OSCAT Network Documentation

Franz Höpfinger

Contents

1. OSCAT Network Documentation

Welcome to the OSCAT Network Library documentation! (Download documentation as PDF)
This library contains function blocks for network communication, protocols, and internet services.

1.1 Categories

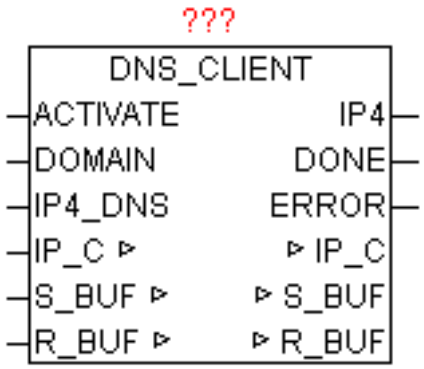
- **NETWORK** - Network protocols and communication
- **NET_VAR** - Network variables
- **OTHER** - Other functions
- **TELNET_VISION** - Telnet and visualization
- **WEATHER** - Weather services and data evaluation

1.2 License

2. NETWORK

2.0.1 DNS_CLIENT

Type	Function module	
IN_OUT IP_C	IP_C (parameterization)	
S_BUF	NETWORK_BUFFER (transmit data)	
R_BUF	NETWORK_BUFFER (receive data)	
INPUT ACTIVATE	BOOL	(Query start by positive edge)
DOMAIN	STRING	(Domain name or IP as String)
IP4_DNS	DWORD (IPv4 address of the DNS server)	
OUTPUT IP4	DWORD (IPv4 address of the requested domain)	
DONE	BOOL (IP of the domain has been queried successfully)	
ERROR	DWORD (error code)	
	DNS_CLIENT determine from the given qualified DOMAIN name the associated IPv4 address eg “www.oscat.de” . For this purpose, a DNS query to a DNS server for configured DOMAIN name with is made. With positive edge of ACTIVATE the specified DOMAIN is stored so that they no longer must be present. If the query provide more IP addresses, so always he highest value of the TTL (Time To Live) is used. As IP4_DNS can be used any public DNS servers. If the PLC is sitting behind a DSL router, this router can be used through its gateway address as a DNS server. Which ultimately leads to faster even with repeated requests response times because they are managed in the router cache. With positive results DONE = TRUE the IP4 contains the requested IP address until the start of the next query by positive edge of ACTIVATE. If in the DOMAIN name a valid IPv4 address is detected, no more DNS query is made and it is passed in converted type to IPv4 and DONE is set to TRUE. ERROR gives, if an error occurs, the exact cause.	
Error Codes		

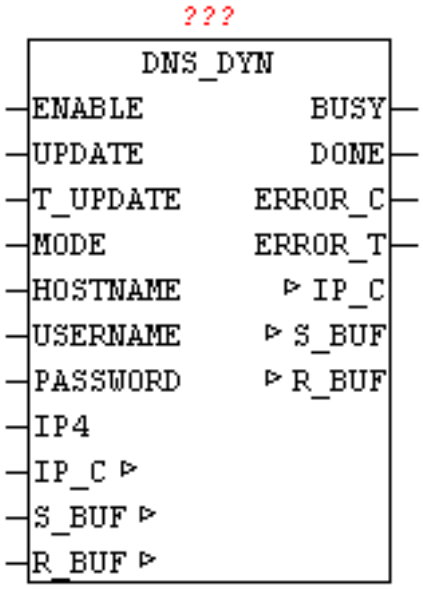


Value	Source	Description
B3	B2	B1
nn	nn	nn
xx	xx	xx
xx	xx	xx
xx	xx	xx
xx	xx	xx
xx	xx	xx
xx	xx	xx
xx	xx	xx
xx	xx	xx
xx	xx	xx
xx	xx	xx
xx	xx	xx
xx	xx	xx
xx	xx	xx

2.0.2 DNS_DYN

Type	Function module
INPUT ENABLE	BOOL (release of the module)
UPDATE	BOOL (Launches new DNS registration immediately)
T_UPDATE	TIME (waiting time for new DNS registration)
MODE	BYTE (selection of DynDNS provider)
HOST NAME	STRING (30) (own domain name)
USER NAME	STRING(20) (name for registration)
PASSWORD	STRING(20) (password for registration)
IP4	DWORD (Optional specify the IP address)
OUTPUT BUSY	BOOL (module is active, query is performed)
DONE	BOOL (performed DNS registration successful)

ERROR_C	DWORD (error code)
ERROR_T	BYTE (error type)
IN_OUT IP_C	IP_C (parameterization)
S_BUF	NETWORK_BUFFER (transmit data)
R_BUF	NETWORK_BUFFER (receive data)
	WIth DNS_DYN dynamic IP addresses are registered as domain names. Many Internet providers assign a dynamic IP address when dialing into the Internet. To be visible and accessible for Internet Participants, one of the ways is to upgrade its current IP address via Dyn-DNS. The process is not standardized, unfortunately, so for every Dyn-DNS provider has to be created a individual solution. The module can be used in conjunction with DynDNS.org and Selfhost.de. These providers offer in addition to paid also free DynDNS services.
	If ENABLE is set to TRUE, then the module is active. Using a positive edge to UPDATE any time an update can be started. If at T_UPDATE a time is specified, always an update is done after that time.
	Caution, most DynDNS providers rates a frequent or unnecessary update as an attack, and block the account for a certain time.
	The time T_UPDATE should not be set below an hour. If the parameter T_UPDATE is not connect it is assumed as an update time of 1 hour. If no update is needed on time, then T#0ms should be passed.
	The MODE parameter allows the selection of DynDNS Provider
	(0 = DynDNS.org, 1 = SELFHOST.DE)
	The own domain name must be passed by the hostname. For security reasons, USERNAME and PASSWORD as authorization data must be specified to the DynDNS provider. If the parameter IP4 is not used, so DynDNS provider automatically adopts the current registration-IP as WAN IP with which the update is performed. By specifying an IP address also an individual IP address may be assigned.
	With flawless execution the parameter DONE = TRUE, else ERROR_C and ERROR_T passes the error code and error type. (See error codes).
ERROR_T	

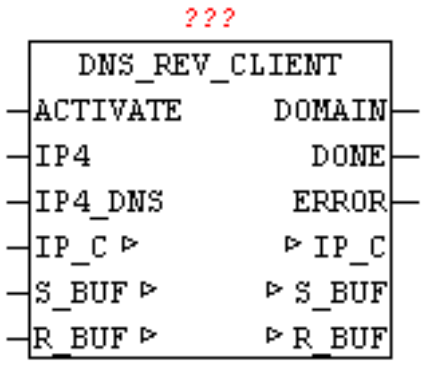


Value	Properties
1	The exact meaning of ERROR_C can be read at module DNS_CLIENT
2	The exact meaning of ERROR_C can be read at module HTTP_GET
3	The DynDNS provider has refused registration

2.0.3 DNS_REV_CLIENT

Type	Function module	
IN_OUT IP_C	IP_C (parameterization)	
S_BUF	NETWORK_BUFFER (transmit data)	
R_BUF	NETWORK_BUFFER (receive data)	
INPUT ACTIVATE	BOOL (query start by positive edge)	
DOMAIN	STRING	(Domain name or IP as String)
IP4_DNS	DWORD (IPv4 address of the DNS server)	
OUTPUT IP4	DWORD (IPv4 address of the requested domain)	
DONE	BOOL (IP of the domain has been queried successfully)	
ERROR	DWORD (error code)	
	DNS_REV_CLIENT determine from the given IP address the officially registered domain name. For this purpose a reverse DNS query on the configured IP address with a DNS server is made. With positive edge of ACTIVATE the specified IP is stored so that they no longer must be present. If the query result in more matches, it will always use the last record. As IP4_DNS can be used any pub-	

	lic DNS servers. If the PLC is sitting behind a DSL router, this router can be used as a DNS server through its gateway address. Which ultimately leads to faster even with repeated requests response times because they are managed in the router cache. With positive results DONE = TRUE the DOMAIN contains the officially registered domain name until the start of the next query by positive edge of ACTIVATE. ERROR gives ao error, the error code. (See error codes).	
Error Codes		



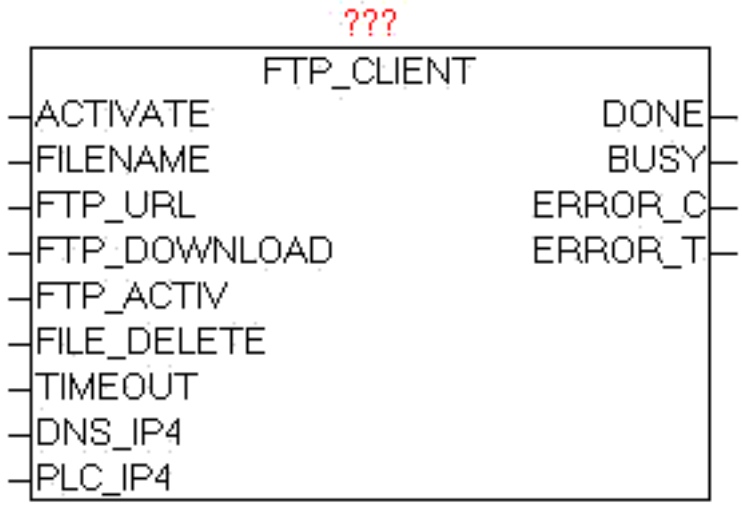
Value	Source	Description
B3	B2	B1
nn	nn	nn
xx	xx	xx
xx	xx	xx
xx	xx	xx
xx	xx	xx
xx	xx	xx
xx	xx	xx
xx	xx	xx
xx	xx	xx
xx	xx	xx
xx	xx	xx
xx	xx	xx

2.0.4 FTP_CLIENT

Type Function module	
INPUT ACTIVATE	BOOL (positive edge starts the query)
FILE NAME	STRING (file path/ filename)

FTP_URL	STRING(STRING_LENGTH) (FTP access path)
FTP_DOWNLOAD	BOOL (UPLOAD = 0 / DOWNLOAD = 1)
FTP_ACTIV	BOOL (PASSIV = 0 / ACTIV = 1)
FILE_DELETE	BOOL (delete files after transfer)
TIMEOUT	TIME (time)
Dns_ip4	DWORD (IP4 address of the DNS server)
Dns_ip4	DWORD (IP4 address of the DNS server)
OUTPUT DONE	BOOL (Transfer completed without error)
BUSY	BOOL (Transfer active)
ERROR_C	DWORD (Error code)
ERROR_T	BYTE (Problem type)
The module FTP_CLIENT is used to transfer files from the PLC to an FTP server and to transmitted from the FTP server to the PLC. A positive edge at ACTIVATE starts the transfer process. In FTP_DOWNLOAD the transmission direction can be specified. The parameter FTP_URL contains the name of the FTP server and pass the optional user name and password, an access path and an additional port number for the data channel. If no username or password is passed, the module trys automatically to register as "Anonymous" . The parameter FTP_ACTIV determines whether the FTP server is operated in active or passive mode. In the ACTIV mode, the FTP server tries to establish the data channel for control, however these may cause problems by security software, firewall, etc. because these could block the connection request. For this purpose, in the firewall a corresponding exception rule has to be defined. In the passive mode, this problem is alleviated since the controller establishes the connection, and can easily pass through the firewall. The control channel is always set up on port 20, and the data channel via standard PORT21, but this is in turn is depending whether active or passive mode is used, or optional PORT number in the FTP-URL is specified. With the parameter FILE_DELETE can be determined whether the source file should be deleted after successful transfer. This works on FTP and even on the control side. In specifying FTP directories the behavior depends on FTP server, whether they exist in this case or are created automatically. Normally, these should be already available. The size of files is no limit per se, but there are practical limits	Space on PLC, FTP storage and the transmission time. With dns_ip4 the IP address of the DNS server must be specified, if in the FTP URL a DNS name is given, alternatively, an IP address can be entered in the FTP URL. At parameters PLC_IP4 the own IP addresses has to be supplied. If errors occur during transmission these are passed to the output ERROR_C and ERROR_T. As long as the transfer is running, BUSY = TRUE, and after an error-free completion of the operation, DONE = TRUE. Once a new transfer is started, DONE, ERROR_T and ERROR_C are reseted.

	The module has integrated the IP_CONTROL and must not be externally linked to this, as it by default would be necessary.
Background	http://de.wikipedia.org/wiki/File_Transfer_Protocol
URL examples	
ftp	// username:password@servername:portnummer/ directory/
ftp	//username:password@servername
ftp	//username:password @ servername / directory /
ftp	//servername
ftp	//username:password@192.168.1.1/directory/
ftp	//192.168.1.1
ERROR_T	

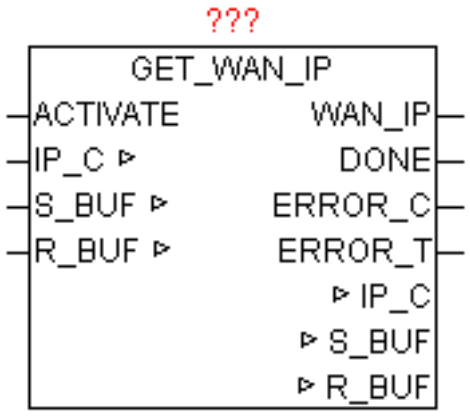


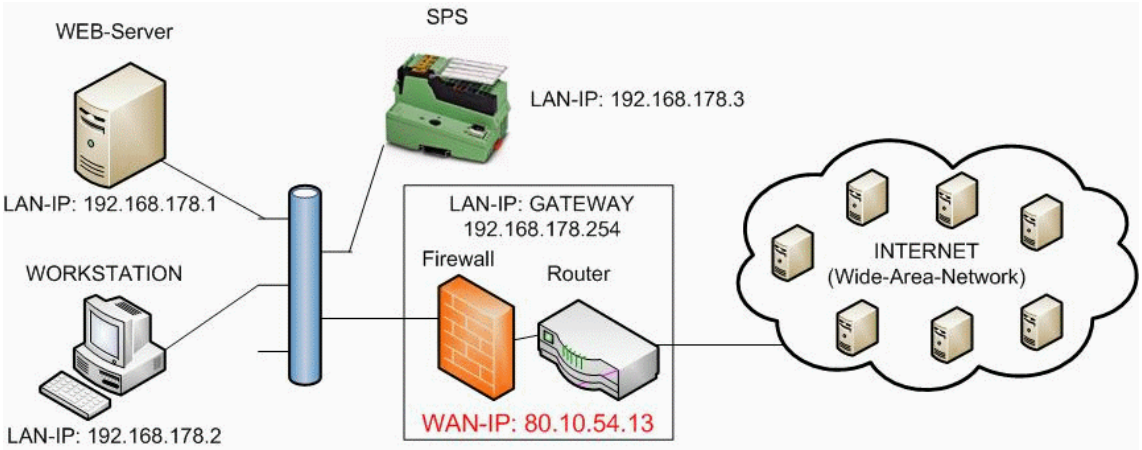
Value	Properties
1	Problem: DNS_CLIENTThe exact meaning of ERROR_C can be read at module DNS_CLIENT
2	Problem: FTP control channelThe exact meaning of ERROR_C can be read at module IP_CONTROL
3	Problem: FTP data channelThe exact meaning of ERROR_C can be read at module IP_CONTROL
4	Problem: FILE_SERVERThe exact meaning of ERROR_C can be read at block FILE_SERVER
5	Problem: END - TIMEOUTERROR_C contains the left WORD of the step number, and the right WORD has the response code received by the FTP server.The parameters must be considered first as a HEX value, divided into two WORDS, and then be considered as a decimal value.Example:ERROR_T = 5ERROR_C = 0x0028_00DCEnd-step number 0x0028 = 40Response-Code 0x00DC = 220

2.0.5 Type Function module:

INPUT ACTIVATE	BOOL (release for query)
OUTPUT	WAN_IP: DWORD (Wide Area Network address)
DONE	BOOL (Query completed without errors)
ERROR_C	DWORD (Error code)
ERROR_T	BYTE (error type)
	The module determines the IP address that the Internet router on the Wide Area Network (Internet) uses. This IP address is necessary for example to to make DynDNS declarations. With a positive edge of the ACTIVATE the request is started. After successful completion of the query DONE = TRUE, and the parameters WAN_IP the current WAN IP address displayed. If an error occurs during the query it is reported in ERROR_C in combination with ERROR_T.
ERROR_T	

IN_OUT IP_C: data structure 'IP_CONTROL ' (Parameterization)
S_BUF: data structure NETWORK_BUFFER '(transmit data)
R_BUF: data structure 'NETWORK_BUFFER '(receive data)

