAN_1st_IsMaster* parameter in the library. If the parameter is '1' the node will be master, otherwise, if it is set to '0' the node will be slave.

Notes
The library is template type in nature. It is essential that this library be used only once per project. Following each alteration, it is necessary to switch the power off and then on for the modifications to take effect.

CAN1st_MyNode



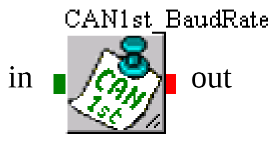
Input	*Type*	*Limits*	*Description*
in	CJ_BYTE	1..127	Sets the current node number

Output	*Type*	*Limits*	*Description*
out	CJ_BYTE	1..127	Returns the current node number

Description
The *CAN1st_MyNode* library represents the physical address of the controller where the program is loaded. In order to decide the default value, adjust the *value* property of the *CAN_1st_MyNode* parameter in the library.

Notes
The library is template type in nature. It is essential that this library be used only once per project. Following each alteration, it is necessary to switch the power off and then on for the modifications to take effect.

CAN1st_BaudRate



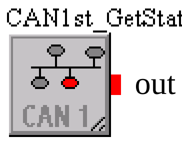
Input	*Type*	*Limits*	*Description*
in	CJ_ BYTE	1..4	Sets the baud rate of the first CAN communication channel

Output	*Type*	*Limits*	*Description*
out	CJ_BYTE	1..4	Returns the baud rate of the first CAN communication channel

Description
The *CAN1st_BaudRate* library allows setting the baud rate for communication over the first CAN channel. The following baud rates are supported: 1. 20 Kbit 2. 50 Kbit 3. 125 Kbit 4. 500 Kbit Set the default value by adjusting the *value* property of the *CAN_1st_BaudRate* parameter.

Notes
The library is template type in nature. It is essential that this library be used only once per project. Depending on the controller, the CAN channel baud rate may be set by means of a parameter, or by means of a jumper. Please refer to the controller hardware documentation. Following each alteration, it is necessary to switch the power off and then on for the modifications to take effect.

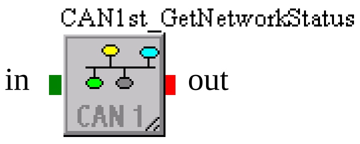
CAN1st_GetStatus



Input	*Type*	*Limits*	*Description*
out	CJ_SHORT	0..3	Returns the status of the communication channel

Description
The *CAN1st_GetStatus* library returns the status of the first CAN channel. The permitted output values are the following: ➢ 0 : CJ_CAN_OK. The channel is operating correctly ➢ 1 : CJ_CAN_WARNING. The channel is operational, but is in a state requiring attention ➢ 2 : CJ_CAN_ERROR. The channel has a fault and is only enabled for reception ➢ 3 : CJ_CAN_BUS_OFF. The channel has a serious fault and both transmission and reception are disabled.

CAN1st_GetNetworkStatus



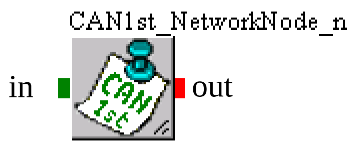
Input	*Type*	*Limits*	*Description*
in	CJ_BYTE	1..32	The number of the logical node whose status is to be verified

Output	*Type*	*Range*	*Description*
out	CJ_BYTE	0..3	Returns the status of the requested node

Description
The *CAN1st_GetNetworkStatus* library returns the status of the logical node requested by the input *in* and connected to the first CAN channel. The permitted output values are the following: ➢ 0 : CJ_CAN_OK. The node is operating correctly ➢ 1 : CJ_CAN_WARNING. The node is operational, but is in a state requiring attention ➢ 2 : CJ_CAN_ERROR. The node has a fault and is only enabled for reception ➢ 3 : CJ_CAN_BUS_OFF. The node has a serious fault and both transmission and reception are disabled.

Notes
This library may be used for each of the 32 potential network nodes, connected to the first CAN communication channel.

CAN1st_NetworkNode_n

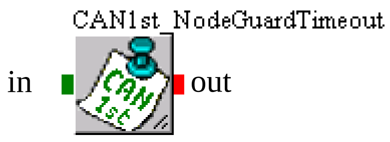


Input	***Type***	***Limits***	***Description***
in	CJ_BYTE	0..127	Sets the address of the network node
Output	***Type***	***Limits***	***Description***
out	CJ_BYTE	0..127	Returns the address of the network node

Description
The *CAN1st_NetworkNode_n* library represents the physical address of the nth node of the first channel of the network (n = 1..32). To decide the default value, adjust the *value* property of the *CAN_1st_NetworkNode_n* parameter in the library. Addresses may assume a value comprised of between 1 and 127 and must be unique for each channel. A value of 0 means that the node is not present.

Notes
The library is template type in nature. It is appropriate to use this library only once for each nth node.

CAN1st_NodeGuardTimeout



Input	***Type***	***Limits***	***Description***
in	CJ_BYTE	1..60	Establishes how often the master node checks the status of its slave nodes

Output	***Type***	***Limits***	***Description***
out	CJ_BYTE	1..60	Returns how often the master node checks the status of its slave nodes

Description
The *CAN1st_NodeGuardTimeout* library is used to set the CAN protocol *guarding* time. To set the default value, adjust the *value* property of the *CAN_1st_NodeGuardTimeout* parameter in the library. Every *value* time, the controller, if it is master, checks all the nodes connected to it as slaves and their relevant status using the *CAN1st_GetNetworkStatus* value. It is possible to read this status for each node of the network. The parameter values are seconds. For example, by setting a value of 5, all the nodes on the network will be interrogated by the master every 5 seconds.

Notes
It is essential that this library be used only once per project.

2nd CAN

The libraries pertaining to the second communication channel are the same as for the first and are as follows:

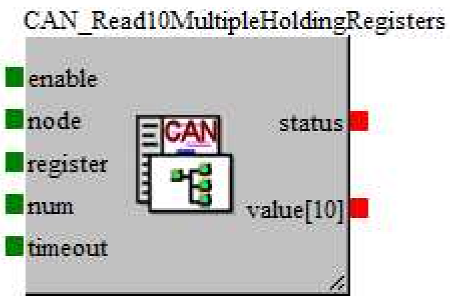
- CAN2nd_GetStatus
- CAN2nd_GetNetworkStatus
- CAN2nd_MyNode
- CAN2nd_IsMaster
- CAN2nd_Networknode_n
- CAN2nd_BaudRate
- CAN2nd_NodeGuardTimeout

As already mentioned, they perform the same function as those pertaining to the first CAN channel, hence they will not be discussed in any further detail, but simply listed. For the methods of use, please refer to the similar entities described previously.

2.7 CAN Master

These libraries allow you to access through the CAN network to the registers exported on the Modbus (through the UNI-PRO Export Entities tool).

CAN_Read10MultipleHoldingRegisters



Input	*Type*	*Limits*	*Description*
enable	CJ_ BIT	0-1	Enable the reading process
node	CJ_ BYTE	1-32	Logical adress in CAN network
register	CJ_WORD	1-65000	Starting modbus register
num	CJ_ BYTE	1-10	Number of consecutive addresses to read
timeout	CJ_ WORD	0-600	Timeout (mS)

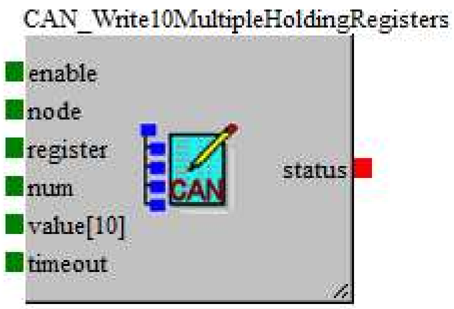
Output	*Type*	*Limits*	*Description*
status	CJ_BYTE	0-6	Return the reading status: 0 = Idle. Free 1 = Wait. Inserted new request 2 = In Progress. 3 = OK. Reading correctly completed 4 = Error. Reading not completed correctly 5 = Timeout. Reading not completed correctly for timeout. 6 = Rejected. Reading rejected.
value[10]	CJ_ WORD	0-65535	Values read

Description

The *CAN_Read10MultipleHoldingRegisters* library allows to read up to 10 Holding Registers exported on Modbus to a node of the CAN network. To start the reading operation must set a trasition from "0" to "1" to the *enable* input: will be added a reading request of the *num* data present in the *register* of node *node*. Once added the message will be processed and the answer will be analyzed monitoring the *status* output: when the output has the value 3 the response will be positive and the value will be in the *value[10]* outputs, instead if the output has the value 4 the response was negative, if the output has the value 5 the response was negative because timeout has expired, if the output has the value 6 the reading for the register is not accessible. Setting *enable* to 0 frees the queue for a next transmission. During all the communication period the *enable* input must be maintained to 1.

Note
It is essential that this library be used only once per project.

CAN_Write10MultipleHoldingRegisters



Ingressi	***Tipo***	***Limiti***	***Description***
enable	CJ_ BIT	0-1	Enable the writing process
node	CJ_ BYTE	1-32	Logical adress in CAN network
register	CJ_WORD	1-65000	Starting modbus register
num	CJ_ WORD	1-10	Number of consecutive addresses to write
value[10]	CJ_ WORD	0-65535	Values to write
timeout	CJ_ WORD	0-600	Timeout (mS)

Uscita	***Tipo***	***Limiti***	***Description***

status	CJ_BYTE	0-6	Return the writing status: 0 = Idle. Free 1 = Wait. Inserted new request 2 = In Progress. 3 = OK. Writing correctly completed 4 = Error. Writing not completed correctly 5 = Timeout. Writing not completed correctly for timeout. 6 = Rejected. Writing rejected.

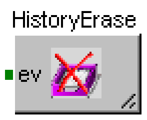
Description
The *CAN_Write10MultipleHoldingRegisters* library allows to write up to 10 Holding Registers exported on Modbus to a node of the CAN network. To start the writing operation must set a trasition from "0" to "1" to the *enable* input: will be added in queue the writing request of the data present in that moment in the *value[10]* input, to send to the *num* consecutive registers starting from the register *register* of the node *node*. Once added the message will be processed and the answer will be analyzed monitoring the *status* output: when the output has the value 3 the response will be positive, instead if the output has the value 4 the response was negative, if the output has the value 5 the response was negative because timeout has expired, if the output has the value 6 the writing for the register is not accessible. Setting *enable* to 0 frees the queue for a next transmission. During all the communication period the *enable* input must be maintained to 1.

Note
It is essential that this library be used only once per project.

2.8 History

The libraries shown below are to be used to manage the history log (cancelling, reading and writing). To be able to use them, it is necessary to enable use of the history log and define its dimensions in the project options. For a more detailed description of this topic please refer to the UNI-PRO software manual.

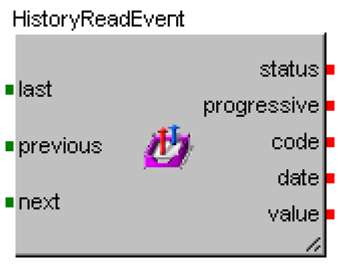
HistoryErase



Input	*Type*	*Limits*	*Description*
ev	CJ_BIT	0-1	Requests cancellation of the log entries. Cancellation is requested when the input value changes from 0 to 1

Description
Using the HistoryErase library it is possible to reset the history log, deleting all the entries present. The operation is performed when the *ev* input changes from 0 to 1 (leading edge).

HistoryReadEvent

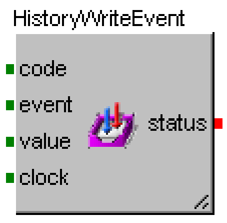


Input	*Type*	*Limits*	*Description*
last	CJ_BIT	0-1	Requests the last item stored in the historian
previous	CJ_BIT	0-1	Requests the previous item stored in the historian
next	CJ_BIT	0-1	Requests the next item stored in the historian

Output	***Type***	***Limits***	***Description***
status	CJ_BYTE	1..4	Indicates the status of the reading operation
progressive	CJ_LONG	-2147483648.. 2147483647	Progressive number of the event
code	CJ_WORD	0..65535	Event code
date	CJ_DATETIME	0.. 2147483647	Date and time of the entry
value	CJ_SHORT	-32768 .. 32767	Value joined to the event

Description
Using the HistoryReadEvent library it is possible to read the entries recorded previously in the event log using the HistoryWriteEvent library. This is an asynchronous operation and hence it is necessary to act on the inputs and then check the *value* output to know the status of the operation. Using the *last* input, it is possible to request the last event, the preceding events can be requested using *previous*, while the succeeding events can be requested using *next* input. Once reading the event has been achieved (*value* = HISTORY_READ_OK) it is possible to have the progressive number, code, date and value eventually stored for the requested event. The output *status* may assume the following values: ➢ 1: *HISTORY_READ_OK*. Reading correct ➢ 2: *HISTORY_READ_NONE*. There are no stored elements ➢ 3: *HISTORY_READ_DIM*. Memory “overflow” error ➢ 4: *HISTORY_READ_FAIL*. Reading failed

HstoryWriteEvent



Input	***Type***	***Limits***	***Description***
code	CJ_WORD	0..65535	Code of the event to be saved
event	CJ_BIT	0-1	Event save request. When this input changes from ‘0’ to ‘1’ then saving the event is initiated
clock	CJ_DATETIME	0..2147483647	Date and time in which the vent occurred (normally associated with the system clock)
value	CJ_SHORT	-32768 .. 32767	Value joined to the event

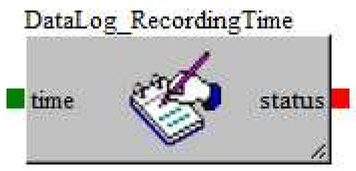
Output	***Type***	***Limits***	***Description***
status	CJ_BYTE	0..2	Operation status

Description
The HistoryWriteEvent library allows saving an event in the log. Associates the event code with the *code* input, the system clock with the *clock* input and field *value* with the value one wants to join to the event to store. When a change from ‘0’ to ‘1’ is detected, the data input save process is activated. To verify the outcome of the operation, check the value of the *status* output. The output may assume the following values: ➢ 0: *HISTORY_WRITE_OK*. Writing correct ➢ 1: *HISTORY_QUEUE_FULL*. Writing suspended because the request queue is full ➢ 2: *HISTORY_WRONG_DATETIME*. Writing failed because the data is incorrect

2.9 Data Log

To be able to use this libraries, it is necessary to enable use of the Data Log in the project options. For a more detailed description of this topic please refer to the UNI-PRO software manual.

DataLog_RecordingTime



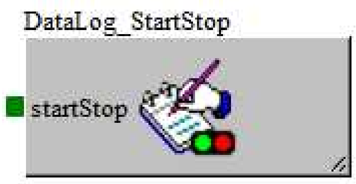
Input	*Type*	*Limits*	*Description*
time	CJ_WORD	10-65535	Recording time (in seconds)

Output	*Type*	*Limits*	*Description*
status	CJ_BIT	0..1	Returns the status of operation: 0 = Update OK. 1 = Update failed.

Description
The library *DataLog_RecordingTime* allows to set the recording frequency (in seconds) of the TIME elements selected in the log table. To set the default value, adjust the *value* property of the parameter *P_LOG_RecordingTime* in the library.

Note
The library is template type in nature. It is essential that this library be used only once per project.

DataLog_StartStop



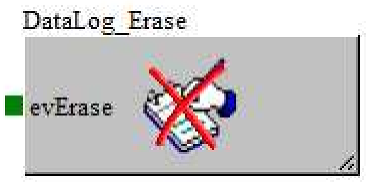
Input	*Type*	*Limits*	*Description*
startSop	CJ_BIT	0-1	It starts(1) or stop(0) the selected data recording.

Description

Through the *DataLog_StartStop* library you can start or stop recording entities in the Log. To enable logging set *startSop=1*, and to stop recording set *startStop=0*.

Note
It is essential that this library be used only once per project.

DataLog_Erase

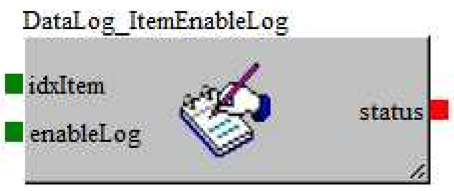


Input	***Type***	***Limits***	***Description***
evErase	CJ_BIT	0-1	Requests cancellation of the Data Log. Erasing is performed when the *ev* input changes from 0 to 1.

Description
Through the *DataLog_Erase* library you can delete alle the records of tha Data Log. The operation is performed when the *evErase* input changes from 0 to 1 (rising edge).

Note
It is essential that this library be used only once per project.

DataLog_ItemEnableLog



Input	***Type***	***Limits***	***Description***
idxItem	CJ_WORD	1-64	Index of Log item to enable/disable.
enableLog	CJ_BIT	0-1	Enabling of item: 0 = Disabled 1 = Enabled

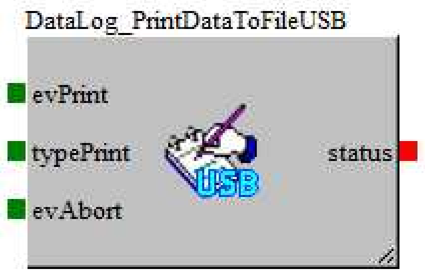
Output	***Type***	***Limits***	***Description***

status	CJ_WORD	0-4	Returns the status of operation: 0 = Operation executed. 1 = Error. Index out of bound. 2 = Error. Item no found 3 = Udate not requested 4 = Generic error.

Description
The *DataLog_ItemEnableLog* library is used to enable the recording of the element represented by the idxItem. The Data Log records only the values of the entities enabled (*enableLog=1*). To verify the correctness of the operation to monitor the status of the *status* output. The enabling/disabling of the Item must be at Log in STOP. Changes made enabling/disabling of the Item, if done on an existing Data Log, involve the formatting of the entire log, with the following consequences: loss of data previously recorded and momentary loss of communication with local and remote displays.

Note
To know the indices of the entities exported in log refer to the export table created with the *Data Log* tool in UNI-PRO.

DataLog_PrintDataToFileUSB



Input	***Type***	***Limits***	***Description***
evPrint	CJ_BIT	0-1	Request of printing of the CSV file on the USB port. Printig is performed when the *evPrint* input changes from 0 to 1.
typePrint	CJ_BIT	0-1	Export type for the printing: 0=Suspensive 1=Not Suspensive
evAbort	CJ_BIT	0-1	Abort request of the CSV file printing (this procedure makes sense only if *typePrint=1*). Abort is performed when the *evAbort* input changes from 0 to 1.

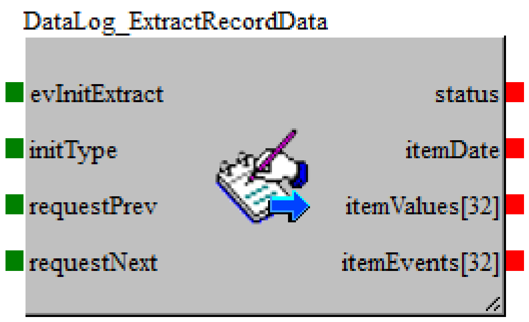
Output	***Type***	***Limits***	***Description***

status	CJ_WORD	0-1	Return status of elaboration: 0 = Operation Done/Print not in progress. 1 = Print in progress.

Description
The library *DataLog_PrintDataToFileUSB* is used to request the print of the files (named *NomeProgetto_YYYY_MM_DD.csv)* containing all the recordings of the Data Log on the USB port of the controller. The operation is carried out when the input *evPrint* changes from 0 to 1 (rising edge). You can perform this procedure in two modes (input *typePrint*): • 0 = Suspensive. The performance of the algorithms of the application is stopped, and the outputs/inputs remain frozen at the present value, until the end of the operation. • 1 = Not Suspesive. The performance of the algorithms of the application continues without interruption. This mode is slower than the previous. The end of the operation can occur by testing the exit *status*. Through input *evAbort* you can stop processing for printing records and print the files on the USB port to the point of interruption (the operation is carried out when the input changes from 0 to 1). Through the exit *status*, you can monitor the progress of the operation. The input *evAbort* and the output *status* can only be used in the Not Suspensive mode (*typePrint=1*).

Note
It is essential that this library be used only once per project. During printing it is advisable to stop the Data Log so as not to stretch the processing time (using the library *DataLog_StartStop*).

DataLog_ExtractRecordData



Input	***Type***	***Limits***	***Description***
evInitExtract	CJ_BIT	0-1	Request of initialization for the data structure to get the first Data Log record. This event makes available the first element of Datalog to the output. Request is performed when the

			evInitExtract input changes from 0 to 1.
initType	CJ_BYTE	1-2	Initialization type: 1 = Positioning to the first record of the Data Log (the oldest). 2 = Positioning to the last record of the Data Log (the most recent).
requestPrev	CJ_BIT	0-1	Request the previous record.
requestNext	CJ_BIT	0-1	Request the next record.

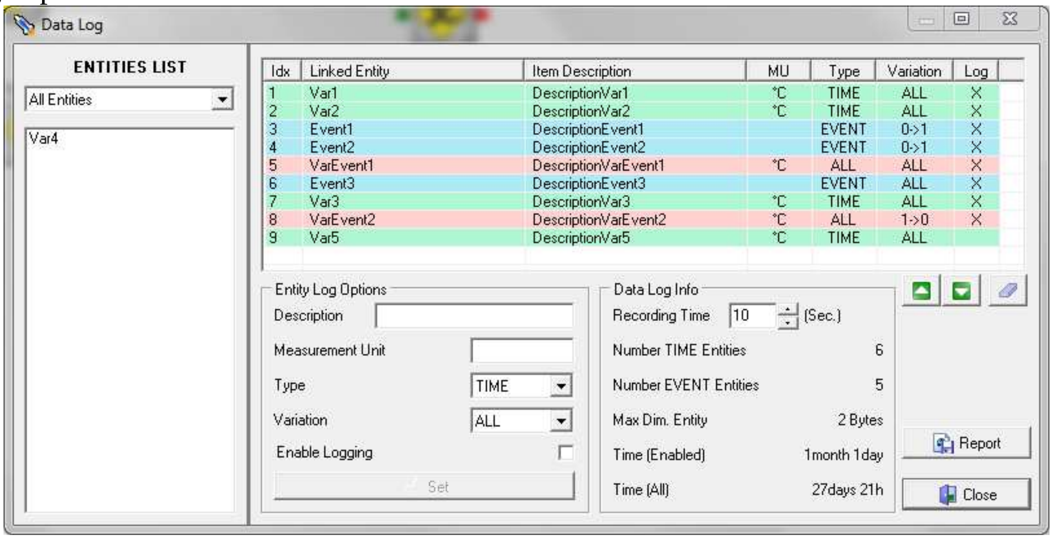
Output	***Type***	***Limits***	***Description***
status	CJ_BYTE	0-2	Status of the operation: 0 = No update 1 = Output data updated 2 = Reaching the end of the log (uphill) 3 = Reaching the end of the log (downhill) When the error is 2 or 3 in output there are not valid data.
itemDate	CJ_DATETIME		Registration Date of the current record.
itemValues[32]	CJ_LONG	-2147483648.. 2147483647	Value of the Time items in the current record.
itemEvents[32]	CJ_BIT	0-1	Value of the Event items in the current record.

Description
The library *DataLog_ExtractRecordData* allows you to read the records stored in the Data Log. First you need to position the pointer on the starting record, using the terminal *evInitExtract*, this is done when the input changes from 0 to 1 (rising edge). It is possible select where start the extracting, using the *initType* input: • =1: Select the first record registered. • =2: Select the last record registered. The operation of init, as well as to prepare structures, fills the outputs with the data of the selected record. Using the inputs *requestNext* and *requestPrev* you can scroll through all the log records. For each request outputs are updated with the data of the selected record: • *itemDate* : recording data of the record. • *itemValues[32]*: value of the Time items in the current record (TIME/ALL). • *itemEvents[32]*: value of the Event items in the current record (EVENT/ALL). • *status*: identifies the status of the request - 0: No request/waiting for data - 1: Data output *(itemDate, itemValues, itemEvents*) are updated. Warning that this state remains valid for a single turn of the main. - 2: After an action of *requestNext* has been exceeded the last element of the LOG, the output data are not valid. The latest data available are those of the previous request.

- 3: After an action of *requestPrev* has been exceeded the first element of the LOG, the output data are not valid. The latest data available are those of the previous request.

Interpretation of the outputs *itemValues[32]* and *itemEvents[32]*
Each index of the two arrays is an item in the exportation log table generated with UNI-PRO, the indexes are consecutive and ordered compared to the Dat Log table. The outputs are arrays of 32 elements, because the maximum number of entities of type TIME and EVENT is 32 for each.
The value of the disabled items in the log, and the elements not present is always 0.

Example.
Data Log exportation table with 6 TIME entities and 5 EVENT entities.


Data Log

ENTITIES LIST
All Entities
Var4

Idx	Linked Entity	Item Description	MU	Type	Variation	Log
1	Var1	DescriptionVar1	°C	TIME	ALL	X
2	Var2	DescriptionVar2	°C	TIME	ALL	X
3	Event1	DescriptionEvent1		EVENT	0->1	X
4	Event2	DescriptionEvent2		EVENT	0->1	X
5	VarEvent1	DescriptionVarEvent1	°C	ALL	ALL	X
6	Event3	DescriptionEvent3		EVENT	ALL	X
7	Var3	DescriptionVar3	°C	TIME	ALL	X
8	VarEvent2	DescriptionVarEvent2	°C	ALL	1->0	X
9	Var5	DescriptionVar5	°C	TIME	ALL	

Entity Log Options
Description
Measurement Unit
Type: TIME
Variation: ALL
Enable Logging
Set

Data Log Info
Recording Time: 10 (Sec.)
Number TIME Entities: 6
Number EVENT Entities: 5
Max Dim. Entity: 2 Bytes
Time (Enabled): 1month 1day
Time (All): 27days 21h

Report
Close

To every request *requestNext* or *requestPrev* outputs *itemValues[32]* and *itemEvents[32]* must be interpreted as:

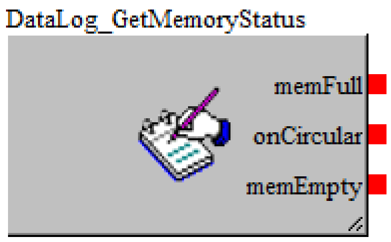
TIME	
itemValues[0]	Value of Var 1
itemValues[1]	Value of Var 2
itemValues[2]	Value of VarEvent 1
itemValues[3]	Value of Var 3
itemValues[4]	Value of VarEvent 2
itemValues[5]	Value of Var 5 (ZERO because disabled)
itemValues[6]	0
…	

EVENT	
itemEvents[0]	Event of Event1
itemEvents[1]	Event of Event 2
itemEvents[2]	Event of VarEvent1
itemEvents[3]	Event of Event 3
itemEvents[4]	Event of VarEvent2
itemEvents[5]	0
itemEvents[6]	0
….	

All the values int the successive indexes that are not used, are 0.

Note
It is essential that this library be used only once per project.

DataLog_GetMemoryStatus

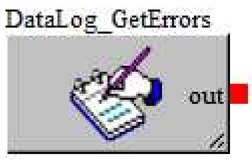


Output	*Type*	*Limits*	*Description*
memFull	CJ_BIT	0-1	When true (1) memory is full.
onCircular	CJ_BIT	0-1	When true (1) memory space is exhausted and is being stored circular.
memEmpty	CJ_BIT	0-1	When true (1) memory is empty, nothing record is stored.

Description
The library *DataLog_GetMemoryStatus* returns the Data Log memory status, identifies the states of full memory, memory empty and circular logging. When recording enters on circular mode (*onCircular=1*), the oldest data in memory is overwritten by new ones.

Note
It is essential that this library be used only once per project.

DataLog_GetErrors



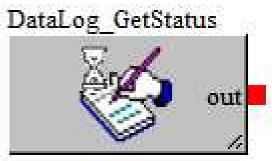
Output	*Type*	*Limits*	*Description*
out	CJ_WORD	0-6	Returns the status of operation of the Data log: 0 = Data Log not configured. 1 = Data Log OK, in STOP. 2 = Data Log OK, in RUN. 3 = Error, memory not compatible. 4 = Error RTC. 5 = Error, memory corrupted. 6 = Error, not enough space.

Description

The library *DataLog_GetErrors* returns the operation status of the Data Log.

Note
It is essential that this library be used only once per project.

DataLog_GetStatus



Output	***Type***	***Limits***	***Description***
out	CJ_BYTE	0-3	Returns the status of operation of the Data log: 0 = Idle/format finished with success. 1 = Format/erase in progress. 2 = Format/erase in request. 3 = Init & check datalog memory.

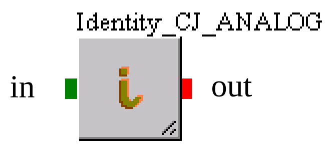
Description
The library *DataLog_GetStatus* returns the operation status of the Data Log, during the erase and format phases.

Note
It is essential that this library be used only once per project.

2.10 *Identity*

These libraries return the data inputting through them. For example, they are useful within a subsheet if it is desired to use an input exported from the subsheet itself several times. A single example is illustrated by way of completeness.

Identity_CJ_ANALOG



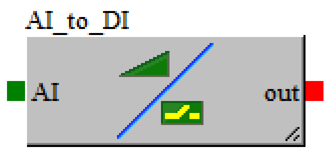
The *out* output represents the *in* input without any changes. In this library, the two “terminals” accept a CJ_ANALOG type structure.

The other libraries belonging to the *Identity* group are the following:

- Identity_CY_BIT
- Identity_CY_BTN
- Identity_CY_BUZZ
- Identity_CY_BYTE
- Identity_CY_CMD
- Identity_CY_DATE
- Identity_CY_DATETIME
- Identity_CY_DWORD
- Identity_CY_LED
- Identity_CY_LONG
- Identity_CY_SHORT
- Identity_CY_S_BYTE
- Identity_CY_TIME
- Identity_CY_WORD

2.11 IO Utils

AI_to_DI

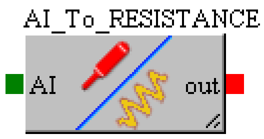


Input	*Type*	*Limits*	*Description*
AI	CJ_ANALOG	Value: -32768..32767 Error: 0..2	Sensor input

Output	*Type*	*Limits*	*Description*
out	CJ_BIT	0..1	Value of digital input

Description
Coverts an analog input to a digital input. To allow to the correctly library functionality, *AI* pin must be directly connected to a real analog input (*out* pin of *ANALOGIN* library) and not to a CJ_ANALOG variable derived from elaborations. The analog input must be configured as NTC. • DI = 1 for contacts with resistance <= 600 ohms • DI = 0 for contacts with resistance >= 60k ohms

AI_to_RESISTANCE

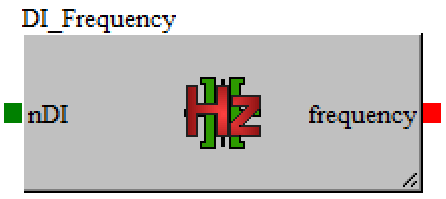


Ingresso	*Tipo*	*Limiti*	*Descrizione*
AI	CJ_ANALOG	Value: -32768..32767 Error: 0..2	Sensor input CJ_ANALOG "RESISTANCE"

Uscita	*Tipo*	*Limiti*	*Descrizione*
out	CJ_LONG	0.. 2147483648	Value of resi stance in OHM

Descrizione
Converts a RESISTANCE type analog inut to a value OHM based. To allow to the correctly library functionality, *AI* pin must be directly connected to a real analog input (RESISTANCE) and not to a CJ_ANALOG variable derived from elaborations

DI_Frequency



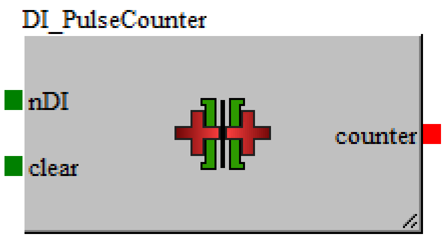
Input	*Type*	*Limits*	*Description*
nDI	CJ_BYTE	0..255	Number of the digital input wich read the frequency

Output	*Type*	*Limits*	*Description*
frequency	CJ_WORD	0.. 65535	Returns the Frequency value (Hz)

Description
Returns the frequency in Hz detected by the digital input selected by *nDI*. The value of *nDI* must be the number of the digital input to be monitored.

Note
The digital inputs of the controllers where reading frequency are different, to use the correct inputs refer to the hardware manual. • Family Micro/Kilo: DI3 and DI4 • Family Node Kilo: DI1 and DI2

DI_PulseCounter



Input	*Type*	*Limits*	*Description*
nDI	CJ_BYTE	0..255	Number of the digital input used ad pulse counter
clear	CJ_BIT	0-1	Clear the counter

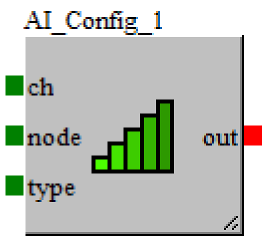
Output	*Type*	*Limits*	*Description*

counter	CJ_DWORD	0.. 4294967295	Counter of the pulses detected by the digital input

Description
Returns the number of the pulsese detected by the the digital input selected by *nDI*. The value of *nDI* must be the number of the digital input to be monitored. To reset the counter set the input value *clear=1*.

Note
The digital inputs used as pulse counter of the controllers are different, to use the correct inputs refer to the hardware manual. • Family Micro/Kilo: DI3 and DI4 • Family Node Kilo: DI1 and DI2

AI_Config

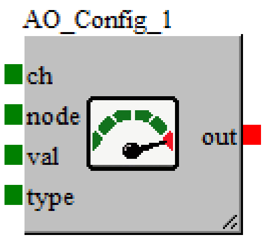


Input	***Type***	***Limits***	***Description***
ch	CJ_BYTE	1..N	Physical channel of the analog input of the controller or expansion
node	CJ_BYTE	0..32	Logical address in the CANbus network of the controller (= 0) or of the expansion (1..32). It must correspond to what is set in the Hardware Expert. If not connected by default it is 0 (controller)
type	CJ_BYTE	1..N	Type of sensor to be configured. Only for the controller. If not connected, the sensor is not configured.

Output	***Type***	***Limits***	***Description***
out	CJ_ANALOG	-	Value read by the analog input *ch* of the controller or of the expansion of the *node* address. If you read the CJ_ERR_REF value in the Error field, the sensor configuration requested with the *type* value was not successful.

Description
It allows you to read and configure an analog input, specifying the physical channel and logical node in the CANbus network. To ensure the correct functioning of the library, all the *AI* entities with which these libraries are used must be allocated in the project (they do not need to be linked).

AO_Config

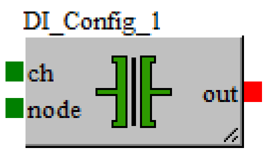


Input	***Type***	***Limits***	***Description***
ch	CJ_BYTE	1..N	Physical channel of the analog output of the controller or expansion
node	CJ_BYTE	0..32	Logical address in the CANbus network of the controller (= 0) or of the expansion (1..32). It must correspond to what is set in the Hardware Expert. If not connected by default it is 0 (controller)
val	CJ_WORD	0..100.00	Value to be provided to the analog output.
type	CJ_BYTE	0..N	Type of actuator to be configured. Only for the controller. If not connected, the output is not configured.

Output	***Type***	***Limits***	***Description***
out	CJ_WORD	0..100.00	Value supplied to the analog output *ch* of the controller or of the expansion of the *node* address. If you read the 0.01 value, the output configuration requested with the *type* value was not successful.

Description
It allows you to control and configure an analog output, specifying the physical channel and logical node in the CANbus network. To ensure the correct functioning of the library, all the *AO* entities of which these libraries are used must be allocated in the project (it is necessary that they are connected to an entity of the VAR type, no matter the value as it will be overwritten).

DI_Config

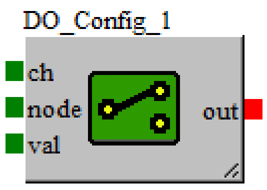


Input	*Type*	*Limits*	*Description*
ch	CJ_BYTE	1..N	Physical channel of the digital input of the controller or expansion
node	CJ_BYTE	0..32	Logical address in the CANbus network of the controller (= 0) or of the expansion (1..32). It must correspond to what is set in the Hardware Expert. If not connected by default it is 0 (controller)

Output	*Type*	*Limits*	*Description*
out	CJ_BIT	0..1	Value read by the digital input *ch* of the controller or of the expansion of the *node* address.

Description
It allows to read a digital input, specifying physical channel and logical node in CANbus network. To ensure the correct functioning of the library, all the *DI* entities with which these libraries are used must be allocated in the project (they do not need to be linked).

DO_Config



Input	*Type*	*Limits*	*Description*
ch	CJ_BYTE	1..N	Physical channel of the digital output of the controller or expansion
node	CJ_BYTE	0..32	Logical address in the CANbus network of the controller (= 0) or of the expansion (1..32). It must correspond to what is set in the Hardware Expert. If not connected by default it is 0 (controller)

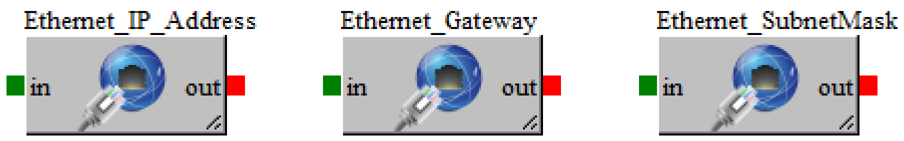
val	CJ_BIT	0..1	Value to be provided to the digital output.

Output	***Type***	***Limits***	***Description***
out	CJ_BIT	0..1	Value provided by the digital output *ch* of the controller or of the expansion of the *node* address.

Description
It allows you to control a digital output, specifying the physical channel and logical node in the CANbus network. To ensure the correct functioning of the library, all the *DO* entities with which these libraries are used must be allocated in the project (it is necessary that they are connected to an entity of the VAR type, no matter the value as it will be overwritten).

2.12 Ethernet TCPIP

Ethernet_IP_Address - Ethernet_Gateway - Ethernet_SubnetMask



Input	*Type*	*Limits*	*Description*
in	CJ_IP_ADDRESS	0.0.0.0 255.255.255.255	IP address, GATEWAY and SUBnet MASK for the device

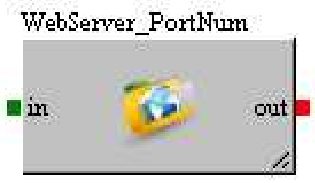
Output	*Type*	*Limits*	*Description*
out	CJ_IP_ADDRESS	0.0.0.0 255.255.255.255	IP address, GATEWAY and SUBnet MASK for the device

Description
The libraries allow setting the IP address of the controller, the Gateway and the Subnet Mask of the network. To set the default value, adjust the *value* property of the parameters in the libraries.

Note
The library is template type in nature. It is essential that this library be used only once per project. The library has meaning only if Ethernet protocol is selected for the relative port.

2.13 WebServer TCP

WebServer_PortNum



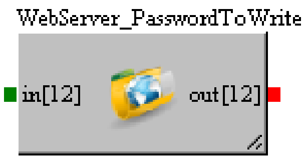
Input	*Type*	*Limits*	*Description*
in	CJ_WORD	0..65535	Web Server port number

Output	*Type*	*Limits*	*Description*
out	CJ_WORD	0..65535	Web Server port number

Description
The library allows setting the port number for the communication with the Web Server over TCP-IP. To set the default value, adjust the *value* property of the parameter *P_WebServer_PortNum* in the library.

Note
The library is template type in nature. It is essential that this library be used only once per project. The library has meaning only if Ethernet protocol is selected on the relative port and the Web Server is enabled.

WebServer_PasswordToWrite



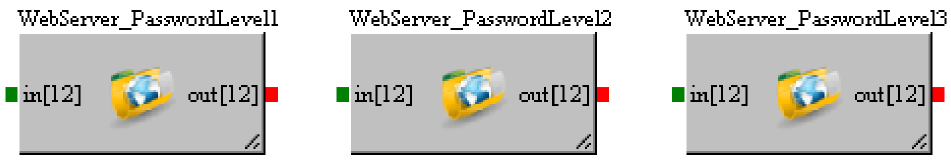
Input	***Type***	***Limits***	***Description***
in[12]	CJ_CHAR		Sets the password string chars (max. 12 characters)

Output	***Type***	***Limits***	***Description***
out[12]	CJ_CHAR		Returns the password string chars (max. 12 characters)

Description
The library allows setting the password to modify the status exported in the web page of the WebServer over TCP-IP. To set the default value, adjust the *value* property of the parameter *P_WebServer_PasswordToWrite* in the library.

Note
The library is template type in nature. It is essential that this library be used only once per project. The library has meaning only if Ethernet protocol is selected on the relative port and the Web Server is enabled.

WebServer_PasswordLevel1 .. WebServer_PasswordLevel3



Input	***Type***	***Limits***	***Description***
in[12]	CJ_CHAR		Sets the password string chars (max. 12 characters) for each level

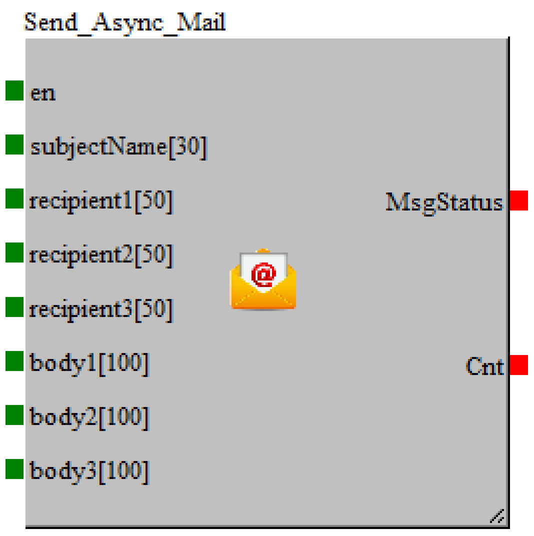
Output	***Type***	***Limits***	***Description***
out[12]	CJ_CHAR		Returns the password string chars (max. 12 characters)

Description
The library allows setting the password to access the web page referred to the protection level of the WebServer over TCP-IP. To set the default value, adjust the *value* property of the parameter *P_WebServer_PasswordLevel..* in the library.

Note
The library is template type in nature. It is essential that this library be used only once per project. The library has meaning only if Ethernet protocol is selected on the relative port and the Web Server is enabled.

2.14 Messages

Send_Async_Mail



Input	*Type*	*Limits*	*Description*
en	CJ_ BIT	0..1	Enable transmission
subjectName[30]	CJ_ CHAR		Mail subject (max 30 chars)
recipient1[50]	CJ_ CHAR		First recipient (max 50 chars)
recipient2[50]	CJ_ CHAR		Second recipient (max 50 chars)
recipient3[50]	CJ_ CHAR		Third recipient (max 50 chars)
body1[100]	CJ_ CHAR		First part of the mail body
body2[100]	CJ_ CHAR		Second part of the mail body
body3[100]	CJ_ CHAR		Third part of the body of the mail

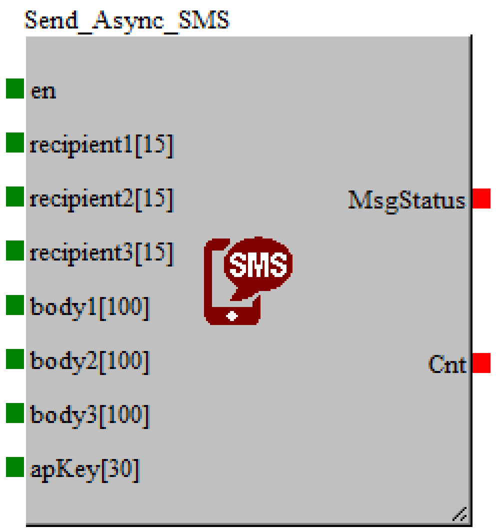
Output	*Type*	*Limits*	*Description*
MsgStatus	CJ_BYTE	0..6	Status of the transmission activity: 0 = IDLE 1 = SENDING (from CPro to Cloud Server) 2 = DELIVERING (from the Cloud Server to Recipients) 3 = OK (Messages reached the recipients) 4 = ERROR 5 = ABORT 6 = WAITING (the channel is unique, if there are more than one consecutive requests the next one queues).

Cnt	CJ_WORD	0..65535	Counter since the message begins to be transmitted until the server returns OK or ERROR. Currently arriving at 300 seconds the message is aborted by default.

Description
The *Send_Async_Mail* library allows you to send e-mail. The input en must be held at 1 to enable the sending and until it has completed successfully (OK) or with error (ERROR). If it is brought to 0 before the state goes into ABORT. At least one of the three recipients (*recipientsN* inputs) must be not null, no validity checks, if the server fails to forward to one of the recipients, retry all recipients. Each body of the mail (*bodyN* inputs) is composed of up to 100 characters, 99 of which are free, while the first one must necessarily indicate the type of body, namely: - 0 = HTML - 1 = Text UTF8(*) - 2 = Table (*) In the case of a character greater than 127 ASCII, it is necessary to encode with the UTF8 notation. E.g. "2°C" must write "2\ xc2\ xb0C"

Note
The library has meaning only if Ethernet protocol is selected on the relative port and the option Mail/SMS is enabled.

Send_Async_SMS



Input	***Type***	***Limits***	***Description***
en	CJ_ BIT	0..1	Enable transmission

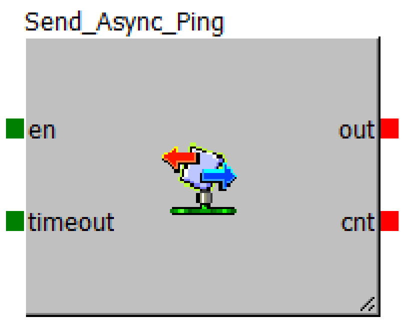
recipient1[15]	CJ_ CHAR		First recipient number (must start with '+' followed by International Prefix)
recipient2[15]	CJ_ CHAR		Second recipient number (must start with '+' followed by International Prefix)
recipient3[15]	CJ_ CHAR		Third recipient number (must start with '+' followed by International Prefix)
body1[100]	CJ_ CHAR		First part of the SMS body
body2[100]	CJ_ CHAR		Second part of the SMS body
body3[100]	CJ_ CHAR		Third part of the SMS body
apKey[30]	CJ_ CHAR		API Key for sending messages

Uscita	***Type***	***Limits***	***Description***
MsgStatus	CJ_BYTE	0..6	Status of the transmission activity: 0 = IDLE 1 = SENDING (from CPro to Cloud Server) 2 = DELIVERING (from the Cloud Server to Recipients) 3 = OK (Messages reached the recipients) 4 = ERROR 5 = ABORT 6 = WAITING (the channel is unique, if there are more than one consecutive requests the next one queues).
Cnt	CJ_WORD	0..65535	Counter since the message begins to be transmitted until the server returns OK or ERROR. Currently arriving at 300 seconds the message is aborted by default.

Description
The *Send_Async_SMS* library allows you to send SMS. The input en must be held at 1 to enable the sending and until it has completed successfully (OK) or with error (ERROR). If it is brought to 0 before the state goes into ABORT. At least one of the three recipients (*recipientsN* inputs) must be not null, no validity checks, if the server fails to forward to one of the recipients, retry all recipients. Each body of the message (*bodyN* inputs) is composed of up to 100 characters, 99 of which are free, while the first one must necessarily indicate the type of body, in this case must always be 1.

Note
The library has meaning only if Ethernet protocol is selected on the relative port and the option Mail/SMS is enabled.

Send_Async_Ping



Input	*Type*	*Limits*	*Description*
en	CJ_ BIT	0..1	Enable operation
timeout	CJ_BYTE	0..255	Operation timeout (s)

Output	*Type*	*Limits*	*Description*
out	CJ_BYTE	0..5	Status of the transmission activity: 0 = IDLE 1 = SEND/RETRY 2 = WAIT 3 = OK (Ping completed successfully) 4 = ERROR 5 = ABORT
cnt	CJ_WORD	0..255	Counter from when the sending of the packet begins, until the "out" output reaches the OK or ERROR state.

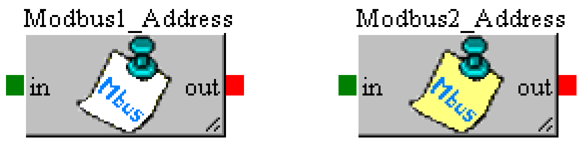
Description
The library *Send_Async_Ping* allows you to send a packet to the network to check its status. The input *en* must be held at 1 to enable the sending and until it has completed successfully (OK) or with error (ERROR). The value of the *cnt* output exceeds that set for the *timeout*, the *out* output switches to the ERROR state.

Note
The library has meaning only if Ethernet protocol is selected on the relative port and the option Mail/SMS is enabled.

2.15 Modbus

These libraries allow the configuration and management of the standard MODBUS protocol.
The UNI-PRO software envisages the possibility for assigning this protocol to one or both of the available UART interfaces on the controller. The UART1 libraries will be identified by the Modbus1 prefix, the UART2 libraries from the Modbus2 prefix.
Please refer to the controller hardware documentation to check the compatibility with this protocol.

Modbus1_Address – Modubus2_Address



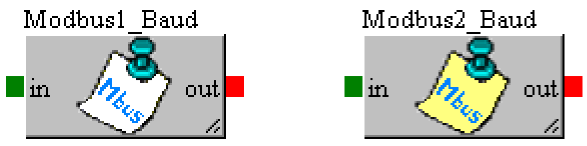
Input	*Type*	*Limits*	*Description*
in	CJ_ BYTE	1..247	Sets the physical address of the controller within the network

Output	*Type*	*Limits*	*Description*
out	CJ_BYTE	1..247	Returns the physical address of the controller within the network

Description
The *ModbusAddress* library represents the physical address of the controller on the MODBUS network. To set the default value, adjust the *value* property of the *ModBus_Address* parameter in the library.

Notes
The library is template type in nature. It is essential that this library be used only once per project. The library has meaning only for Modbus Slave.

Modbus1_Baud – Modbus2_Baud



Input	*Type*	*Limits*	*Description*
in	CJ_ BYTE	0..7	Sets the baud rate

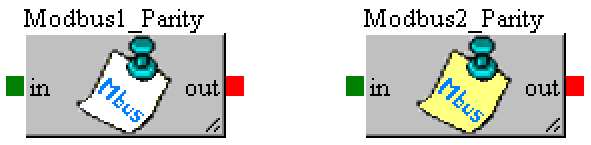
Output	*Type*	*Limits*	*Description*

out	CJ_BYTE	0..7	Returns the baud rate

Description
The *ModbusBaud* library allows setting the communication baud rate. The following baud rates are supported: ➢ 0: 1200 Kbit ➢ 1: 2400 Kbit ➢ 2: 4800 Kbit ➢ 3: 9600 Kbit ➢ 4: 19200 Kbit ➢ 5: 28800 Kbit ➢ 6: 38400 Kbit ➢ 7: 57600 Kbit To set the default value, adjust the *value* property of the *ModBus_Baud* parameter in the library.

Notes
The library is template type in nature. It is essential that this library be used only once per project. The library can be used for Modubus Master and Modbus Slave.

Modbus1_Parity – Modbus2_Parity



Input	***Type***	***Limits***	***Description***
in	CJ_ BYTE	0..2	Sets the type of parity for data transmission

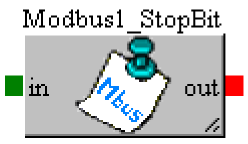
Output	***Type***	***Limits***	***Description***
out	CJ_BYTE	0..2	Returns the type of parity for data transmission

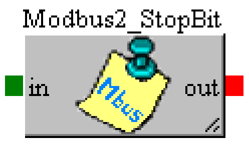
Description
The *ModbusParity* library allows setting the parity for data transmission. Parity is a code used to check the integrity of transmitted data and may be of the following types: *0.* *NONE*. No parity *1.* *ODD*. Odd parity *2.* *EVEN*. Even parity To set the default value, adjust the *value* property of the *ModBus_Parity* parameter in the library.

Notes

The library is template type in nature. It is essential that this library be used only once per project. The library can be used for Modubus Master and Modbus Slave.

Modbus1_StopBit – Modbus2_StopBit





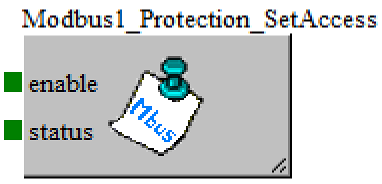
Input	***Type***	***Limits***	***Description***
in	CJ_ BIT	0-1	Sets the type of “stop bit”

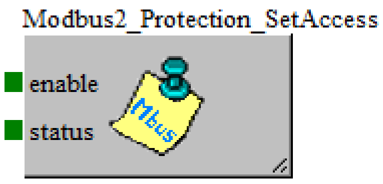
Output	***Type***	***Limits***	***Description***
out	CJ_BIT	0-1	Returns the type of “stop bit”

Description
The *ModbusStopBit* library allows setting the type of “stop bit” for data transmission. This technique is used to allow the transmitting/receiving device to detect the end of a communication frame. The number association is as follows: ➢ 0 = 1 Stop bit. ➢ 1 = 2 Stop bits. To set the default value, adjust the *value* property of the *ModBus_StopBit* parameter in the library.

Notes
The library is template type in nature. It is essential that this library be used only once per project. The library can be used for Modubus Master and Modbus Slave.

Modbus1_Protection_SetAccess – Modbus2_Protection_SetAccess



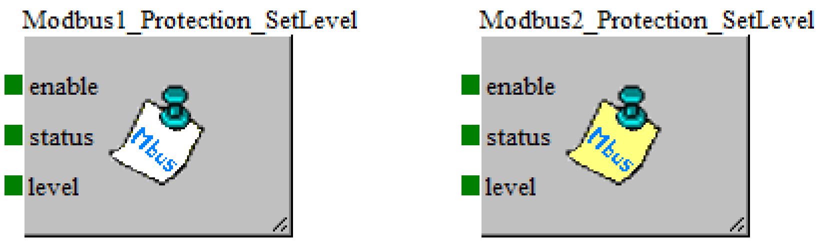


Input	***Type***	***Limits***	***Description***
enable	CJ_ BIT	0-1	Enable the modification of the reading access
status	CJ_ BIT	0-1	Sets the reading access status: 0=Not accessible 1=Accessible

Description
The *Modbus_Protection_SetAccess* library allows setting the reading protection for the relative Modbus uart; the modification is permorfed with enable=1, otherwise if enable=0 is not made any changes.

Notes
The library can be used for Modbus Slave.

Modbus1_Protection_SetLevel – Modbus2_Protection_SetLevel

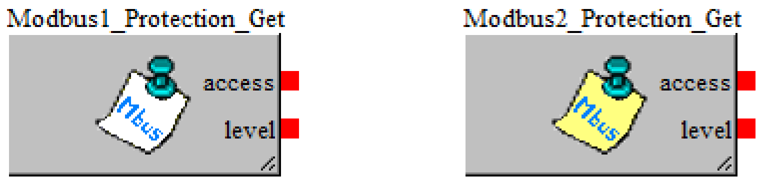


Input	***Type***	***Limits***	***Description***
enable	CJ_ BIT	0-1	Enable the modification of the level
status	CJ_ BIT	0-1	Sets the lock/unlock of the level: 0=Level locked 1=Level unlocked
level	CJ_ BYTE	1-5	Level to lock/unlock

Description
The *Modbus_Protection_SetLevel* library allows setting the lock/unlock (writing protection) of the level passed as input on the relative Modbus uart; the modification is permorfed with enable=1, otherwise if enable=0 is not made any changes.

Notes
The library can be used for Modbus Slave.

Modbus1_Protection_Get – Modbus2_Protection_Get



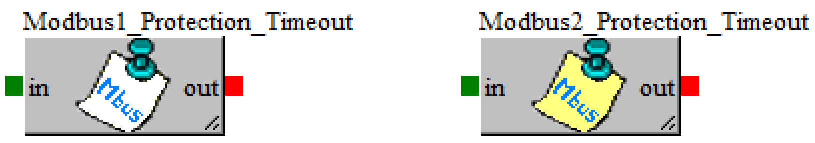
Output	***Tipo***	***Limiti***	***Descrizione***

access	CJ_ BIT	0-1	Staus of reading access protection: 0=Not accessible 1=Accessible
level	CJ_ BYTE	0-5	Max unlocked level

Description
The *Modbus_Protection_Get* library returns the status of the Modbus protection. The output *access* takes these values: • 0= Reading access protection not active. • 1= Reading access protection active. The output *level* returns the unlocked level (The level lock management is hierarchical, then the previous levels are unlocked).

Notes
The library can be used for Modbus Slave.

Modbus1_Protection_Timeout – Modbus2_Protection_Timeout



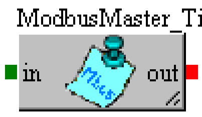
Input	***Type***	***Limits***	***Description***
in	CJ_ WORD	0-65535	Unlock level timeout (Seconds)

Output	***Type***	***Limits***	***Description***
in	CJ_ WORD	0-65535	Unlock level timeout (Seconds)

Description
The *ModbusTCP_Protection_Timeout* library allows setting the duration of the unlock of the levels, the count is recalculated at every access made on the serial. To set the default value, adjust the *value* property of the *P_Modbus_ProtectionTimeout* parameter in the library. If the value is 0 levels are always unlocked.

Notes
The library is template type in nature. It is essential that this library be used only once per project. The library can be used for Modbus Slave.

ModbusMaster_Timeout



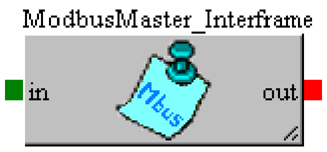
Input	*Type*	*Limits*	*Description*
in	CJ_ BYTE	1..240	Sets the timeout of the response during a communication with the slave.

Output	*Type*	*Limits*	*Description*
out	CJ_BYTE	1..240	Returns the timeout of the response during a communication with the slave.

Description
The *ModbusMasterTimeout* library represents the maximum time at disposal of the slave to answer to the master. After the expiration of this timeout, if there isn't any answer from the slave, the master set a communication error and can start another communication process. To set the default value, adjust the *value* property of the *ModBusMaster_Timeout* parameter in the library. The value is in hundreds of milliseconds (e.g. a value of 10 means 1 second)

Notes
The library is template type in nature. It is essential that this library be used only once per project. The library has meaning only for Modbus Master.

ModbusMaster_Interframe



Input	*Type*	*Limits*	*Description*
in	CJ_ BYTE	0..240	Sets the interframe time in 100ms.

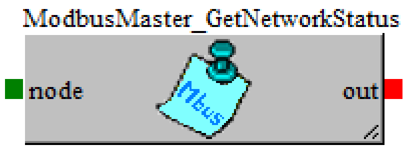
Output	*Type*	*Limits*	*Description*
out	CJ_BYTE	0..240	Returns the interframe time in 100ms.

Description

The library *ModbusMaster_Interframe* allows setting the minimum time between a data package and the following in the Modbus communication.
To set the the default value operate on property *value* of the parameter *P_ModbusMaster_Interframe* into the library.
The value is in 100 x milliseconds (it means that 10 is 1 second).

Notes
The library is a template library. It is necessary to use this library once a project.

ModbusMaster_GetNetworkStatus



Input	***Type***	***Limits***	***Description***
node	CJ_ BYTE	0..247	Logical node to verify.

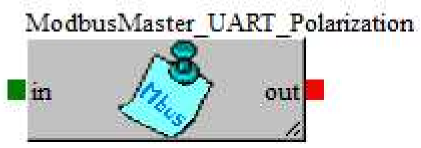
Output	***Type***	***Limits***	***Description***
out	CJ_S_BYTE	-1..2	Returns the Modbus Master communication status with *node* (logical node of the network) -1 = Device not avail 0 = Communication OK 1 = Device in NoLink 2 = Device does not communicate - At the beginning if the slave has never been queried - If it was a 0 (OK) and passed at least one minute after the last communication with the master - If it was a 1 (NoLink) and remains in the error status after the timeout of a minute.

Description
The *ModbusMaster_GetNetworkStatus* library allows to verify the Modbus Master communication status with the logical node setted by di input *node*. The library is applicable only to logical nodes of the devices configurated through ModbusMaster tool in the UNI-PRO project.

Note

It is possibile to use more times this library in the project. The library can be used only for Modbus Master.

ModbusMaster_UART_Polarization



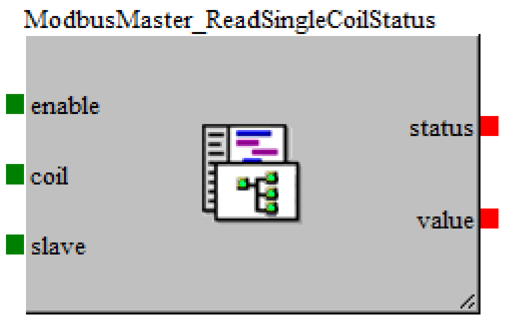
Input	*Type*	*Limits*	*Description*
in	CJ_ BIT	0-1	Sets the polarization of the ModbusMaster port: 0: UART Not polarized 1: UART Polarized

Output	*Type*	*Limits*	*Description*
out	CJ_BIT	0-1	Returns the polarization of the ModbusMaster port.

Description
The *ModbusMaster_UART_Polarization* allows to set the polarization of the communication port for the ModbusMaster protocol. To set the default value, adjust the *value* property of the *P_ModbusMaster_UART_Polarization* parameter in the library.

Note
The library is a template library. It is necessary to use this library once a project.

ModbusMaster_ReadSingleCoilStatus



Inputs	*Type*	*Limits*	*Description*
enable	CJ_ BIT	0-1	Enables the master/slave transmission process of a reading frame

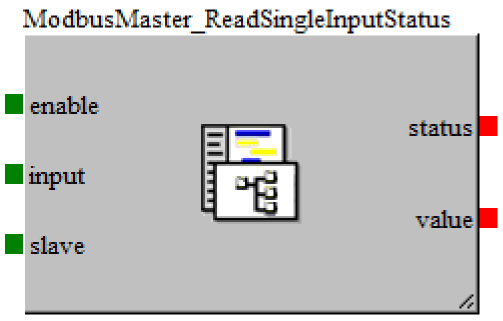
coil	CJ_WORD	1-65000	Identifier of the coil that you want to access
slave	CJ_ BYTE	0-247	Slave node address. 0 = Broadcast.

outputs	***Type***	***Limits***	***Description***
status	CJ_BYTE	0-5	Returns the master/slave communication status: 0 = free for a communication 1 = added a communication request 2 = communication in progress 3 = communication correctly completed 4 = communication not completed correctly (i.e. NAK reply or timeout) 5 = communication not correctly concluded because of ModbusMaster requests queue saturation
value	CJ_ BIT	0-1	Read value

Description
The *ModbusMaster_ReadSingleCoilStatus* library allows to write a value in a Coil of a slave net node. This library implements the dispatch of a Modbus message wich implements the function code 0x01. To start the writing operation must set a trasition from “0” to “1” to the *enable* input: will be added in queue the writing request of the data present in that moment in the *value* input, to send to the *coil* of slave *node*. Once added in queue the message will be processed and the answer will be analyzed monitoring the *status* output: when the output has the value 3 the slave response will be positive, instead if the output has the value 4 the response was negative or the slave didn’t response. Setting *enable* to 0 frees the queue for a next transmission. During all the communication period the *enable* input must be maintained 1.