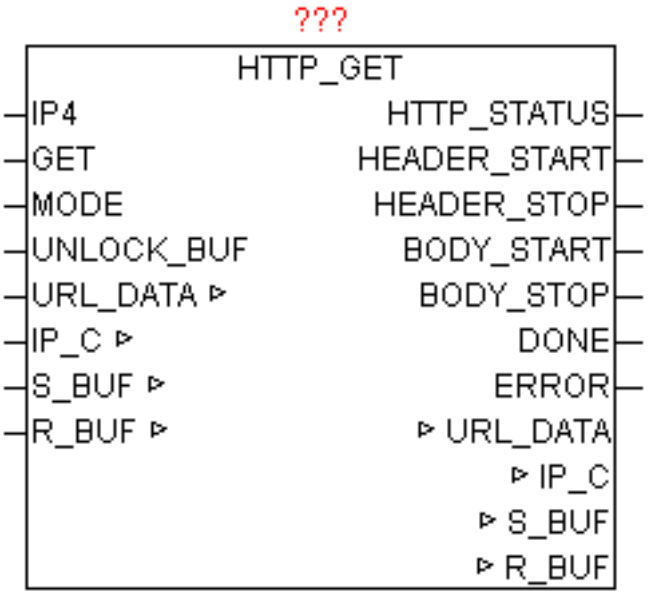




Value	Properties
1	The exact meaning of ERROR_C can be read at module DNS_CLIENT
2	The exact meaning of ERROR_C can be read at module HTTP_GET

2.0.6 HTTP_GET

Type Function module	
IN_OUT URL_DATA	URL (data STRING_TO_URL)
IP_C	IP_C (parameterization)
S_BUF	NETWORK_BUFFER (transmit data)
R_BUF	NETWORK_BUFFER (receive data)
INPUT IP4	DWORD (IP address of the HTTP server)
GET	BOOL (Starts the HTTP query)
MODE	BYTE (version of the HTTP GET query)
UNLOCK_BUF	BOOL (release of the receive data buffer)
OUTPUT HTTP_STATUS	STRING (HTTP status code)
HTTP_START	UINT (start position of the message header)
HTTP_STOP	UINT (stop position of the message header)
BODY_START	UINT (start position of the message body)
BODY_STOP	UINT (stop position of the message body)
DONE	BOOL (task performed without error)
ERROR	DWORD (error code)
	HTTP_GET does at positive edge of Get a GET-command on an HTTP server. IWith MODE the HTTP protocol version can be specified. The requested URL (web link) must be submitted completely processed in the URL_DATA structure. The full URL should therefore be processed before the module call by “STRING_TO_URL. After a successful query DONE = TRUE, and the parameters HTTP_START and HTTP_STOP point to the data area in which the message header data for further processing and evaluation are to be found. Normally, a message body is present, which in turn is transmitted via BODY_START and BODY_STOP. Also, on HTTP_STATUS is reported the HTTP status code as a string. One of the difficulties in receiving the HTTP data is the end of the data stream. This module pursued multiple strategies. In the process of the HTTP/1.0 the end of receiving data is detected by disconnection of the host. Furthermore, always the information in the header “Content-Length” is checked, and with this can be clearly recognized, that all data is received. If none of the previous versions is true, so a simple Receive Timeout Error detectes the end of data transmission. The only downside is, that this takes time. Sometimes, depending on the timeout value longer than desired. Therefore it is not bad if a reasonable timeout value is set at IP_CONTROL. ERROR gives at errors, the exact cause (See module IP_CONTROL)
The following MODE can be used	

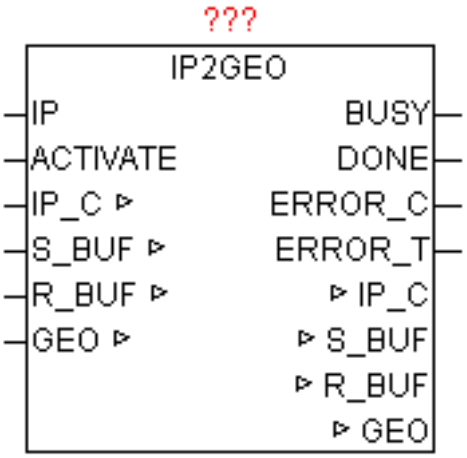


Mode	Protocol Version	Properties
0	HTTP/1.0	The host terminates automatically the TCP connection, after the transfer of data.
1	HTTP/1.0	By applying “Connection: Keep-Alive”, the host is instructed to use a persistent connection. The client should end of the connection after stopping activities.
2	HTTP/1.1	The host uses a persistent connection and must be stopped by client.
3	HTTP/1.1	By use of “Connection: Close” the host is instructed to stop transmission of data, the TCP connection.

2.0.7 IP2GEO

Type Function module	
IN_OUT IP_C	IP_C (parameterization)
S_BUF	NETWORK_BUFFER (transmit data)
R_BUF	NETWORK_BUFFER (receive data)
GEO	IP2GEO (Geographic Data)
INPUT ACTIVATE	BOOL (release for query)
OUTPUT BUSY	BOOL (Query is active)
DONE	BOOL (Query completed without errors)
ERROR_C	DWORD (Error code)
ERROR_T	BYTE (error type)
	The device supplies because of the wide-area network IP address, the geographic information of the Internet access. As the WAN IP addresses are registered worldwide, therefore can be determined the approximate geographical position of

	the PLC. Should access runs through a proxy server, so its geographic position is determined and not the PLC. Usually, but normally it is in the nearness of the PLC, and thus the deviation is not relevant. This results in calculated positions differ only a few miles from the true position, and is relatively accurate.
	If the parameter “IP” specifies no IP address, automatically the current WAN IP is used, otherwise the information of the configured IP delivered. With a positive edge of the ACTIVATE the request is started. As long as the query is not complete, BUSY = TRUE is passed. After successful completion of the query DONE = TRUE, and the parameters WAN_IP the current WAN IP address displayed. If an error occurs during the query it is reported in ERROR_C in combination with ERROR_T.
ERROR_T	
The “country_code is coded according to ISO 3166 country code ALPHA-2”. http	//www.iso.org/iso/english_country_names_and_code_elements http://de.wikipedia.org/wiki/ISO-3166-1-Kodierliste
The “REGION_CODE” is coded to “FIPS region code”. http	//en.wikipedia.org/wiki/List_of_FIPS_region_codes

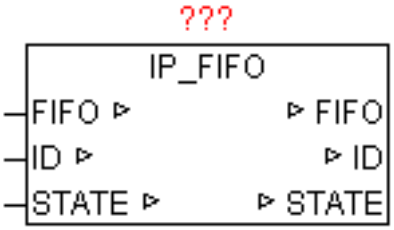


Value	Properties
1	The exact meaning of ERROR_C can be read at module DNS_CLIENT
2	The exact meaning of ERROR_C can be read at module HTTP_GET

2.0.8 IP_FIFO

Type Function module	
IN_OUT FIFO	IP_FIFO_DATA (IP-FIFO management data)
ID	BYTE (current ID assigned by IP_FIFO module)
STATE	BYTE (control commands and status messages)

	This module is used in combination with IP_CONTROL for resource management. This makes it possible that client applications request exclusive access permissions and can also give back. By the FIFO is ensured that each participant equally often gets the resource access assigned.
	In the first call of the module automatically a new unique application ID is assigned, which one the administration in FIFO is managed. The STATE parameter is changed by the application as well as from IP_FIFO module. Each application may register by default only once in the FIFO.
STATE	
Procedure	
	1. application set the STATE to 1
	2. Access permission are required as is the STATE = 2
	3. if resource is free, and access rights are present, then STATE=3
	4. If the application has the resource resp. the access needs not anymore the application sets STATE to 4. Thereafter IP_FIFO releases the resource again and set STATE to 0.
	5. Process is repeated (same or other application)



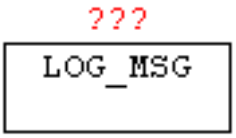
Example:
Example is found in the module IP_CONTROL!

Value	State Message
0	no action
1	Privilege request
2	Privilege request has been accepted in FIFO
3	Privilege obtained (allowing resource access)
4	Privilege remove
5	Privilege was again removed from FIFO

2.0.9 LOG_MSG

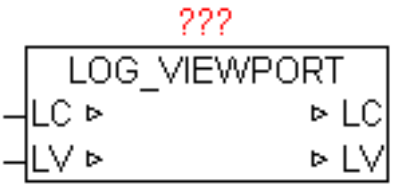
Type Function module	
IN_OUT LOG_CL	LOG_CONTROL (log-data)
	With LOG_MSG messages (STRINGS) are stored in a ring buffer. The messages can be provided with additional parameters such as the front color and back color for the output to TELNET and a filter by specifying an entry-level news. Is the level of the message larger than the default log level, the message is discarded. Furthermore,with Enable the logging will be disabled in general. Thus, it is not a problem to archive many messages

	per PLC cycle. The message buffer can be passed to a telnet client with the module TELNET_LOG. Details on the interface are shown in the table below.
	If messages are applied from various module instances to the same LOG_BUFFER, then the “LOG_CL” data structure has to be created Global.



2.0.10 LOG_VIEWPORT

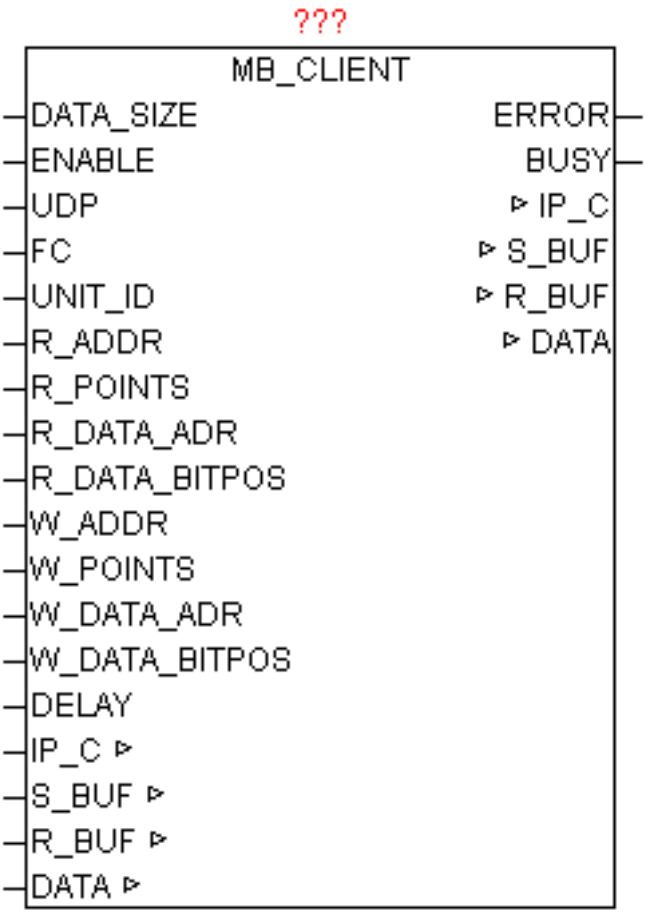
Type	Function module
IN_OUT LC	LOG_CONTROL
LV	us_LOG_VIEWPORT
The module LOG_VIEWPORT is used to index a list of LOG_CONTROL messages, which are currently in the virtual view. To move around within the message list (scroll), a scroll offset can be specified by LV.MOVE_TO_X. A positive value scroll in direction of newer reports and a negative value in the direction of the earlier messages. The number of lines in the message list of the virtual view is given by LV.COUNT. The current messages in the virtual view are stored in LV.LINE_ARRAY [x], and are available for further processing. A change in the message list is always announced with LV.UPDATE	= TRUE, and the user has to reset.
	The following LV.MOVE_TO_X values produce a special behavior.
	+30000 = display previous Messages (beginning of the ring buffer)
	+30001 = display latest messages (end of the ring buffer)
	+30002 = one full page in direction of recent messages.
	+30003 = One full page in direction of older messages



2.0.11 MB_CLIENT (OPEN MODBUS)

Type Function module	
IN_OUT IP_C	IP_C (parameterization)
S_BUF	NETWORK_BUFFER_SHORT (transmit data)
R_BUF	NETWORK_BUFFER_SHORT (receive data)
DATA	ARRAY [0..255] OF WORD (MODBUS Register)
INPUT DATA_SIZE	INT (number of data words in structure MB_DATA)
ENABLE	BOOL (release)
UDP	BOOL (Prefix TCP / UDP, UDP = TRUE)
FC	INT (function number)
UNIT_ID	BYTE (Device ID)
R_ADDR	INT (Read command: MODBUS data point address)
R_POINTS	INT (Read command: MODBUS number of data points)
R_DATA_ADR	INT (Read command: DATA data point address)
R_DATA_BITPOS	INT (read command: DATA data point bitpos.)
W_ADDR	INT (Read command: MODBUS data point address)
W_POINTS	INT (Read command: MODBUS number of data points)
W_DATA_ADR	INT (Read command: DATA data point address)
W_DATA_BITPOS	INT (read command: DATA data point bit pos.)
DELAY	TIME (repetition time)
OUTPUT ERROR	DWORD (error code)
BUSY	BOOL (module is active)
	The module provides access to Ethernet devices, the MODBUS TCP or MODBUS UDP supported, or MODBUS RS232/485 devices are connected via Ethernet Modbus gateway. There commands from Classes 0,1,2 are supported. The parameters IP_C, S_BUF, R_BUF this form the interface to the module IP_CONTROL and used here for processing and coordination. The desired IP address and port number (for MODBUS default is 502) must be specified on IP_CONTROL centrally. The DATA structure is designed as a WORD array and contains the MODBUS data for reading and writing. The size of the WORD_ARRAY is given by DATA_SIZE. By ENABLE, the module is released, and by remove of the release a possibly still active query is ended. For devices that support MODBUS with UDP = TRUE this mode can be activated. The

	parameter UNIT_ID must only at use of Ethernet Modbus provided. The desired function is specified by FC (see function code table). Depending on the function, the R_xxx and W_xxx parameters has to be supplied with data. By specifying the DELAY the repetition time can be specified. If not specify the time an attempt is made as often as possible to execute the command. The integrated access management automatically guarantees to get the other module instances also to the series. A negative command execution is reported by ERROR (see ERROR-table). If the module actively performs a query, then BUSY = TRUE will be passed during this time.
Supported function codes and parameters used	
ERROR	



Function Code	Bit Access	16 Bit Access (Register)	Group	Function Description	R_ADDR	R_POINTS	R_DATA_ADR	R_DATA_BITPOS	W_ADDR	W_POINTS	W_DATA_ADR	W_DATA_BITPOS
1	x		Coils	Read Coils	x	x	x	x				

2	x		Input Discrete	Read Discrete Inputs	x	x	x	x				
3		x	Holding Register	Read Holding Registers	x	x	x					
4		x	Input Register	Read Input Register	x		x					
5	x		Coils	Write Single Coil					x		x	x
6		x	Holding Register	Write Single Register					x		x	
15	x		Coils	Write Multiple Coils					x	x	x	x
16		x	Holding Register	Write Multiple Register					x	x	x	
22		x	Holding Register	Mask Write Register					x		x	
23		x	Holding Register	Read/Write Multiple Register	x	x	x		x	x	x	

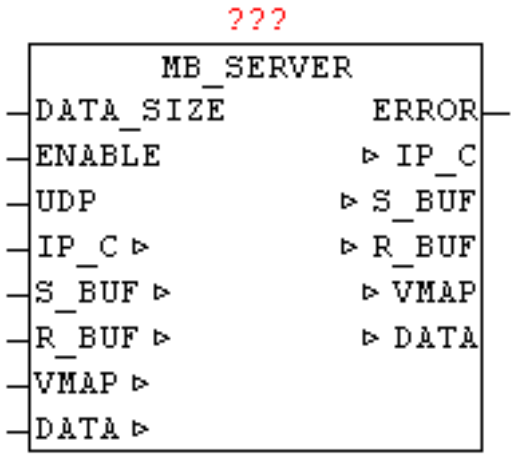
Value	Source	Description
B3	B2	B1
nn	nn	nn
xx	xx	xx
xx	xx	xx
xx	xx	xx
xx	xx	xx

xx	xx	xx
xx	xx	xx
xx	xx	xx
xx	xx	xx
xx	xx	xx
xx	xx	xx

2.0.12 MB_SERVER (OPEN-MODBUS)

Type Function module	
IN_OUT IP_C	IP_C (parameterization)
S_BUF	NETWORK_BUFFER_SHORT (transmit data)
R_BUF	NETWORK_BUFFER_SHORT (receive data)
VMAP	ARRAY [1..10] OF VMAP_DATA (virtual address table)
DATA	ARRAY [0..255] OF WORD (MODBUS Register)
INPUT DATA_SIZE	INT (number of data words in DATA)
ENABLE	BOOL (release)
UDP	BOOL (Prefix TCP / UDP, UDP = TRUE)
OUTPUT ERROR	DWORD (error code)
	The module provides access from external to local MODBUS data tables via Ethernet. It supports commands in categories 0,1,2. The parameters IP_C, S_BUF, R_BUF this form the interface to the module IP_CONTROL and used here for processing and coordination. The desired port number (for MODBUS default is 502) must be specified on IP_CONTROL centrally. The IP address is not required on IP_CONTROL, as this one operates in the PASSIVE mode. The DATA structure is designed as a WORD array and contains the MODBUS data. DATA_SIZE specifies the size of DATA . By ENABLE, the module is released, and by remove of the release a possibly still active query is ended. For devices that support MODBUS with UDP = TRUE this mode can be activated. A negative command execution is reported by ERROR (see ERROR table).
	WIth entries in the data structure VMAP, virtual data areas are created, and the access to certain function codes and data regions is parameterized. Thus, it is very easy to map virtual address spaces into a coherent Data block (DATA), or write data areas. Or provide areas, that are connected to output peripherals, with a watchdog.

	The handling of the VMAP data is described in more detail in the module MB_VMAP.
ERROR	
Supported function codes and parameters used	



Value	Source	Description
B3	B2	B1
nn	nn	nn
xx	xx	xx
xx	xx	xx
xx	xx	xx
xx	xx	xx

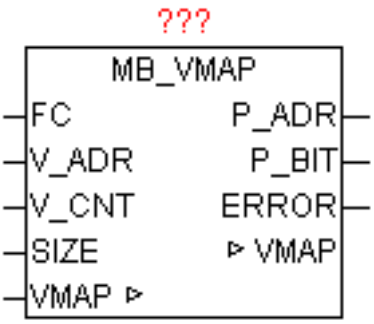
Function Code	Bit Access	16 Bit Access (Register)	Group	Function Description
1	x		Coils	Read Coils
2	x		Input Discrete	Read Discrete Inputs
3		x	Holding Register	Read Holding Registers
4		x	Input Register	Read Input Register
5	x		Coils	Write Single Coil
6		x	Holding Register	Write Single Register
15	x		Coils	Write Multiple Coils
16		x	Holding Register	Write Multiple Register
22		x	Holding Register	Mask Write Register
23		x	Holding Register	Read/Write Multiple Register

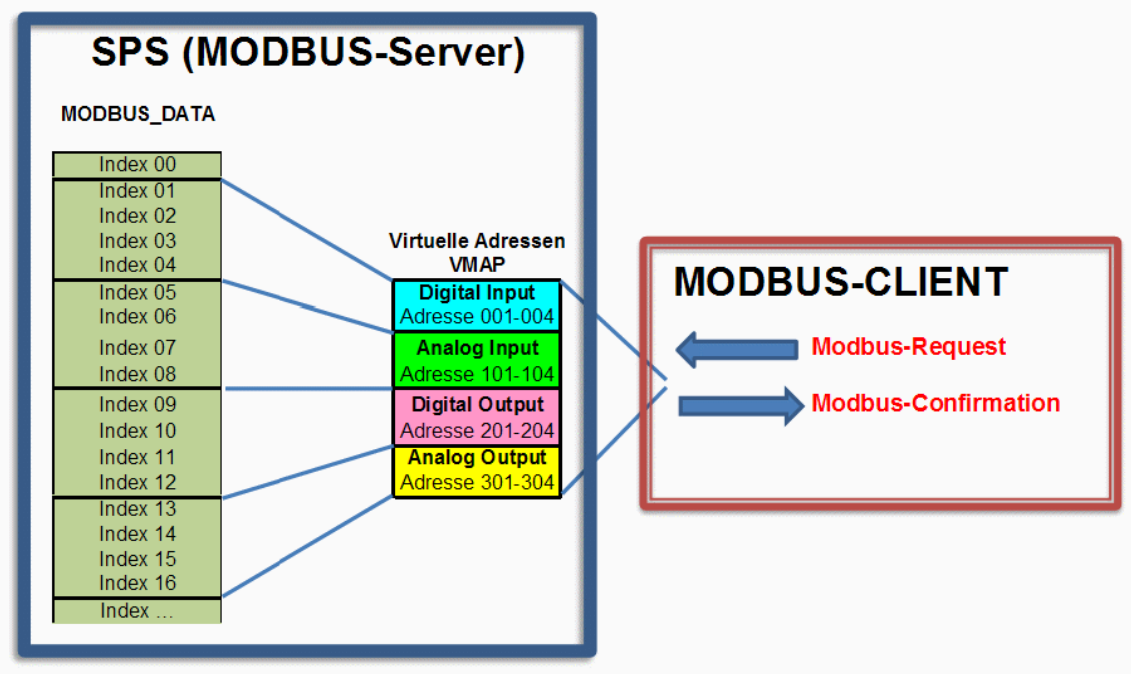
2.0.13 MB_VMAP

Type Function module	

IN_OUT VMAP	ARRAY [1..10] OF VMAP_DATA (VIRTUAL_MAP Data)
INPUT FC	INT (function number)
V_ADR	INT (virtual address range start address)
V_CNT	INT (Virtual address space: number of data points)
SIZE	INT (number of MODBUS registers in structure DATA)
OUTPUT	P_ADR: INT (Real address space: Start address)
P_BIT	INT (real address range: bit position)
ERROR	DWORD (error code)
	The module allows the conversion of virtual addresses at a real address space in the MODBUS DATA Structure. Virtual address ranges are defined in the VMAP data array. If the module is called and found that nothing in the VMAP data is entered, automatically a block is created, allowing full access to all the MODBUS data. In each address block also a watchdog timer is maintained that sets each time you access this block on the timer to zero. Thus, simply by comparing the TIME_OUT value to a cutoff value, at communication error (no update) can be responded.
	By the parameter FC is detected the functional code and whether the register (16 bit) or individual bits must be processed. The bit number corresponds to the function code. This means that Bit5 = 1 in FC the function code 5 (Write Single Coil) enables. By V_ADR by the virtual start address is specified (At 16bit commands this is a register address and at bit commands an absolute bit number within a defined block.) The parameter V_CNT defines the number of data points (unit 16-bit or bits depending on the function code). The overall size is given by MODBUS_ARRAY SIZE (number WORDS). By using these parameters, the module searched the VMAP data table for a matching block of data, and passes from the correct data block P_ADR as a result. The value corresponds to the real index for MODBUS_DATA array. At a function code with bit access in addition the bit position within P_ADR is passed as well. A potential error occurring in the analysis is reported for the parameter “error” (see error table). The watchdog timer is reseted at each access to a function code from the group of write commands.
If no special treatment required, so in VMAP are not settings required, and then MODBUS_ARRAY is mapped 1	1 with the access.
ERROR	
	! Note the special treatment of function code 23!

	The Modbus Function Code 23 is a combined command, because it consists of two actions. First register are written and then the register are read. Found that the write or read parameter is not allowed, so neither of these actions is performed.
	To distinguish between reading and writing by VMAP, the read command is checked in VMAP at FC 23 as BIT23 (Read/Write Multiple registers), and the write command on the other hand, is tested in Bit16 (Write multiple registers).





Example:

Example Configuration
(* Virtual block 1 *)
VMAP[1].FC := DWORD#2#00000000_10000000_00000000_00011100); (FC 2,3,4,23)
VMAP[1].V_ADR := 1; (* Virtual Address Range: Start address *)
VMAP[1].V_SIZE := 4; (* Virtual address space: number of WORD *)
VMAP[1].P_ADR := 1; (* Real address space: Start address *)
(* Virtual Block 2 *)
VMAP[2].FC := DWORD#2#00000000_10000000_00000000_00011000); (FC 3,4,23)
VMAP [2] V_ADR. = 101; (* Virtual Address Range: Start address *)
VMAP[2].V_SIZE := 4; (* Virtual address space: number of WORD *)
VMAP[2].P_ADR := 5; (* Real address space: Start address *)
(* Virtual Block 3 *)
VMAP[3].FC := DWORD#2#00000000_11000001_10000000_01111010);(FC1,3-6,15-16,23)

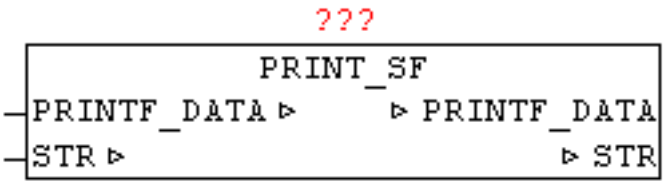
VMAP [3] V_ADR. = 201; (* Virtual Address Range: Start address *)
VMAP[3].V_SIZE := 4; (* Virtual address space: number of WORD *)
VMAP[3].P_ADR := 9; (* Real address space: Start address *)
(* Virtual Block 4 *)
VMAP[4].FC := DWORD#2#00000000_11000001_00000000_01011000); (FC 3,4,6,16,23)
VMAP [4] V_ADR. = 301; (* Virtual Address Range: Start address *)
VMAP[4].V_SIZE := 4; (* Virtual address space: number of WORD *)
VMAP[4].P_ADR := 12; (* Real address space: home address *)
The configuration is following access matrix:

Value	Description
0	No error
1	Invalid function code
2	Invalid Data Address

Function Description	Function Code	Bit Access	16 Bit Access (Register)	Read / Write	Digital Input	Analog Input	Digital Output	Analog Output
Read Coils	1	x		Read			x	
Read Discrete Inputs	2	x		Read	x			
Read Holding Registers	3		x	Read	x	x	x	x
Read Input Register	4		x	Read	x	x	x	x
Write Single Coil	5	x		Schreiben			x	
Write Single Register	6		x	Schreiben			x	x
Write Multiple Coils	15	x		Schreiben			x	
Write Multiple Register	16		x	Schreiben			x	x
Mask Write Register	22		x	Schreiben				
Read/ Write Multiple Register	23		x	Read / Write	x	x	x	x

2.0.14 PRINT_SF

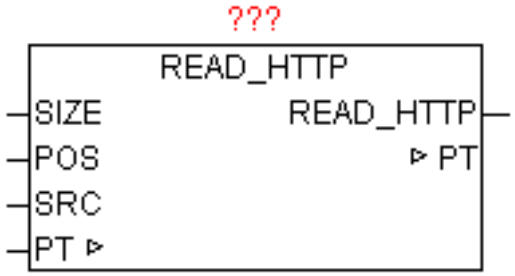
Type Function module		
IN_OUT PRINTF_DATA	ARRAY[1..11] OF STRING(string_length)	(Parameter data)
STR	STRING (String_length) (String result)	
	With PRINT_SF a STRING can be added dynamically with a part of a string. The position of the substring to be inserted is indicated by ‘ ‘ tilde character and the subsequent number defines the parameter number. ‘ 1’ to ‘ 9’ are thus processed automatically. If the insert of the substring reached the maximum number of characters, so instead of the substring ‘..’ is inserted.	
	VAR	
LITER	REAL := 545.4;	
FUELLZEIT	INT := 25;	
NAME	STRING: = ‘tank content’;	
PARA	ARRAY[1..11] OF STRING(string_length);	
PS	PRINT_SF;	
	END_VAR	
PARA[1]	= REAL_TO_STRING(liters); (* Parameter 1: string to convert *)	
PARA[2]	= INT_TO_STRING(filling time); (* Parameter 2: string to convert *)	
PARA[3]	= NAME; (* Parameter 3: *)	
PS.STR	= ’ 3: 1 Liter, filling time: 2 Min.’ ; (* Text output-mask *)	
PS.PRINTF_DATA	= PARA; (* Pass parameter data structure *)	
	PS(); (* Module version *)	
	The string PS. STR then has the following content	
’Tank Capacity	545.4 liters, filling time: 25 min’	



2.0.15 READ_HTTP

Type Function module		
INPUT SIZE	UINT	(Buffer size)

POS	INT	(position as of that the search is started)
SRC	STRING	(Search string)
IN_OUT PT	POINTER	(Address of the buffer)
OUTPUT	VALUE (Parameters of the header information)	
	After a successful HTTP-GET Request always a HTTP header (message header) and a message body (message body) is available in the buffer. In the HTTP header various information about the requested HTTP page is stored. The following message body contains the actual requested data. With READ_HTTP the HTTP header information can be analyzed. The module searches any array of bytes on the contents of a string and then evaluates the following parameters, and returns that string as its result. The data in the buffer are automatically converted to upper case, so all search string at SRC has to be too, given in capital letters. With POS it can begin its search at any position. The first element in the array is at position number 1	



Example:

Example of an HTTP response (header information):
HTTP/1.0 200 OK
Content-Length: 2165
Content-Type: text/html
Date: Mon, 15 Sep 2008 16:59:08 GMT
Last-Modified: Wed, 18 Jun 2008 12:35:52 GMT
Mime-Version: 1.0

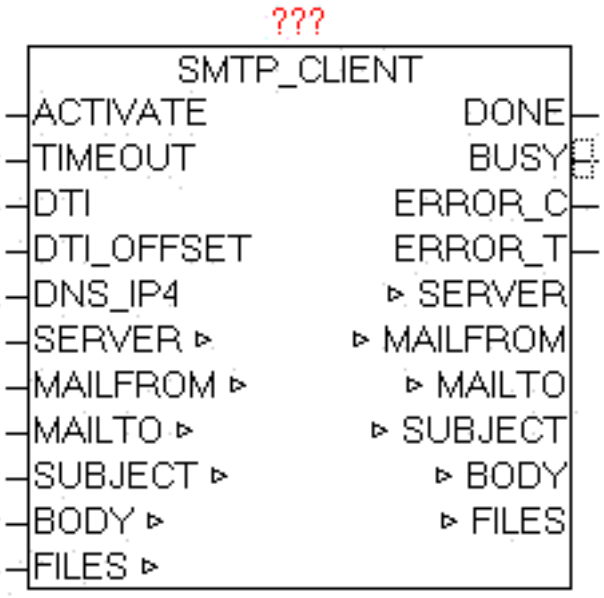
If SRC does not include a search term, automatically the HTTP version and the HTTP status code in the buffer is searched and evaluated. As a result, according to the above example “1.0 200 OK” is returned. If SRC is in a search term, this header information is searched in the buffer and the value as a string eg ‘Content-Length’ = “2165” is returned.

2.0.16 SMTP_CLIENT

Type Function module	

IN_OUT SERVER	STRING (URL of the SMTP server)
MAIL FROM	STRING (return address)
MAILTO	STRING (string_length) (recipient address)
SUBJECT	STRING (subject text)
SUBJECT	STRING (subject text)
FILES	STRING (string_length) (attached files)
INPUT ACTIVATE	BOOL (positive edge starts the query)
TIMEOUT	TIME (time)
DTI	DT (current date-time)
DTI_OFFSET	INT (time zone offset from UTC)
Dns_ip4	DWORD (IP4 address of the DNS server)
OUTPUT DONE	BOOL (Transfer completed without error)
BUSY	BOOL (Transfer active)
ERROR_C	DWORD (Error code)
ERROR_T	BYTE (Problem type)
	The module SMTP_CLIENT is used to send of classic emails.
Following features are supported	
	SMTP protocol
	Extended SMTP protocol
	Sending the subject line, text and content
Indication of email sender address (From	), including “Display Name”
Indication of the recipient (s) (To	)
Indication of carbon copy recipient (s) (Cc	)
Indication of blind copy recipient (s) (bc	)
	Sending file (s) as an attachment
Authentication method	NO, PLAIN, LOGIN, CRAM-MD5
	Specifying the port number
	When positive edge at ACTIVATE the transfer process is started. The SERVER parameter contains the name of the SMTP server and optionally the user name and password and a port number. If you pass a user name and password, the procedure is according to standard SMTP.
SERVER	URL Examples:
username	password@smtp_server
username	password@smtp_server:portnumber
	smtp_server
Special case	

	If in the username is a '@' included this must be passed as '%' - character, and is then automatically corrected by the module again.
By specifying user and password the Extend-SMTP is used, and automatically the safest possible Authentication method is used. If parameter is to specify the MAIL FROM sender address	
	i.e. oscat@gmx.net
	Optionally, an additional "Display Name" be added This is displayed the email client automatically instead of the real return address. Therefore, always an easily recognizable name to be used.
	i.e.. oscat@gmx.net;Station_01
	The email client shows as the sender then "Station_01". Thus, more people will use the same email address but send a own "Alias".
	At the MAILTO parameter can To, Cc, Bc be specified. The different groups of recipients are specified by '#' as the separator in a list. Multiple addresses within the same group are divided with the separator ";" . In each group can be defined unlimited count of recipients, the only limitation is the length of the mailto string.
	To;To..#Cc;Cc...#Bc;Bc...