Notes
It is possibile to use more times this library in the project. The library can be used only for Modbus Master.

ModbusMaster_ReadSingleInputStatus



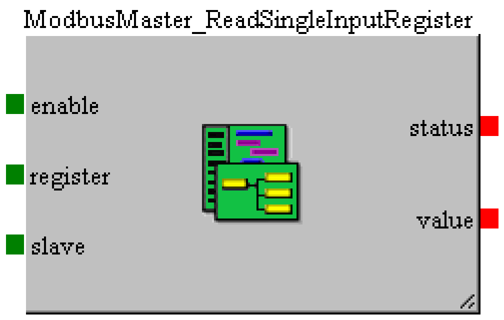
Inputs	***Type***	***Limits***	***Description***
enable	CJ_ BIT	0-1	Enables the master/slave transmission process of a reading frame
inpu	CJ_WORD	1-65000	Identifier of the input status that you want to access
slave	CJ_ BYTE	0-247	Slave node address. 0 = Broadcast.

outputs	***Type***	***Limits***	***Description***
status	CJ_BYTE	0-5	Returns the master/slave communication status: 0 = free for a communication 1 = added a communication request 2 = communication in progress 3 = communication correctly completed 4 = communication not completed correctly (i.e. NAK reply or timeout) 5 = communication not correctly concluded because of ModbusMaster requests queue saturation
value	CJ_ BIT	0-1	Read value

Description
The *ModbusMaster_ReadSingleInputStatus* library allows to read a value of an Input Status of a slave net node. This library implements the dispatch of a Modbus message wich implements the function code 0x02. To start the writing operation must set a trasition from “0” to “1” to the *enable* input: will be added in queue the writing request of the data present in that moment in the *value* input, to send to the *register* of slave *node*. Once added in queue the message will be processed and the answer will be analyzed monitoring the *status* output: when the output has the value 3 the slave response will be positive, instead if the output has the value 4 the response was negative or the slave didn’t response. Setting *enable* to 0 frees the queue for a next transmission. During all the communication period the *enable* input must be maintained 1.

Notes
It is possibile to use more times this library in the project. The library can be used only for Modbus Master.

ModbusMaster_ReadSingleInputRegister



Inputs	***Type***	***Limits***	***Description***
enable	CJ_ BIT	0-1	Enables the master/slave transmission process of a reading frame
register	CJ_WORD	1-65000	Identifier of the register that you want to access
slave	CJ_ BYTE	0-247	Slave node address. 0 = Broadcast.

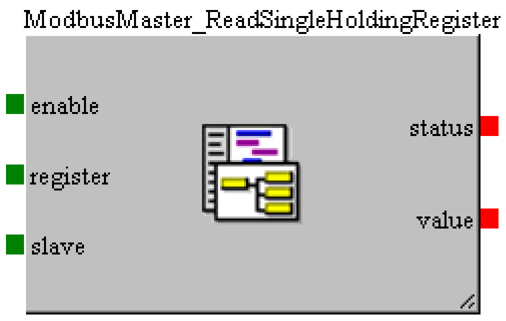
outputs	***Type***	***Limits***	***Description***
status	CJ_BYTE	0-5	Returns the master/slave communication status: 0 = free for a communication 1 = added a communication request 2 = communication in progress 3 = communication correctly completed 4 = communication not completed correctly (i.e. NAK reply or timeout) 5 = communication not correctly concluded because of ModbusMaster requests queue saturation
value	CJ_ WORD	0-65000	Read value

Description

The *ModbusMaster_ReadSingleInputRegister* library allows to write a value in a Input Register of a slave net node. This library implements the dispatch of a Modbus message wich implements the function code 0x04. To start the writing operation must set a trasition from “0” to “1” to the *enable* input: will be added in queue the writing request of the data present in that moment in the *value* input, to send to the *register* of slave *node*. Once added in queue the message will be processed and the answer will be analyzed monitoring the *status* output: when the output has the value 3 the slave response will be positive, instead if the output has the value 4 the response was negative or the slave didn’t response. Setting *enable* to 0 frees the queue for a next transmission. During all the communication period the *enable* input must be maintained 1.

Notes
It is possibile to use more times this library in the project. The library can be used only for Modbus Master. It is possible to write also more registers in succession through the libraries: • *ModbusMaster_Read10MultipleInputRegisters* • *ModbusMaster_Read100MultipleInputRegisters* They allow reading respectively, up to 10 and 100 holding registers in succession, selecting on input *num* the number of registers to read.

ModbusMaster_ReadSingleHoldingRegister



Input	***Type***	***Limits***	***Description***
enable	CJ_ BIT	0-1	Enable the master/slave transmission process of a Modbus reading frame
register	CJ_WORD	1-65000	Identifier of the registry that you want to access.
slave	CJ_ BYTE	0-247	Slave node address. 0 = Broadcast.

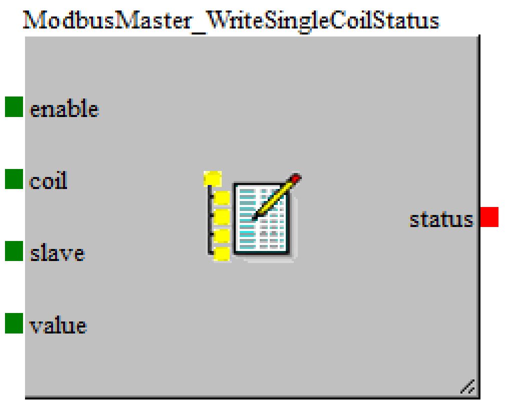
Outpu	***Type***	***Limits***	***Description***

status	CJ_BYTE	0-5	Returns the Master/Slave communication status: 0 = free for a communication 1 = added a communication request 2 = communication in progress 3 = communication correctly completed 4 = comunication not completed correctly (ie. NAK reply or timeout). 5 = communication not correctly concluded because of ModbusMaster requests queue saturation
value	CJ_ WORD	0-65000	Readed value.

Description
The *ModbusMaster_ReadSingleHoldingRegister* library allows to read a value in a Holding Register of a slave net node. This library implements the dispatch of a Modbus message wich implements the function code 0x03. To start the reading operation must set a trasition from “0” to “1” to the *enable* input: will be added in queue the reading request of the data present in the *register* of slave *node*. Once added in queue the message will be processed and the answer will be analyzed monitoring the *status* output: when the output has the value 3 the slave response will be positive and the value will be in the *value* output, instead if the output has the value 4 the response was negative or the slave didn’t response. Setting *enable* to 0 frees the queue for a next transmission. During all the communication period the *enable* input must be maintained to 1.

Notes
It is possibile to use more times this library in the project. The library can be used only for Modbus Master. It is also possible to read more registers in succession through libraries: • *ModbusMaster_Read10MultipleHoldingRegisters* • *ModbusMaster_Read100MultipleHoldingRegisters* They allow reading respectively, up to 10 and 100 holding registers in succession, selecting on input *num* the number of registers to read.

ModbusMaster_WriteSingleCoilStatus



Inputs	*Type*	*Limits*	*Description*
enable	CJ_ BIT	0-1	Enable the master/slave transmission process of a Modbus writing frame
coil	CJ_WORD	1-65000	Identifier of the registry that you want to access.
slave	CJ_ BYTE	0-247	Slave node address. 0 = Broadcast.
value	CJ_ BIT	0-1	Value that you want to write

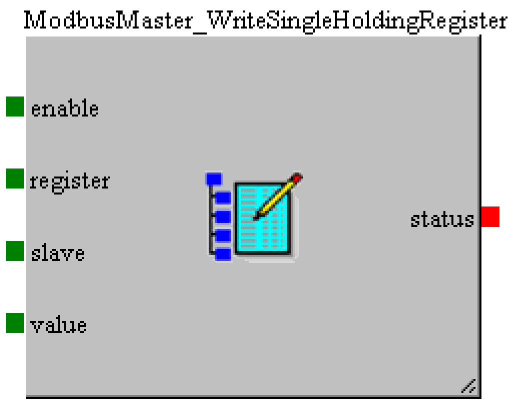
Output	*Type*	*Limits*	*Description*
status	CJ_BYTE	0-5	Returns the Master/Slave communication status: 0 = free for a communication 1 = added a communication request 2 = communication in progress 3 = communication correctly completed 4 = comunication not completed correctly (ie. NAK reply or timeout). 5 = communication not correctly concluded because of ModbusMaster requests queue saturation

Description

The *ModbusMaster_WriteSingleCoilStatus* library allows to write a value in a Coil of a slave net node. This library implements the dispatch of a Modbus message wich implements the function code 0x05.
To start the writing operation must set a trasition from “0” to “1” to the *enable* input: will be added in queue the writing request of the data present in that moment in the *value* input, to send to the *coil* of slave *node*. Once added in queue the message will be processed and the answer will be analyzed monitoring the *status* output: when the output has the value 3 the slave response will be positive, instead if the output has the value 4 the response was negative or the slave didn’t response. Setting *enable* to 0 frees the queue for a next transmission. During all the communication period the *enable* input must be maintained 1.

Notes
It is possibile to use more times this library in the project. The library can be used only for Modbus Master.

ModbusMaster_WriteSingleHoldingRegister



Inputs	***Type***	***Limits***	***Description***
enable	CJ_ BIT	0-1	Enable the master/slave transmission process of a Modbus writing frame
register	CJ_WORD	1-65000	Identifier of the registry that you want to access.
slave	CJ_ BYTE	0-247	Slave node address. 0 = Broadcast.
value	CJ_ WORD	0-65000	Value that you want to write

Output	***Type***	***Limits***	***Description***

status	CJ_BYTE	0-5	Returns the Master/Slave communication status: 0 = free for a communication 1 = added a communication request 2 = communication in progress 3 = communication correctly completed 4 = comunication not completed correctly (ie. NAK reply or timeout). 5 = communication not correctly concluded because of ModbusMaster requests queue saturation

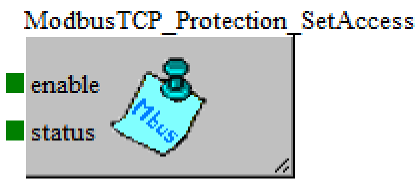
Description
The *ModbusMaster_WriteSingleHoldingRegister* library allows to write a value in a Holding Register of a slave net node. This library implements the dispatch of a Modbus message wich implements the function code 0x06. To start the writing operation must set a trasition from “0” to “1” to the *enable* input: will be added in queue the writing request of the data present in that moment in the *value* input, to send to the *register* of slave *node*. Once added in queue the message will be processed and the answer will be analyzed monitoring the *status* output: when the output has the value 3 the slave response will be positive, instead if the output has the value 4 the response was negative or the slave didn’t response. Setting *enable* to 0 frees the queue for a next transmission. During all the communication period the *enable* input must be maintained 1.

Notes
It is possibile to use more times this library in the project. The library can be used only for Modbus Master. It is possible to write also more registers in succession through the libraries: • *ModbusMaster_Write10MultipleHoldingRegisters* • *ModbusMaster_Write100MultipleHoldingRegisters* They allow reading respectively, up to 10 and 100 holding registers in succession, selecting on input *num* the number of registers to read. These libraries implements the dispatch of a Modbus message wich implements the function code 0x10.

2.16 Modbus TCP

These libraries allow the configuration and management of the standard MODBUS protocol via TCP. Please refer to the controller hardware documentation to check the compatibility with this protocol.

ModbusTCP_Protection_SetAccess

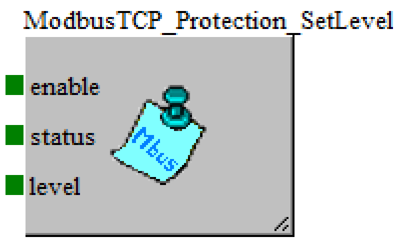


Input	*Type*	*Limits*	*Description*
enable	CJ_ BIT	0-1	Enable the modification of the reading access
status	CJ_ BIT	0-1	Sets the reading access status: 0=Not accessible 1=Accessible

Description
The *ModbusTCP_Protection_SetAccess* library allows setting the reading protection for the relative Modbus uart; the modification is permorfed with enable=1, otherwise if enable=0 is not made any changes.

Notes
The library can be used for Modubus Master and Modbus Slave.

ModbusTCP_Protection_SetLevel

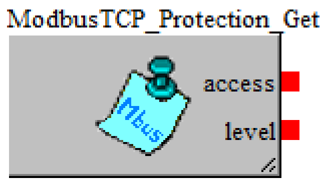


Input	*Type*	*Limits*	*Description*
enable	CJ_ BIT	0-1	Enable the modification of the level
status	CJ_ BIT	0-1	Sets the lock/unlock of the level: 0=Level locked 1=Level unlocked
level	CJ_ BYTE	1-5	Level to lock/unlock

Description
The *ModbusTCP_Protection_SetLevel* library allows setting the lock/unlock (writing protection) of the level passed as input on the relative Modbus uart; the modification is permorfed with enable=1, otherwise if enable=0 is not made any changes.

Notes
The library can be used for Modubus Master and Modbus Slave.

ModbusTCP_Protection_Get

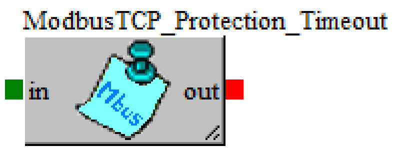


Output	***Tipo***	***Limiti***	***Descrizione***
access	CJ_ BIT	0-1	Staus of reading access protection: 0=Not accessible 1=Accessible
level	CJ_ BYTE	0-5	Max unlocked level

Description
The *ModbusTCP_Protection_Get* library returns the status of the Modbus protection. The output *access* takes these values: • 0= Reading access protection not active. • 1= Reading access protection active. The output *level* returns the unlocked level (The level lock management is hierarchical, then the previous levels are unlocked).

Notes
The library can be used for Modubus Master and Modbus Slave.

ModbusTCP_Protection_Timeout



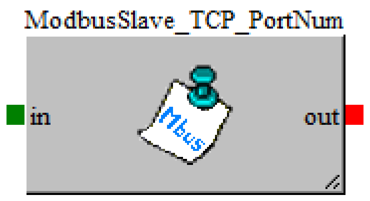
Input	***Type***	***Limits***	***Description***
in	CJ_ WORD	0-65535	Unlock level timeout (Seconds)

Output	***Type***	***Limits***	***Description***
in	CJ_ WORD	0-65535	Unlock level timeout (Seconds)

Description
The *ModbusTCP_Protection_Timeout* library allows setting the duration of the unlock of the levels, the count is recalculated at every access made on the serial. To set the default value, adjust the *value* property of the *P_Modbus_ProtectionTimeout* parameter in the library. If the value is 0 levels are always unlocked.

Notes
The library is template type in nature. It is essential that this library be used only once per project. The library can be used for Modubus Master and Modbus Slave.

ModbusSlave_TCP_PortNum



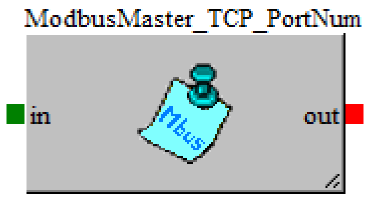
Ingresso	***Tipo***	***Limiti***	***Descrizione***
in	CJ_WORD	0..65535	Communication port number Modbus slave via TCP.

Uscita	***Tipo***	***Limiti***	***Descrizione***
out	CJ_WORD	0..65535	Communication port number Modbus slave via TCP.

Descrizione
The library allows setting the port number for the communication with the protocol Modbus Salve over TCP-IP. To set the default value, adjust the *value* property of the parameter *P_MbSlaveTCPIP_PortNum* in the library.

Note
The library is template type in nature. The library has meaning only if Ethernet protocol is selected on the relative port and the protocol Modbus Slave over TCP-IP is enabled.

ModbusMaster_TCP_PortNum



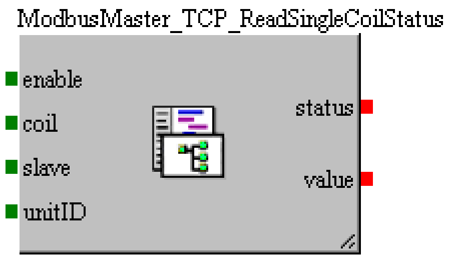
Ingresso	*Tipo*	*Limiti*	*Descrizione*
in	CJ_WORD	0..65535	Modbus master connection port number

Uscita	*Tipo*	*Limiti*	*Descrizione*
out	CJ_WORD	0..65535	Modbus master connection port number

Descrizione
The library allows setting the port number for the communication with the protocol Modbus Master over TCP-IP. To set the default value, adjust the *value* property of the parameter *P_ MbMasterTCPIP _PortNum* in the library.

Note
The library is template type in nature. The library has meaning only if Ethernet protocol is selected on the relative port and the protocol Modbus Master over TCP-IP is enabled.

ModbusMaster_TCP_ReadSingleCoilStatus



Inputs	*Type*	*Limits*	*Description*
enable	CJ_ BIT	0-1	Enables the master/slave transmission process of a reading frame
coil	CJ_WORD	1-65000	Identifier of the coil that you want to access
slave	CJ_IP_ADDRESS	0.0.0.0 255.255.255.255	Slave node address

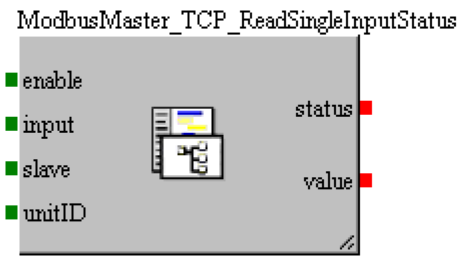
unitID	CJ_BYTE	1-247	Remote address to select the slave in a RS485 network accessible through a gateway (Note: that identifier 255 is generally used to address the gateway device itself)

outputs	***Type***	***Limits***	***Description***
status	CJ_BYTE	0-5	Returns the master/slave communication status: 0 = free for a communication 1 = added a communication request 2 = communication in progress 3 = communication correctly completed 4 = communication not completed correctly (i.e. NAK reply or timeout) 5 = communication not correctly concluded because of ModbusMaster requests queue saturation
value	CJ_ BIT	0-1	Read value

Description
The *ModbusMaster_TCP_ReadSingleCoilStatus* library allows to write a value in a Coil of a slave net node. This library implements the dispatch of a Modbus TCP message wich implements the function code 0x01. To start the writing operation must set a trasition from “0” to “1” to the *enable* input: will be added in queue the writing request of the data present in that moment in the *value* input, to send to the *coil* of slave *node*. Once added in queue the message will be processed and the answer will be analyzed monitoring the *status* output: when the output has the value 3 the slave response will be positive, instead if the output has the value 4 the response was negative or the slave didn’t response. Setting *enable* to 0 frees the queue for a next transmission. During all the communication period the *enable* input must be maintained 1.

Notes
It is possibile to use more times this library in the project. The library can be used only for Modbus Master via TCP in the controllers with Ethernet connection enabled.

ModbusMaster_TCP_ReadSingleInputStatus



Inputs	*Type*	*Limits*	*Description*
enable	CJ_ BIT	0-1	Enables the master/slave transmission process of a reading frame
inpu	CJ_WORD	1-65000	Identifier of the input status that you want to access
slave	CJ_IP_ADDRESS	0.0.0.0 255.255.255.255	Slave node address
unitID	CJ_BYTE	1-247	Remote address to select the slave in a RS485 network accessible through a gateway (Note: that identifier 255 is generally used to address the gateway device itself)

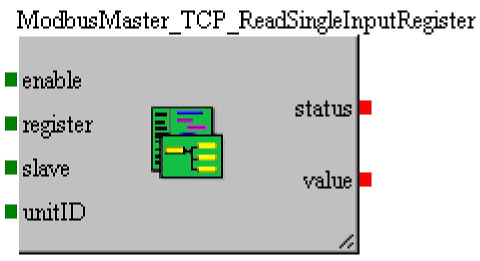
outputs	*Type*	*Limits*	*Description*
status	CJ_BYTE	0-5	Returns the master/slave communication status: 0 = free for a communication 1 = added a communication request 2 = communication in progress 3 = communication correctly completed 4 = communication not completed correctly (i.e. NAK reply or timeout) 5 = communication not correctly concluded because of ModbusMaster requests queue saturation
value	CJ_ BIT	0-1	Read value

Description

The *ModbusMaster_TCP_ReadSingleInputStatus* library allows to read a value of an Input Status of a slave net node. This library implements the dispatch of a Modbus TCP message wich implements the function code 0x02. To start the writing operation must set a trasition from “0” to “1” to the *enable* input: will be added in queue the writing request of the data present in that moment in the *value* input, to send to the *register* of slave *node*. Once added in queue the message will be processed and the answer will be analyzed monitoring the *status* output: when the output has the value 3 the slave response will be positive, instead if the output has the value 4 the response was negative or the slave didn’t response. Setting *enable* to 0 frees the queue for a next transmission. During all the communication period the *enable* input must be maintained 1.

Notes
It is possibile to use more times this library in the project. The library can be used only for Modbus Master via TCP in the controllers with Ethernet connection enabled.

ModbusMaster_TCP_ReadSingleInputRegister



Inputs	***Type***	***Limits***	***Description***
enable	CJ_ BIT	0-1	Enables the master/slave transmission process of a reading frame
register	CJ_WORD	1-65000	Identifier of the register that you want to access
slave	CJ_IP_ADDRESS	0.0.0.0 255.255.255.255	Slave node address
unitID	CJ_BYTE	1-247	Remote address to select the slave in a RS485 network accessible through a gateway (Note: that identifier 255 is generally used to address the gateway device itself)

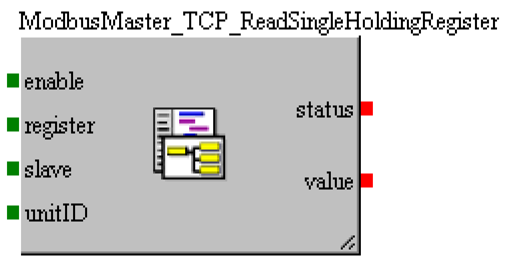
outputs	***Type***	***Limits***	***Description***

status	CJ_BYTE	0-5	Returns the master/slave communication status: 0 = free for a communication 1 = added a communication request 2 = communication in progress 3 = communication correctly completed 4 = communication not completed correctly (i.e. NAK reply or timeout) 5 = communication not correctly concluded because of ModbusMaster requests queue saturation
value	CJ_ WORD	0-65000	Read value

Description
The *ModbusMaster_TCP_ReadSingleInputRegister* library allows to write a value in a Input Register of a slave net node. This library implements the dispatch of a Modbus TCP message wich implements the function code 0x04. To start the writing operation must set a trasition from “0” to “1” to the *enable* input: will be added in queue the writing request of the data present in that moment in the *value* input, to send to the *register* of slave *node*. Once added in queue the message will be processed and the answer will be analyzed monitoring the *status* output: when the output has the value 3 the slave response will be positive, instead if the output has the value 4 the response was negative or the slave didn’t response. Setting *enable* to 0 frees the queue for a next transmission. During all the communication period the *enable* input must be maintained 1.

Notes
It is possibile to use more times this library in the project. The library can be used only for Modbus Master via TCP in the controllers with Ethernet connection enabled. It is possible to write also more registers in succession through the libraries: • *ModbusMaster_TCP_Read10MultipleInputRegisters* • *ModbusMaster_TCP_Read100MultipleInputRegisters* They allow reading respectively, up to 10 and 100 holding registers in succession, selecting on input *num* the number of registers to read.

ModbusMaster_TCP_ReadSingleHoldingRegister



Input	*Type*	*Limits*	*Description*
enable	CJ_ BIT	0-1	Enable the master/slave transmission process of a Modbus reading frame
register	CJ_WORD	1-65000	Identifier of the registry that you want to access.
slave	CJ_IP_ADDRESS	0.0.0.0 255.255.255.255	Slave node address
unitID	CJ_BYTE	1-247	Remote address to select the slave in a RS485 network accessible through a gateway (Note: that identifier 255 is generally used to address the gateway device itself)

Output	*Type*	*Limits*	*Description*
status	CJ_BYTE	0-5	Returns the Master/Slave communication status: 0 = free for a communication 1 = added a communication request 2 = communication in progress 3 = communication correctly completed 4 = comunication not completed correctly (ie. NAK reply or timeout). 5 = communication not correctly concluded because of ModbusMaster requests queue saturation
value	CJ_ WORD	0-65000	Readed value.

Description
The *ModbusMaster_TCP_ReadSingleHoldingRegister* library allows to read a value in a Holding Register of a slave net node. This library implements the dispatch of a Modbus TCP message wich implements the function code 0x03. To start the reading operation must set a trasition from “0” to “1” to the *enable* input: will be added in queue the reading request of the data present in the *register* of slave *node*. Once added in queue the message will be processed and the answer will be analyzed monitoring the *status* output: when the output has the value 3 the slave response will be positive and the value will be in the *value* output, instead if the output has the value 4 the response was negative or the slave didn’t response. Setting *enable* to 0 frees the queue for a next transmission. During all the communication period the *enable* input must be maintained to 1.

Notes

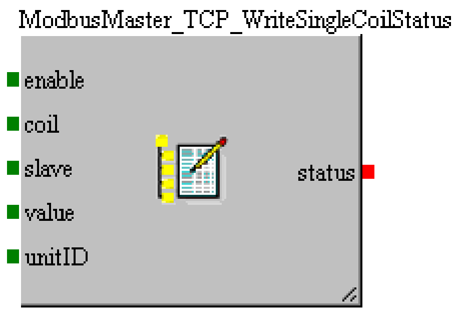
It is possibile to use more times this library in the project.
The library can be used only for Modbus Master via TCP in the controllers with Ethernet connection enabled.

It is also possible to read more registers in succession through libraries:

- *ModbusMaster_TCP_Read10MultipleHoldingRegisters*
- *ModbusMaster_TCP_Read100MultipleHoldingRegisters*

They allow reading respectively, up to 10 and 100 holding registers in succession, selecting on input *num* the number of registers to read.

ModbusMaster_TCP_WriteSingleCoilStatus



Inputs	*Type*	*Limits*	*Description*
enable	CJ_ BIT	0-1	Enable the master/slave transmission process of a Modbus writing frame
coil	CJ_WORD	1-65000	Identifier of the registry that you want to access.
slave	CJ_IP_ADDRESS	0.0.0.0 255.255.255.255	Slave node address
value	CJ_ BIT	0-1	Value that you want to write
unitID	CJ_BYTE	1-247	Remote address to select the slave in a RS485 network accessible through a gateway (Note: that identifier 255 is generally used to address the gateway device itself)

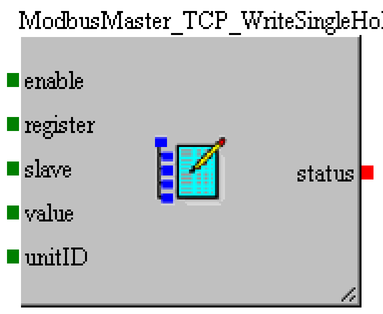
Output	*Type*	*Limits*	*Description*

status	CJ_BYTE	0-5	Returns the Master/Slave communication status: 0 = free for a communication 1 = added a communication request 2 = communication in progress 3 = communication correctly completed 4 = comunication not completed correctly (ie. NAK reply or timeout). 5 = communication not correctly concluded because of ModbusMaster requests queue saturation

Description
The *ModbusMaster_TCP_WriteSingleCoilStatus* library allows to write a value in a Coil of a slave net node. This library implements the dispatch of a Modbus TCP message wich implements the function code 0x05. To start the writing operation must set a trasition from "0" to "1" to the *enable* input: will be added in queue the writing request of the data present in that moment in the *value* input, to send to the *coil* of slave *node*. Once added in queue the message will be processed and the answer will be analyzed monitoring the *status* output: when the output has the value 3 the slave response will be positive, instead if the output has the value 4 the response was negative or the slave didn't response. Setting *enable* to 0 frees the queue for a next transmission. During all the communication period the *enable* input must be maintained 1.

Notes
It is possibile to use more times this library in the project. The library can be used only for Modbus Master via TCP in the controllers with Ethernet connection enabled.

ModbusMaster_TCP_WriteSingleHoldingRegister



Inputs	***Type***	***Limits***	***Description***
enable	CJ_ BIT	0-1	Enable the master/slave transmission process of a Modbus writing frame
register	CJ_WORD	1-65000	Identifier of the registry that you want to access.

slave	CJ_IP_ADDRESS	0.0.0.0 255.255.255.255	Slave node address
value	CJ_ WORD	0-65000	Value that you want to write
unitID	CJ_BYTE	1-247	Remote address to select the slave in a RS485 network accessible through a gateway (Note: that identifier 255 is generally used to address the gateway device itself)

Output	***Type***	***Limits***	***Description***
status	CJ_BYTE	0-5	Returns the Master/Slave communication status: 0 = free for a communication 1 = added a communication request 2 = communication in progress 3 = communication correctly completed 4 = comunication not completed correctly (ie. NAK reply or timeout). 5 = communication not correctly concluded because of ModbusMaster requests queue saturation

Description
The *ModbusMaster_TCP_WriteSingleHoldingRegister* library allows to write a value in a Holding Register of a slave net node. This library implements the dispatch of a Modbus TCP message wich implements the function code 0x06. To start the writing operation must set a trasition from “0” to “1” to the *enable* input: will be added in queue the writing request of the data present in that moment in the *value* input, to send to the *register* of slave *node*. Once added in queue the message will be processed and the answer will be analyzed monitoring the *status* output: when the output has the value 3 the slave response will be positive, instead if the output has the value 4 the response was negative or the slave didn’t response. Setting *enable* to 0 frees the queue for a next transmission. During all the communication period the *enable* input must be maintained 1.

Notes

It is possibile to use more times this library in the project.
The library can be used only for Modbus Master via TCP in the controllers with Ethernet connection enabled.

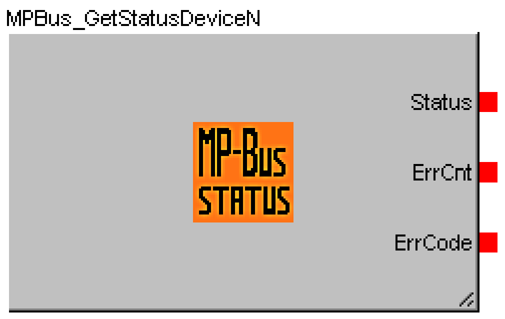
It is possible to write also more registers in succession through the libraries:

- *ModbusMaster_TCP_Write10MultipleHoldingRegisters*
- *ModbusMaster_TCP_Write100MultipleHoldingRegisters*

They allow reading respectively, up to 10 and 100 holding registers in succession, selecting on input *num* the number of registers to read. These libraries implements the dispatch of a Modbus message wich implements the function code 0x10.

2.17 MPBus Belimo

MPBus_GetStatusDeviceN

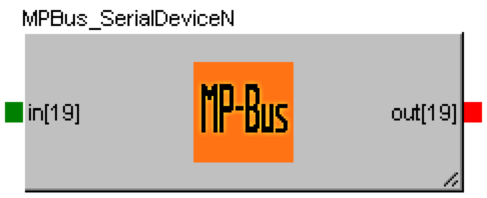


Output	*Type*	*Limits*	*Description*
Status	CJ_BYTE	Out: 0..5	Returns the status of the device N of the MPBus network 0 = device not used 1 = device not found 2 = device offline 3 = device in inizialization 4 = device in configuration 5 = device online
ErrCnt	CJ_BYTE	Out: 0..255	Returns the communication errors counter with the nth device.
ErrCode	CJ_BYTE	Out: 0..255	Returns the error code in case of MP-Bus command not approved or not feasible (look at Belimo's codes)

Description
The library allows monitoring the status of the nth device of the MP-Bus network.

Notes
It is necessary to use these libraries once a project.

MPBus_SerialDeviceN



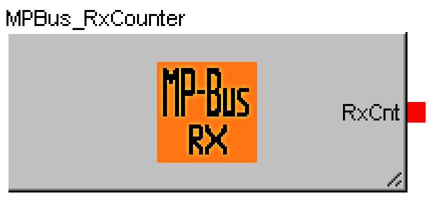
Input	*Type*	*Limits*	*Description*
In[19]	CJ_BYTE	In[x]: 0..9	Sets the serial number of the device in order to set it in the MP-Bus network [XXXXX-YYYYY-DDD-ZZZ]

Output	*Type*	*Limits*	*Description*
out	CJ_BYTE	Out[x]: 0..9	Returns the serial number of the device

Description
The library allows configuring the serial number of the device in order to connect and configure it properly in the MP-Bus network. To set the default value operate on the property value of the parameter *Serial_DeviceN* there is into the library.

Notes
The library is a template. It is necessary to use these libraries once a project.

MPBus_RxCounter

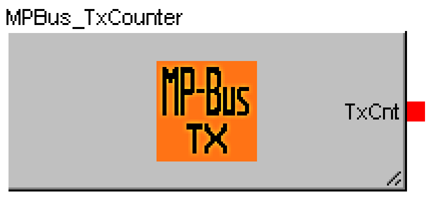


Output	*Type*	*Limits*	*Description*
out	CJ_BYTE	Out: 0..255	The number of packages received by the MP-Bus network

Description
The library allows monitoring the status of the received packages of the MP-Bus network.

Notes
This library must be used joined to the *MPBus_TxCounter* library: the network works in normal condition and on condition that there are not lost packages, or if the difference between the transmitted packages and the received ones differs of few unities.

MPBus_TxCounter



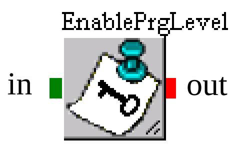
Output	*Type*	*Limits*	*Description*
out	CJ_BYTE	Out: 0..255	The number of packages received by the MP-Bus network

Description
The library allows monitoring the number of transmitted packages.

Notes
This library must be used joined to the *MPBus_RxCounter* library: the network works in normal condition and on condition that there are not lost packages, or if the difference between the transmitted packages and the received ones differs of few unities.

2.18 Password

EnablePrgLevel



Input	*Type*	*Limits*	*Description*
in	CJ_BIT	0-1	Sets the value of the “EnablePrgLevel” driver parameter

Output	*Type*	*Limits*	*Description*
out	CJ_BIT	0-1	Reads the value of the “EnablePrgLevel” driver parameter

Description

The *EnablePrgLevel* library allows automatically enabling the request to send the first page of levels 1-5, whenever the relevant command is intercepted (see the command list in the UNI-PRO software manual).
Some of these commands are associated with specific terminal key combinations:

- pressing and holding the “PRG” key for approx. 3 seconds: *PrgLevel 1 command*
- pressing and holding the “ENTER+ESC” keys for approx. 3 seconds: PrgLevel 2 command
- pressing and holding the “LEFT+RIGHT” keys for approx. 3 seconds: PrgLevel 3 command

If ‘1’, then by using a combination of the above mentioned keys, it is possible to access the first page of levels 1-3. If ‘0’ then this mode is deactivated.
To set the default value, adjust the *value* property of the *Enable_PrgLevel* parameter in the library.

Notes

The library is template type in nature.
It is essential that this library be used only once per project.

PasswordIndependent

PasswordIndependent
in
out

Input	*Type*	*Limits*	*Description*
in	CJ_BIT	0-1	Sets the value of the "Password Independent" driver parameter

Output	*Type*	*Limits*	*Description*
out	CJ_BIT	0-1	Reads the value of the "PasswordIndependent" driver parameter

Description
The *PasswordIndependent* library permits management of the page level passwords, either independently or hierarchically (higher level passwords also have access to lower levels). If this parameter assumes a value of '1' then the levels will be considered to be independent. To set the default value, adjust the *value* property of the *Password_Independent* parameter in the library.

Notes
The library is template type in nature. It is essential that this library be used only once per project.

PasswordTimeout

PasswordTimeout
in
out

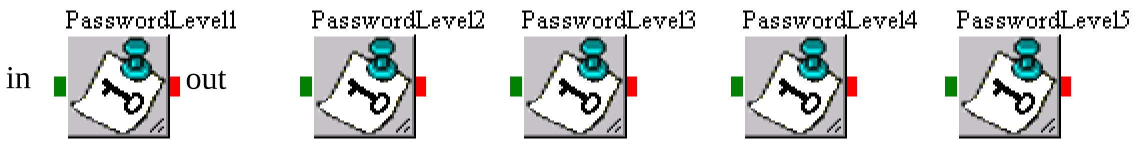
Input	*Type*	*Limits*	*Description*
in	CJ_BYTE	0..240	Sets the number of seconds to configure the timeout delay

Output	***Type***	***Limits***	***Description***
out	CJ_BYTE	0..240	Returns the number of seconds set for the timeout delay

Description
The *PasswordTimeout* library permits setting/returning the number of seconds of keyboard inactivity after which entering the password once more will be required to gain access to the various levels of the system.

Notes
The library is template type in nature. It is essential that this library be used only once per project. For further details on configuring levels and passwords, please refer to the UNI-PRO software manual.

PasswordLevel1 – PasswordLevel5



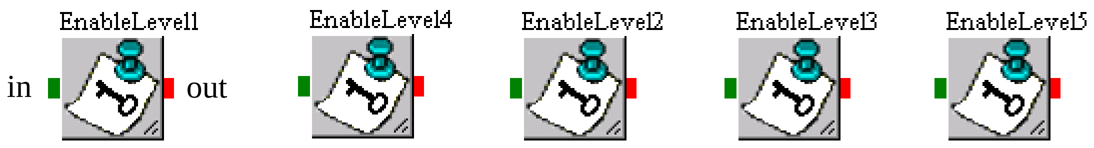
Input	***Type***	***Limits***	***Description***
in	CJ_SHORT	-32768..32767	Sets the password for the various levels of protection

Output	***Type***	***Limits***	***Description***
out	CJ_SHORT	-32768..32767	Reads the values of the passwords for the various levels of protection

Description
The *PasswordLeveln* libraries permit reading and writing the passwords pertaining to the various levels of protection. Each password may assume values within the range established by the CJ_SHORT data type (from -32768 to 32767). The value 0 corresponds to no protection for that level. To set the default password value for each level, adjust the *value* property of the *Password_Leveln* parameter in the library.

Notes
The library is template type in nature. It is essential that this library be used only once per project. For further details on configuring levels and passwords, please refer to the software manual.

EnableLevel1 – EnableLevel5



Input	***Type***	***Limits***	***Description***
in	CJ_BIT	0-1	Sets enabling for level n

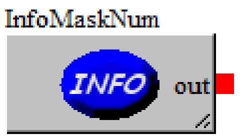
Output	***Type***	***Limits***	***Description***
out	CJ_BIT	0-1	Returns enabling for level n

Description
The *EnableLeveln* libraries permit setting the enabling status in relation to the various levels of protection. To set the default password value for each level, adjust the *value* property of the *Enable_Leveln* parameter in the library. If ‘1’ then the level is accessible (optionally password protected). If ‘0’ then the level is not accessible.

Notes
The library is template type in nature. It is essential that this library be used only once per project. For further details on configuring levels and passwords, please refer to the software manual.

2.19 System

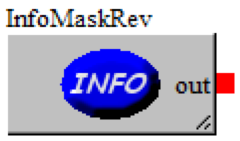
InfoMaskNum



Output	*Type*	*Limits*	*Description*
out	CJ_WORD	0..65535	Returns the firmware number

Description
The *InfoMaskNum* library has the purpose of returning the firmware code of the hardware being used.

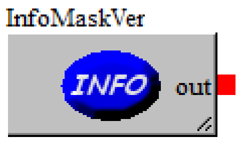
InfoMaskRev



Output	*Type*	*Limits*	*Description*
out	CJ_WORD	0..65535	Returns the firmware revision number

Description
The *InfoMaskRev* library has the purpose of returning the firmware revision number of the hardware being used.

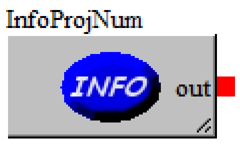
InfoMaskVer



Output	*Type*	*Limits*	*Description*
out	CJ_WORD	0..65535	Returns the firmware version

Description
The *InfoMaskVer* library has the purpose of returning the firmware version of the hardware being used.

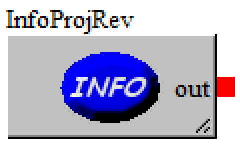
InfoProjNum



Output	*Type*	*Limits*	*Description*
out	CJ_WORD	0..65535	Returns the number of the current project

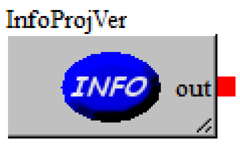
Description
The *InfoProjNum* library has the purpose of returning the number assigned to the current project.

InfoProjRev



Output	*Type*	*Limits*	*Description*
out	CJ_WORD	0..65535	Returns the current project's revision number
Description			
The *InfoProjRev* library has the purpose of returning the revision number of the current project.			

InfoProjVer



Output	*Type*	*Limits*	*Description*
out	CJ_WORD	0..65535	Returns the current project version

Description
The *InfoProjVer* library has the purpose of returning the current project version.

E2_GetStatus

E2_GetStatus
out

Output	*Type*	*Limits*	*Description*
out	CJ_SHORT	0..3	Returns a status code for the E2 memory

Description
The *E2_GetStatus* library has the purpose of returning a code representing the status of the controller EEPROM (E2) at the time the request is made. The codes returned are the following: ▪ *0 – CJ_E2_OK*. Operating correctly ▪ *1 – CJ_E2_READ_ERROR*. An E2 access error has been detected ▪ *2 – CJ_E2_WRITE_ERROR*. An E2 write error has been detected ▪ *3 – CJ_E2_CRC_ERROR* . Inconsistent data in the memory.

RCT_GetStatus

RTC_GetStatus
out

Output	*Type*	*Limits*	*Description*
out	CJ_SHORT	0..2	Returns a code for the status of the RTC

Description
The *RTC_GetStatus* library has the purpose of returning a code representing the status of the controller RTC at the time the request is made. The codes returned are the following: ➢ *0 : CJ_RCT_OK*. Operating correctly ➢ *1 : CJ_RCT_READ_ERROR*. A RTC access read error has been encountered ➢ *2 : CJ_RCT_LOW_VOLTAGE*. The RTC chip has passed the minimum threshold voltage necessary for maintaining information. The data present may no longer be valid.

Language

Language
in
out

Input	*Type*	*Limits*	*Description*
in	CJ_BYTE	0..255	The identifier of the language which it is desired to set

Output	*Type*	*Limits*	*Description*
out	CJ_BYTE	0..255	The identifier of the language which it is desired to set

Description
The *Language* library allows altering the system language. The language identifiers are associated as follows: ➢ 0: English ➢ 1: Italian ➢ 2: French ➢ 3: Spanish ➢ 4: German ➢ 5: Russian ➢ 6: Portuguese ➢ 7..255 : To be defined

LCDBacklight

LCDBacklight
in
out

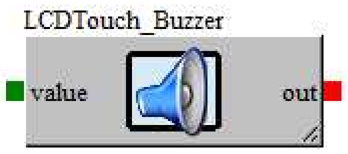
Input	*Type*	*Limits*	*Description*
in	CJ_BYTE	0..2	Sets the LCD backlight mode

Output	*Type*	*Limits*	*Description*
out	CJ_BYTE	0..2	Returns the LCD backlight mode

Description
The library *LCDBacklight* allows to modify the controller LCD backlight mode: ➢ 0: OFF. Backlight always off ➢ 1: ON. Backlight always on ➢ 2: TIME. The backlight is on for a defined time (default 60 seconds)

Note
The library is template type in nature. It is essential that this library be only used once per project.

LCDTouch_Buzzer



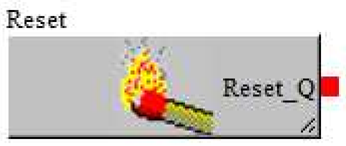
Input	***Type***	***Limits***	***Description***
in	CJ_BYTE	0..2	Sets the “beep” mode for the touchscreen display

Output	***Type***	***Limits***	***Description***
out	CJ_BYTE	0..2	Returns the “beep” mode for the touchscreen display

Description
The library *LCDTouch_Buzzer* allows to modify the “beep” mode when using the touchscreen: ➢ 0: No beep ➢ 1: Beep on always area ➢ 2: Beep only for sensible area

Note
The library is template type in nature. It is essential that this library be only used once per project.

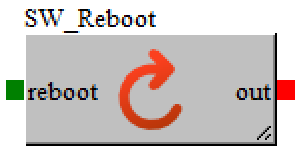
Reset



Output	***Type***	***Limits***	***Description***
Reset_Q	CJ_BIT	0-1	Returns the Reset status. Returns ‘1’ if there is a Reset, otherwise the output is ‘0’

Description
The *Reset* library permits performing actions on the *reset* status of the system. This state is an initial stage which the system processes prior to initiating the entire application.

SW_Reboot



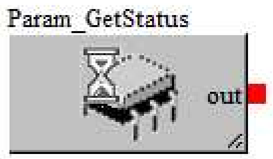
Input	***Type***	***Limits***	***Description***
reboot	CJ_BIT	0..1	When it becomes=1 software reboot starts.

Output	***Type***	***Limits***	***Description***
out	CJ_BIT	0..1	Rturn the rebbot status.

Description
The *SW_Reboot* library allows a soft reset of the application. A software reset is performed when the *reboot* input changes from "0" to "1", at reset the output *out* becomes equal to "1" for that loop of main.

Note
It is essential that this library be only used once per project.

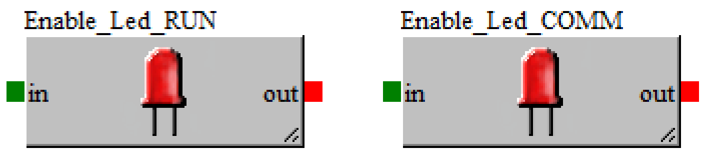
Param_GetStatus



Output	***Type***	***Limits***	***Description***
out	CJ_Byte	0-1	Returns the status of the parameter area. 0 = idle or upload finished with success 1 = one of these conditions: - Init & check datalog memory - upload in progress - wait for Reset

Description
The *Param_GetStatus* library allows you to check the status of the parameter areas in memory; in fact, at the first reboot, after a compilation or when the default reset command is used, the times are long, especially if the parameters are many and it is useful to know when this activity is finished.

Enable_Led_RUN e Enable_Led_COMM



Input	***Type***	***Limits***	***Description***
in	CJ_BIT	0..1	Enable/Disable controller diagnostic led management.

Output	***Type***	***Limits***	***Description***
out	CJ_BIT	0..1	Returns enable/disable status of the diagnostic led.

Description
The libraries *Enable_Led_RUN* e *Enable_Led_COMM* allow to enable/disable management of the diagnostic led RUN and COMMUNICATION (CAN).

Note
It is essential that this library be only used once per project.

3 STANDARD LIBRARIES

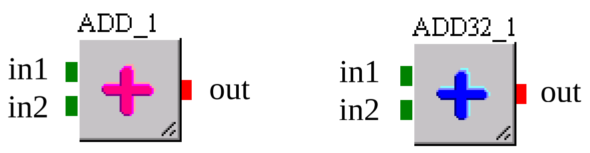
3.1 Arithmetic

These libraries allow performing arithmetical operations between 16 and 32 bit operands.
For each arithmetical library is tested the possible overflow condition; depending on the error type the relevant flags, wich can be read with the dedicated firmware functions, are defined:

- *CJ_BIT CJ_Math_Error_Read (void)*
- *CJ_BIT CJ_DivByZero_Error_Read (void)*
- *CJ_BIT CJ_Overflow_Error_Read (void)*
- *CJ_BIT CJ_Underflow_Error_Read (void)*
- *CJ_BIT CJ_NaN_Error_Read (void)*

If the function returns "1" the related error is active.

ADD



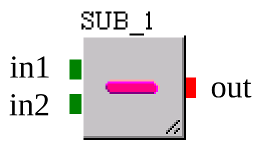
Input	*Type*	*Limits*	*Description*
in1	CJ_SHORT	-32768..32767	16 bit input
in1	CJ_LONG	-2147483648..2147483647	32 bit input
in2	CJ_SHORT	-32768..32767	16 bit input
in2	CJ_LONG	-2147483648..2147483647	32 bit input

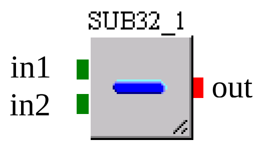
Output	*Type*	*Limits*	*Description*
out	CJ_SHORT	-32768..32767	Result of the 16 bit addition
out	CJ_LONG	-2147483648.. 2147483647	Result of the 32 bit addition

Description
The *ADD* and *ADD32* libraries allow performing 16 and 32 bit addition operations respectively.

Notes
To simplify operations, the libraries have been combined. *CJ_SHORT* type inputs/outputs refer to 16 bit operations. *CJ_LONG* type inputs/outputs refer to 32 bit operations.

SUB





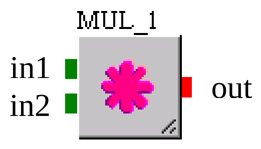
Input	***Type***	***Limits***	***Description***
in1 *in1*	CJ_SHORT CJ_LONG	-32768..32767 -2147483648..2147483647	16 bit input 32 bit input
in2 *in2*	CJ_SHORT CJ_LONG	-32768..32767 -2147483648..2147483647	16 bit input 32 bit input

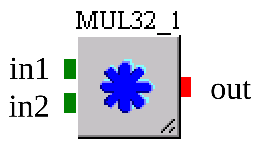
Output	***Type***	***Limits***	***Description***
out *out*	CJ_SHORT CJ_LONG	-32768..32767 -2147483648.. 2147483647	Result of the 16 bit subtraction Result of the 32 bit subtraction

Description
The *SUB* and *SUB32* libraries allow performing 16 and 32 bit subtraction operations respectively.

Notes
Operations are performed using the following syntax: in1 - in2. To simplify operations, the libraries have been combined. *CJ_SHORT* type inputs/outputs refer to 16 bit operations. *CJ_LONG* type inputs/outputs refer to 32 bit operations.

MUL





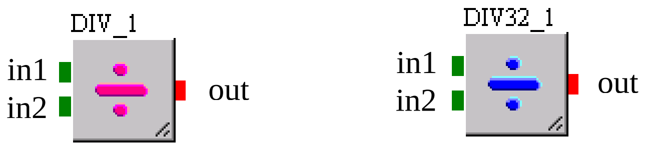
Input	***Type***	***Limits***	***Description***
in1 *in1*	CJ_SHORT CJ_LONG	-32768..32767 -2147483648..2147483647	16 bit input 32 bit input
in2 *in2*	CJ_SHORT CJ_LONG	-32768..32767 -2147483648..2147483647	16 bit input 32 bit input

Output	***Type***	***Limits***	***Description***
out *out*	CJ_SHORT CJ_LONG	-32768..32767 -2147483648.. 2147483647	Result of the 16 bit multiplication Result of the 32 bit multiplication

Description
The *MUL* and *MUL32* libraries allow performing 16 and 32 bit multiplication operations respectively.

Notes
To simplify operations, the libraries have been combined. *CJ_SHORT* type inputs/outputs refer to 16 bit operations. *CJ_LONG* type inputs/outputs refer to 32 bit operations.

DIV



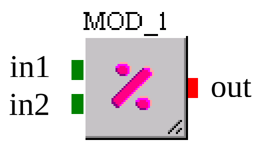
Input	***Type***	***Limits***	***Description***
in1 *in1*	CJ_SHORT CJ_LONG	-32768..32767 -2147483648..2147483647	16 bit input 32 bit input
in2 *in2*	CJ_SHORT CJ_LONG	-32768..32767 -2147483648..2147483647	16 bit input 32 bit input

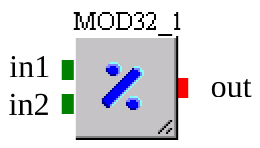
Output	***Type***	***Limits***	***Description***
out	CJ_SHORT	-32768..32767	Result of the 16 bit division
out	CJ_LONG	-2147483648.. 2147483647	Result of the 32 bit division

Description
The *DIV* and *DIV32* libraries allow performing 16 and 32 bit division operations respectively.

Notes
Operations are performed using the following syntax: in1 / in2. To simplify operations, the libraries have been combined. *CJ_SHORT* type inputs/outputs refer to 16 bit operations. *CJ_LONG* type inputs/outputs refer to 32 bit operations.

MOD





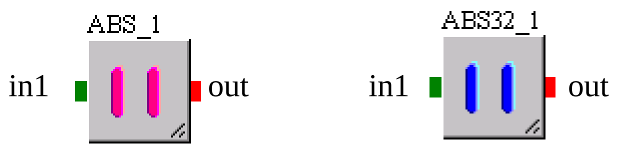
Input	***Type***	***Limits***	***Description***
in1	CJ_SHORT	-32768..32767	16 bit input
in1	CJ_LONG	-2147483648..2147483647	32 bit input
in2	CJ_SHORT	-32768..32767	16 bit input
in2	CJ_LONG	-2147483648..2147483647	32 bit input

Output	***Type***	***Limits***	***Description***
out	CJ_SHORT	-32768..32767	The remainder from the division between 16 bit inputs
out	CJ_LONG	-2147483648.. 2147483647	The remainder from the division between 32 bit inputs

Description
The *MOD* and *MOD32* libraries allow obtaining the remainder from the division between two 16 and 32 bit inputs, respectively.

Notes
Operations are performed using the following syntax: in1 MOD in2. To simplify operations, the libraries have been combined. *CJ_SHORT* type inputs/outputs refer to 16 bit operations. *CJ_LONG* type inputs/outputs refer to 32 bit operations.

ABS



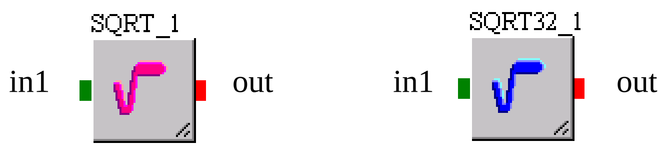
Input	***Type***	***Limits***	***Description***
in1	CJ_SHORT	-32768..32767	16 bit input
in1	CJ_LONG	-2147483648..2147483647	32 bit input

Output	***Type***	***Limits***	***Description***
out	CJ_SHORT	0..32767	Absolute 16 bit value
out	CJ_LONG	0.. 2147483647	Absolute 32 bit value

Description
The *ABS* and *ABS32* libraries allow performing *in* input 16 and 32 bit absolute value operations respectively.

Notes
To simplify operations, the libraries have been combined. *CJ_SHORT* type inputs/outputs refer to 16 bit operations. *CJ_LONG* type inputs/outputs refer to 32 bit operations.

SQRT



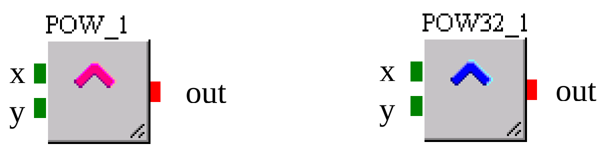
Input	*Type*	*Limits*	*Description*
in1	CJ_SHORT	0..32767	16 bit input
in1	CJ_LONG	0..2147483647	32 bit input

Output	*Type*	*Limits*	*Description*
out	CJ_SHORT	0..181	16 bit square root
out	CJ_LONG	0.. 46340	32 bit square root

Description
The *SQRT* and *SQRT32* libraries allow performing 16 and 32 bit square root operations respectively on *in1* input data.

Notes
To simplify operations, the libraries have been combined. *CJ_SHORT* type inputs/outputs refer to 16 bit operations. *CJ_LONG* type inputs/outputs refer to 32 bit operations.

POW



Input	*Type*	*Limits*	*Description*
x	CJ_SHORT	-32768..32767	16 bit base input
x	CJ_LONG	-2147483648..2147483647	32 bit base input

y	CJ_SHORT	-32768..32767	16 bit exponent input
y	CJ_LONG	-2147483648..2147483647	32 bit exponent input

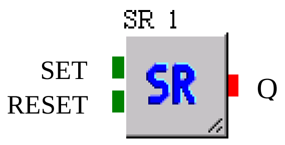
Output	***Type***	***Limits***	***Description***
out	CJ_SHORT	-32768..32767	16 bit result
out	CJ_LONG	-2147483648.. 2147483647	32 bit result

Description
The *POW* and *POW32* libraries allow performing 16 and 32 bit power raising operations respectively.

Notes
Operations are performed using the following syntax: x ^y. To simplify operations, the libraries have been combined. *CJ_SHORT* type inputs/outputs refer to 16 bit operations. *CJ_LONG* type inputs/outputs refer to 32 bit operations.

3.2 Bistable

SR

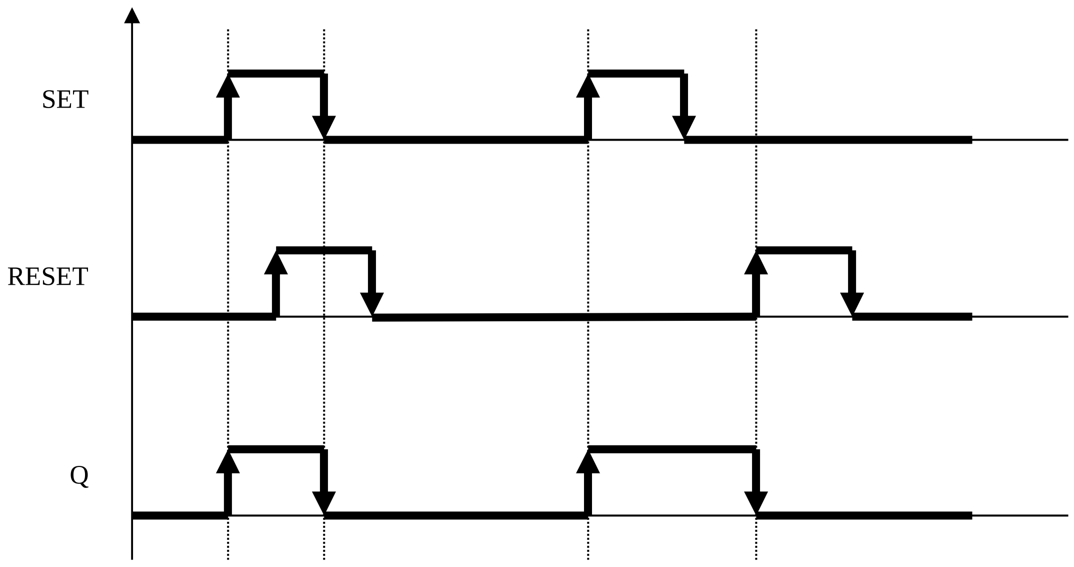


Input	*Type*	*Limits*	*Description*
SET	CJ_BIT	0-1	Set input
RESET	CJ_BIT	0-1	Reset input

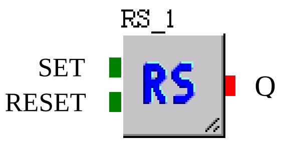
Output	*Type*	*Limits*	*Description*
Q	CJ_BIT	0-1	Output value

Description
The *SR* library performs SR functions, *i.e.* sets the output value according to the following logic: - Q is set to the value ‘1’ each time SET is equal to ‘1’ - Q is set to the value ‘0’ each time RESET is equal to ‘1’, except when the value of SET is ‘1’. In practice, the output is set giving priority to the value of SET.

Notes
Please refer to the following graph.



RS

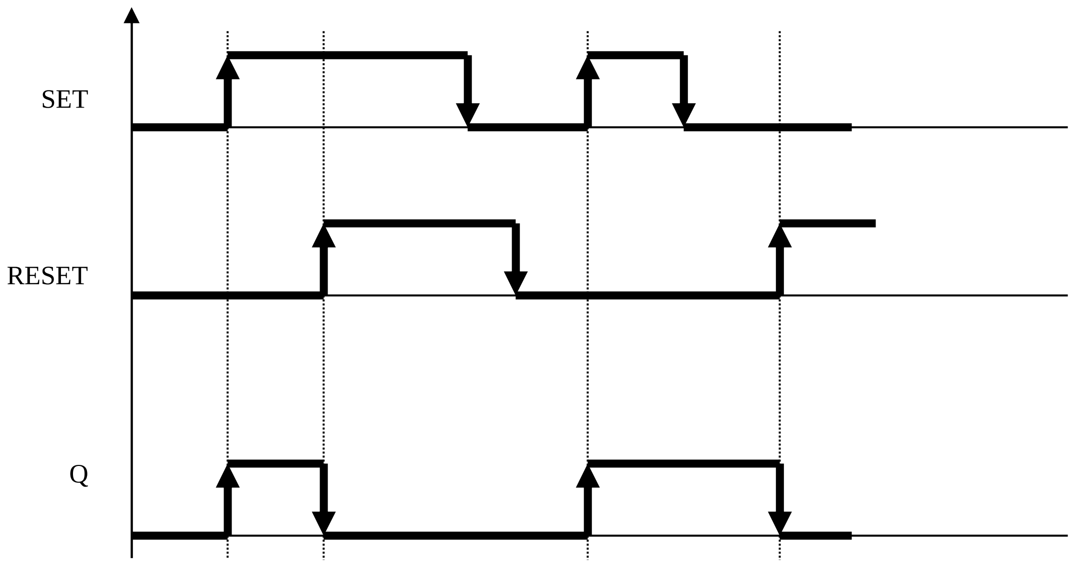


Input	*Type*	*Limits*	*Description*
SET	CJ_BIT	0-1	Set input
RESET	CJ_BIT	0-1	Reset input

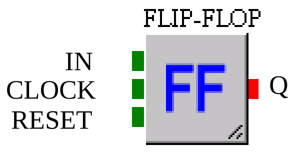
Output	*Type*	*Limits*	*Description*
Q	CJ_BIT	0-1	Output value

Description
The *RS* library performs RS functions, *i.e.* sets the output value according to the following logic: - Q is set to the value '1' each time RESET is equal to '1' - Q is set to the value '0' each time SET is equal to '1', except when the value of RESET is '1'. In practice, the output is set giving priority to the value of RESET.

Notes
Please refer to the following graph.