Example:

Examples.
o1@gmx.net;o2@gmx.net#o1@gmx.net#o2@gmx.net
defines two TO-addresses, one CC-address and a Bc-address
##o2@gmx.net
defines only one BC-address.

With subject, a subject text will be specified, as well as with BODY an email text content. The current Date / Time value must be definded at DTI, and at DTI_OFFSET the correction value as an offset in minutes from UTC (Universal Time). If the DTI in UTC time is passed, at DTI_OFFSET a 0 must be passed.

It can be sent files as attachment. The files must be passed in list form for parameter FILES. Any number of files are given, only limitation is the length of the file-strings, and the space of the e-mailbox (in practice 50-30 megabytes).

By an additional optional information of '#DEL# deleting the files can be triggered on the controller after the successful transfer of files via email.

eg

FILES: 'log1.csv ; log2.csv ; #DEL# '

The two files are deleted after successful transfer.

The monitoring time can be specified with parameter TIMEOUT. At dns_ip4 must be specified the IP address of the DNS server, if in SERVER a DNS name is specified. If errors occur during the transmission, they are passed at ERROR_C and ERROR_T. As long as the transfer is running, BUSY = TRUE, and after an error-free completion of the operation, DONE = TRUE. Once a new transfer is started, DONE, ERROR_T and ERROR_C are reseted.

The module has integrated the IP_CONTROL and must not be externally linked to this, as it by default would be necessary.

Basics:

http://de.wikipedia.org/wiki/SMTP-Auth

http://de.wikipedia.org/wiki/Simple_Mail_Transfer_Protocol

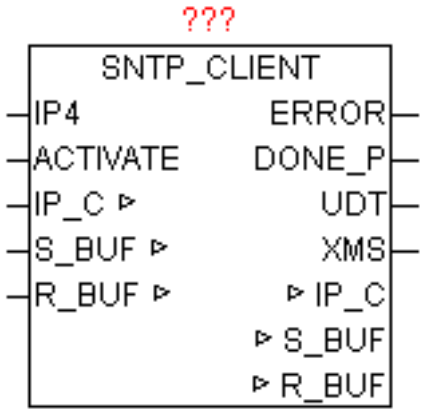
ERROR_T:

Value	Properties
1	Problem: DNS_CLIENTThe exact meaning of ERROR_C can be read at module DNS_CLIENT
2	Problem: SMTP ChannelThe exact meaning of ERROR_C can be read at module IP_CONTROL
4	Problem: FILE_SERVERThe exact meaning of ERROR_C can be read at block FILE_SERVER
5	Problem: END - TIMEOUTERROR_C contains the left WORD the end of the step number, and in the right WORD the last response code received by the SMTP server.The parameters must be considered first as a HEX value, divided into two WORDS, and then be considered as a decimal value.Example:ERROR_T = 5ERROR_C = 0x0028_00FAEnd-step number 0x0028 = 40Response-Code 0x00DC = 250

2.0.17 SNTP_CLIENT

Type Function module	
IN_OUT IP_C	IP_C (parameterization)
S_BUF	NETWORK_BUFFER (transmit data)
R_BUF	NETWORK_BUFFER (receive data)
INPUT IP4	DWORD (IP address of the SNTP server)
ACTIVATE	BOOL (Starts the query)
OUTPUT ERROR	DWORD (error code)
DONE_P	BOOL (positive edge finish without error)
UDT	DT (Date and time output as Universal Time)
XMS	INT (millisecond of Universal Time UDT)
	The SNTP_CLIENT is used to synchronize local time with an SNTP server. For this, the Simple Network Time Protocol is used which is designed to provide a reliable time information over networks with variable packet delay. The SNTP is technically completely identical with NTP, which here

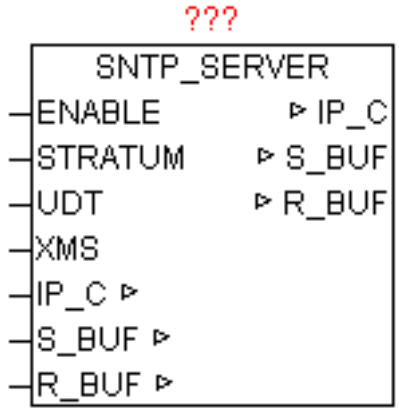
	means no differences. Therefore, all known SNTP and NTP server can be used, whether it be on the local network or via the Internet. For IP4 a IP-address of a SNTP / NTP server is specified. A positive edge at ACTIVATE starts the query. The elapsed time between sending and receiving of the time is measured and a time correction is calculated. Then, the received time will be corrected by this value. Upon successful completion DONE_P is one positive edge, and the current time is passed at UDT. On XMS the associated fractional seconds as milliseconds are passed. The values of UDT and XMS are only valid when DONE_P = TRUE, since this is a static time value, and is only used for setting of pulse-controlled time. ERROR gives at error the exact cause (See block IP_CONTROL).

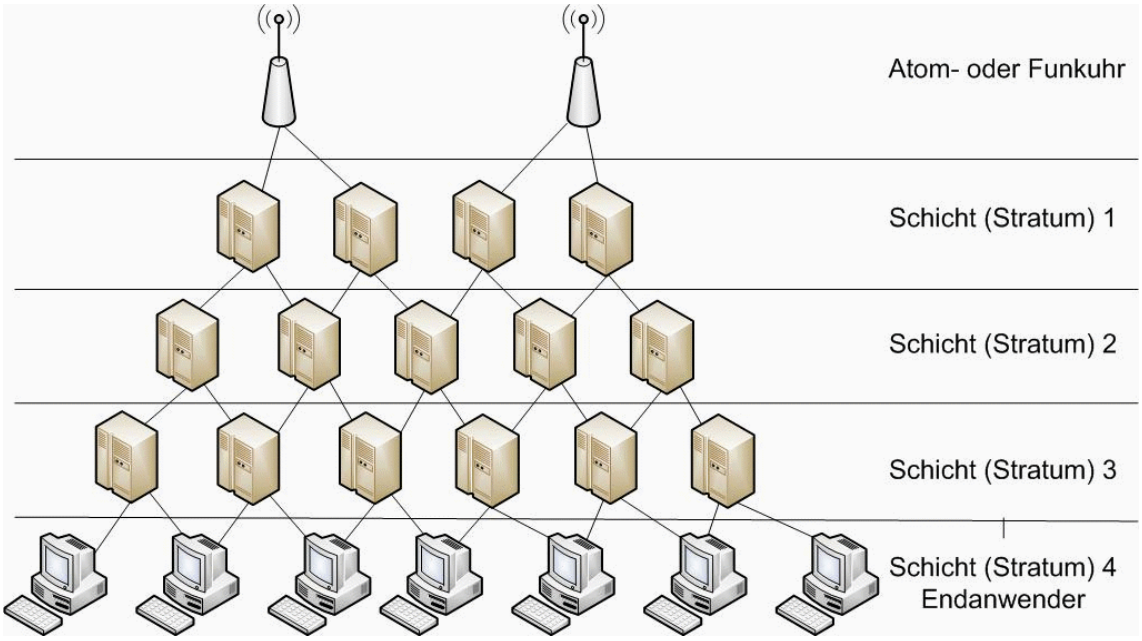


2.0.18 SNTP_SERVER

Type Function module		
IN_OUT IP_C	IP_C (parameterization)	
S_BUF	NETWORK_BUFFER (transmit data)	
R_BUF	NETWORK_BUFFER (receive data)	
INPUT ENABLE	BOOL (Starts SNTP server)	
STRATUM	BYTE (specify the hierarchical level or	accuracy)
UDT	DT (Date and time input as Universal Time)	
XMS	INT (millisecond of Universal Time UDT)	
	The module provides the functionality of an SNTP (NTP) server. With ENABLE = TRUE the module logs in at IP_CONTROL and waits for the release of the resource, if it occupied by other subscribers for now. Then the module is waiting for requests from other SNTP clients and answers it with the current time of UDT and XMS. As long as ENABLE = TRUE, the Ethernet access of this resource is permanently locked for other users (Exclusive Access - due to passive UDP mode). SNTP uses a hierarchical system of different strata. As stratum 0 is defined as the exact time standard. The directly coupled systems, such as NTP, GPS or DCF77 time signals are called Stratum 1.Each additional dependent unit causes an additional time lag of 10-100ms and is designate with a higher number (Stratum 2, Stratum 3 to 255). If no STRATUM is specified at the module, STRATUM = 1 is used as a standard.	

	If an SNTP client itself has a time with a higher stratum than an SNTP server, the time of this is sometimes rejected because it is less accurate than their own reference. It is therefore important to specify a logically correct STRATUM. The module SNTP_CLIENT ignores deliberately the STRATUM and synchronizes in each case with the SNTP server, because pretty much everyone SNTP server as a more precise time than a PLC.	

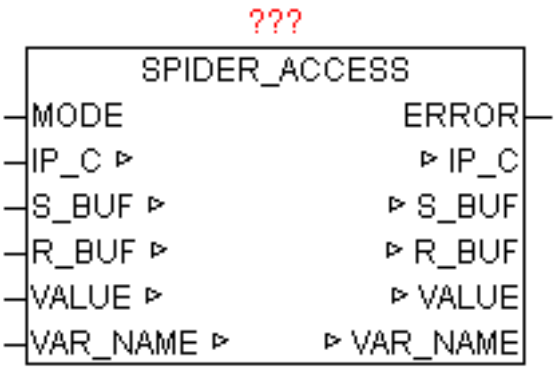




2.0.19 SPIDER_ACCESS

Type Function module	
IN_OUT IP_C	IP_C (parameterization)
S_BUF	NETWORK_BUFFER (transmit data)
R_BUF	NETWORK_BUFFER (receive data)
VALUE	STRING (Value of the variable)
NAME	STRING(40) (Variable Name)
INPUT MODE	BYTE (operating mode: 1 = read / write = 2)

ERROR ERROR	DWORD (Error code)
ERROR	
With SPIDER_ACCESS variables can be read and written from the PLC, which are provided by visualizations of web servers based on “spider control” from the company iniNet integrated Solution GmbH. For the following PLC is this web server integration available	
	Simatic S7 200/300/400, SAIA-Burgess PCD, Wago (750-841), Beckhoff (CX series), Phoenix Contact (ILC Reihe), Selectron, Berthel, Tbox, Beck IPC
	In the PLC program of target PLC, the desired variables must be released for web access. Since the communication is performed via HTTP (port 80), the data exchange is no problem, even across firewalls. Global and instance variables can be processed.
Format of variables	
	At global variables, only the regular variable names has to be given. An instance variable must be specified below: “instance name. variable name”
Mode	Read
	If the MODE parameter is set to “1” and the variable name is quoted in “NAME”, so cyclically a request to the HTTP to Web Server (PLC) is performed and the result is displayed the “VLAUE” as a string.
Mode	Write
	If the parameter MODE is set to “2” and at “VALUE” the variable value and in “NAME” the variable name as string, then cyclically an HTTP request to the Web Server (PLC) is performed
	The mode resp. the variable name can be changed in the cyclic mode at any time. If several variables have to be processed, thus only a many module instances as needed must be called.

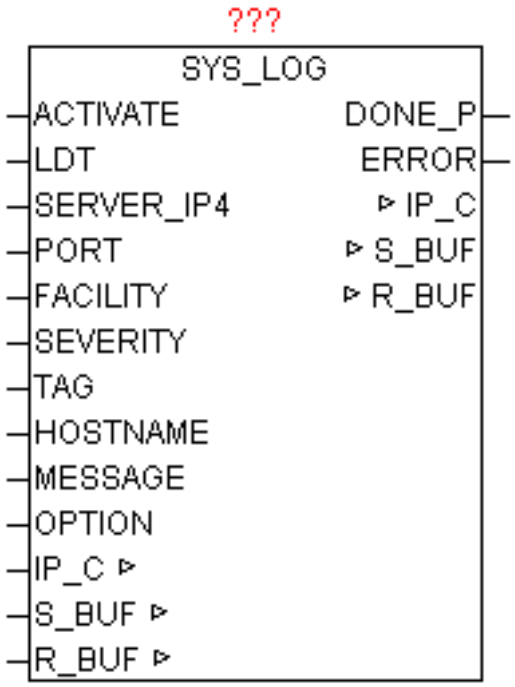


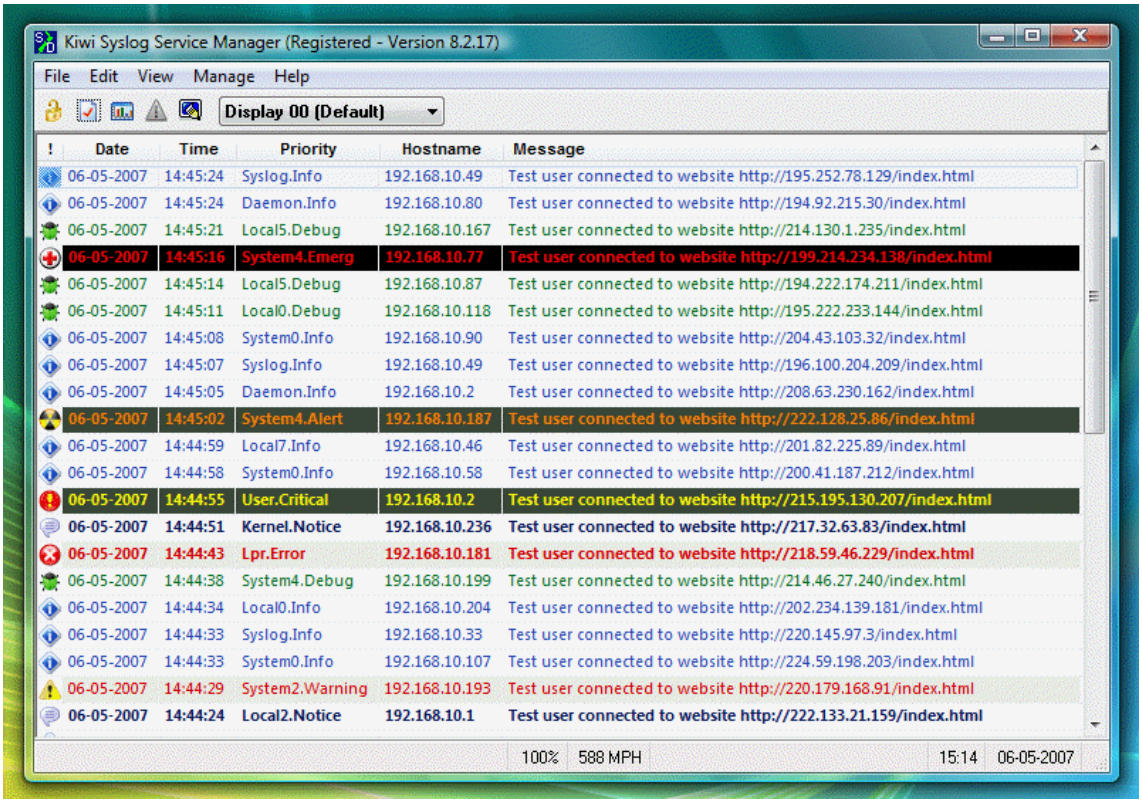
Value	Properties

1	At writing variable values an error occured.
> 1	The exact meaning of ERROR is read at module HTTP_GET

2.0.20 SYS_LOG

Type Function module	
IN_OUT IP_C	IP_C (parameterization)
S_BUF	NETWORK_BUFFER (transmit data)
R_BUF	NETWORK_BUFFER (receive data)
INPUT ACTIVATE	BOOL (positive edge starts the query)
LDT	DT (local time)
SERVER_IP4	DWORD (IP address of the syslog server)
PORT	WORD (Port number of the syslog server)
FACILITY	BYTE (specifies the service or component)
SEVERITY	BYTE (Classification of severity)
TAG	STRING(32) (Process name, ID, etc.)
HOST NAME	STRING (Name or IP address of the sender)
MESSAGE	STRING(string_length) (Message)
	OPTION BYTE (Various)
OUTPUT DONE	BOOL (Query completed without errors)
ERROR	DWORD (Error code)
	SYSLOG is a standard for transmitting messages in an IP computer network. The protocol is very simple - the client sends a short text message to the syslog receiver. The receiver is also called “syslog daemon” or “syslog server”. The messages are sent using UDP port 514 or TCP port 1468 and includes the message in plain text. SYSLOG is typical used for computer systems management and security surveillance. This enables the easy integration of various log sources to a central syslog server. The server software is available for all platforms, sometimes known as free / shareware. Unix or Linux systems have a syslog server already integrated. Through a positive edge at ACTIVATE from the parameters of LDT, FACILITY, SEVERITY, TAG, HOST NAME, MESSAGE a syslog message is generated and sent to the SERVER_IP4 mail address. With OPTION various properties can still be controlled (See Table OPTION). After successfully sending DONE gets TRUE, otherwise ERROR is issued when the actual error message (See ERROR of module IP_CONTROL).
	FACILITY,SEVERITY,TIMESTAMP,HOSTNAME,TAG,MESSAGE





Example:

Example:

MAIL.ERR: Sep 10 08:31:10 149.100.100.02 PLANT2_PLC1 This is a test message generated by OSCAT SYSLOG

The following options can be used

Severity is defined as the following standard:

The following facility is defined as standard:

For general syslog messages, the facility values 16-23 are provided (local0 to local7). But it is quite permissible to use the predefined values from 0 to 15 for own purposes.

With Facility and Severity can be filtered on the SYSLOG server (database) according to certain reports, such as: “Record all error messages from the mail server with severity level.

Example (screenshot) of a syslog server for Windows

BIT	Function
0	FALSE = with Facility, Severity codeTRUE = No Facility, Severity code
1	FALSE = with RFC headerTRUE = without RFC Header (only the MESSAGE alone sent)
2	FALSE = with CR,LF at endTRUE = without CR,LF end
3	FALSE = UDP ModusTRUE = TCP Modus

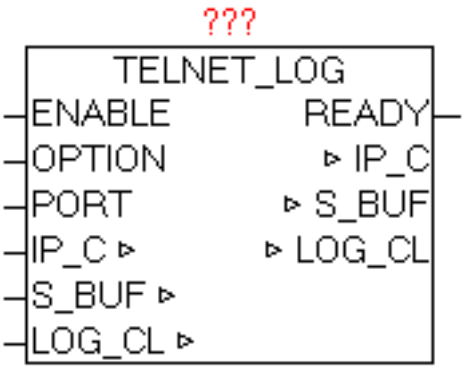
Severity	Description
0	Emergency
1	Alert
2	Critical
3	Error
4	Warning
5	Notice
6	Informational
7	Debug

Facility	Description
00	Kernel message
01	user-level messages
02	mail system
03	system daemons
04	security/authorization messages
05	messages generated internally by syslogd
06	line printer subsystem
07	network news subsystem
08	UUCP subsystem
09	clock daemon
10	security/authorization messages
11	FTP daemon
12	NTP subsystem
13	log audit
14	log alert
15	clock daemon
16	local10
17	local11
18	local12

19	local13
20	local14
21	local15
22	local16
23	local17

2.0.21 TELNET_LOG

Type Function module	
IN_OUT IP_C	IP_C (parameterization)
S_BUF	NETWORK_BUFFER (transmit data)
LOG_CL	LOG_CONTROL (log-data)
INPUT S_BUF_SIZE	UINT (Size of S_BUF)
ENABLE	BOOL (TELNET server released)
OPTION	BYTE (Send Options)
PORT	WORD (Port Nummer)
OUTPUT READY	BOOL (TELNET client has established connection))
	TELNET_LOG is used to pass all the messages in the ring LOG_CONTROL-buffer over TELNET. By “ENABLE”, the module can be activated. At parameter PORT the desired port number can be defined. If the parameter is not defined the default port is 23.
With OPTION various properties can still be controlled (See Table OPTION). If the parameter OPTION is not connected the following default is assumed	
	OPTION = BYTE#2#1000_1100;
	As soon as a Telnet client connects this is indicated by parameter “READY”. Then be automatically all messages are passed to TELNET. Once occurred new reports in the course in LOG_CONTROL they are always passed automatically. When a new connection from/to rebuilds, all messages will be passed again. Most TELNET clients offer the opportunity to redirect the data stream to a file, just to make a long-term data archiving.
OPTION	

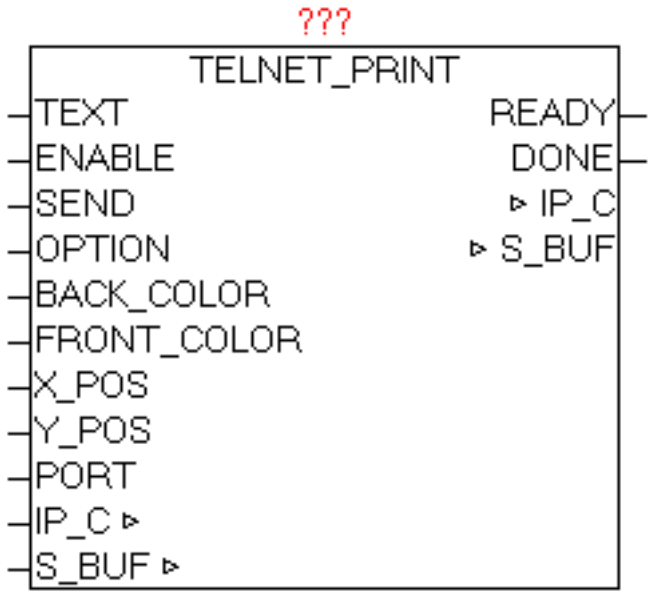


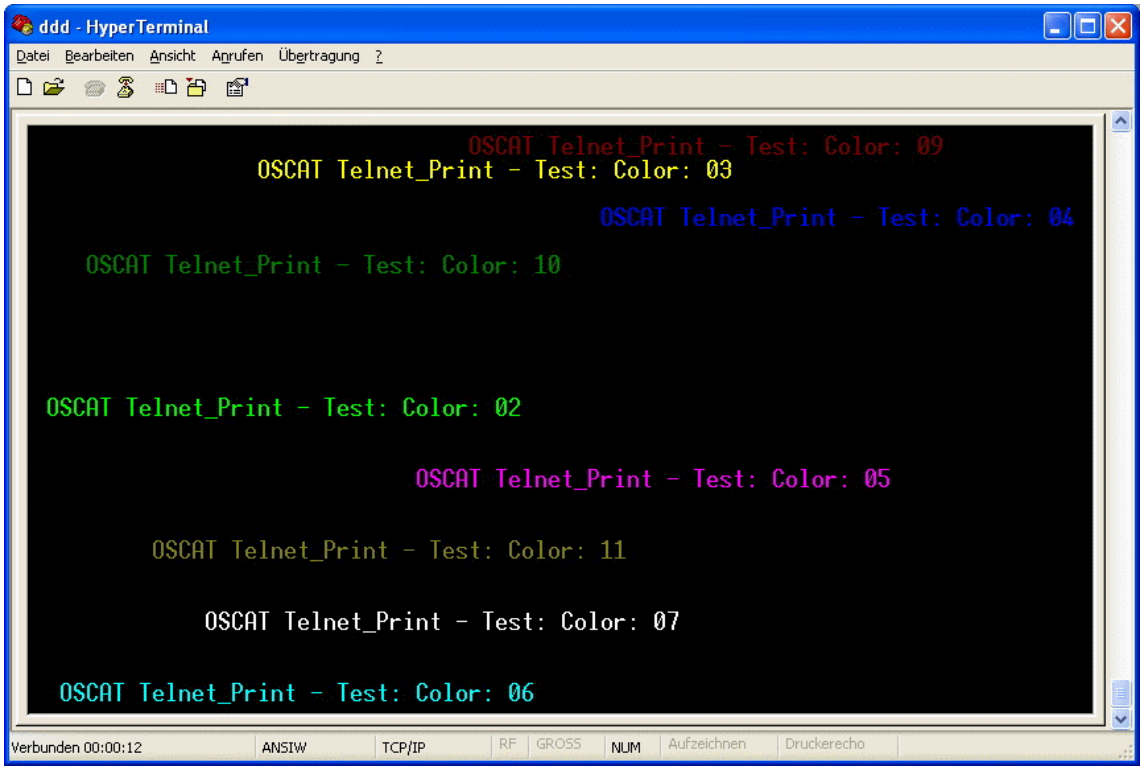
BIT	Function	Description
0	SCREEN_INIT	After connecting to the TELNET console the entire screen is cleared. If the COLOR OPTION is selected, the screen BACK_COLOR will be deleted.
1	AUTOWRAP	In AUTOWRAP = 1, the write cursor is on reaching the end of line is automatically set to a next line. If the text output the X,Y positions are always specified with, it is better when AUTOWRAP = 0.
2	COLOR	Enables the color mode, it will apply BACK_COLOR and FRONT_COLOR to the output.
3	NEW_LINE	In NEW_LINE = 1 is automatically a carriage return and line feed added to the end of the text. So the next text output starts a new line. But this is only useful if no X_pos and Y_pos be specified.
4	RESERVE	
5	RESERVE	
6	RESERVE	
7	NO_BUF_FLUSH	Prevents the data in the buffer to be sent immediately. Only if the buffer is completely full, or this option is disabled, the data is sent. Allows fast sending many texts in the same cycle

2.0.22 TELNET_PRINT

Type Function module	
IN_OUT IP_C	IP_C (parameterization)
S_BUF	NETWORK_BUFFER (transmit data)
INPUT TEXT	STRING(string_length) (output text)
S_BUF_SIZE	UINT ((Size of the buffer S_BUF)
ENABLE	BOOL (enable communication)
SEND	BOOL (positive edge - Send offense)
OPTION	BYTE (Send Options)
BACK_COLOR	BYTE (background color)
FRONT_COLOR	BYTE (foreground color)

X_pos	BYTE (X-coordinate of the cursor position)
Y_pos	BYTE (Y-coordinate of the cursor position)
PORT	WORD (port-number)
OUTPUT	READY: BOOL (module ready)
DONE	BOOL (positive edge - Transmission completed)
	The module enables easy output of text to a TELNET console. At the parameter TEXT is passed the desired string. To unlock the module for communication, ENABLE = 1 must be set, so that the registration takes place at IP_CONTROL. With parameter PORT can be defined the port number you want, if not value is specified the default port 23 is activated. With BACK_COLOR and FRONT_COLOR can be defined the colors you want, if the function parameter OPTION is activated. The parameters X_pos and Y_pos pass the desired coordinates of the text. If indicated in X_pos and Y_pos the value “0”, the text position is inactive, and the text are always appended at the current cursor position. The standard Telnet console allows X_pos (horizontal) from 1 to 80 and a Y_pos (Vertical) 1 to 25. The behavior here can in turn be modified by OPTION (Autowrap, carriage return, line feed, Buf_Flush etc..). If a large quantity of text will be issued, there may be a buffering enabled, so the data are written if either the buffer is full (this is from the module induced independently) or this is signaled by the amended OPTION parameter. By SEND = 1, the data is written into the buffer. The parameters may only be changed again if READY is 1, and with DONE the data acquisition was displayed as a positive edge.
OPTION	
FRONT_COLOR	
BACK_COLOR	





BIT	Function	Description
0	SCREEN_INIT	After connecting to the TELNET console the entire screen is cleared. If the COLOR OPTION is selected, the screen BACK_COLOR will be deleted.
1	AUTOWRAP	In AUTOWRAP = 1, the write cursor is on reaching the end of line is automatically set to a next line. If the text output the X,Y positions are always specified with, it is better when AUTOWRAP = 0.
2	COLOR	Enables the color mode, it will apply BACK_COLOR and FRONT_COLOR to the output.
3	NEW_LINE	In NEW_LINE = 1 is automatically a carriage return and line feed added to the end of the text. So the next text output starts a new line. But this is only useful if no X_pos and Y_pos be specified.
4	RESERVE	
5	RESERVE	
6	RESERVE	
7	NO_BUF_FLUSH	Prevents the data in the buffer to be sent immediately. Only if the buffer is completely full, or this option is disabled, the data is sent. Allows fast sending many texts in the same cycle

Byte	Color	Byte	Color
0	Black	16	Flashing Black
1	Light Red	17	Flashing Light Red
2	Light Green	18	Flashing Light Green
3	Yellow	19	Flashing Yellow
4	Light Blue	20	Flashing Light Blue
5	Pink / Light Magenta	21	Flashing Pink / Light Magenta

6	Light Cyan	22	Flashing Light Cyan
7	White	23	Flashing White
8	Black	24	Flashing Black
9	Red	25	Flashing Red
10	Green	26	Flashing Green
11	Brown	27	Flashing Brown
12	Blue	28	Flashing Blue
13	Purple / Magenta	29	Purple / Magenta
14	Cyan	30	Flashing Cyan
15	Gray	31	Flashing Gray

Byte	Color
0	Black
1	Red
2	Green
3	Brown
4	Blue
5	Purple / Magenta
6	Cyan
7	Gray

2.0.23 XML_READER

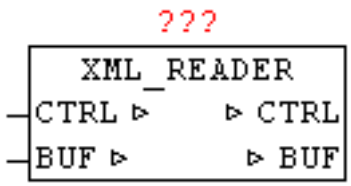
Type Function module		
IN_OUT CTRL	XML_CONTROL	
	(Control and status data)	
BUF	NETWORK_BUFFER (Receive data)	
	XML_READER means it is possible to parse so-called 'well-formed' XML documents. Here, not as usual at high-level languages, the whole XML data is read as a data structure and stored in memory, but a very resource-friendly version is used. The XML_READER reads XML data as a sequential data stream from the buffer and signals the in COMMAND defined element types automatically back.	
	With XML is a strict distinction between upper and lower case.	

	An XML document consists of just elements, attributes, their assignments, and the contents of the elements that can be text or child elements, which in turn can have attributes with assigned values and content. There are elements with and without attributes, elements can consist of many other elements, and those that may occur within the text only, and even empty elements that may have no content. The structure that emerges from these elements and their principles can be understood as a tree structure. Elements always consist of tags and end tags. Attributes are additional information about items. Comment elements are also allowed; however, these may not be between the start and end tags of elements in XML_READER. A possible DTD (Document Type Definition) will only be reported as DTD, but not further evaluated and applied by XML_READER. With a CDATA section, a parser is told that no markup follows, but normal text which is reported by start-end block.	
	Before the first call of the XML_READER a few parameters in the CTRL data structure needs to be initialized. CTRL.START_POS and CTRL.STOP_POS defines the beginning and the end of the XML data in the buffer. CTRL.COMMAND with one hand, can be an initialization (Bit15 = TRUE) and with bit 0-14 can be defined which element / data types are reported. Here the type codes in the following table corresponds just the bit number, which has to be set to True in CTRL.COMMAND.	
	It is tried to pass the text of element, attribute, Value and Path in total length to the accompanying STRINGS. In STRINGS greater than 255 characters this will be cut off flush left, but with block-start and block-end	

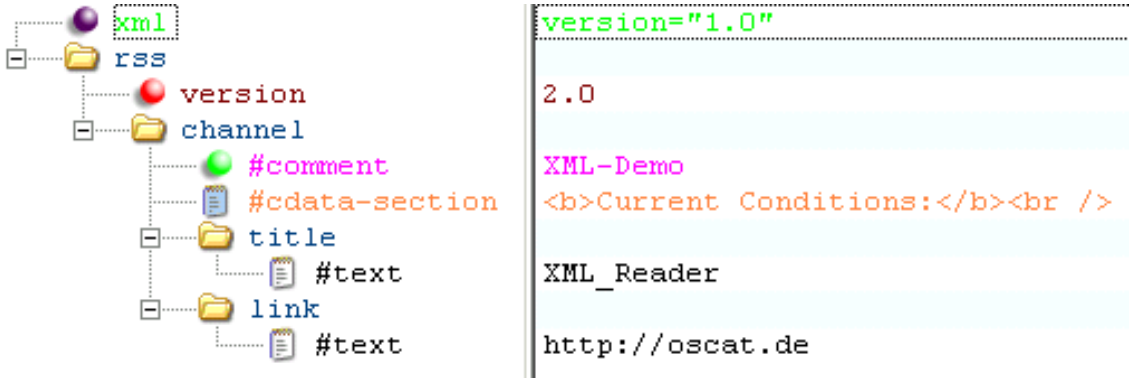
	parameters is reported back, so that they can subsequently be evaluated yet complete. The BLOCK-START/STOP index is always passed parallel to the STRINGS. If the PATH STRING is greater than 255 characters so the PATH tracking is disabled and only “OVERFLOW” is entered as text.	
	Since for very large and complex XML data is not clear, how long it takes until the module find data to report back, an WATCHDOG function is integrated. A maximum processing time can be parameterized. When reaching the time limit the module call is automatically canceled, and the next cycle resumes at the same point. The type code 98 is returned.	
	The following type codes are defined.	
	Sample XML	
	flat display	
<![CDATA[CurrentConditions	]]>XML_Reader http://oscat.de	
	Representation of the levels (without processing Instruction)	
	View as tree of item types	
Legend		
Application example		
	CASE STATE OF	
00		
STATE	= 10;	
CTRL.START_POS	= HTTP_GET.BODY_START; (Index of first character *)	
CTRL.STOP_POS	= HTTP_GET.BODY_STOP; (Index of last character *)	
CTRL.COMMAND	= WORD#2#11111111_11111111; (* Init + report all elements *)	
10		
	(* XML * data read serial)	
XML_READER.CTRL	= CTRL;	