FLIP-FLOP



Input	*Type*	*Limits*	*Description*
IN	CJ_BIT	0-1	Set input
CLOCK	CJ_BIT	0-1	Clock
RESET	CJ_BIT	0-1	Reset input

Output	*Type*	*Limits*	*Description*
Q	CJ_BIT	0-1	Output value

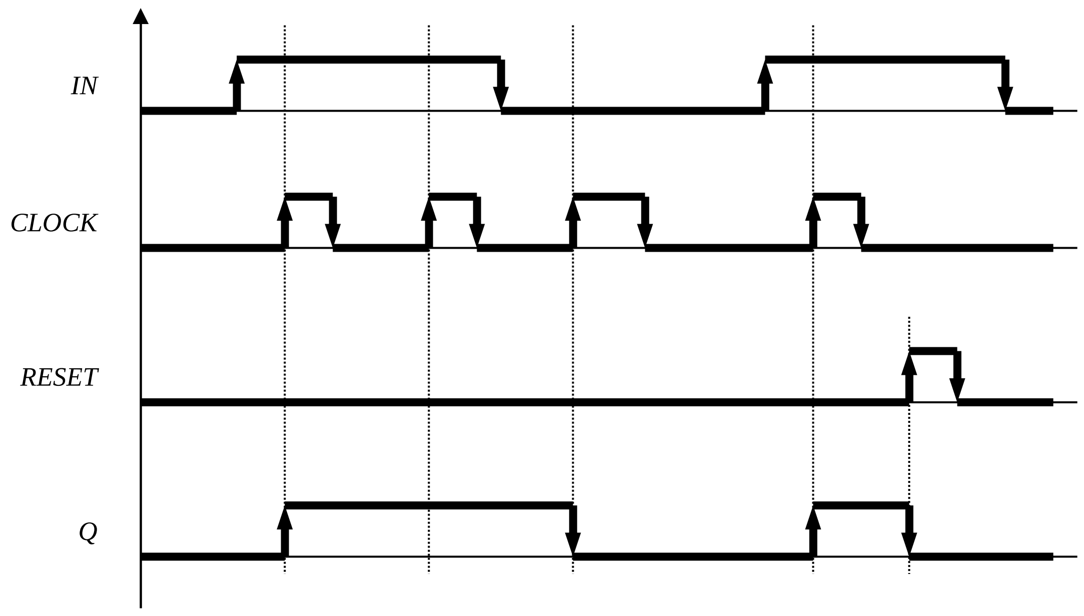
Description

The *FLIP_FLOP* library performs SR flip-flop functions, *i.e.* for each leading edge of the clock it stores the input value and returns it as output.
- Q is set to the value of IN for each CLOCK input leading edge
- Q is set to the value ‘0’ for each RESET input leading edge
- In all other cases Q returns the previous value.

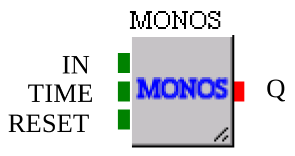
In practice, this function stores the input and maintains it until the conditions change.

Notes

Please refer to the following graph.



MONOS

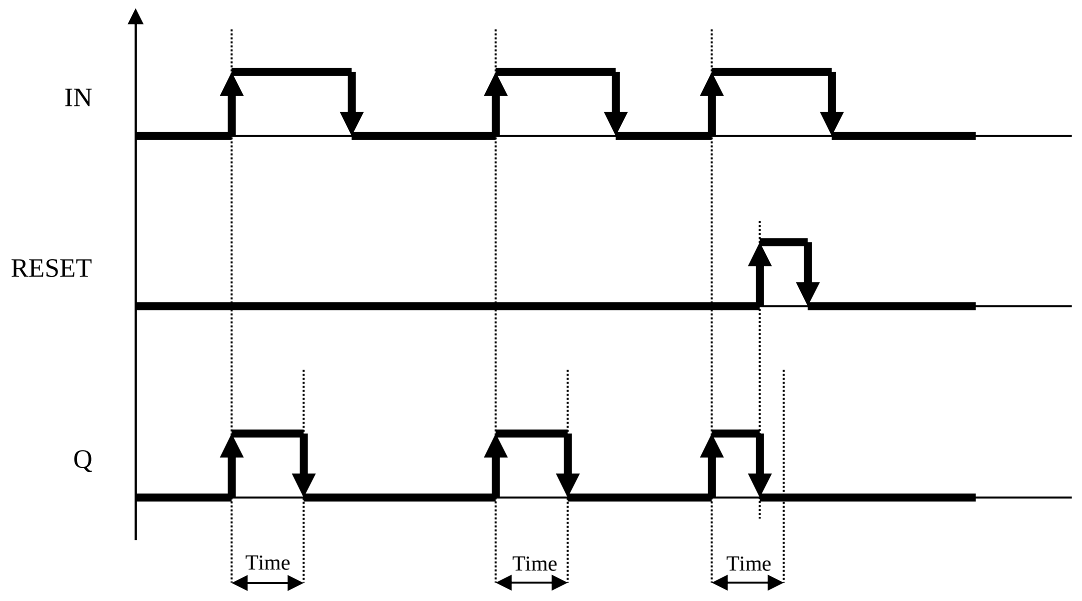


Input	*Type*	*Limits*	*Description*
IN	CJ_BIT	0-1	Set input
TIME	CJ_WORD	0..65535	the time in 100mS for which the output will remain at the value ‘1’
RESET	CJ_BIT	0-1	Reset input

Output	*Type*	*Limits*	*Description*
Q	CJ_BIT	0-1	Output value

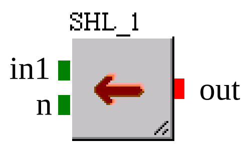
Description
The *MONOS* library performs the monostable function, i.e. for each input leading edge, the output remains ‘1’ for a determined amount of time, set by the TIME input. If during this phase there should be a RESET then the output Q immediately returns to ‘0’ even if the set time has not elapsed. Once the pre-established time has elapsed, the output returns to the value ‘0’. A RESET when the IN input is ‘0’ has no effect on the output Q.

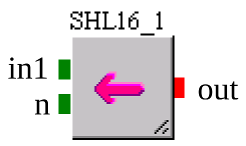
Notes
Please refer to the following graph.



3.3 Bit Shift

SHL (Shift Left)





SHL32_1
in1
n
out

Input	*Type*	*Limits*	*Description*
in1	CJ_BYTE	0..255	8 bit input
in1	CJ_WORD	0..65535	16 bit input
in1	CJ_DWORD	0..4294967295	32 bit input
n (identical for all three types)	CJ_BYTE	0..255	The number of bits on which the shift is to be performed

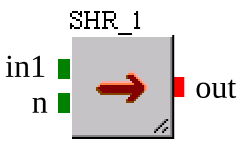
Output	*Type*	*Limits*	*Description*
out	CJ_BYTE	0..255	8 bit result
out	CJ_WORD	0..65535	16 bit result
out	CJ_DWORD	0..4294967295	32 bit result

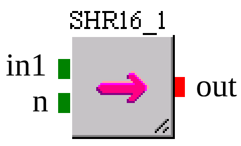
Description

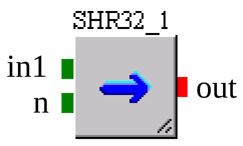
The *SHL*, *SHL16* and *SHL32* libraries allow performing left shift operations on 8, 16 and 32 bit inputs respectively.
By adjusting *n* it is possible to make n bits enter from the right. The result will be that the least significant bits of the binary string will be translated leftwards n positions.
8 Bit example:
in1 = 10011101 (157 or 0x9D)
n = 2
SHL = 011101**00** (116 or 0x74)

Notes
To make operations more uniform, the libraries have been combined.

SHR (Shift Right)







Input	*Type*	*Limits*	*Description*
in1	CJ_BYTE	0..255	8 bit input
in1	CJ_WORD	0..65535	16 bit input
in1	CJ_DWORD	0..4294967295	32 bit input
n (identical for all three types)	CJ_BYTE	0..255	The number of bits on which the shift is to be performed

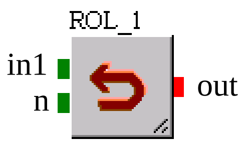
Output	*Type*	*Limits*	*Description*
out	CJ_BYTE	0..255	8 bit result
out	CJ_WORD	0..65535	16 bit result
out	CJ_DWORD	0..4294967295	32 bit result

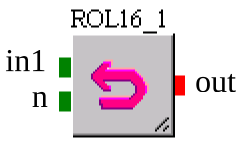
Description

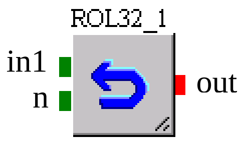
The *SHR*, *SHR16* and *SHR32* libraries allow performing right shift operations on 8, 16 and 32 bit inputs respectively.
By adjusting *n* it is possible to make n bits enter from the left. The result will be that the most significant bits of the binary string will be translated rightwards n positions.
8 Bit example:
in1 = 10011101 (157 or 0x9D)
n = 2
SHR = **00**100111 (39 or 0x27)

Notes
To make operations more uniform, the libraries have been combined.

ROL (Rotary Left)







Input	***Type***	***Limits***	***Description***
in1	CJ_BYTE	0..255	8 bit input
in1	CJ_WORD	0..65535	16 bit input
in1	CJ_DWORD	0..4294967295	32 bit input
n (identical for all three types)	CJ_BYTE	0..255	The number of bits on which the rotation is to be performed

Output	***Type***	***Limits***	***Description***
out	CJ_BYTE	0..255	8 bit result
out	CJ_WORD	0..65535	16 bit result
out	CJ_DWORD	0..4294967295	32 bit result

Description

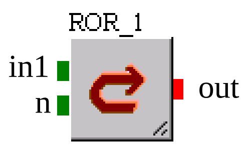
The *ROL*, *ROL16* and *ROL32* libraries allow performing left rotation operations on 8, 16 and 32 bit inputs respectively.
By adjusting *n* it is possible to make the last n bits rotate. The result is that the least significant n bits will be inserted at the beginning of the binary string and the remaining bits translated rightwards n positions.
8 Bit example:

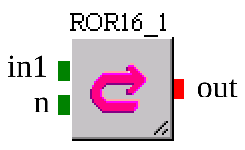
in1	= **10**011101 (157 or 0x9D)
n	= 2
ROL	= 011101**10** (118 or 0x76)

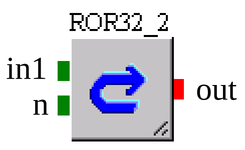
Notes

To make operations more uniform, the libraries have been combined.

ROR (Rotary Right)







Input	*Tipo*	*Limits*	*Description*
in1	CJ_BYTE	0..255	8 bit input
in1	CJ_WORD	0..65535	16 bit input
in1	CJ_DWORD	0..4294967295	32 bit input
n (identical for all three types)	CJ_BYTE	0..255	The number of bits on which the rotation is to be performed

Output	*Type*	*Limits*	*Description*
out	CJ_BYTE	0..255	8 bit result
out	CJ_WORD	0..65535	16 bit result
out	CJ_DWORD	0..4294967295	32 bit result

Description

The *ROR*, *ROR16* and *ROR32* libraries allow performing right rotation operations on 8, 16 and 32 bit inputs respectively.
By adjusting *n* it is possible to make the first n bits rotate. The result is that the most significant n bits will be inserted at the end of the binary string and the remaining bits translated leftwards n positions.
8 Bit example:

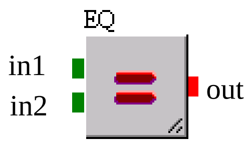
in1 = 100111**01** (157 or 0x9D)
n = 2
ROR = **01**100111 (103 or 0x67)

Notes

To make operations more uniform, the libraries have been combined.

3.4 Comparators

EQ (Equal Comparator)



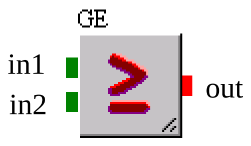
Input	*Type*	*Limits*	*Description*
in1	CJ_LONG	-2147483648..2147483647	Input to be compared
in2	CJ_LONG	-2147483648..2147483647	Input to be compared

Output	*Type*	*Limits*	*Description*
out	CJ_BIT	0-1	Output value

Description
Inputs *in1* and *in2* are compared: if both values are equal, the output assumes a value of ‘1’, otherwise, a value of ‘0’.

Notes
Input data are converted to 32 bit values with signs.

GE (Greater or Equal Comparator)



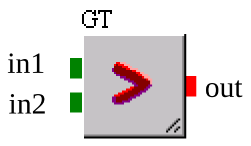
Input	*Type*	*Limits*	*Description*
in1	CJ_LONG	-2147483648..2147483647	Input to be compared
in2	CJ_LONG	-2147483648..2147483647	Input to be compared

Output	*Type*	*Limits*	*Description*
out	CJ_BIT	0-1	Output value

Description
Inputs *in1* and *in2* are compared: if the value of *in1* is greater than or equal to the value of *in2* the output assumes the value ‘1’, otherwise the value ‘0’.

Notes
Input data are converted to 32 bit values with signs.

GT (Greater Comparator)



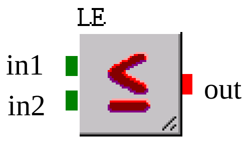
Input	***Type***	***Limits***	***Description***
in1	CJ_LONG	-2147483648..2147483647	Input to be compared
in2	CJ_LONG	-2147483648..2147483647	Input to be compared

Output	***Type***	***Limits***	***Description***
out	CJ_BIT	0-1	Output value

Description
Inputs *in1* and *in2* are compared: if the value of *in1* is greater than the value of *in2* the output assumes the value ‘1’, otherwise the value ‘0’.

Notes
Input data are converted to 32 bit values with signs.

LE (Less or Equal Comparator)



Input	***Type***	***Limits***	***Description***
in1	CJ_LONG	-2147483648..2147483647	Input to be compared

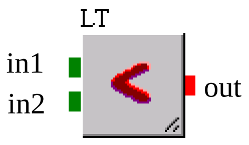
in2	CJ_LONG	-2147483648..2147483647	Input to be compared

Output	***Type***	***Limits***	***Description***
out	CJ_BIT	0-1	Output value

Description
Inputs *in1* and *in2* are compared: if the value of *in1* is less than or equal to the value of *in2* the output assumes the value ‘1’, otherwise the value ‘0’.

Notes
Input data are converted to 32 bit values with signs.

LT (Less Comparator)



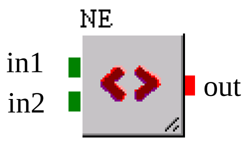
Input	***Type***	***Limits***	***Description***
in1	CJ_LONG	-2147483648..2147483647	Input to be compared
in2	CJ_LONG	-2147483648..2147483647	Input to be compared

Output	***Type***	***Limits***	***Description***
out	CJ_BIT	0-1	Output value

Description
Inputs *in1* and *in2* are compared: if the value of *in1* is less than the value of *in2* the output assumes the value ‘1’, otherwise the value ‘0’.

Notes
Input data are converted to 32 bit values with signs.

NE (Not Equal Comparator)



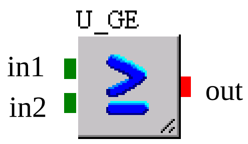
Input	*Type*	*Limits*	*Description*
in1	CJ_LONG	-2147483648..2147483647	Input to be compared
in2	CJ_LONG	-2147483648..2147483647	Input to be compared

Output	*Type*	*Limits*	*Description*
out	CJ_BIT	0-1	Output value

Description
Inputs *in1* and *in2* are compared: if the value of *in1* differs from the value of *in2* the output assumes the value '1', otherwise the value '0'.

Notes
Input data are converted to 32 bit values with signs.

U_GE (Unsigned Greater or Equal Comparator)



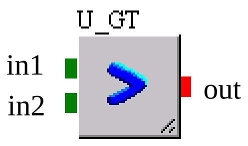
Input	*Type*	*Limits*	*Description*
in1	CJ_DWORD	0..4294967295	Input to be compared
in2	CJ_DWORD	0..4294967295	Input to be compared

Output	*Type*	*Limits*	*Description*
out	CJ_BIT	0-1	Output value

Description
Inputs *in1* and *in2* are compared: if the value of *in1* is greater than or equal to the value of *in2* the output assumes the value '1', otherwise the value '0'.

Notes
Input data are converted to 32 bit values without signs.

U_GT (Unsigned Greater Comparator)



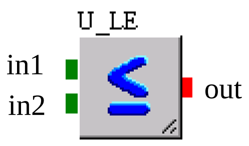
Input	***Type***	***Limits***	***Description***
in1	CJ_DWORD	0..4294967295	Input to be compared
in2	CJ_DWORD	0..4294967295	Input to be compared

Output	***Type***	***Limits***	***Description***
out	CJ_BIT	0-1	Output value

Description
Inputs *in1* and *in2* are compared: if the value of *in1* is greater than the value of *in2* the output assumes the value '1', otherwise the value '0'.

Notes
Input data are converted to 32 bit values without signs.

U_LE (Unsigned Less or Equal Comparator)



Input	***Type***	***Limits***	***Description***
in1	CJ_DWORD	0..4294967295	Input to be compared
in2	CJ_DWORD	0..4294967295	Input to be compared

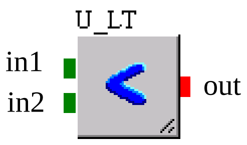
Output	***Type***	***Limits***	***Description***

out	CJ_BIT	0-1	Output value

Description
Inputs *in1* and *in2* are compared: if the value of *in1* is less than or equal to the value of *in2* the output assumes the value ‘1’, otherwise the value ‘0’.

Notes
Input data are converted to 32 bit values without signs.

U_LT (Unsigned Less Comparator)



Input	***Type***	***Limits***	***Description***
in1	CJ_DWORD	0..4294967295	Input to be compared
in2	CJ_DWORD	0..4294967295	Input to be compared

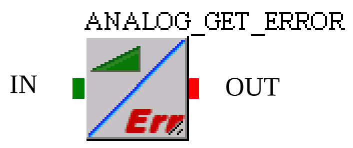
Output	***Type***	***Range***	***Description***
out	CJ_BIT	0-1	Output value

Description
Inputs *in1* and *in2* are compared: if the value of *in1* is less than the value of *in2* the output assumes the value ‘1’, otherwise the value ‘0’.

Notes
Input data are converted to 32 bit values without signs.

3.5 Conversion

ANALOG_GET_ERROR

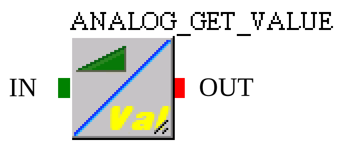


Input	*Type*	*Limits*	*Description*
IN	CJ_ANALOG	Value: -32768..32767 Error: 0..2	Analogue input

Output	*Type*	*Limits*	*Description*
OUT	CJ_BYTE	0..2	Error type

Description
Returns the analogue input error type, in particular: 0. No error 1. Sensor short circuit 2. Damaged or missing sensor

ANALOG_GET_VALUE

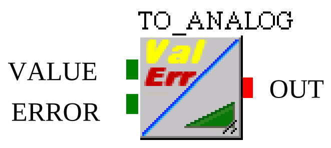


Input	*Type*	*Limits*	*Description*
IN	CJ_ANALOG	Value: -32768..32767 Error: 0..2	Analogue input

Output	*Type*	*Limits*	*Description*
OUT	CJ_SHORT	-32768..32767	Sensor value

Description
Returns the value measured by the sensor. If there is an error with the sensor, the value returned in the maximum (32767).

TO_ANALOG

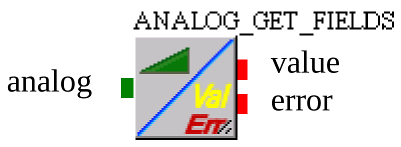


Input	***Type***	***Limits***	***Description***
VALUE	CJ_SHORT	-32768..32767	Measured value
ERROR	CJ_BYTE	0..2	Error

Output	***Type***	***Limits***	***Description***
OUT	CJ_ANALOG	Value: -32768..32767 Error: 0..2	Analogue type

Description
Constructs a CJ_ANALOG structure using the values entered as input.

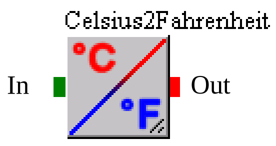
ANALOG_GET_FIELDS



Input	***Type***	***Limits***	***Description***
analogue	CJ_ANALOG	value: -32768..32767 error: 0..2	Sensor input

Output	***Tipo***	***Limits***	***Description***
value:	CJ_SHORT	-32768..32767	Measured value
error:	CJ_BYTE	0..2	Error
Description			
Returns the fields of the inputting CJ_ANALOG structure.			

Celsius2Fahrenheit

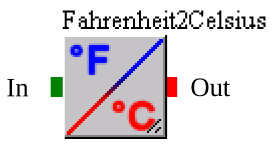


Input	*Type*	*Limits*	*Description*
In	CJ_SHORT	-32768..32767	Temperature in °C

Output	*Type*	*Limits*	*Description*
Out	CJ_SHORT	-32768..32767	Temperature in °F

Description
Performs the conversion °C - °F.

Fahrenheit2Celsius

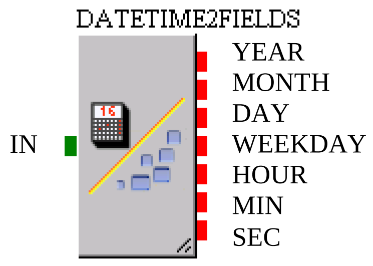


Input	*Type*	*Limits*	*Description*
In	CJ_SHORT	-32768..32767	Temperature in °F

Output	*Type*	*Limits*	*Description*
Out	CJ_SHORT	-32768..32767	Temperature in °C

Description
Performs the conversion °F - °C.

DATETIME2FIELDS

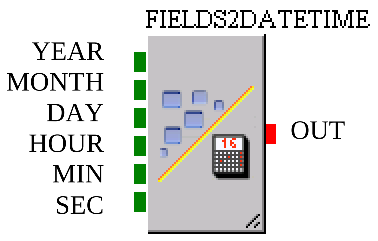


Input	*Type*	*Limits*	*Description*
IN	CJ_DATETIME	0..2147483647	Date and time

Output	*Type*	*Limits*	*Description*
YEAR, *MONTH,* *DAY,* *WEEKDAY,* *HOUR,* *MIN, SEC*	CJ_BYTE	00..68 1..12 1..31 0..6 0..23 0..59	Year month day day of the week hour minutes and seconds

Description
Returns the individual fields of the input date and time.

FIELDS2DATETIME

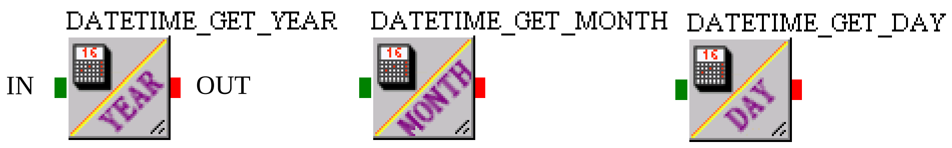


Input	*Type*	*Limits*	*Description*
YEAR, *MONTH,* *DAY,* *HOUR,* *MIN, SEC*	CJ_BYTE	00..68 1..12 1..31 0..23 0..59	Year month day hour minutes and seconds
Output	***Type***	***Limits***	***Description***

OUT	CJ_DATETIME	0..2147483647	Date and time

Description
Constructs the date and time from the input fields.

DATETIME_GET_YEAR, DATETIME_GET_MONTH and DATETIME_GET_DAY

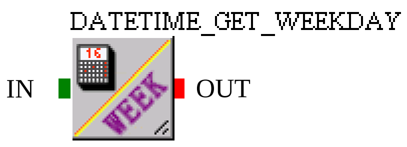


Input	*Type*	*Limits*	*Description*
IN	CJ_DATETIME	0..2147483647	Date and time

Output	*Type*	*Limits*	*Description*
OUT	CJ_BYTE	0..68 1..12 1..31	Year month day

Description
Returns the year, month and day individually from the input date.

DATETIME_GET_WEEKDAY

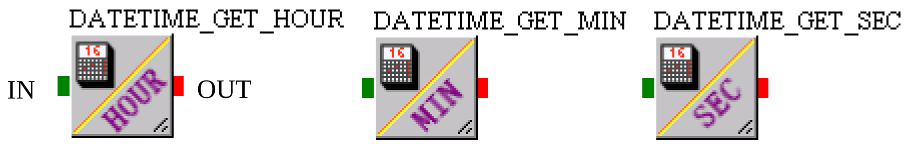


Input	*Type*	*Limits*	*Description*
IN	CJ_DATETIME	0..2147483647	Date and time

Output	*Type*	*Limits*	*Description*
OUT	CJ_BYTE	0..6	Day of the week

Description
Returns the number of the day of the week. [0 = Sunday, …, 6 = Saturday]

DATETIME_GET_HOUR, DATETIME_GET_MIN and DATETIME_GET_SEC

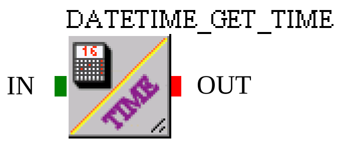


Input	*Type*	*Limits*	*Description*
IN	CJ_DATETIME	0..2147483647	Date and time

Output	*Type*	*Limits*	*Description*
OUT	CJ_BYTE	0..23 0..59 0..59	Hour minute second

Description
Returns the hour, minute and second individually from the input time.

DATETIME_GET_TIME

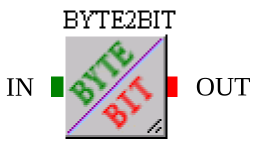


Input	*Type*	*Limits*	*Description*
IN	CJ_DATETIME	0..2147483647	Date and time

Output	*Type*	*Limits*	*Description*
OUT	CJ_TIME	0..86400	Returns the time (hour, minute, second)

Description
This library returns the time (hour, minute and second) from a CJ_DATETIME type input The value 86400 represents the number of seconds in a day.

BYTE2BIT

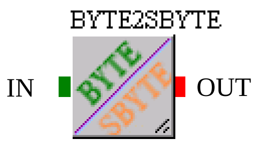


Input	***Type***	***Limits***	***Description***
IN	CJ_BYTE	0..255	8 bit input

Output	***Type***	***Limits***	***Description***
OUT	CJ_BIT	0-1	Binary output

Description
Returns ‘1’ if *IN* is other than zero, otherwise returns ‘0’.

BYTE2SBYTE

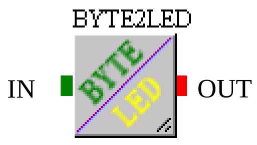


Input	***Type***	***Limits***	***Description***
IN	CJ_BYTE	0..255	8 bit input (without sign)

Output	***Type***	***Limits***	***Description***
OUT	CJ_S_BYTE	-128..127	8 bit output (with sign)

Description
Performs the conversion of 8 bit data without sign to 8 bit data with sign.

BYTE2LED

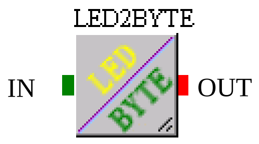


Input	*Type*	*Limits*	*Description*
IN	CJ_BYTE	0..3	LED type value

Output	*Type*	*Limits*	*Description*
OUT	CJ_LED	0..3	LED type

Description
Sets the LED operation type, in particular: 0. LED off 1. LED continually on 2. LED on at low frequency 3. LED on at high frequency

LED2BYTE



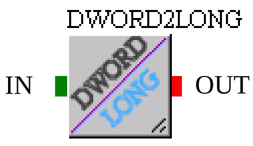
Input	*Type*	*Limits*	*Description*
IN	CJ_LED	0..3	LED type

Output	*Type*	*Limits*	*Description*
OUT	CJ_BYTE	0..3	LED type value

Description

Returns the input LED operation type, in particular: 4. LED off 5. LED continually on 6. LED on at low frequency 7. LED on at high frequency

DWORD2LONG

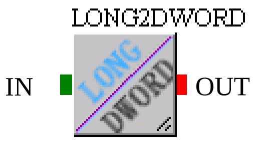


Input	*Type*	*Limits*	*Description*
IN	CJ_DWORD	0..4294967295	32 bit input (with sign)

Output	*Type*	*Limits*	*Description*
OUT	CJ_LONG	-2147483647.. 2147483648	32 bit output (without sign)

Description
Performs the conversion of 32 bit data without sign to 32 bit data with sign.

LONG2DWORD



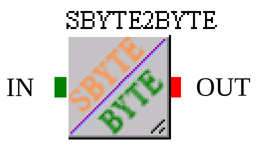
Input	*Type*	*Limits*	*Description*
IN	CJ_LONG	-2147483647.. 2147483648	32 bit input (without sign)

Output	*Type*	*Limits*	*Description*

OUT	CJ_DWORD	0..4294967295	32 bit output (with sign)

Description
Performs the conversion of 32 bit data with sign to 32 bit data without sign.

SBYTE2BYTE

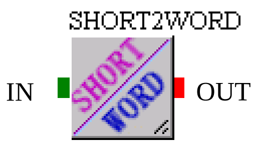


Input	***Type***	***Limits***	***Description***
IN	CJ_S_BYTE	-128..127	8 bit input (with sign)

Output	***Type***	***Limits***	***Description***
OUT	CJ_BYTE	0..255	8 bit output (without sign)

Description
Performs the conversion of 8 bit data with sign to 8 bit data without sign.

SHORT2WORD

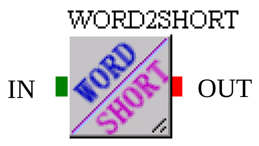


Input	***Type***	***Limits***	***Description***
IN	CJ_SHORT	-32768..32767	16 bit input (with sign)

Output	***Type***	***Limits***	***Description***
OUT	CJ_WORD	0..65535	16 bit output (without sign)

Description
Performs the conversion of 16 bit data with sign to 16 bit data without sign

WORD2SHORT

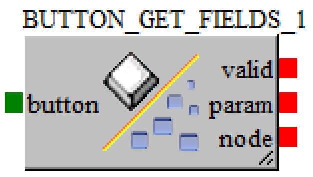


Input	***Type***	***Limits***	***Description***
IN	CJ_WORD	0..65535	16 bit input (without sign)

Output	***Type***	***Limits***	***Description***
OUT	CJ_SHORT	-32768..32767	16 bit output (with sign)

Description
Performs the conversion of 16 bit data without sign to 16 bit data with sign.