XML_READER.BUF	= BUFFER;	
	XML_READER();	
CTRL	= XML_READER.CTRL;	
BUFFER	= XML_READER.BUF;	
	IF CTRL.TYP = 99 THEN	
STATE	= 20; (* Exit – no further elements available *)	
	ELSIF CTRL.TYP < 98 THEN (* do nothing at timeout(Code 98) *)	
	(* Evaluation of the XML elements by accessing the CTRL data structure *)	
	END_IF;	
20		
	(* miscellaneous..... *)	
	END_CASE;	
	The following information is passed via the CTRL-data structure	
––––First pass –––– COUNT	1 TYPE:	5 (OPEN ELEMENT - PROCESSING INSTRUCTION) LEVEL: 1 ELEMENT: ‘xml’ PATH: ‘/xml’ ––––Next cycle–––– COUNT: 2 TYPE: 4 (ATTRIBUTE) LEVEL: 1 ELEMENT: ‘xml’ ATTRIBUTE: ‘version’ VALUE: ‘1.0’ PATH: ‘/xml’ ––––Next cycle–––– COUNT: 3 TYPE: 2 (CLOSE ELEMENT) LEVEL: 0 ELEMENT: ‘xml’ PATH: ’’ ––––Next cycle–––– COUNT: 4 TYPE: 1 (OPEN ELEMENT - Standard) LEVEL: 1 ELEMENT: ‘rss’ PATH: ‘/rss’ ––––Next cycle–––– COUNT: 5 TYPE: 4 (ATTRIBUTE) LEVEL: 1 ELEMENT: ‘rss’ ATTRIBUTE: ‘version’ VALUE: ‘2.0’ PATH: ‘/rss’ ––––Next cycle–––– COUNT: 6 TYPE: 1 (OPEN ELEMENT - Standard) LEVEL: 2 ELEMENT: ‘channel’ PATH: ‘/rss/channel’ ––––Next cycle–––– COUNT: 7 TYPE: 13 (COMMENT-ELEMENT) LEVEL: 2 VALUE: ’ XML-Demo ’ PATH: ‘/rss/channel’ ––––Next cycle–––– COUNT: 8 TYPE: 12 (CDATA) LEVEL: 2 VALUE: ‘Current Con-

		ditions:’ PATH: ‘/rss/channel’ ––––Next cycle–––– COUNT: 9 TYPE: 1 (OPEN ELEMENT - Standard) LEVEL: 3 ELEMENT: ‘title’ PATH: ‘/rss/channel/title’ ––––Next cycle–––– COUNT: 10 TYPE: 3 (TEXT) LEVEL: 3 ELEMENT: ‘title’ VALUE: ’ XML_Reader’ PATH: ‘/rss/channel/title’ ––––Next cycle–––– COUNT: 11 TYPE: 2 (CLOSE ELEMENT) LEVEL: 2 ELEMENT: ‘title’ PATH: ‘/rss/channel’ ––––Next cycle–––– COUNT: 12 TYPE: 1 (OPEN ELEMENT - Standard) LEVEL: 3 ELEMENT: ‘link’ PATH: ‘/rss/channel/link’ ––––Next cycle–––– COUNT: 13 TYPE: 3 (TEXT) LEVEL: 3 ELEMENT: ‘link’ VALUE: ‘http://oscat.de’ PATH: ‘/rss/channel/link’ ––––Next cycle–––– COUNT: 14 TYPE: 2 (CLOSE ELEMENT) LEVEL: 2 ELEMENT: ‘link’ PATH: ‘/rss/channel’ ––––Next cycle–––– COUNT: 15 TYPE: 2 (CLOSE ELEMENT) LEVEL: 1 ELEMENT: ‘channel’ PATH: ‘/rss’ ––––Next cycle–––– COUNT: 16 TYPE: 2 (CLOSE ELEMENT) LEVEL: 0 ELEMENT: ‘rss’ PATH: ’’ ––––Next cycle–––– COUNT: 17 TYPE: 99 (EXIT – END OF DATA)




```
-<rss version="2.0">
 -<channel>
    <!-- XML-Demo  -->
    <b>Current Conditions:</b><br />
    <title>XML_Reader</title>
    <link>http://oscat.de</link>
  </channel>
</rss>
```

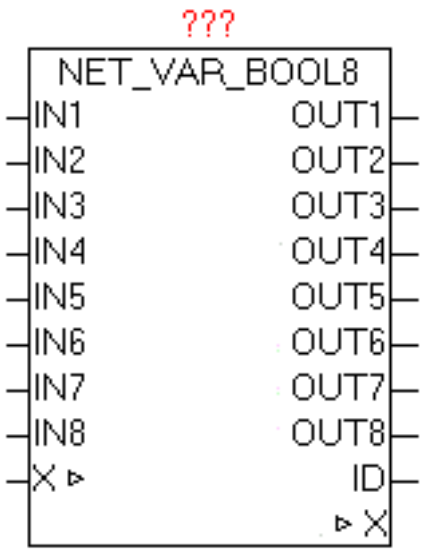


Type (Code)	Data Type	Description
00	Unknown	Undefined item found
01	TAG (element)	Start - of ElementData pointer for the element of BLOCK1
02	END-TAG (Element)	End - of the element
03	TEXT	Content of an elementData pointer for value on BLOCK1
04	ATTRIBUTE	Attributes of an elementData pointer for attributes of BLOCK1Data pointer for value on BLOCK2
05	TAG (Processing Instruction)	Instructions for processingData pointer for the element of BLOCK1
12	CDATA	not analyzed content TEXTData pointer for value on BLOCK1
13	COMMENT	COMMENTData pointer for value on BLOCK1
14	DTD	Document Type DeclarationData pointer for value on BLOCK1
98	WATCHDOG	Maximum processing time reached - cancel
99	END	No more items available

3. NET VAR

3.0.1 NET_VAR_BOOL8

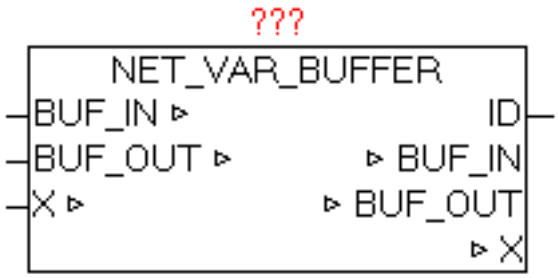
Type Function module	
IN_OUT X	NET_VAR_DATA (NET_VAR data structure)
INPUT	IN1 ..8 BOOL (signal input)
OUTPUT	OUT1 ..8 BOOL (signal output)
ID	BYTE (ID)
	The module is used for bidirectional transmission of NET_VAR_BOOL8 8 binary signals from the master to slave and vice versa. The signals IN 1..8 are collected and passed to the other side (control) on the same module at the same position as OUT1..8 again.
	Simultaneously, the on the opposite side (other control) passed input data passed here as a OUT1..8 again.
	ID parameter indicates the current identification number of the module instance. If the configuration of the master and the slave program is differently (incorrectly) that ID number is passed as a fault in the module NET_VAR_CONTROL.



3.0.2 NET_VAR_BUFFER

Type Function module	
IN_OUT X	NET_VAR_DATA (NET_VAR data structure)
BUF_IN	ARRAY [1..64] OF BYTE (input data buffer)

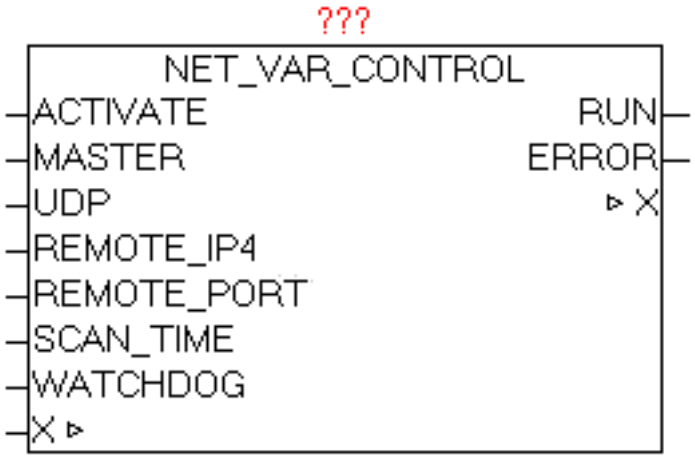
BUF_OUT	ARRAY [1..64] OF BYTE (output data buffer)
OUTPUT ID	BYTE (ID)
	The module NET_VAR_BUFFER is used for bidirectional transmission of 64 bytes from the master to slave and vice versa. The data from BUF_IN be recorded and passed on the other side (other plc) on the same module at the same position as BUF_OUT.
	Simultaneously, the input data on the opposite side (other control) is passed here as BUF_OUT again.
	ID parameter indicates the current identification number of the module instance. If the configuration of the master and the slave program is differently (incorrectly) that ID number is passed as a fault in the module NET_VAR_CONTROL.



3.0.3 NET_VAR_CONTROL

Type Function module	
IN_OUT X	NET_VAR_DATA (NET_VAR data structure)
INPUT ACTIVATE	BOOL (Enables the exchange of data)
MASTER	BOOL (FALSE = SLAVE / MASTER = TRUE)
UDP	BOOL (FALSE=TCP / TRUE = UDP)
REMOTE_IP4	DWORD (IP4-address of the other SPS)
REMOTE_PORT	WORD (PORT number of other PLC)
SCAN_TIME	TIME (update time)
WATCHDOG	TIME (monitoring time)
OUTPUT RUN	BOOL (active data exchange - no error)
	ERROR DWORD ((error code)
	The module NET_VAR_CONTROL coordinates the data exchange between the two controllers and the satellite components NET_VAR_*. With ACTIVATE = TRUE, the data exchange will be released. The module must be invoked on both controllers, with the parameter MASTER must be assigned once with TRUE and once must be FALSE. Thus determines which side the active connection will establish. With UDP (FALSE / TRUE) can be specified whether a UDP or TCP connection is used. The the IP address of the other side must be specified in REMOTE-IP4, and alternatively, the port address (default port is 10000). The SCAN TIME determines a data refresh interval (default is T # 1s). WIth WATCHDOG the monitoring time

	is set (default is T # 2s). When data exchange runs, the parameter RUN = TRUE. If the data exchange is longer than the watchdog time not possible, RUN = FALSE and an error is passed. The error will not be acknowledged, because the module automatically tries to restore the data exchange. Once no more error exists, RUN = TRUE and the error code is cleared.
ERROR	(regarded as a HEX value!)

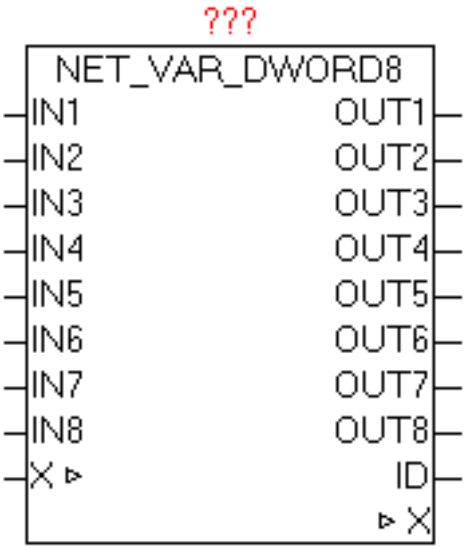


DWORD	Message Type	Description
B3	B2	B1
XX	..	..
..	XX	..
..	..	XX
..	..	..

3.0.4 NET_VAR_DWORD8

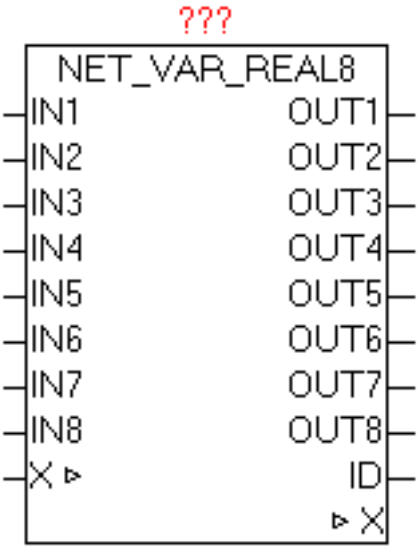
Type Function module	
IN_OUT X	NET_VAR_DATA (NET_VAR data structure)
INPUT IN 1..8	DWORD (input DWORD)
OUTPUT OUT 1..8	DWORD (output DWORD)
ID	BYTE (ID)
	The module NET_VAR_DWORD8 is used for bidirectional transmission of eight DWORD from the master to slave and vice versa. The signals DWORD IN1..8 are collected and passed to the other side (control) on the same module at the same position as OUT1..8 again.
	Simultaneously, the on the opposite side (other control) passed input datawords passed here as a OUT1..8 again.
	ID parameter indicates the current identification number of the module instance. If the configuration of the master and the slave program is

	differently (incorrectly) that ID number is passed as a fault in the module NET_VAR_CONTROL.



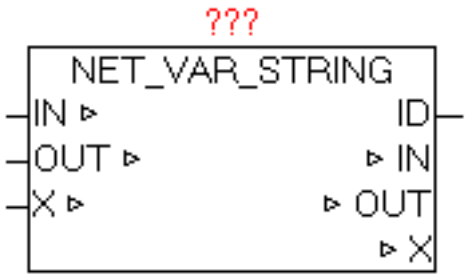
3.0.5 NET_VAR_REAL8

Type Function module	
IN_OUT X	NET_VAR_DATA (NET_VAR data structure)
INPUT IN 1 .. 8	REAL (input)
OUTPUT OUT 1 .. 8	REAL (output value)
ID	BYTE (ID)
	The module NET_VAR_REAL8 is used for bidirectional transmission of eight REAL-values from the master to slave and vice versa. The REAL values IN1..8 are collected and passed to the other side (control) on the same module at the same position as OUT1..8 again.
	Simultaneously, the on the opposite side (other control) passed input REAL values are passed here as a OUT1..8 again.
	ID parameter indicates the current identification number of the module instance. If the configuration of the master and the slave program is differently (incorrectly) that ID number is passed as a fault in the module NET_VAR_CONTROL.



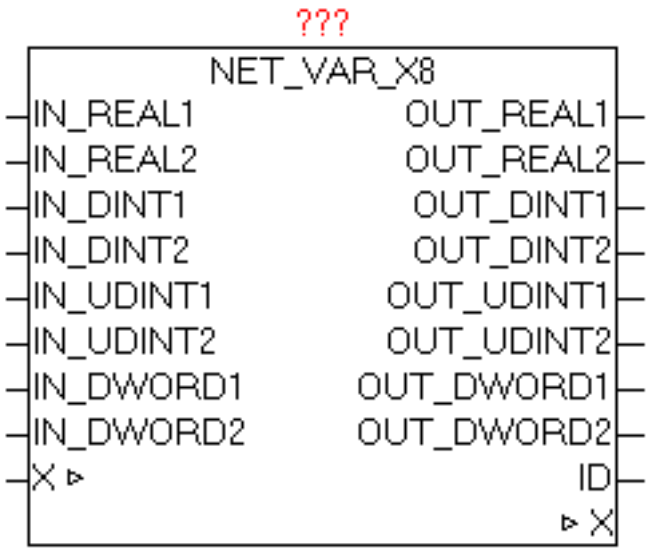
3.0.6 NET_VAR_STRING

Type Function module	
IN_OUT X	NET_VAR_DATA (NET_VAR data structure)
IN	STRING(string_length) (input string)
OUT	STRING (string_length) (output-string)
OUTPUT ID	BYTE (ID)
	The module NET_VAR_STRING is used for bidirectional transmission of STRING from the master to slave and vice versa. The STRING in the parameters IN will be recorded and passed on the other side (control) on the same module at the same position as OUT parameter.
	At the same time the input String on the opposite side of the (other control) is passed here as a OUT value again.
	ID parameter indicates the current identification number of the module instance. If the configuration of the master and the slave program is differently (incorrectly) that ID number is passed as a fault in the module NET_VAR_CONTROL.



3.0.7 NET_VAR_X8

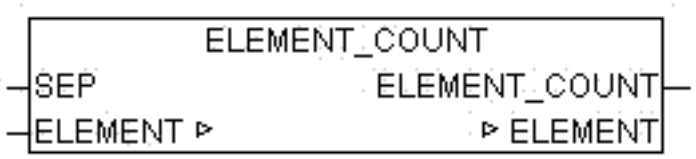
Type Function module	
IN_OUT X	NET_VAR_DATA (NET_VAR data structure)
INPUT IN_REAL1	REAL (input)
IN_REAL2	REAL (input)
IN_DINT1	DINT (input)
IN_DINT2	DINT (input)
IN_UDINT1	DINT (input)
IN_UDINT2	DINT (input)
IN_DWORD1	DINT (input)
IN_DWORD2	DINT (input)
OUTPUT OUT_REAL1	REAL (ouput)
OUT_REAL2	REAL (output)
OUT_DINT1	DINT (output)
OUT_DINT2	DINT (output)
OUT_UDINT1	DINT (output)
OUT_UDINT2	DINT (output)
OUT_DWORD1	DINT (output)
OUT_DWORD2	DINT (output)
ID	BYTE (ID)
	The module NET_VAR_X8 is used for bidirectional transmission of each two REAL, DINT, UINT, DWORD values from the master to slave and vice versa. The signals IN1..8 are collected and passed to the other side (control) on the same module at the same position as OUT1..8 again.
	Simultaneously, the input data on the opposite side (other control) is passed here as BUF_OUT again.
	ID parameter indicates the current identification number of the module instance. If the configuration of the master and the slave program is differently (incorrectly) that ID number is passed as a fault in the module NET_VAR_CONTROL.


???
NET_VAR_X8
IN_REAL1
IN_REAL2
IN_DINT1
IN_DINT2
IN_UDINT1
IN_UDINT2
IN_DWORD1
IN_DWORD2
X
OUT_REAL1
OUT_REAL2
OUT_DINT1
OUT_DINT2
OUT_UDINT1
OUT_UDINT2
OUT_DWORD1
OUT_DWORD2
ID
X

4. OTHER

4.0.1 ELEMENT_COUNT

Type Function	INT
Input SEP	BYTE (separation character of the elements)
I / O ELEMENT	STRING(ELEMENT_LENGTH) (input list)
Output	INT (number of items in the list)
	ELEMENT_COUNT determines the number of items in a list.
	If the parameter ELEMENT is an empty string 0 is passed as result. If at least one character is in ELEMENT it is evaluated as a single element and ELEMENT_COUNT = 1 is passed to output.



Example:

Examples:
ELEMENT_COUNT('0,1,2,3',44) = 4
ELEMENT_COUNT('',44) = 0
ELEMENT_COUNT('x',44) = 1

4.0.2 ELEMENT_GET

Type Function	STRING(ELEMENT_LENGTH)
Input SEP	BYTE (separation character of the elements)
POS	INT (of the item)
I / O ELEMENT	STRING(ELEMENT_LENGTH) (input list)
Output	STRING (String output)
	ELEMENT_GET passes the item at the position POS from a list. The list consists of strings which are separated by the separation character SEP. The first element of the list has the position 0

Example:
Examples:
ELEMENT_GET(‘ABC,23,,NEXT’, 44, 0) = ‘ABC’
ELEMENT_GET(‘ABC,23,,NEXT’, 44, 1) = ‘23’
ELEMENT_GET(‘ABC,23,,NEXT’, 44, 2) = ’’
ELEMENT_GET(‘ABC,23,,NEXT’, 44, 3) = ‘NEXT’
ELEMENT_GET(‘ABC,23,,NEXT’, 44, 4) = ’’
ELEMENT_GET(’’, 44, 0) = ’’

4.0.3 NETWORK_VERSION

Type Function	DWORD
Input IN	BOOL (if TRUE the module provides the release date)
Output	(Version of the library)
	NETWORK_VERSION provides if IN = FALSE the current version number as DWORD. If IN is set to TRUE then the release date of the current version as a DWORD is returned.

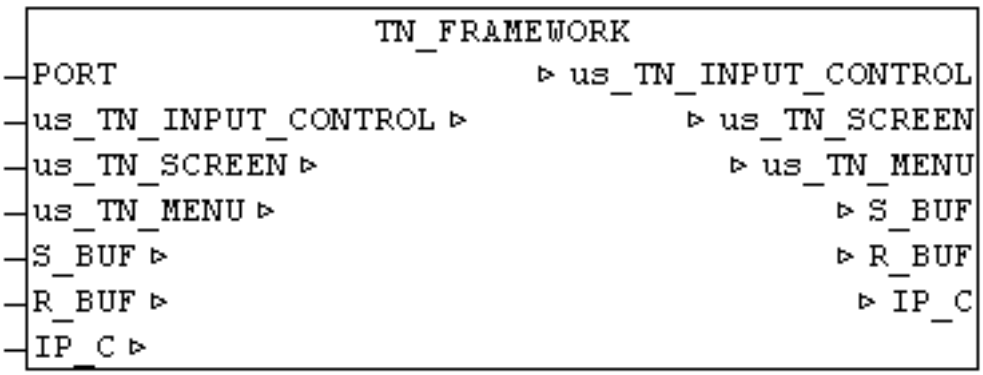
Example:
Example: NETWORK_VERSION(FALSE) = 111 for version 1.11
DWORD_TO_DATE(NETWORK_VERSION(TRUE)) = 2011-2-3

5. TELNET VISION

5.0.1 TN_FRAMEWORK

Type	Function module
IN_OUT Xus_TN_INPUT_CONTROL	Us_TN_INPUT_CONTROL
Xus_TN_SCREEN	Us_TN_SCREEN
Xus_TN_MENU	us_TN_MENU
S_BUF	NETWORK_BUFFER (transmit data)
R_BUF	NETWORK_BUFFER (receive data)
IP_C	IP_CONTROL (parameterization)
	The module TN_FRAMEWORK is a frame structure, which provides a finished maturity model for TELNET-Vision .
	The following tasks and functions are treated.
	Connection setup and breakdown with Telnet Client
	Send and receive data
	Data structures for graphics functions
	INPUT_CONTROL elements
	Intelligent automatic updating of the Telnet display
	Menu bar display
	Direct access to all data structures for user program

???

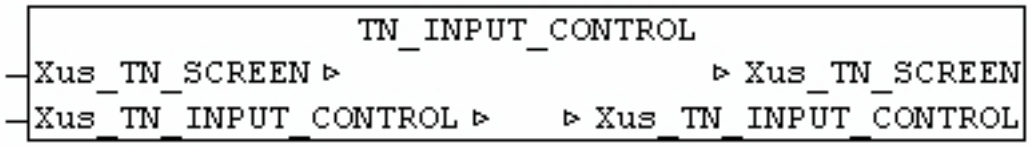


5.0.2 TN_INPUT_CONTROL

Type	Function module

IN_OUT Xus_TN_SCREEN	Us_TN_SCREEN
Xus_TN_INPUT_CONTROL	us_TN_INPUT_CONTROL
	The module TN_INPUT_CONTROL is used to manage the INPUT_CONTROL elements. If Xus_TN_INPUT_CONTROL.bo_Reset_Fokus = TRUE then the FOCUS is disabled on all elements and the first item gets to the focus. Using the cursor up / down buttons and tab, the individual elements can be selected or changed. The current element loses focus and then the next following item gets the input focus reallocated. At the focus change of the elements automatically a redraw of the respective elements is triggered. The image/flashing cursor is always positioned at each active element and is displayed. It always automatically displays and updates the ToolTip text, as this has been configured.
	It supports the following elements.
	TN_INPUT_EDIT_LINE
	TN_INPUT_SELECT_TEXT
	TN_INPUT_SELECT_POPUP

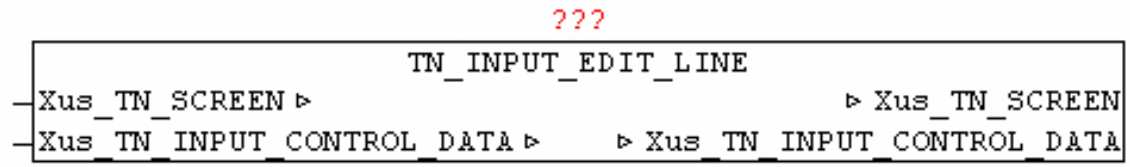
???



5.0.3 TN_INPUT_EDIT_LINE

Type	Function module	
IN_OUT Xus_TN_SCREEN	Us_TN_SCREEN	
Xus_TN_INPUT_CONTROL	us_TN_INPUT_CONTROL	
	The module TN_INPUT_EDIT_LINE is used to manage a command line. This must be set *. in_TYPE = 1.	
	The item will be provided as *.in_X and *.in_Y. Every entry line can be provided with a title text. With *.in_Title_Y_Offset and *.in_Title_X_Offset the position relative to the element coordinates is expressed. The color can be determined with *.by_Title_Attr, and the text by *.st_Title_String. If a tool tip should appear at the element *. st_Input_ToolTip the text hast to be specified.	
	If the item has focus, using the keyboard cursor left / right the flashing cursor can be moved within the line. The backspace key can delete entered character. By pressing the Enter / Return key the input text is issued at *.st_Input_String and *.bo_Input_Entered	

	ist set to TRUE. The input flag must be reset after receive by the user. Using *.bo_Input_Hidden = TRUE the hidden input is activated, thus, all input characters represented with a ”.*	
	Using *.st_Input_Mask determines at which position and how many characters can be entered. At each position which a space, character can be entered. During initialization *.st_Input_Mask must be copied once to *.st_Input_Data.	
	Is *.bo_Input_Only_Num = TRUE only numeric keys are accepted and adopted.	
***.in_Type**	= INT#01; *.in_Y := INT#16; *.in_X := INT#09; *.by_Attr_mF := BYTE#16#72; (* white, green *) *.by_Attr_oF := BYTE#16#74; (* white, blue *) *.in_Cursor_Pos := INT#0; *.bo_Input_Only_Num := FALSE; *.bo_Input_Hidden := FALSE; *.st_Input_Mask := ’ ’; *.st_Input_Data := *.st_Input_Mask; *.st_Input_ToolTip := ’inputline active	SCROLL F1/F2/F3/F4

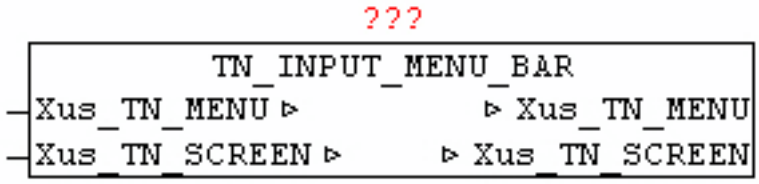


Example:
Example:
The following output:

5.0.4 TN_INPUT_MENU_BAR

Type	Function module
IN_OUT Xus_TN_SCREEN	Us_TN_SCREEN
Xus_TN_MENU	us_TN_MENU
	The module TN_INPUT_MENU_BAR is used to manage and view the Menu_Bar. The element is shown in *.in_X and *.in_Y. The menu items are stored as elements within verschachelte *.st_MENU_TEXT. Two different separators are used. A ‘$’ separates the different menu lists, and each menu list is further divided by ‘#’ into individual menu items. The first menu list is the actual menu bar, this implies the number of sub-

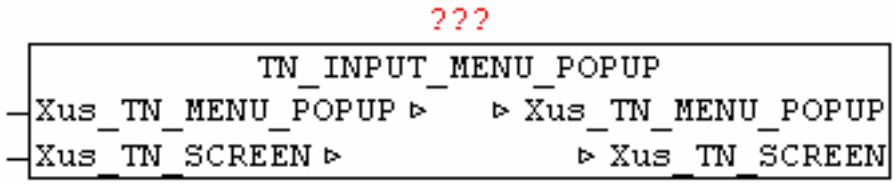
	menus, and the titles of the elements. Then all the sub-menu lists are follow and are separated by '%'. To devide individual sub-menu items from each other or providing them with a cut line, an '-' has to submitted as text menu-element.
By pressing the Escape key, the menu bar activated and the respective sub-menu is displayed using the module TN_INPUT_MENU_POPUP. Within the sub-menu can be navigated with up / down key. Within the sub-menu can be navigated with up / down cursor. If a sub-menu item is confirmed by pressing Enter / Return key, then in .in_Menu_Selected the number of the selected menu-point is passed. The calculation of the menu item number is as following	Main menu index * 10 + Submenu-index. The entry in *.in_Menu_Selected needs set again to 0 after acceptance by users.
	Thus, a maximum of 9 main menu items and each 9-Submenu items are executable. Means of escape key at any time the menu can be hided again.
	Active Menu automatically backs up the background before it is drawn, and restores the background after ending.
	As long as a menu is display, the user program may not make graphical changes. This can be checked by TN_SCREEN.bo_Menue_Bar_Dialog = TRUE.
***.in_X**	= INT#00;
***.in_Y**	= INT#00;
***.by_Attr_mF**	= BYTE#16#33; (* yellow + brown *)
***.by_Attr_oF**	= BYTE#16#0F; (* black + grey *)
***.st_MENU_TEXT**	= 'File#Edit#View#End';
***.st_MENU_TEXT**	= CONCAT(*.st_MENU_TEXT,
***.st_MENU_TEXT**	= CONCAT(*.st_MENU_TEXT,
***.bo_Create**	= TRUE;



Example:
Example:
‘%oeffnen#-#speichern#beenden%loeschen#-#einfuegen#-#kopieren’);
‘%alles#detail#kopieren%Logout’);
The following output:

5.0.5 TN_INPUT_MENU_POPUP

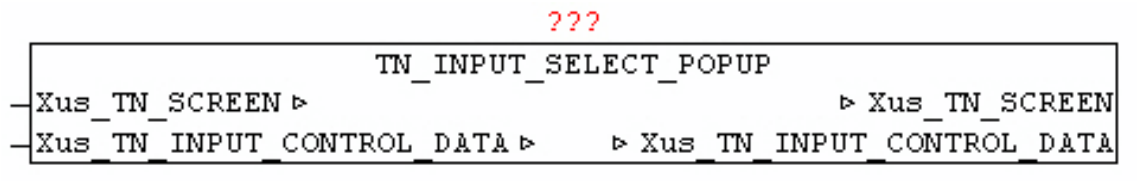
Type	Function module
IN_OUT Xus_TN_SCREEN	Us_TN_SCREEN
Xus_TN_MENU	us_TN_MENU
	The module TN_INPUT_MENU_POPUP is used to manage and display the Menu_Bar Submenu and for the representation of TN_INPUT_SELECT_POPUP elements. The element is shown in *.in_X and *.in_Y. The menu items are stored as elements within *.st_Menu_Text. The individual element are devided from each other using ‘#’. To devide individual sub-menu items from each other or providing them with a cut line, an ‘-’ has to submitted as text menu-element.
	Within the sub-menu can be navigated with up / down key. If a sub-menu item is confirmed by pressing Enter / Return key, then in *.in_Menu_Selected the number of the selected menu-point is passed.
	An active Menu automatically backs up the background before it is drawn, and restores the background after ending.
	As long as a menu is display, the user program may not make graphical changes. This can be checked by TN_SCREEN.bo_Menue_Bar_Dialog = TRUE or TN_SCREEN.bo_Modal_Dialog = TRUE.
	The module is primarily from TN_INPUT_MENU_BAR and TN_INPUT_SELECT_POPUP used internally, and need not be executed directly by the user.