BUTTON_GET_FIELDS



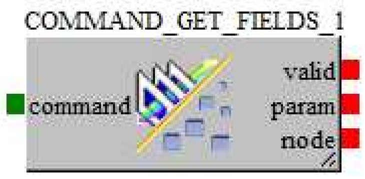
Input	***Type***	***Limits***	***Description***
button	CJ_BTN	Valid: 0-1 Node: 0..127 Param: 0..32767	Button to process

Output	***Type***	***Limits***	***Description***
valid	CJ_BIT	0-1	Valid field
param	CJ_WORD	0..65535	Parameter field
node	CJ_BYTE	0..127	Node field

Description

Returns the CJ_BTN structured fields. ➢ *valid*. Establishes whether the *button* event has been executed. If '1' – event executed ➢ *param*. Represents the number of seconds the relevant key persists ➢ *node*. Logical node in which the action has been verified

COMMAD_GET_FIELDS

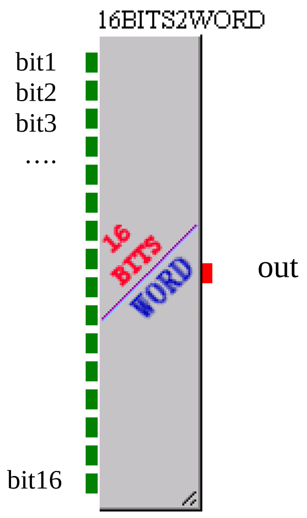


Input	*Type*	*Limits*	*Description*
command	CJ_CMD	Valid: 0-1 Node: 0..127 Param: 0..32767	Command to be processed

Output	*Type*	*Limits*	*Description*
valid	CJ_BIT	0-1	Valid field
param	CJ_WORD	0..65535	Parameter field
node	CJ_BYTE	0..127	Node field

Description
Returns the CJ_CMD structured fields. ➢ *valid*. Establishes whether the *command* event has been executed. If '1' – event executed ➢ *param*. Represents the command parameter ➢ *node*. Logical node in which the action has been verified

16BITS2WORD

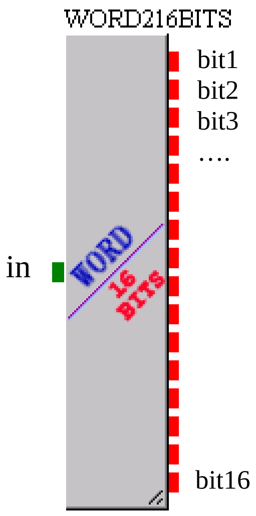


Input	*Type*	*Limits*	*Description*
bit1	CJ_BIT	0-1	First bit
bit2	CJ_BIT	0-1	Second bit
bitN	CJ_BIT	0-1	nth bit
bit16	CJ_BIT	0-1	Sixteenth bit

Output	*Type*	*Limits*	*Description*
out	CJ_WORD	0..65535	Words extracted from the string of bits inserted in input

Description
Constructs the value in Words starting from the 16 bits inserted in input.

WORD216BITS

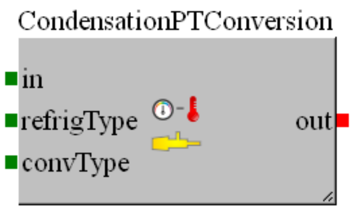


Input	*Type*	*Limits*	*Description*
in	CJ_WORD	0..65535	Input value

Output	*Type*	*Limits*	*Description*
bit1	CJ_BIT	0-1	First bit
bit2	CJ_BIT	0-1	Second bit
bitN	CJ_BIT	0-1	nth bit
bit16	CJ_BIT	0-1	Sixteenth bit

Description
Returns 16 single bits calcuated from a Word input value.

CondensationP/TConversion



Ingresso	*Tipo*	*Limiti*	*Descrizione*
in	CJ_ANALOG	Value:-32768..32767 Error: 0..2	Value to be converted, according to the parameter *convType*: - evaporation pressure **in Barg with two decimal places** - Saturation temperature **in °C with a decimal point**

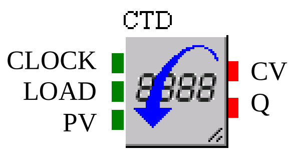
refrType	CJ_BYTE	0..19	Used GAS: 0: R-22 1: R-134A 2: R-402A 3: R-404A 4: R-407A 5: R-407C 6: R-410A 7: R-417A 8: R-422A 9: R-422D 10: R-507A 11: R-744 12: R-438A 13: R-401B 14: R-290 15: R-717 16: R-1270 17: R-32 18: R-407F 19:R-1234ZE
convType	CJ_BIT	0-1	Conversion type: 0: P →T conversion 1: T →P conversion

Uscita	***Tipo***	***Limiti***	***Descrizione***
out	CJ_ANALOG	Value:-32768..32767 Error: 0..2	Converted value, according to the parameter *convType*: - evaporation pressure **in Barg with two decimal places** - Saturation temperature **in °C with a decimal point**

Descrizione
Calculates the saturation temperature [° C] on the pressure [barg] or the saturation pressure [barg] on the temperature in [° C] for a specific gas.

3.6 Counters

CTD / MCTD (Down Counter)

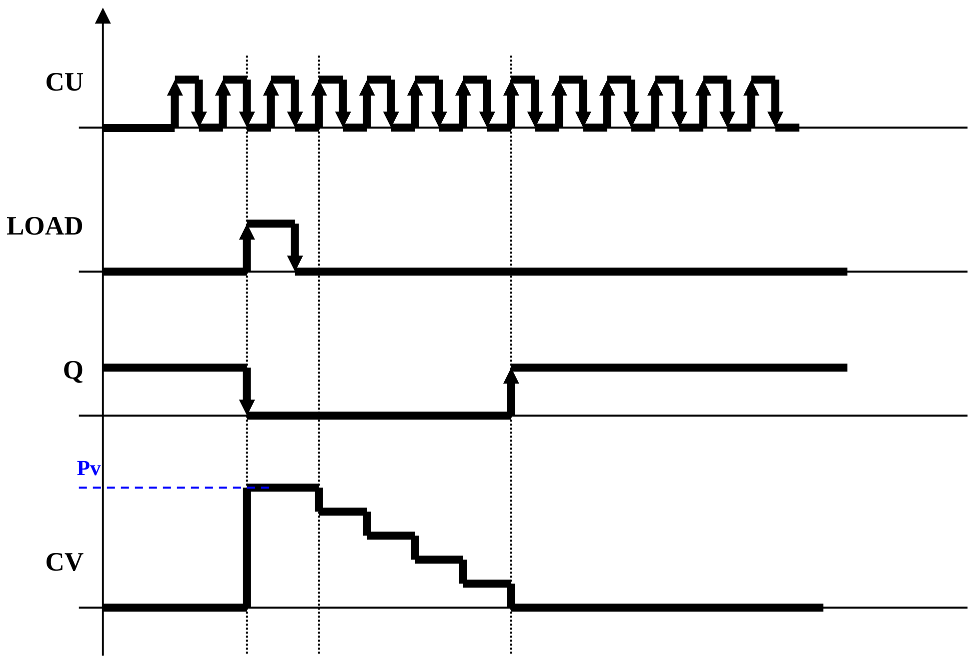


Input	***Type***	***Limits***	***Description***
CLOCK	CJ_BIT	0-1	Count Down. For each leading edge detected in the *CLOCK* input, the current CV value is decreased, with the exception of when the *LOAD* input is active, or when *CV* has reached the value 0
LOAD	CJ_BIT	0-1	When the *LOAD* input is set to ‘1’ the counter status is loaded with the value of *PV*
PV	CJ_WORD	0..65535	Preset Value. Represents the counter starting value which is loaded each time the *LOAD* input is set to the value ‘1’

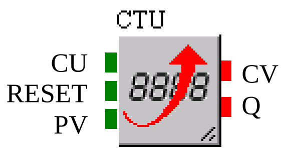
Output	***Type***	***Limits***	***Description***
CV	CJ_WORD	0..65535	Current Value. Returns the current counter value
Q	CJ_BIT	0-1	Represents termination of count. This is set to ‘1’ when the counter internal status is set to zero (*CV* = 0)

Description
Causes an event counter to count down from a set value, *PV*. MCTD save in memory the Current Value e restart from this value after a power cycle.

Notes
Please refer to the following graph.



CTU / MCTU (Up Counter)



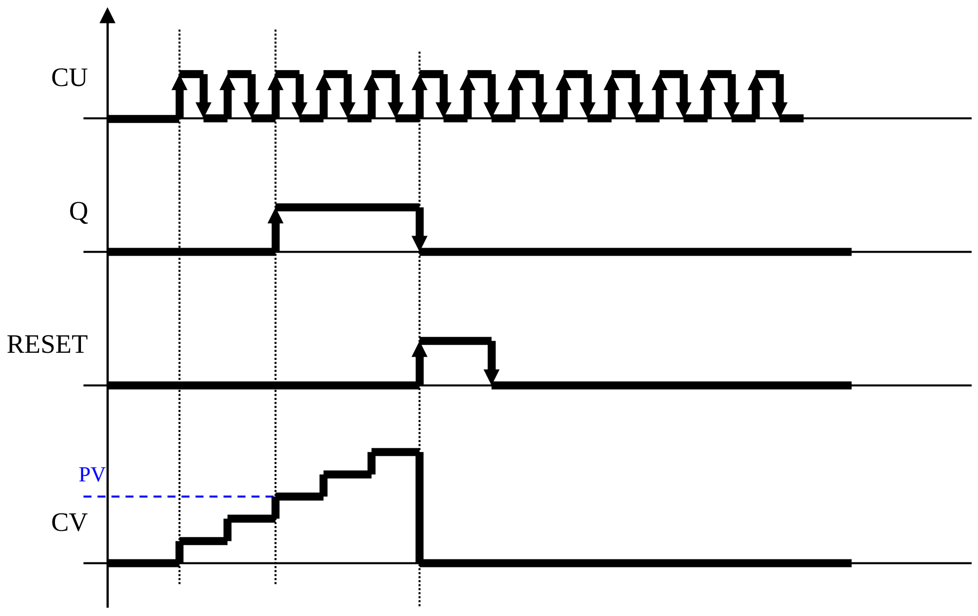
Input	*Type*	*Limits*	*Description*
CU	CJ_BIT	0-1	Count Up. For each leading edge detected in the *CU* input, the current CV value is increased, with the exception of when the *LOAD* input is active, until reaching the maximum value of 65635
RESET	CJ_BIT	0-1	When the *RESET* input is set to ‘1’ the internal state of the counter *CV* is set to zero
PV	CJ_WORD	0..65535	Preset Value. When the *CV* counter internal status exceeds the PV, then output *Q* is set to ‘1’

Output	*Type*	*Limits*	*Description*
CV	CJ_WORD	0.65535	Current Value. Returns the current counter value

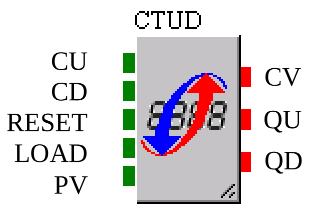
Q	CJ_BIT	0-1	Represents termination of count. This is set to '1' when the counter internal status is greater than or equal to the PV input

Description
Causes an event counter to count up from a value of 0. MCTU save in memory the Current Value e restart from this value after a power cycle.

Notes
Please refer to the following graph



CTUD (Up Down Counter)

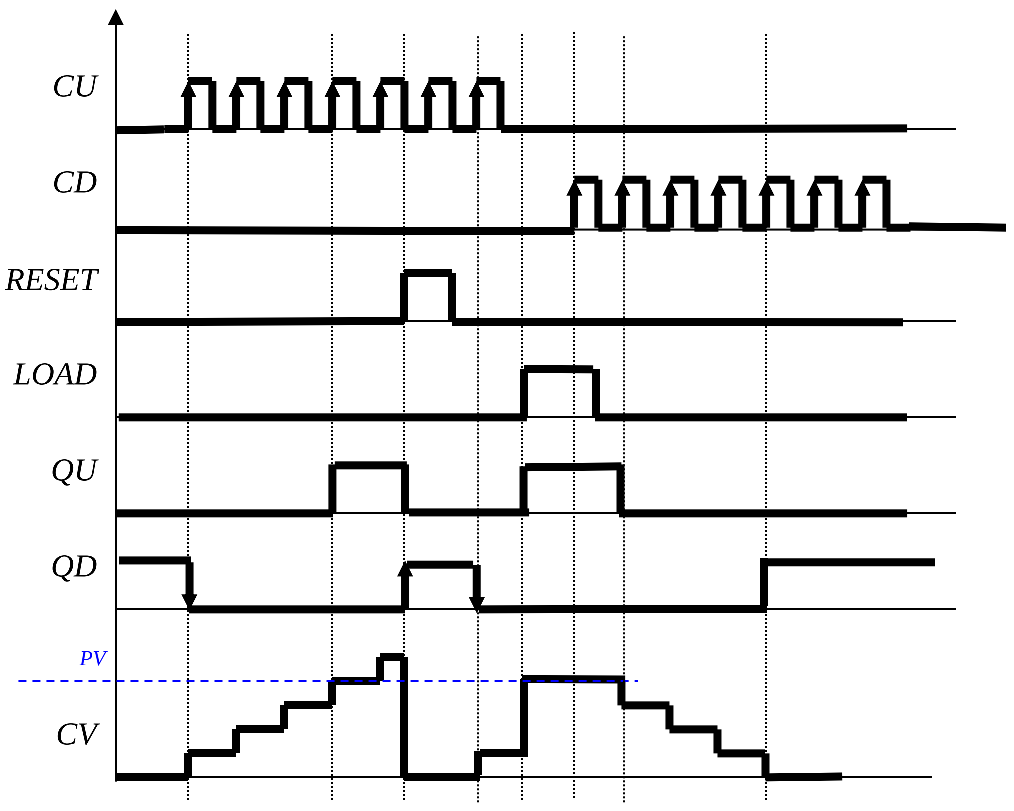


Input	*Type*	*Limits*	*Description*
CU	CJ_BIT	0-1	Count Up. For each leading edge detected in the *CU* input, the current value of CV is increased, except when at least one of the *LOAD* and *RESET inputs* is active, and until reaching the maximum value of 65635. If *CU* and *CD* are active simultaneously, the counter is increased
CD	CJ_BIT	0-1	Count Down. For each leading edge detected in the *CLOCK* input, the current CV value is decreased, with the exception of when at least one of the *LOAD* and *RESET* inputs is active, or when *CV* has reached the value 0 If *CU* and *CD* are active simultaneously, the counter is increased.
RESET	CJ_BIT	0-1	When the *RESET* input is set to ‘1’ the internal state of the counter *CV* is set to zero
LOAD	CJ_BIT	0-1	When the *LOAD* input is set to ‘1’ the counter status is loaded with the value of *PV*. If the *RESET* input is simultaneously active, it is ignored
PV	CJ_WORD	0..65535	Preset Value. When the *CV* counter internal status exceeds the PV, then output *QD* is set to ‘1’

Output	*Type*	*Limits*	*Description*
CV	CJ_WORD	0..65535	Current value. Returns the current counter value
QU	CJ_BIT	0-1	This set to ‘1’ if *CV* is greater than or equal to *PV*
QD	CJ_BIT	0-1	This set to ‘1’ if *CV* is equal to zero

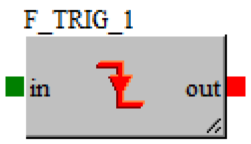
Description
Causes an event counter to count up and down.

Notes
Please refer to the following graph



3.7 Edge detection

F_TRIG

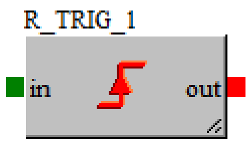


Input	*Type*	*Limits*	*Description*
in	CJ_BIT	0-1	Clock

Output	*Type*	*Limits*	*Description*
out	CJ_BIT	0-1	When a trailing edge is detected at the *in* input (the value changes from ‘1’ to ‘0’), the output is set to ‘1’ for the duration of one processing cycle (main or 100 ms timed)

Description
Use this library in the case where it is desired to detect a digital magnitude changing from high to low states.

R_TRIG



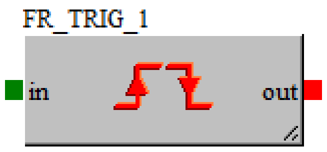
Input	*Type*	*Limits*	*Description*
in	CJ_BIT	0-1	Clock

Output	*Type*	*Limits*	*Description*
out	CJ_BIT	0-1	When a leading edge is detected at the *in* input (the value changes from ‘0’ to ‘1’), the output is set to ‘1’ for the duration of one processing cycle (main, 5 ms timed or 100 ms timed)

Description

Use this library in the case where it is desired to detect a digital magnitude changing from low to high states.

FR_TRIG



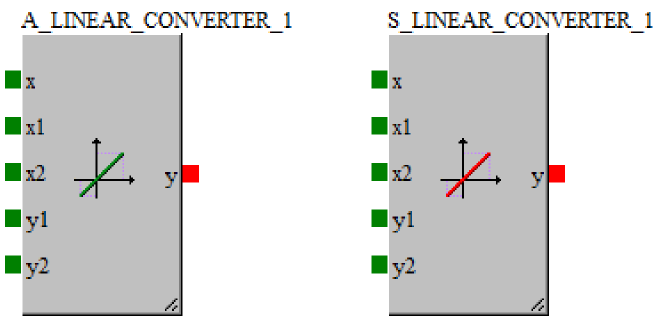
Input	***Type***	***Limits***	***Description***
in	CJ_BIT	0-1	Clock

Output	***Type***	***Limits***	***Description***
out	CJ_BIT	0-1	When a trailing or leading edge is detected at the *in* input (the value changes from ‘1’ to ‘0’ or viceversa), the output is set to ‘1’ for the duration of one processing cycle (main or 100 ms timed)

Description
Use this library in the case where it is desired to detect a digital magnitude changing from high to low states

3.8 *Linear*

A_LINEAR_CONVERTER

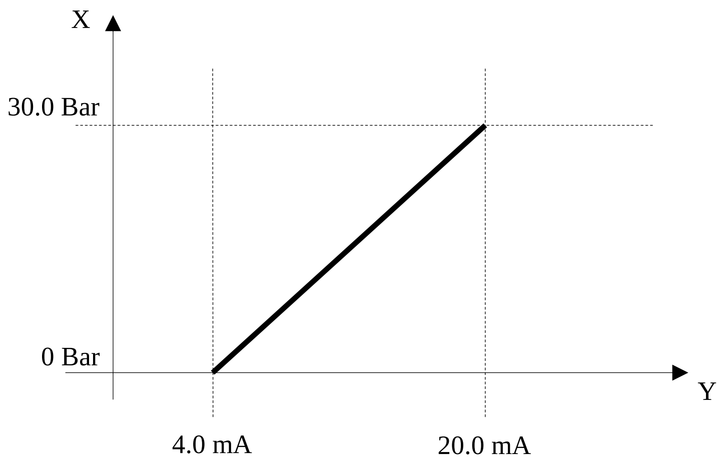


Input	*Type*	*Limits*	*Description*
x	CJ_ANALOG	Value: -32768..32767 Error: 0..2	Analogue input signal
x	CJ_SHORT	-32768..32767	Digital input signal
x1	CJ_SHORT	-32768..32767	Coordinate x1
x2	CJ_SHORT	-32768..32767	Coordinate x2
y1	CJ_SHORT	-32768..32767	Coordinate y1
y2	CJ_SHORT	-32768..32767	Coordinate y2

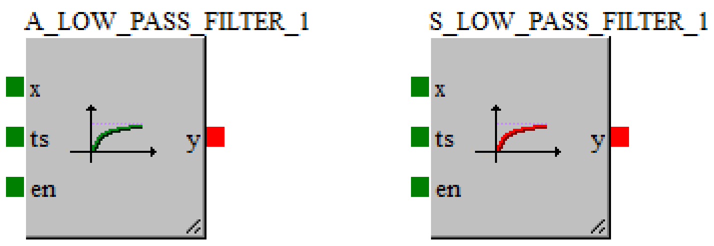
Output	*Type*	*Limits*	*Description*
y	CJ_ANALOG	Value: -32768..32767 Error: 0..2	Analogue signal linearised according to the set inputs
y	CJ_SHORT	-32768..32767	Digital signal, linearised according to the set inputs

Description
The *A_Linear_Converter* library performs a linear transformation on an inputting analogue/digital signal. The parameters for the function are set by the user through the x1, x2, y1 and y2 inputs. In output, the function represents the linearised analogue signal.

Notes
If the input is in an error state than the output will also be in an error state (*A_ Linear_Converter* library). Please refer to the following graph. Linear conversion from 4-20 mA to 0-30 Bar. The ordinate axis represents the pressure (Bar) and the abscissa axis current (mA).



A_LOW_PASS_FILTER - S_LOW_PASS_FILTER



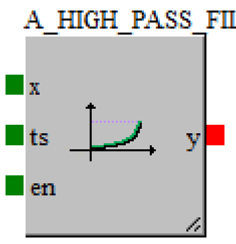
Input	***Type***	***Limits***	***Description***
x	CJ_ANALOG	Value: -32768..32767 Error: 0..2	Analogue input signal
x	CJ_SHORT	-32768..32767	Digital input signal
ts	CJ_WORD	0..65535	Settling time, in 100 ms
en	CJ_BIT	0-1	If ‘1’ the filter is active If ‘0’ the output corresponds to the input

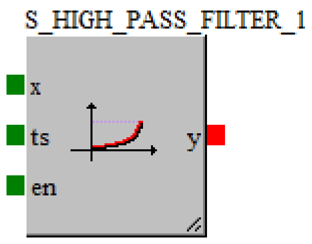
Output	***Type***	***Limits***	***Description***
y	CJ_ANALOG	Value: -32768..32767 Error: 0..2	Analogue signal settled according to the input settings.
y	CJ_SHORT	-32768..32767	Digital signal, settled according to the input settings.

Description
The *LOW_PASS_FILTER* library allows filtering out input fluctuations by means of a low-pass filter. In the case of any step changes in the input, this library allows the inclusion of a settling time prior to the output value being adjusted to that of the input *x*. If the function is active (*en* = ‘1’) the output signal will reach the value of the input signal after a period of time equal to *ts* * 100 ms.

Notes
For the library *LOW_PASS_FILTER* if the input is in an error state than the output will also be in an error state.

A_HIGH_PASS_FILTER - S_HIGH_PASS_FILTER





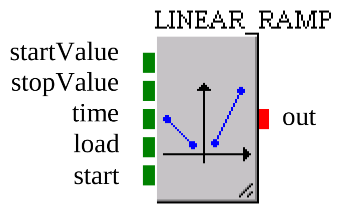
Input	*Type*	*Limits*	*Description*
x	CJ_ANALOG	Value: -32768..32767 Error: 0..2	Analogue input signal
x	CJ_SHORT	-32768..32767	Digital input signal
ts	CJ_WORD	0..65535	Settling time, in 100 ms
en	CJ_BIT	0-1	If ‘1’ the filter is active If ‘0’ the output corresponds to the input

Output	*Type*	*Limits*	*Description*
y	CJ_ANALOG	Value: -32768..32767 Error: 0..2	Analogue signal settled according to the input settings.
y	CJ_SHORT	-32768..32767	Digital signal, settled according to the input settings.

Description
The *HIGH_PASS_FILTER* library allows filtering out input fluctuations by means of a high-pass filter. In the case of any step changes in the input, this library allows the inclusion of a settling time prior to the output value being adjusted to that of the input *x*. If the function is active (*en* = ‘1’) the output signal will reach the value of the input signal after a period of time equal to *ts* * 100 ms.

Notes
For the library *HIGH_PASS_FILTER* if the input is in an error state than the output will also be in an error state.

LINEAR_RAMP

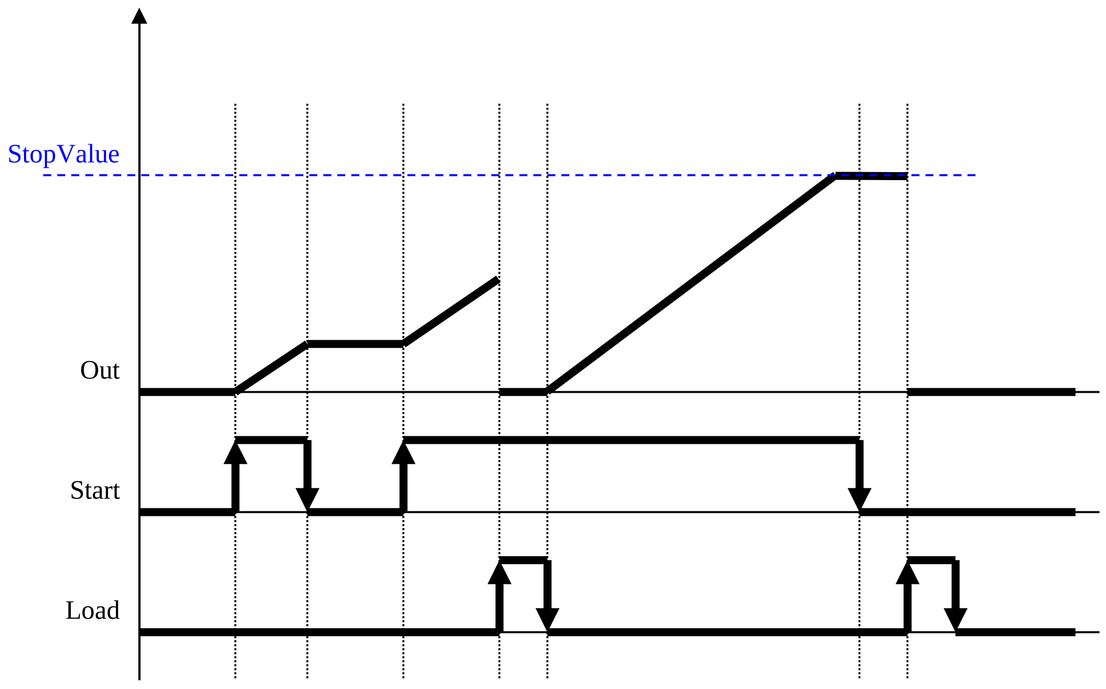


Input	*Type*	*Limits*	*Description*
startValue	CJ_SHORT	-32768..32767	Initial value
stopValue	CJ_SHORT	-32768..32767	Final value
time	CJ_WORD	0..65535	The time, in seconds, required for going from the *startValue* to the *stopValue*
load	CJ_BIT	0-1	Loads the initial value and resets the output to that value
start	CJ_BIT	0-1	Enables the function

Output	*Type*	*Limits*	*Description*
out	CJ_SHORT	-32768..32767	The output of the function

Description
The *LINEAR_RAMP* library performs the ramp function y=kx + c, where the slope k is a function of the *stopValue*, *startValue* and *time* parameters. The *time* parameter indicates the number of seconds the function uses to reach the *stopValue* value, starting from any given initial value *startValue*. According to the above considerations, the slope of the ramp is defined as follows: *stopValue-startValue/time*. The start input has the following functions: ➢ if set to the value ’1’ it initiates the function ➢ if set to the value ‘0’ it stops the function, storing the output in memory. If start changes once more to the value ‘1’ the output starts off again from where it had stopped The *load* input is used to reset the output to the initial value. If *load*=1 the ramp function is inhibited and the output assumes the *startValue* value.

Notes
Please refer to the following graph. The graph in the example considers the startValue value = 0.



3.9 Logic

AND (0-1 AND)

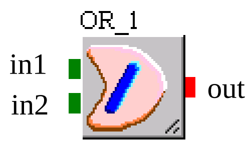
AND_1
in1
in2
out

Input	*Type*	*Limits*	*Description*
in1	CJ_BIT	0-1	Digital input
in2	CJ_BIT	0-1	Digital input

Output	*Type*	*Limits*	*Description*
out	CJ_BIT	0-1	Output value

Description
Returns the logical AND of inputs *in1* and *in2*. If both values are ‘1’, the output assumes the value ‘1’, otherwise, the value ‘0’.

OR (0-1 OR)

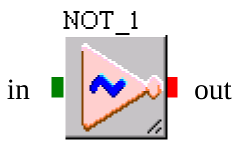


Input	*Type*	*Limits*	*Description*
in1	CJ_BIT	0-1	Digital input
in2	CJ_BIT	0-1	Digital input

Output	*Type*	*Limits*	*Description*
out	CJ_BIT	0-1	Output value

Description
Returns the logical OR of inputs *in1* and *in2*. If at least one of the two values is ‘1’, the output assumes the value ‘1’, otherwise, the value ‘0’.

NOT (0-1 NOT)

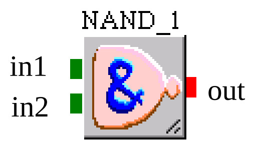


Input	*Type*	*Limits*	*Description*
in	CJ_BIT	0-1	Digital input

Output	*Type*	*Limits*	*Description*
out	CJ_BIT	0-1	Output value

Description
Returns the negation of the digital input *in*. If the input is ‘0’, the output assumes the value ‘1’, otherwise, the value ‘0’.

NAND (0-1 NAND)

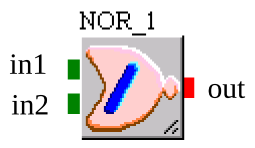


Input	*Type*	*Limits*	*Description*
in1	CJ_BIT	0-1	Digital input
in2	CJ_BIT	0-1	Digital input

Output	*Type*	*Limits*	*Description*
out	CJ_BIT	0-1	Output value

Description
Returns the logical NAND of inputs *in1* and *in2*. If both values are ‘1’, the output assumes the value ‘0’, otherwise, the value ‘1’.

NOR (0-1 NOR)

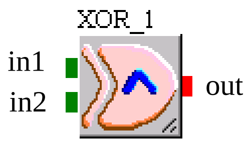


Input	*Type*	*Limits*	*Description*
in1	CJ_BIT	0-1	Digital input
in2	CJ_BIT	0-1	Digital input

Output	*Type*	*Limits*	*Description*
out	CJ_BIT	0-1	Output value

Description
Returns the logical NOR of inputs *in1* and *in2*. If at least one of the two values is ‘1’, the output assumes the value ‘0’, otherwise, the value ‘1’.

XOR (0-1 XOR)

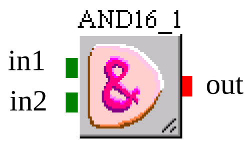


Input	*Type*	*Limits*	*Description*
in1	CJ_BIT	0-1	Digital input
in2	CJ_BIT	0-1	Digital input

Output	*Type*	*Limits*	*Description*
out	CJ_BIT	0-1	Output value

Description
Returns the exclusive OR of inputs *in1* and *in2*. If the values are different, the output assumes the value ‘1’, otherwise, the value ‘0’.

AND16 (16 bit AND)

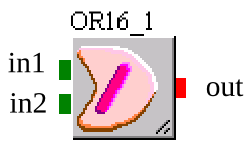


Input	*Type*	*Limits*	*Description*
in1	CJ_WORD	0..65535	Digital input
in2	CJ_WORD	0..65535	Digital input

Output	*Type*	*Limits*	*Description*
out	CJ_WORD	0..65535	Output value

Description
Returns the AND of the 16 bits of inputs *in1* and *in2*. For each bit, if both values are ‘1’ the corresponding output bit assumes the value ‘1’, otherwise it assumes the value ‘0’.

OR16 (16 bit OR)

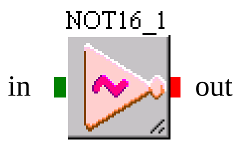


Input	*Type*	*Limits*	*Description*
in1	CJ_WORD	0..65535	Digital input
in2	CJ_WORD	0..65535	Digital input

Output	*Type*	*Limits*	*Description*
out	CJ_WORD	0..65535	Output value

Description
Returns the OR of the 16 bits of inputs *in1* and *in2*. For each bit, if at least one of the two values is '1' the corresponding output bit assumes the value '1', otherwise it assumes the value '0'.

NOT16 (16 bit NOT)



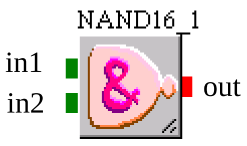
Input	*Type*	*Limits*	*Description*
in1	CJ_WORD	0..65535	Input

Output	*Type*	*Limits*	*Description*
out	CJ_WORD	0..65535	Output value

Description

Returns the negation of the 16 bits of the digital input *in*. For each bit, if its value is '0' the corresponding output bit assumes the value '1', otherwise it assumes the value '0'.

NAND16 (16 bit NAND)

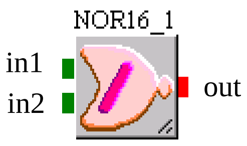


Input	*Type*	*Limits*	*Description*
in1	CJ_WORD	0..65535	Input
in2	CJ_WORD	0..65535	Input

Output	*Type*	*Limits*	*Description*
out	CJ_WORD	0..65535	Output value

Description
Returns the NAND of the 16 bits of inputs *in1* and *in2*. For each bit, if both values are '1' the corresponding output bit assumes the value '0', otherwise it assumes the value '1'.

NOR16 (16 bit NOR)

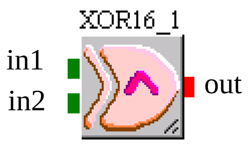


Input	*Type*	*Limits*	*Description*
in1	CJ_WORD	0..65535	Input
in2	CJ_WORD	0..65535	Input

Output	*Type*	*Limits*	*Description*
out	CJ_WORD	0..65535	Output value

Description
Returns the NOR of the 16 bits of inputs *in1* and *in2*. For each bit, if at least one of the two values is ‘1’ the corresponding output bit assumes the value ‘0’, otherwise it assumes the value ‘1’.

NOR16 (16 bit NOR)

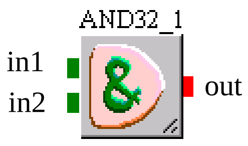


Input	***Type***	***Limits***	***Description***
in1	CJ_WORD	0..65535	Input
in2	CJ_WORD	0..65535	Input

Output	***Type***	***Limits***	***Description***
out	CJ_WORD	0..65535	Output value

Description
Returns the exclusive OR of the 16 bits of inputs *in1* and *in2*. For each bit, if the input values are different, the corresponding output bit assumes the value ‘1’, otherwise it assumes the value ‘0’.

AND32 (32 bit AND)

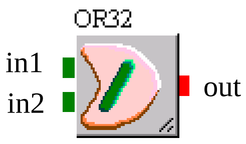


Input	***Type***	***Limits***	***Description***
in1	CJ_DWORD	0..4294967295	Input
in2	CJ_DWORD	0..4294967295	Input

Output	***Type***	***Limits***	***Description***
out	CJ_DWORD	0..4294967295	Output value

Description
Returns the AND of the 32 bits of inputs *in1* and *in2*. For each bit, if both values are '1' the corresponding output bit assumes the value '1', otherwise it assumes the value '0'.

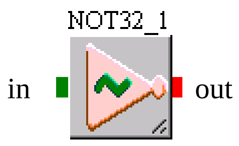
OR32 (32 bit OR)



Input	***Type***	***Limits***	***Description***
in1	CJ_DWORD	0..4294967295	Input
in2	CJ_DWORD	0..4294967295	Input
Output	***Type***	***Limits***	***Description***
out	CJ_DWORD	0..4294967295	Output value

Description
Returns the OR of the 32 bits of inputs *in1* and *in2*. For each bit, if at least one of the two values is '1' the corresponding output bit assumes the value '1', otherwise it assumes the value '0'.

NOT32 (32 bit NOT)

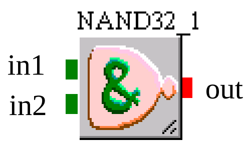


Input	***Type***	***Limits***	***Description***
in1	CJ_DWORD	0..4294967295	Input
in2	CJ_DWORD	0..4294967295	Input

Output	***Type***	***Limits***	***Description***
out	CJ_DWORD	0..4294967295	Output value

Description
Returns the negation of the 32 bits of the digital input *in*. For each bit, if its value is '0' the corresponding output bit assumes the value '1', otherwise it assumes the value '0'.

NAND32 (32 bit NAND)

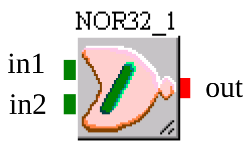


Input	***Type***	***Limits***	***Description***
in1	CJ_DWORD	0..4294967295	Input
in2	CJ_DWORD	0..4294967295	Input

Output	***Type***	***Limits***	***Description***
out	CJ_DWORD	0..4294967295	Output value

Description
Returns the NAND of the 32 bits of inputs *in1* and *in2*. For each bit, if both values are '1' the corresponding output bit assumes the value '0', otherwise it assumes the value '1'.

NOR32 (32 bit NOR)

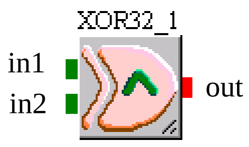


Input	***Type***	***Limits***	***Description***
in1	CJ_DWORD	0..4294967295	Input
in2	CJ_DWORD	0..4294967295	Input

Output	***Type***	***Limits***	***Description***
out	CJ_DWORD	0..4294967295	Output value

Description
Returns the NOR of the 32 bits of inputs *in1* and *in2*. For each bit, if at least one of the two values is '1' the corresponding output bit assumes the value '0', otherwise it assumes the value '1'.

XOR32 (32 bit XOR)



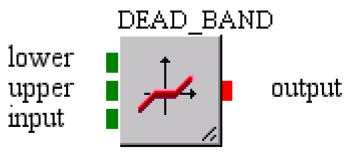
Input	***Type***	***Limits***	***Description***
in1	CJ_DWORD	0..4294967295	Input
in2	CJ_DWORD	0..4294967295	Input

Output	***Type***	***Limits***	***Description***
out	CJ_DWORD	0..4294967295	Output value

Description
Returns the exclusive OR of the 32 bits of inputs *in1* and *in2*. For each bit, if the input values are different, the corresponding output bit assumes the value '1', otherwise it assumes the value '0'.

3.10 Not Linear

DEAD_BAND



Input	*Type*	*Limits*	*Description*
lower	CJ_SHORT	-32768..upper	Lower band limit
upper	CJ_SHORT	lower..32767	Upper band limit
input	CJ_SHORT	-32768..32767	Input to be checked

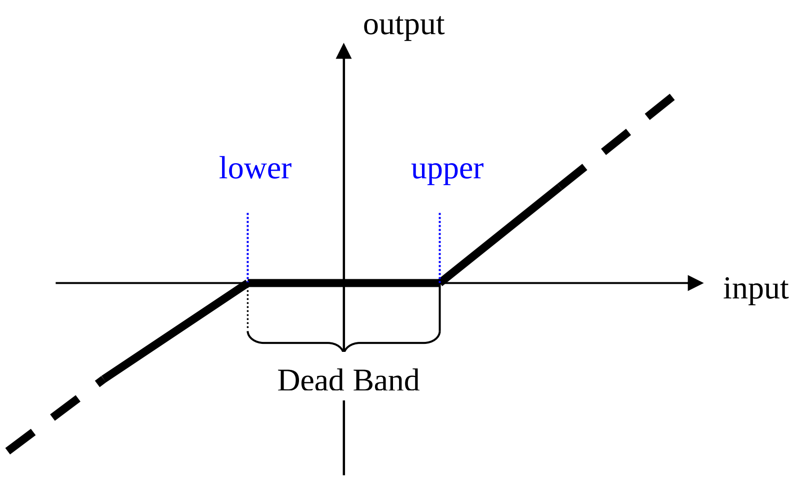
Output	*Type*	*Limits*	*Description*
output	CJ_SHORT	-32768..32767	Controlled output

Description

The *DEAD_BAND* library compares the input value with a “dead band”, which is specified by the *lower* and *upper* inputs. The result of the function is returned as output according to the following criteria:

- If the input value *input* is less than the lower band threshold, the output will correspond to the input, from which the *lower* value will be subtracted.
- If the input value *input* is greater than the upper band threshold, the output will correspond to the input, to which the *upper* value will be added.
- If the input value *input* falls within he specified band, then the output is zero.

(Refer to the following graph).



Input	*Type*	*Limits*	*Description*
lower	CJ_SHORT	-32768..upper	Lower limit
upper	CJ_SHORT	lower..32767	Upper limit
input	CJ_SHORT	-32768..32767	Input to be checked

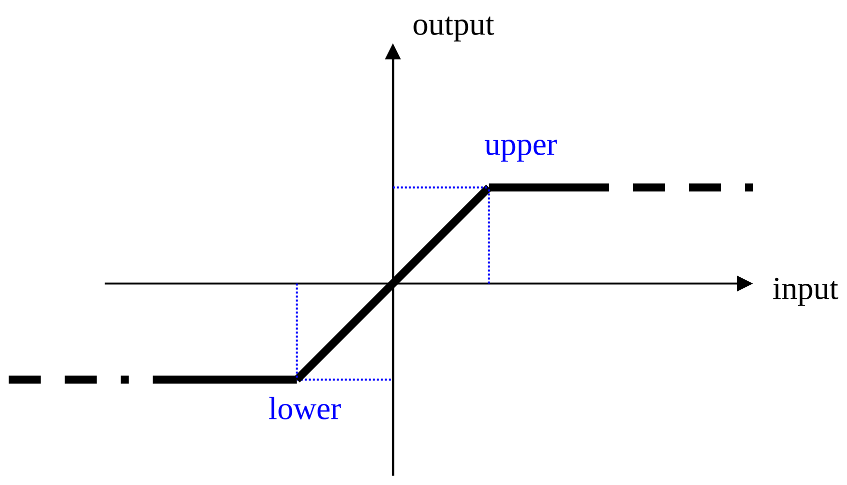
Output	*Type*	*Limits*	*Description*
output	CJ_SHORT	-32768..32767	Controlled output

Description

The *LIMIT* library compares the input value with upper and lower limits. The lower limit is specified by the *lower* input, and the upper limit by the *upper* input. The result of the function is returned as output according to the following criteria:

- If the input value is less than the lower limit *lower*, then the output reports the value of the lower limit.
- If the input value is greater than the upper limit *upper*, then the output reports the value of the upper limit.
- if the value is comprised between the two *lower* and *upper* limits, then the unaltered input value is reported as output.

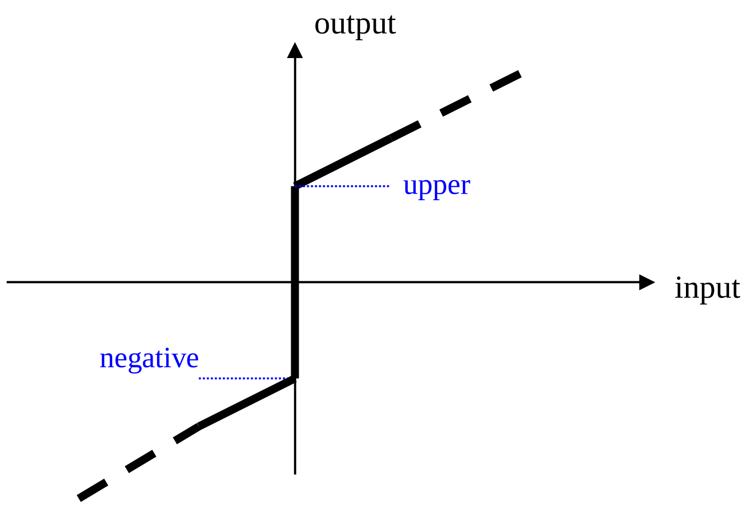
(Refer to the following graph).



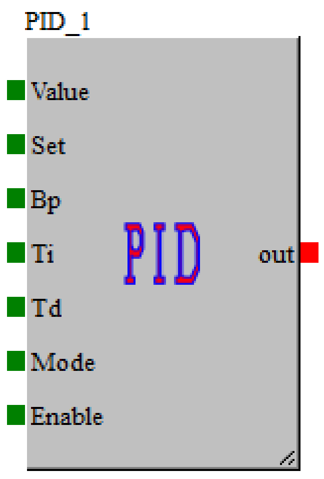
Input	*Type*	*Limits*	*Description*
negative	CJ_SHORT	-32768..positive	Negative offset
positive	CJ_SHORT	negative..32767	Positive offset
input	CJ_SHORT	-32768..32767	Input to be checked

Output	*Type*	*Limits*	*Description*
output	CJ_SHORT	-32768..32767	Controlled output

Description
The *ZONE* library adds an offset value to the input value *input*. The negative and positive offset values are assigned by the *negative* and *positive* inputs respectively. The result of the function is returned as output according to the following criteria: ➢ if the input value is less than zero, then the input value, to which the negative offset has been added, is reported as output. ➢ if the input value is zero, then zero is reported as output. ➢ if the input value is greater than zero, then the input value, to which the positive offset has been added, is reported as output. (Refer to the following graph).



PID

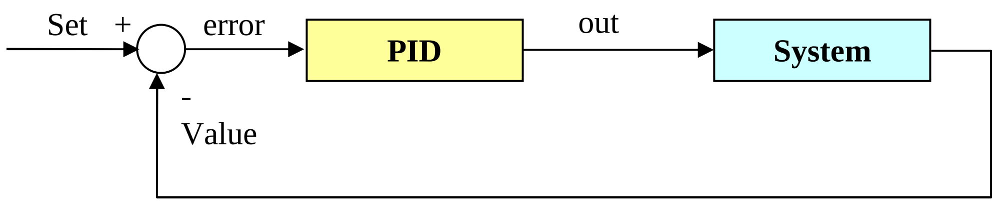


Input	*Type*	*Limits*	*Description*
Value	CJ_SHORT	-32768..32767	Value to be controlled
Set	CJ_SHORT	-32768..32767	Set-point
Bp	CJ_WORD	0..65535	Proportional band
Ti	CJ_WORD	0..999	Integration time (in seconds)
Td	CJ_WORD	0..999	Differentiation time (in seconds)
Mode	CJ_BIT	0-1	Mode of operation 0 = Direct action 1 = Inverse action
Enable	CJ_BIT	0-1	Enable PID regulation

Output	*Type*	*Limits*	*Description*
out	CJ_WORD	0..10000	Percentage output, to two decimal places

Description

This library performs the function of the PID regulator, which produces a control signal which is the sum of a term proportional to the error (part P), one proportional to its integral (part I) and one to its derivative (part D). The logic provided by the regulator follows the idea defined in the following figure:



Depending on how the integration and differentiation times are set, the general PID regulator may be simplified into three types of derivative regulators. In particular:

- If the *Ti* parameter is equal to zero, the calculation of the integral part is not performed, and the regulator will behave as a proportional-derivative regulator, PD
- If the *Td* parameter is equal to zero, the calculation of the derivative part is not performed, and the regulator will behave as a proportional-integral regulator, PI
- If both parameters, *Td* and *Ti*, are set to zero, then the regulator will behave as a purely proportional regulator P

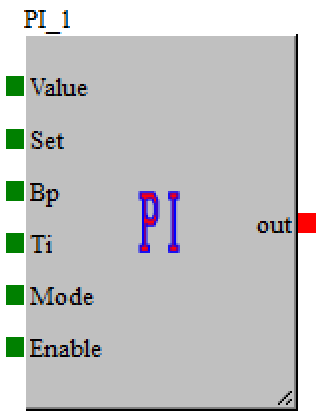
In the case where all the parameters are set to values other than zero, the final function will be the sum of three contributions, each of which having a different action.
The *proportional action* returns an output in proportion with the error measured. It is a function of the error, with a gain inversely proportional to the band width.
The *integral action* is used to bring the regulator output to a set-point value and reduces the error due to the proportional action alone.
The *derivative action* has the role of reducing fluctuations in the reference value and bring the output to set-point as quickly as possible. Depending on the reference value displacement, the action will have a braking or accelerating effect on the final output.
The output of the three components will produce a term, in percentage points to two decimal places, which will correspond to the quantity of signal to be produced to ensure that the output is brought 'close' to the established value (S*et* parameter).
The *Mode* parameter distinguishes the regulator into direct (*Mode* = *0*), or inverse (*Mode* = *1*) acting system control. For example:

- *Mode* = *1*, if it desired to control a heating process
- *Mode* = *0*, if it desired to control a refrigeration process

Notes

For correct functioning of the library, the *Bp*, *Ti* e *Td* parameters are set on the basis of the dynamic characteristics of the system being controlled (for example by using one of the various PID auto-tuning methods)

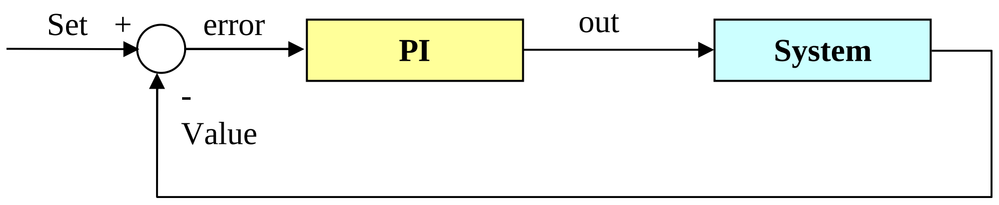
PI



Input	*Type*	*Limits*	*Description*
Value	CJ_SHORT	-32768..32767	Value to be controlled
Set	CJ_SHORT	-32768..32767	Set-point
Bp	CJ_WORD	0..65535	Proportional band
Ti	CJ_WORD	0..999	Integration time (in seconds)
Mode	CJ_BIT	0-1	Mode of operation 0 = Direct action 1 = Inverse action
Enable	CJ_BIT	0-1	Enable PI regulation

Output	*Type*	*Limits*	*Description*
out	CJ_WORD	0..10000	Percentage output, to two decimal places

Description
This library performs the function of the PI regulator, which produces a control signal which is the sum of a term proportional to the error (part P) and one proportional to its integral (part I). The logic provided by the regulator follows the idea defined in the following figure:



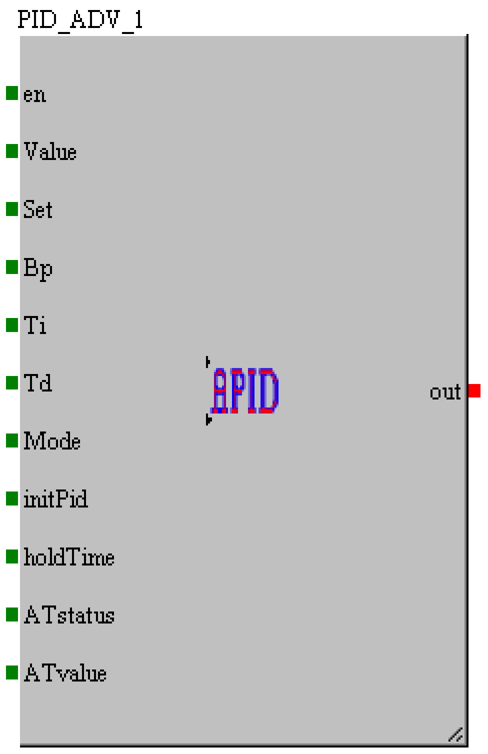
If the *Ti* parameter is equal to zero, the calculation of the integral part is not performed, and the regulator will behave as a simple proportional regulator, P. In the case where all the parameters are set to values other than zero, the final function will be the sum of two contributions, each of which having a different action. The *proportional action* returns an output in proportion with the error measured. It is a function of the error, with a gain inversely proportional to the band width.

The *integral action* is used to bring the regulator output to a set-point value and reduces the error due to the proportional action alone.
The output of the two components will produce a term, in percentage points to two decimal places, which will correspond to the quantity of signal to be produced to ensure that the output is brought 'close' to the established value (S*et* parameter).
The *Mode* parameter distinguishes the regulator into direct (*Mode = 0*), or inverse (*Mode = 1*) acting system control. For example:

- *Mode = 1*, if it desired to control a heating process
- *Mode = 0*, if it desired to control a refrigeration process

Notes
For correct functioning of the library, the *Bp* and *Ti* parameters are set on the basis of the dynamic characteristics of the system being controlled (for example by using one of the various PI auto-tuning methods)

PID_ADV



Input	*Type*	*Limits*	*Description*
en	CJ_BIT	0-1	Enable PID regulation
Value	CJ_SHORT	-32768..32767	Value to be controlled
Set	CJ_SHORT	-32768..32767	Set-point
Bp	CJ_WORD	0..65535	Proportional band
Ti	CJ_WORD	0..999	Integration time (in seconds)

Td	CJ_WORD	0..999	Differentiation time (in seconds)
Mode	CJ_BIT	0-1	Mode of operation 0 = Direct action (cooling) 1 = Inverse action (heating)
initPid	CJ_WORD	0..10000 [0.00 .. 100.00]	Percentage output initial value, to two decimal places
holdTime	CJ_WORD	0..65535	Timeout, in seconds, to maintain the output to the initial value, before the PID regulation start to work
ATstatus	CJ_BYTE	0-3	This pin must be connected to the *ATstatus* output in the library PID_AT. Possible values: 0 = Autotuning disabled 1 = Autotuning in progress 2 = Autotuning OK 3 = Autotuning failed
ATvalue	CJ_WORD	0..10000	This pin must be connected to the *ATvalue* output in the library PID_AT

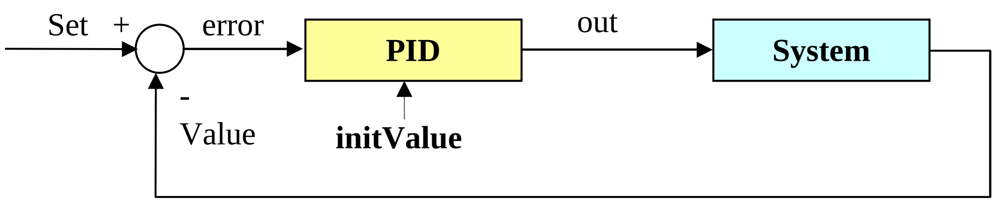
Output	***Type***	***Limits***	***Description***
out	CJ_WORD	0..10000 [0.00 .. 100.00]	Percentage output, to two decimal places

Description

This library realize the classic PID regulation, but with some advanced features:

- Until the PID regulator is not enabled (input *en* = 0) the output remains to 0.
- When the PID regulator is enabled (input *en* = 1), the output value is equal to the value present in the input *initValue*
- If the input *holdTime* is not connected, or is value is 0, the PID regulator starts to work from this value, else it waits that the *holdTime value* expires.

The logic provided by the regulator follows the idea defined in the following figure:

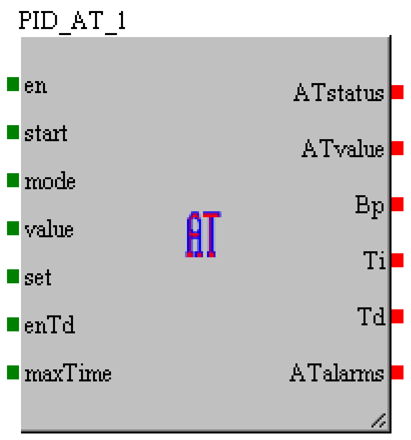


For an explanation of the PID regulation and their parameters, please refer to the PID library documentation.
The enable *en* input is optional: if it is not connected the functional block is always enabled.
The parameters *initValue* and *holdTime* are optionals: if not connected their default values are 0.
The *Mode* parameter distinguishes the regulator into direct (*Mode = 0*), or inverse (*Mode = 1*) acting system control. For example:

- *Mode* = *1*, if it desired to control a heating process
- *Mode* = *0*, if it desired to control a refrigeration process

Notes
For correct functioning of the library, the *Bp* and *Ti* parameters are set on the basis of the dynamic characteristics of the system being controlled (for example by using one of the various PI auto-tuning methods). This library can be estende with the *Autotuning* function, connecting in cascade it with the library PID_AT.

PID_AT

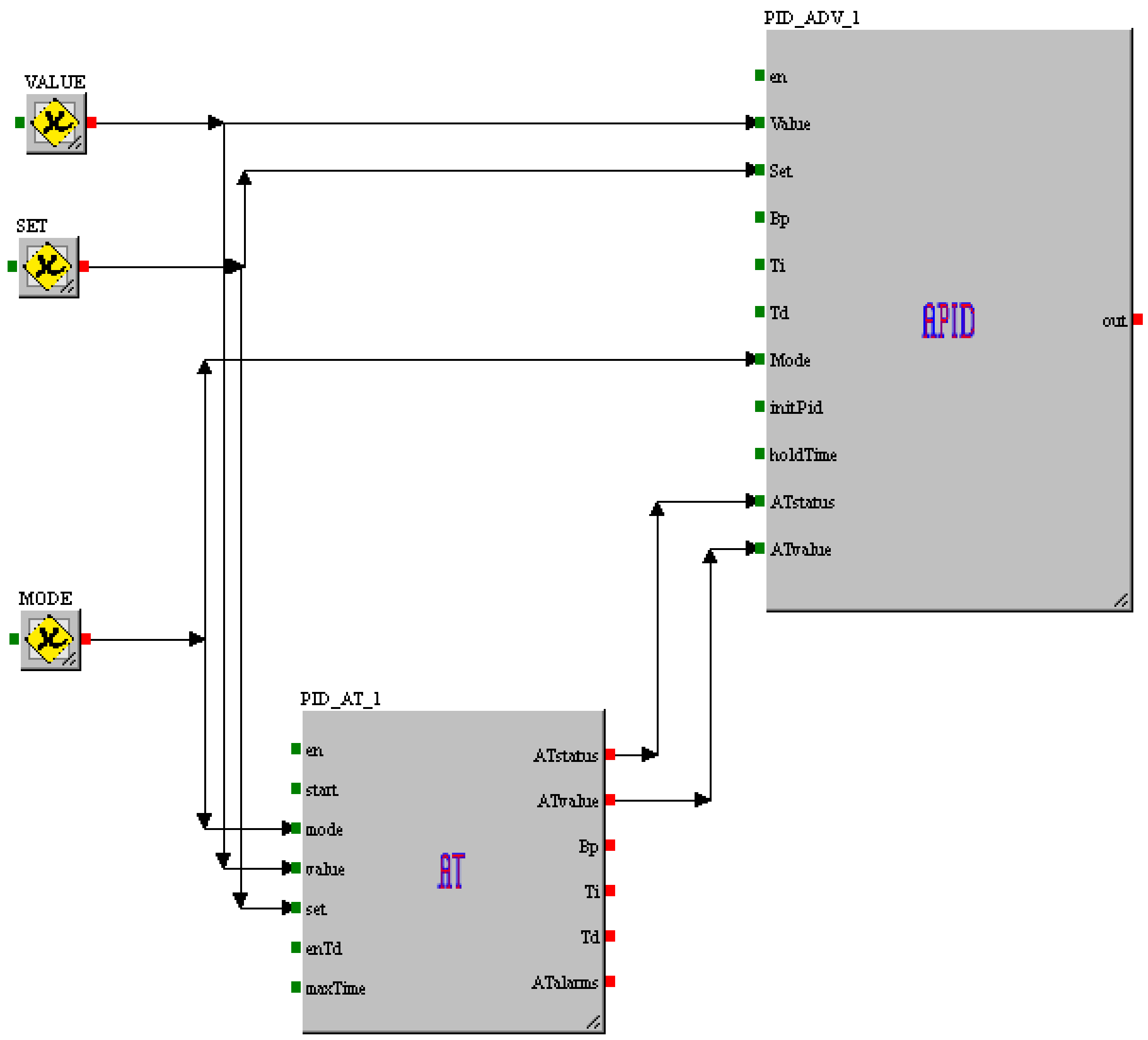


Input	***Type***	***Limits***	***Description***
en	CJ_BIT	0-1	Enable Autotuning procedure
start	CJ_BIT	0-1	Start Autotuning procedure (trigger input)
mode	CJ_BIT	0-1	Mode of operation 0 = Direct action (cooling) 1 = Inverse action (heating)
value	CJ_SHORT	-32768..32767	Value to be controlled
set	CJ_SHORT	-32768..32767	Set-point
enTd	CJ_BIT	0-1	Enable derivative action
maxTime	CJ_WORD	2-1440	Maximum time (in minutes) of PID parameters calculation after which the Auto-tuning procedure stops with failure. Default 30 minutes.

Output	***Type***	***Limits***	***Description***

ATstatus	CJ_BYTE	0-3	This pin must be connected to the ATstatus of PID_ADV library. Possible values: 0 = Auto-tuning disabled 1 = Auto-tuning in progress 2 = Auto tuning OK 3 = autotuning failed
ATvalue	CJ_WORD	0..10000	Output force value during the autotuning. This pin must be connected to the input *ATvalue* prices of PID_ADV library
Bp	CJ_WORD	0..30000	calculated proportional band
Ti	CJ_WORD	0..999	calculated integral time (in seconds)
Td	CJ_WORD	0..999	calculated derivative time (in seconds)
ATalarms	CJ_WORD	0-65535	It describes the possible cause of Auto-tuning operation failure. *bit0*: the time since the start has exceeded *maxTime* and it was unable to calculate new parameters *bit1*: application cycle time too long (2s) *bit2*: change of mode or set parameters when Auto-tuning was in progress *bit3*: they were measured invalid time values to calculate the parameters *bit4*: the calculated values of the parameters Pb, Ti and / or Td are off limits

Description
This library measure the dynamic response of a controlled system, and automatically calculates the PID control parameters: proportional band Pb, integral time Ti and, if enabled, derivative time Td. This library must be connected in cascade to the PID_ADV library. The Auto-tuning operation is based on the realization of forced oscillations (limit cycles) around the setpoint. At the end of 3 limit cycles, if the calculation conditions are sufficient, the new values of Pb, Ti and Td are calculated and the autotuning is successful. Otherwise it is diagnosed the type of error that prevented the calculation. Preparation of the system before starting the autotuning ➢ Prepare the connections between PID_ADV and PID_AT so that they use the same *set*, *value* and *mode*. Furthermore *ATstatus* and *ATvalue* of PID_AT must be connected to the respective inputs *ATstatus* and *ATvalue* of PID_ADV (see figure below). ➢ Ensure that the system to be controlled by PID is unstable or has a signal *value* does not fluctuate too much. In this case, filtering or mediate the signal. ➢ Pay attention to the setpoint with which you want to perform auto-tuning, it could be dangerous during the tuning process to overcome the setpoint minimum or maximum that could damage the process or some subsidiary parts. ➢ Do this tuning procedure in maximum security conditions. ➢ Only in the case of success (ATstatus = OK) copy the calculated values of Pb, Ti and Td in the corresponding control parameters of PID_ADV For a complete example, see the project PID_AT test.ucjp supplied with UniPro.

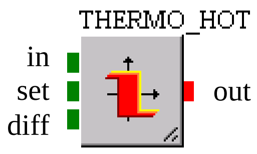


For an explanation of the PID control and parameters are documented by the PID library.
The *en* parameter is optional: if not connected the function block is always enabled.
The *maxTime* parameter is optional: if not connected its default maximum time is 30 minutes.
The *Mode* parameter distinguishes the regulator into direct (*Mode = 0*), or inverse (*Mode = 1*) acting system control. For example:

- *Mode = 1*, if it desired to control a heating process
- *Mode = 0*, if it desired to control a refrigeration process

Note
The *Autotuning* function implemented in this library is associated only to PID_ADV library.

THERMO_HOT

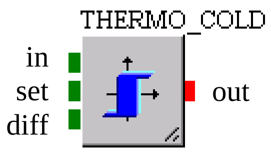


Input	*Type*	*Limits*	*Description*
in	CJ_SHORT	-32768..32767	Input value
set	CJ_SHORT	-32768..32767	Set-point
diff	CJ_SHORT	-32768..32767	Differential

Output	*Type*	*Limits*	*Description*
out	CJ_BIT	0-1	Regulator status

Description
This library performs the role of an inverse-acting on-off regulator, *i.e.*: ➢ if the input value *in*, is less than the set-point value *set*, having subtracted the differential *diff*, the output (*out* = 1) is activated ➢ if the input value *in* is greater than or equal to the set-point value *set*, the output is deactivated (*out* = 0) For example, this library may be used to regulate a heating process, *i.e.* a thermostat with hysteresis below the set-point.

THERMO_COLD

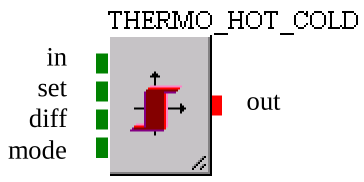


Input	*Type*	*Limits*	*Description*
in	CJ_SHORT	-32768..32767	Input value
set	CJ_SHORT	-32768..32767	Set-point
diff	CJ_SHORT	-32768..32767	Differential

Output	*Type*	*Limits*	*Description*
out	CJ_BIT	0-1	Regulator status

Description
This library performs the role of an direct-acting on-off regulator, *i.e.*: ➢ if the input value *in*, is greater than the set-point value *set* summed with the differential *diff*, the output (*out* = 1) is activated ➢ if the input value *in* is less than or equal to the set-point value *set*, the output is deactivated (*out* = 0) For example, this library may be used to regulate a refrigeration process, *i.e.* a thermostat with hysteresis above the set-point.

THERMO_HOT_COLD



Input	***Type***	***Limits***	***Description***
in	CJ_SHORT	-32768..32767	Input value
set	CJ_SHORT	-32768..32767	Set-point
diff	CJ_SHORT	-32768..32767	Differential
mode	CJ_BIT	0-1	Mode of operation 0 = Direct action 1 = Inverse action

Output	***Type***	***Limits***	***Description***
out	CJ_BIT	0-1	Regulator status

Description

This library permits performing both direct and inverse actions, by altering the *mode* input.
With *mode = 0,* direct action.

- if the input value *in,* is greater than the set-point value *set* summed with the differential *diff*, the output (*out* = 1) is activated
- if the input value *in* is less than or equal to the set-point value *set*, the output is deactivated (*out* = 0)

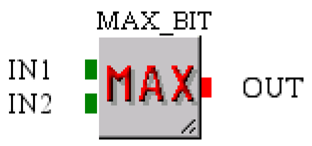
Direct action is necessary when dealing with a thermostat with hysteresis above the set-point, for example for a refrigeration process.
With *mode = 1,* inverse action.

- if the input value *in,* is less than the set-point value *set*, by subtracting the differential *diff*, the output (*out* = 1) is activated
- if the input value *in* is greater than or equal to the set-point value *set*, the output is deactivated (*out* = 0)

Inverse action is necessary when dealing with a thermostat with hysteresis below the set-point, for example for a heating process.

3.11 Selection

MAX_BIT

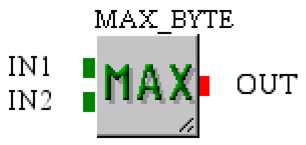


Input	*Type*	*Limits*	*Description*
IN1	CJ_BIT	0-1	Data input 1
IN2	CJ_BIT	0-1	Data input 2

Output	*Type*	*Limits*	*Description*
OUT	CJ_BIT	0-1	Output value

Description
Compares inputs *IN1* and *IN2* and returns the largest value.

MAX_BYTE

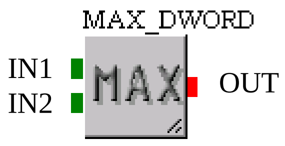


Input	*Type*	*Limits*	*Description*
IN1	CJ_BYTE	0..255	Data input 1
IN2	CJ_BYTE	0..255	Data input 2

Output	*Type*	*Limits*	*Description*
OUT	CJ_BYTE	0..255	Output value

Description
Compares inputs *IN1* and *IN2* and returns the largest value.

MAX_DWORD

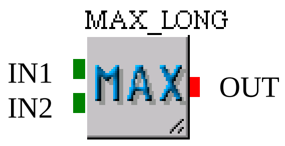


Input	*Type*	*Limits*	*Description*
IN1	CJ_DWORD	0..4294967295	Data input 1
IN2	CJ_DWORD	0..4294967295	Data input 2

Output	*Type*	*Limits*	*Description*
OUT	CJ_DWORD	0..4294967295	Output value

Description
Compares inputs *IN1* and *IN2* and returns the largest value.

MAX_LONG

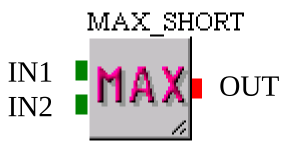


Input	*Type*	*Limits*	*Description*
IN1	CJ_LONG	-2147483648.. 2147483647	Data input 1
IN2	CJ_LONG	-2147483648.. 2147483647	Data input 2

Output	*Type*	*Limits*	*Description*
OUT	CJ_LONG	-2147483648.. 2147483647	Output value

Description
Compares inputs *IN1* and *IN2* and returns the largest value.

MAX_SHORT

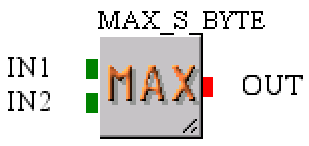


Input	*Type*	*Limits*	*Description*
IN1	CJ_SHORT	-32768..32767	Data input 1
IN2	CJ_SHORT	-32768..32767	Data input 2

Output	*Type*	*Limits*	*Description*
OUT	CJ_SHORT	-32768..32767	Output value

Description
Compares inputs *IN1* and *IN2* and returns the largest value.

MAX_S_BYTE

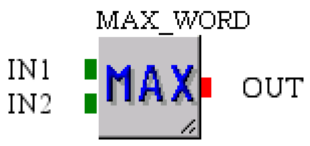


Input	*Type*	*Limits*	*Description*
IN1	CJ_S_BYTE	-128..127	Data input 1
IN2	CJ_S_BYTE	-128..127	Data input 2

Output	*Type*	*Limits*	*Description*
OUT	CJ_S_BYTE	-128..127	Output value

Description
Compares inputs *IN1* and *IN2* and returns the largest value.

MAX_WORD

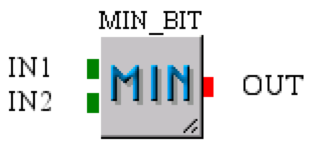


Input	*Type*	*Limits*	*Description*
IN1	CJ_WORD	0..65535	Data input 1
IN2	CJ_WORD	0..65535	Data input 2

Output	*Type*	*Limits*	*Description*
OUT	CJ_WORD	0..65535	Output value

Description
Compares inputs *IN1* and *IN2* and returns the largest value.

MIN_BIT

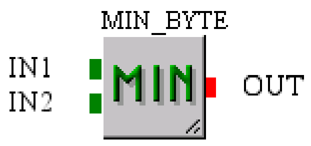


Input	*Type*	*Limits*	*Description*
IN1	CJ_BIT	0-1	Data input 1
IN2	CJ_BIT	0-1	Data input 2

Output	*Type*	*Limits*	*Description*
OUT	CJ_BIT	0-1	Output value

Description
Compares inputs *IN1* and *IN2* and returns the smaller value.

MIN_BYTE

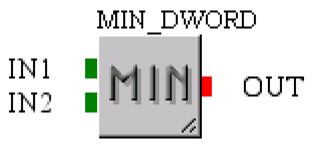


Input	*Type*	*Limits*	*Description*
IN1	CJ_BYTE	0..255	Data input 1
IN2	CJ_BYTE	0..255	Data input 2

Output	*Type*	*Limits*	*Description*
OUT	CJ_BYTE	0..255	Output value

Description
Compares inputs *IN1* and *IN2* and returns the smaller value.

MIN_DWORD

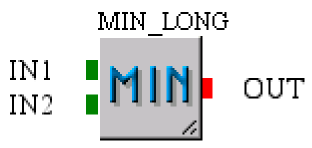


Input	*Type*	*Limits*	*Description*
IN1	CJ_DWORD	0..4294967295	Data input 1
IN2	CJ_DWORD	0..4294967295	Data input 2

Output	*Type*	*Limits*	*Description*
OUT	CJ_DWORD	0..4294967295	Output value

Description
Compares inputs *IN1* and *IN2* and returns the smaller value.

MIN_LONG

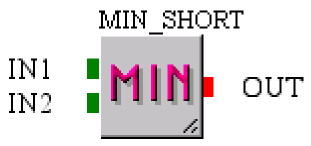


Input	*Type*	*Limits*	*Description*
IN1	CJ_LONG	-2147483648.. -2147483647	Data input 1
IN2	CJ_LONG	-2147483648.. -2147483647	Data input 2

Output	*Type*	*Limits*	*Description*
OUT	CJ_LONG	-2147483648.. -2147483647	Output value

Description
Compares inputs *IN1* and *IN2* and returns the smaller value.

MIN_SHORT

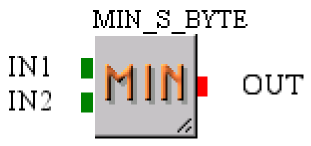


Input	*Type*	*Limits*	*Description*
IN1	CJ_SHORT	-32768..32767	Data input 1
IN2	CJ_SHORT	-32768..32767	Data input 2

Output	*Type*	*Limits*	*Description*
OUT	CJ_SHORT	-32768..32767	Output value

Description
Compares inputs *IN1* and *IN2* and returns the smaller value.

MIN_S_BYTE



Input	*Type*	*Limits*	*Description*
IN1	CJ_S_BYTE	-128..127	Data input 1
IN2	CJ_S_BYTE	-128..127	Data input 2

Output	*Type*	*Limits*	*Description*
OUT	CJ_S_BYTE	-128..127	Output value

Description
Compares inputs *IN1* and *IN2* and returns the smaller value.

MIN_WORD

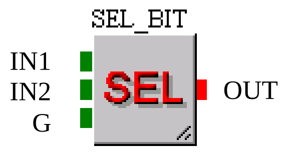
MIN_WORD
IN1
IN2
MIN
OUT

Input	*Type*	*Limits*	*Description*
IN1	CJ_WORD	0..65535	Data input 1
IN2	CJ_WORD	0..65535	Data input 2

Output	*Type*	*Limits*	*Description*
OUT	CJ_WORD	0..65535	Output value

Description
Compares inputs *IN1* and *IN2* and returns the smaller value.

SEL_BIT

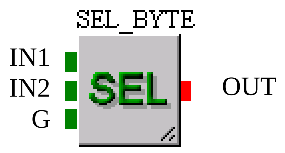


Input	*Type*	*Limits*	*Description*
IN1	CJ_BIT	0-1	Data input 1
IN2	CJ_BIT	0-1	Data input 2
dd	CJ_BIT	0-1	Selection input

Output	*Type*	*Limits*	*Description*
OUT	CJ_BIT	0-1	Output value

Description
If the digital input *G* is set to ‘0’ the output *OUT* will assume the value of *IN1* otherwise it will assume the value of *IN2*.

SEL_BYTE

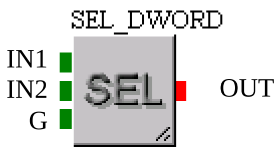


Input	*Type*	*Limits*	*Description*
IN1	CJ_BYTE	0..255	Data input 1
IN2	CJ_BYTE	0..255	Data input 2
dd	CJ_BIT	0-1	Selection input

Output	*Type*	*Limits*	*Description*
OUT	CJ_BYTE	0..255	Output value

Description
If the digital input *G* is set to ‘0’ the output *OUT* will assume the value of *IN1* otherwise it will assume the value of *IN2*.

SEL_DWORD

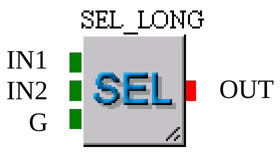


Input	*Type*	*Limits*	*Description*
IN1	CJ_DWORD	0..4294967295	Data input 1
IN2	CJ_DWORD	0..4294967295	Data input 2
dd	CJ_BIT	0-1	Selection input

Output	*Type*	*Limits*	*Description*
OUT	CJ_DWORD	0..4294967295	Output value

Description
If the digital input *G* is set to ‘0’ the output *OUT* will assume the value of *IN1* otherwise it will assume the value of *IN2*.

SEL_LONG

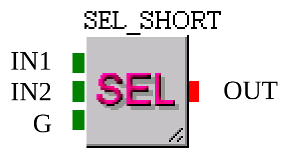


Input	*Type*	*Limits*	*Description*
IN1	CJ_LONG	-2147483648..2147483647	Data input 1
IN2	CJ_LONG	-2147483648..2147483647	Data input 2
dd	CJ_BIT	0-1	Selection input

Output	*Type*	*Limits*	*Description*
OUT	CJ_LONG	-2147483648..2147483647	Output value

Description
If the digital input *G* is set to ‘0’ the output *OUT* will assume the value of *IN1* otherwise it will assume the value of *IN2*.

SEL_SHORT

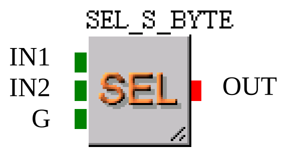


Input	*Type*	*Limits*	*Description*
IN1	CJ_SHORT	-32768..32767	Data input 1
IN2	CJ_SHORT	-32768..32767	Data input 2
dd	CJ_BIT	0-1	Selection input

Output	*Type*	*Limits*	*Description*
OUT	CJ_SHORT	-32768..32767	Output value

Description
If the digital input *G* is set to ‘0’ the output *OUT* will assume the value of *IN1* otherwise it will assume the value of *IN2*.

SEL_S_BYTE

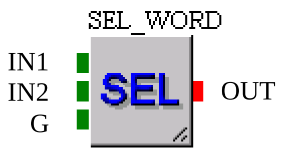


Input	*Type*	*Limits*	*Description*
IN1	CJ_S_BYTE	-128..127	Data input 1
IN2	CJ_S_BYTE	-128..127	Data input 2
dd	CJ_BIT	0-1	Selection input

Output	*Type*	*Limits*	*Description*
OUT	CJ_S_BYTE	-128..127	Output value

Description
If the digital input *G* is set to ‘0’ the output *OUT* will assume the value of *IN1* otherwise it will assume the value of *IN2*.

SEL_WORD



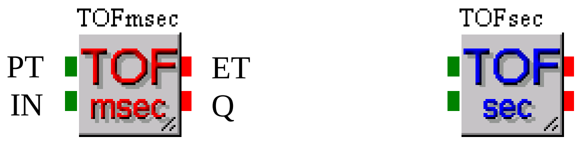
Input	***Type***	***Limits***	***Description***
IN1	CJ_WORD	0..65535	Data input 1
IN2	CJ_WORD	0..65535	Data input 2
dd	CJ_BIT	0-1	Selection input

Output	***Type***	***Limits***	***Description***
OUT	CJ_WORD	0..65535	Output value

Description
If the digital input *G* is set to ‘0’ the output *OUT* will assume the value of *IN1* otherwise it will assume the value of *IN2*.

3.12 Timers

TOF (Off Delay Timer)

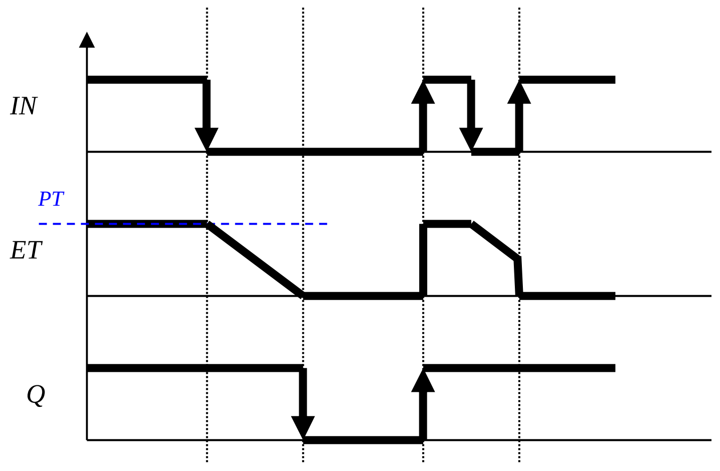


Input	*Type*	*Limits*	*Description*
PT	CJ_WORD	0..65535	Preset Time. A 16 bit value which identifies the off delay time (hundreds of ms for TOFmsec and seconds for TOFsec)
IN	CJ_BIT	0-1	Timer On. When a trailing edge is detected, the internal timer is made to start. If a leading edge is detected before the internal timer has reached the value *PT*, then the output *Q* will not be turned off.

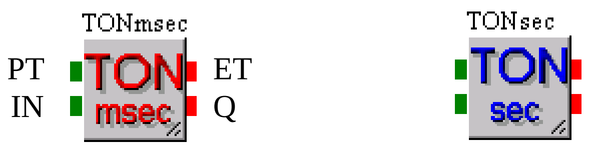
Output	*Type*	*Limits*	*Description*
ET	CJ_WORD	0..65535	Elapsed Time. Represents the current elapsed time value (hundreds of ms for TOFmsec and seconds for TOFsec)
Q	CJ_BIT	0-1	Signal Output. Represents reaching the pre-established time. Assumes a value of ‘0’ if *PT* = *ET*, otherwise‘1’

Description
The TOF library allows the insertion of an off delay time, for example to switch off a machine fan a predetermined length of time after the machine itself is switched off.

Notes
Please refer to the following graph.



TON (On Delay Timer)

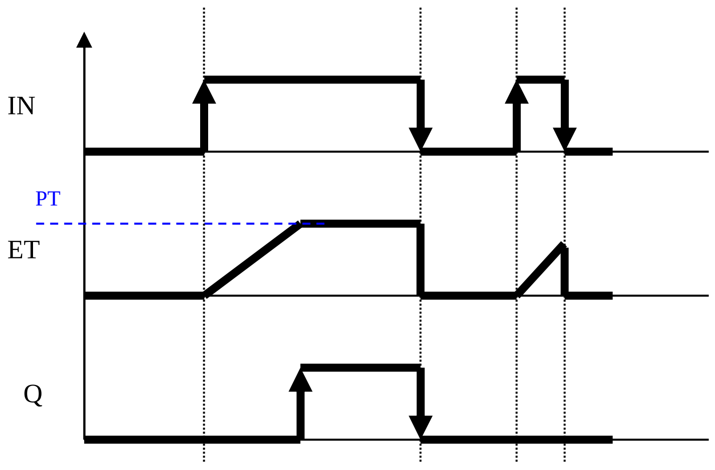


Input	*Type*	*Limits*	*Description*
PT	CJ_WORD	0..65535	Preset Time. A 16 bit value which identifies the on delay time (hundreds of ms for TONmsec and seconds for TONsec)
IN	CJ_BIT	0-1	Timer On. When a leading edge is detected, the internal timer is made to start.

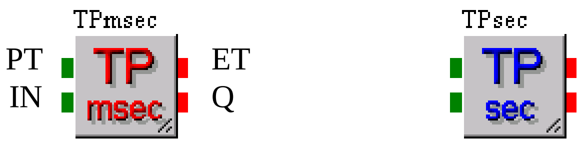
Output	*Type*	*Limits*	*Description*
ET	CJ_WORD	0..65535	Elapsed Time. Represents the current elapsed time value (hundreds of ms for TONmsec and seconds for TONsec)
Q	CJ_BIT	0-1	Signal Output. Represents reaching the pre-established time. Assumes a value of ‘1’ if *PT* = *ET*, otherwise‘0’

Description
The TON library allows inserting an on delay time, for example to switch on a compressor only after a prefixed period of time following the switch-on request.

Notes
Please refer to the following graph.



TP (Pulse Timer)

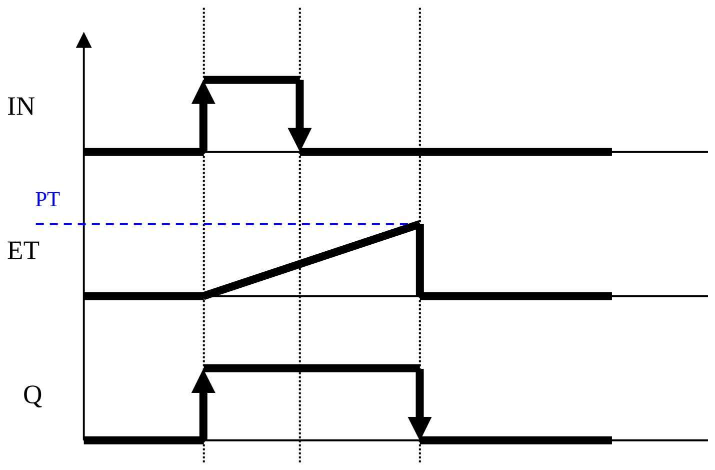


Input	*Type*	*Limits*	*Description*
PT	CJ_WORD	0..65535	Preset Time. A 16 bit value which identifies the time counted by the counter (hundreds of ms for TPmsec and seconds for TPsec)
IN	CJ_BIT	0-1	Pulse Timer. With each leading edge detected in the input *IN* the timer is loaded with a time *PT*. If a leading edge is detected during counting, the timer is not loaded

Output	***Type***	***Limits***	***Description***
ET	CJ_WORD	0-1	Elapsed Time. Represents the current elapsed time value (hundreds of ms for TPmsec and seconds for TPsec)
Q	CJ_BIT	0-1	Signal Output. The output *Q* is at '1' for the period of time *TP* until a new leading edge is detected in input *IN*

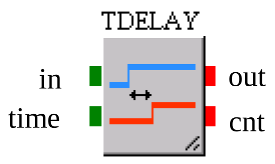
Description
The TP library allows inserting a pulse time, for example for times operation of a light.

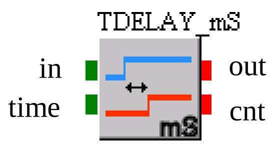
Notes
Please refer to the following graph.



3.13 Timing

TDELAY



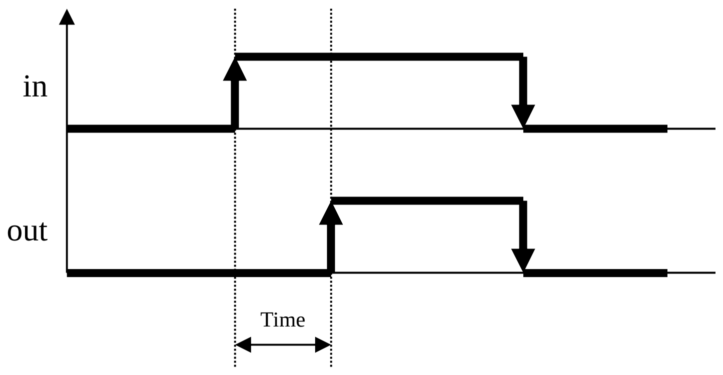


Input	*Type*	*Limits*	*Description*
in	CJ_BIT	0-1	Input signal
time	CJ_WORD	0..65535	Delay time (on the basis of seconds for TDELAY and on 100 ms for TDELAY_mS)

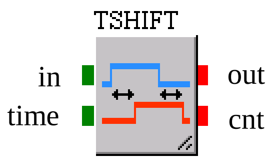
Output	*Type*	*Limits*	*Description*
cnt	CJ_WORD	0..65535	Current count for the delay to be applied
out	CJ_BIT	0-1	Output signal

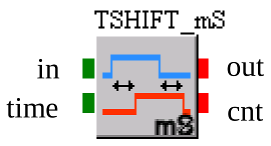
Description
The *TDELAY* and *TDELAY_mS* libraries allow adding a delay to the input signal. The leading edge of the output signal will be translated by a length of time *time* with respect to the input signal.

Notes
Please refer to the following graph.



TSHIFT



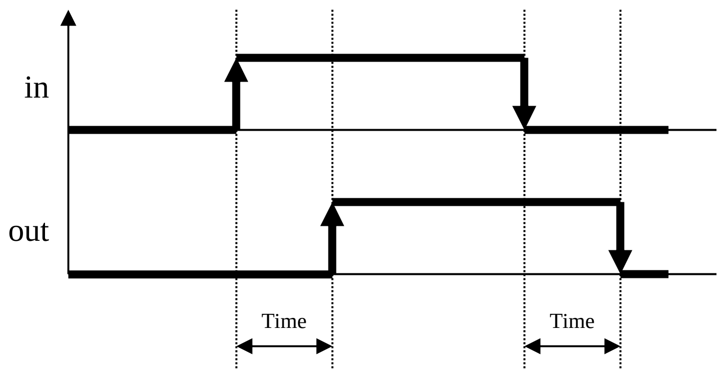


Input	*Type*	*Limits*	*Description*
in	CJ_BIT	0-1	Input signal
time	CJ_WORD	0..65535	Delay time (on the basis of seconds for TSHIFT and on 100 ms for TSHIFT_mS)

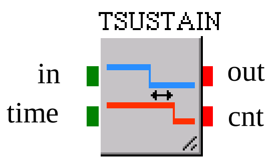
Output	*Type*	*Limits*	*Description*
cnt	CJ_WORD	0..65535	Current count for the delay to be applied
out	CJ_BIT	0-1	Output signal

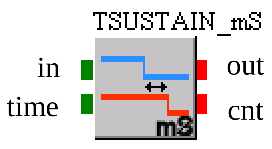
Description
The *TSHIFT* and *TSHIFT_mS* libraries allow adding a delay to the input signal. Both edges (leading and trailing) of the output signal will be translated by a length of time *time* with respect to the input signal.

Notes
Please refer to the following graph.



TSUSTAIN



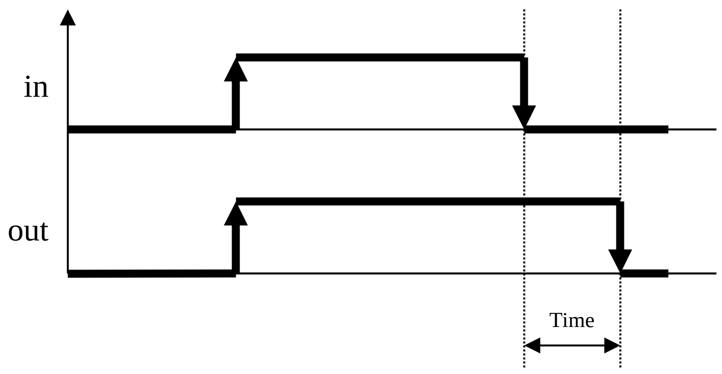


Input	***Type***	***Limits***	***Description***
in	CJ_BIT	0-1	Input signal
time	CJ_WORD	0..65535	Delay time (on the basis of seconds for TSUSTAIN and on 100 ms for TSUSTAIN_mS)

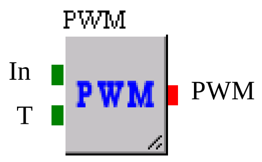
Output	***Type***	***Limits***	***Description***
cnt	CJ_WORD	0..65535	Current count for the delay to be applied
out	CJ_BIT	0-1	Output signal

Description
The *TSUSTAIN* and *TSUSTAIN_mS* libraries allow adding a delay to the input signal. The trailing edge of the output signal will be translated by a length of time *time* with respect to the input signal.

Notes
Please refer to the following graph.



PWM

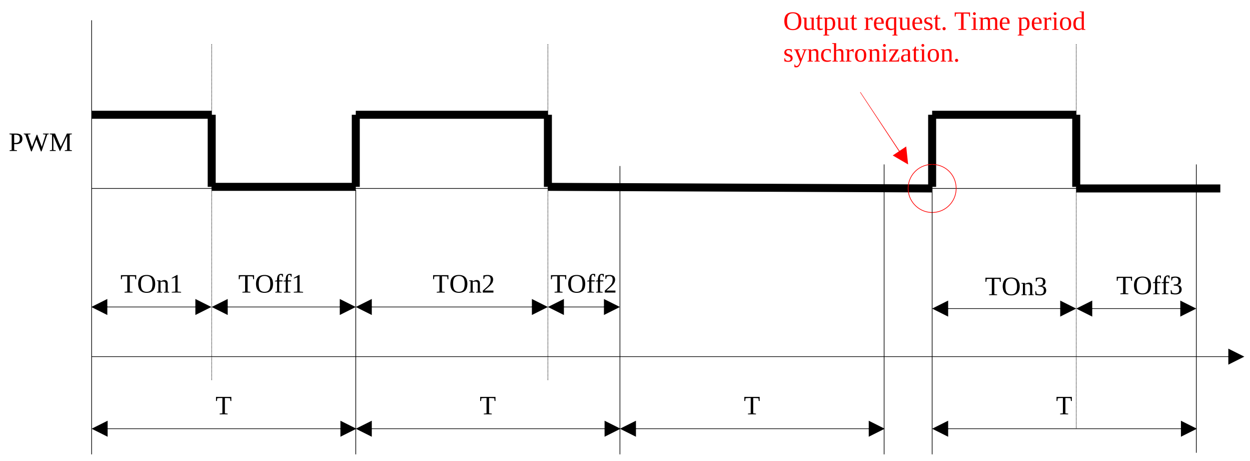


Ingresso	*Tipo*	*Limiti*	*Descrizione*
In	CJ_WORD	0..10000	Percentage input, to two decimal places
T	CJ_BYTE	0..255	Function time period (seconds)

Uscita	*Tipo*	*Limiti*	*Descrizione*
PWM	CJ_BIT	0-1	Function output

Descrizione
The PWM (Pulse Width Modulation) library uses two inputs to calculate a binary output (on/off). This is a periodic function with time period T. It provides a time-modulated output based on the value of the input *In*. This input determines the percentage of the period T that the PWM output will be at logic '1' (on). For the remainder of the period the output will be at logic'0'(Toff = T – Ton). The output can assume the logic value '1' only once within each period. If the value of *In* changes from zero (output permanently off) to some other value, the start of the period is re-synchronised and the output is turned on immediately. This feature is useful to avoid delays when immediate output activation is needed after an idle period.

Note
Internally counts are controlled under a 100mS interrupt. It's advisable to set the *timed* property of the entity connected at the PWM library output (for example a DIGITALOUT) on *"Timed 100mS"*. Please refer to the following graph.



UNI-PRO - Standard libraries manual.
Version 4.2 - July 2020.
Code 114UPROSLE42.
File 114UPROSLE42.pdf.

HEADQUARTERS
Evco
Via Mezzaterra 6, 32036 Sedico Belluno ITALY
Tel. +39 0437-852468
Fax +39 0437-83648
info@evco.it
www.evco.it

OVERSEAS OFFICES
Control France
155 Rue Roger Salengro, 92370 Chaville Paris FRANCE
Tel. 0033-1-41159740
Fax 0033-1-41159739
control.france@wanadoo.fr

Evco Latina
Larrea, 390 San Isidoro, 1609 Buenos Aires ARGENTINA
Tel. 0054-11-47351031
Fax 0054-11-47351031
evcolatina@anykasrl.com.ar

Evco Pacific
59 Premier Drive Campbellfield, 3061, Victoria Melbourne, AUSTRALIA
Tel. 0061-3-9357-0788
Fax 0061-3-9357-7638
everycontrol@pacific.com.au

Evco Russia
111141 Russia Moscow 2-oy Proezd Perova Polya 9
Tel. 007-495-3055884
Fax 007-495-3055884
info@evco.ru

Every Control do Brasil
Rua Marino Félix 256, 02515-030 Casa Verde São Paulo SÃO PAULO BRAZIL
Tel. 0055-11-38588732
Fax 0055-11-39659890
info@everycontrol.com.br

Every Control Norden
Cementvägen 8, 136 50 Haninge SWEDEN
Tel. 0046-8-940470
Fax 0046-8-6053148
mail2@unilec.se

Every Control Shangai
B 302, Yinhai Building, 250 Cao Xi Road, 200235 Shangai CHINA
Tel. 0086-21-64824650
Fax 0086-21-64824649
evcosh@online.sh.cn

Every Control United Kingdom
Unit 19, Monument Business Park, OX44 7RW Chalgrowe, Oxford, UNITED KINGDOM
Tel. 0044-1865-400514
Fax 0044-1865-400419
info@everycontrol.co.uk