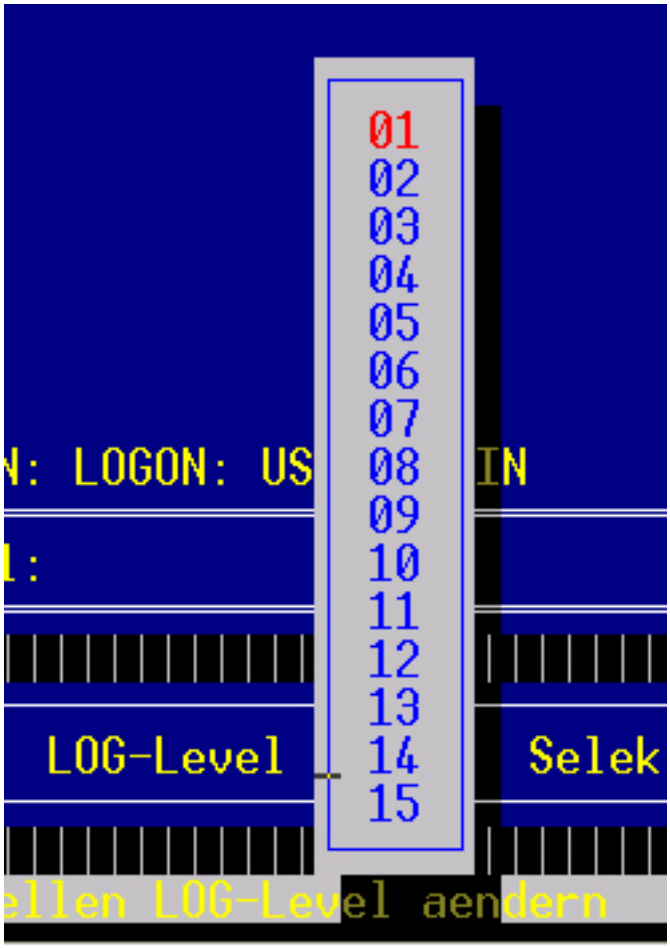


5.0.6 TN_INPUT_SELECT_POPUP

Type	Function module	
IN_OUT Xus_TN_SCREEN	Us_TN_SCREEN	
Xus_TN_INPUT_CONTROL	us_TN_INPUT_CONTROL	
	The module TN_INPUT_SELECT_POPUP is used to manage a selection of texts, by displaying a pop-up dialogue. This must be set *.IN_TYPE = 3.	
	The item will be provided as *.in_X and *.in_Y. Every entry line can be provided with a title text. With *.in_Title_Y_Offset and *.in_Title_X_Offset the position relative to the element coordinates is expressed. The color can be determined with *.by_Title_Attr, and the text by *.st_Title_String. If a tool tip should appear at the element *. st_Input_ToolTip the text hast to be specified.	
	The selection of texts will be handed over in *.st_Input_Data. The text element should be separated from each other by the character '#'.	
	If the focus is on an element, using the Enter / Return key selection dialog can be activated.	
	With the cursor up/down can be changed between the individual elements. If the beginning or the end of the list will be reachted, it continues at the opposite side.	
	The text-element is connected by means *.st_Input_Mask, meaning that the output text length are affected later.	
	By pressing the Enter / Return key is the text of the selected element is passed to *. st_Input_String and *. bo_Input_Entered = TRUE. The input flag must be reset after receive by the user.	

	An active selection (selection dialog) can always be canceled with the Escape key.	
***.in_Type**	= 03; *.in_Y := 20; *.in_X := 18; *.by_Attr_mF := 16#17; *.by_Attr_oF := 16#47; *.st_Input_ToolTip :=	'Change the current log level



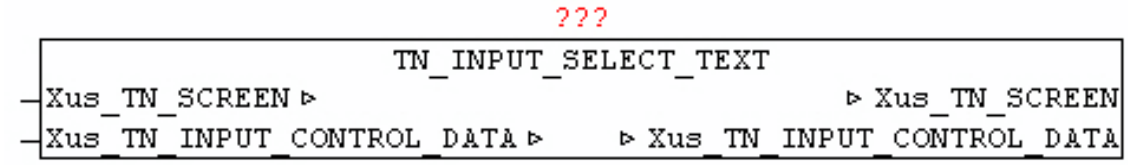


Example:
Example:
The following output:

5.0.7 TN_INPUT_SELECT_TEXT

Type	Function module	
IN_OUT Xus_TN_SCREEN	Us_TN_SCREEN	
Xus_TN_INPUT_CONTROL	us_TN_INPUT_CONTROL	

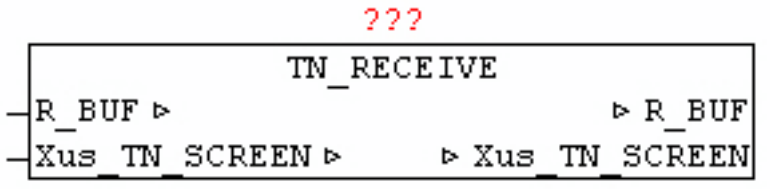
	The module TN_INPUT_SELECT_TEXT is used to manage a selection of texts. This must be set *.IN_TYPE = 2.	
	The item will be provided as *.in_X and *.in_Y. Every entry line can be provided with a title text. With *.in_Title_Y_Offset and *.in_Title_X_Offset the position relative to the element coordinates is expressed. The color can be determined with *.by_Title_Attr, and the text by *.st_Title_String. If a tool tip should appear at the element *.st_Input_ToolTip the text hast to be specified.	
	The selection of texts will be handed over in *.st_Input_Data. The text element should be separated from each other by the character '#'.	
	If the Element has the focus, by using the spacebar (space) can be changed between the individual texts. The text-element is connected by means *.st_Input_Mask, meaning that the output text length are affected later.	
	By pressing the Enter / Return key the input text is issued at *.st_Input_String and *.bo_Input_Entered ist set to TRUE. The input flag must be reset after receive by the user.	
***.in_Type**	= 2; *.in_Y := 20; *.in_X := 58; *.by_Attr_mF := 16#17; *.by_Attr_oF := 16#47; *.st_Input_ToolTip := ' selection text active	press space to select



Example:
Example:
The following output:

5.0.8 TN_RECEIVE

Type	Function module	
IN_OUT Xus_TN_SCREEN	us_TN_SCREEN	
R_BUF	NETWORK_BUFFER	(Telnet receive buffer)
	The module TN_RECEIVE receives input data from the Telnet client, and evaluates the key codes.	
	If the key code in the range 32-126 it shall be stored as ASCII code under Xus_TN_SCREEN, by_Input_ASCII_Code. In addition, Xus_TN_SCREEN.bo_Input_ASCII_IsNum = TRUE if this corresponds to a number between 0 and 9.	
	If the key code is of the following extended code then this is filed under Xus_TN_SCREEN,by_Input_Exten_Code.	

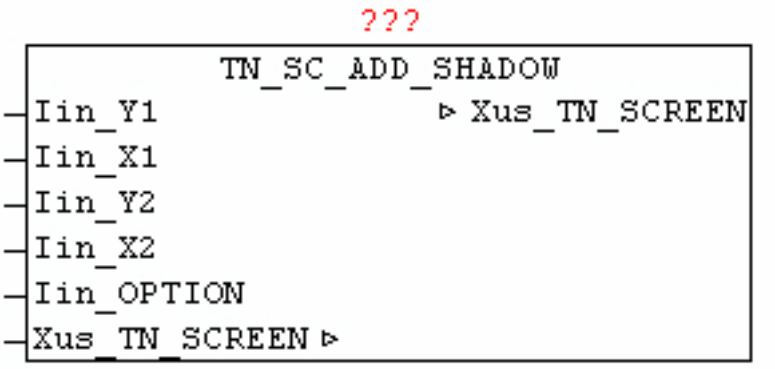


Exten_code	Button name
65	Cursor up
66	Cursor down
67	Cursor RIGHT
68	Cursor left
72	Pos1
75	End
80	F1
81	F2
82	F3
83	F4
8	Backspace
9	Tabulator
13	Return (Enter)
27	Escape

5.0.9 TN_SC_ADD_SHADOW

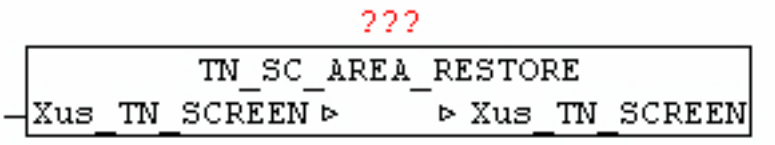
Type	Function module
INPUT	Iin_Y1: INT (Y1 coordinate of the area)

Iin_X1	INT: (X1 coordinate of the area)
Iin_Y2	INT (Y2 coordinate of the area)
Iin_X2	INT: (X2-coordinate of the area)
Iin_OPTION	INT: (kind of the shadow)
IN_OUT Xus_TN_SCREEN	Us_TN_SCREEN
	The module TN_SC_ADD_SHADOW allows you to add optical shadow to rectangular glyphs. By specifying a rectangular area by means of the parameters X1, Y1 and X2, Y2, a basic framework is defined, at which at the right and bottom color darkened lines are drawn (shadow). The shadow coordinates X1, Y1 and X2, Y2 are always given +1 for proper primitive. OPTION means you can choose between two shadow variations. If OPITION = 0 then the shadow is reached by pure color adjustment (darkening of the character) . If an OPTION > 0, in the area of the shadow all the characters replaced by black filled characters.



5.0.10 TN_SC_AREA_RESTORE

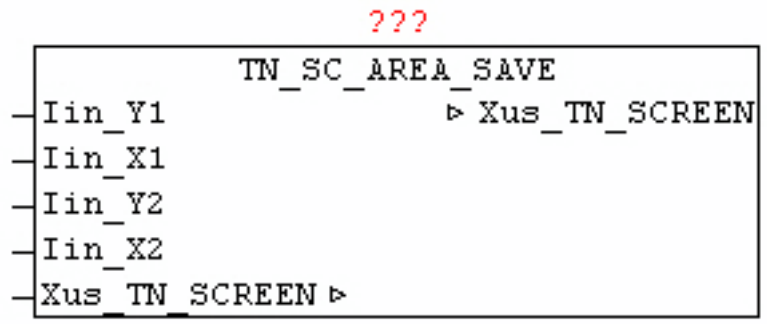
Type	Function module
IN_OUT Xus_TN_SCREEN	Us_TN_SCREEN
	The module TN_SC_AREA_RESTORE enables recovery of previously saved screen area. The screen data in Xus_TN_SCREEN.bya_BACKUP [x] is restored using the stored coordinates. This is done mainly done after the call from the module MENU-BAR amd MENU-POPUP, to restore the modified screen.



5.0.11 TN_SC_AREA_SAVE

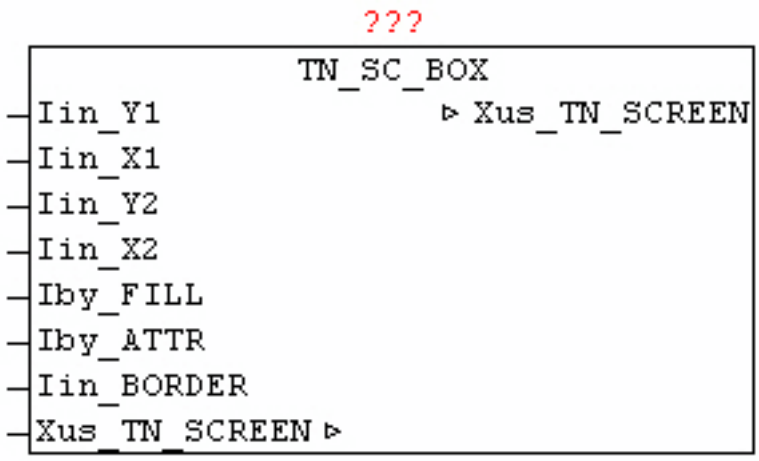
Type	Function module
INPUT	Iin_Y1: INT (Y1 coordinate of the area)

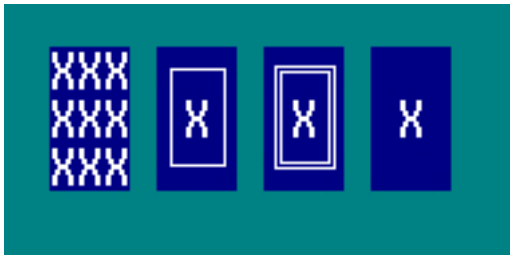
Iin_X1	INT: (X1 coordinate of the area)
Iin_Y2	INT (Y2 coordinate of the area)
Iin_X2	INT: (X2-coordinate of the area)
IN_OUT Xus_TN_SCREEN	Us_TN_SCREEN
	The module TN_SC_AREA_SAVE allows you to save of rectangular areas of the screen before it is modified by other drawing operations. This is mainly done before the call from the module BAR-MENU and MENU-POPUP , because these are the elements as an overlay graphic. Means X1, Y1 and X2, Y2 are given the coordinates of the secured area of the screen. The data are saved in the data area Xus_TN_SCREEN.bya_BACKUP [x]. Here the coordinates and the actual characters and color information is stored. The buffer can hold up half the area of the screen.



5.0.12 TN_SC_BOX

Type	Function module
INPUT	Iin_Y1: INT (Y1 coordinate of the area)
Iin_X1	INT: (X1 coordinate of the area)
Iin_Y2	INT (Y2 coordinate of the area)
Iin_X2	INT: (X2-coordinate of the area)
Iby_FILL	BYTE: (fill in the character of the area)
Iby_ATTR	BYTE: (color code to fill the area)
Iby_BORDER	BYTE: (type of frame)
IN_OUT Xus_TN_SCREEN	Us_TN_SCREEN
	The module TN_SC_BOX is used to draw a rectangular area, that is filled with the specified character in Iby_FILL. With parameter Iby_ATTR fill color can be specified. The fill area is drawn with a border that is given by Iin_BORDER.
Border types	
	0 = no border
	1 = frame with a single line
	2 = frame double line
	3 = frame with spaces

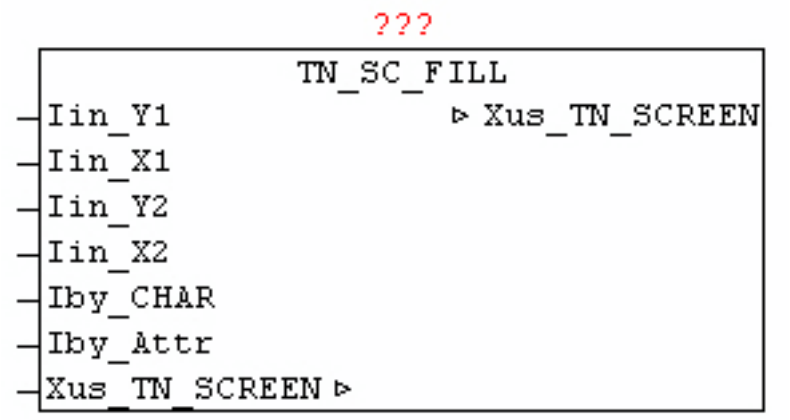




Example:
Example: Box with leaders 'X' and white color to blue
Representation with Iin_BORDER value 0,1,2 and 3 (from left to right)

5.0.13 TN_SC_FILL

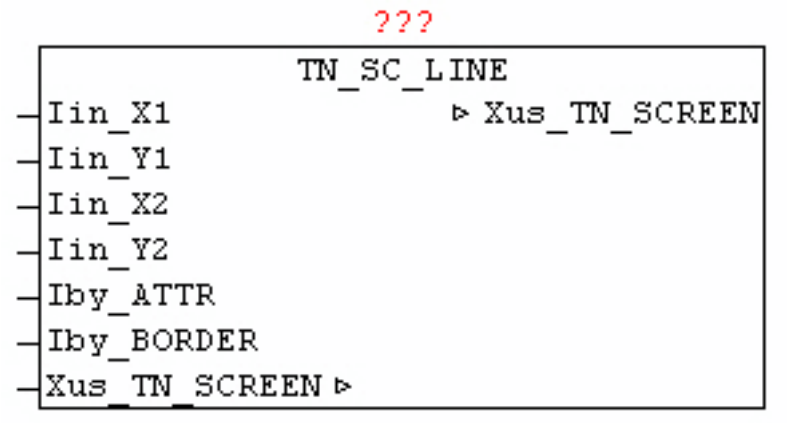
Type	Function module
INPUT	Iin_Y1: INT (Y1 coordinate of the area)
Iin_X1	INT: (X1 coordinate of the area)
Iin_Y2	INT (Y2 coordinate of the area)
Iin_X2	INT: (X2-coordinate of the area)
Iby_CHAR	BYTE: (character to fill in the the area)
Iby_ATTR	BYTE: (color code to fill the area)
IN_OUT Xus_TN_SCREEN	Us_TN_SCREEN
	The module TN_SC_FILL is used to draw a rectangular area, that is filled with the specified character in Iby_FILL.



Example:
Example: Box with leaders 'X' and white color to blue

5.0.14 TN_SC_LINE

Type	Function module
INPUT	Iin_Y1: INT (Y1 coordinate of the line)
Iin_X1	INT: (X1 coordinate of the line)
Iin_Y2	INT (Y2 coordinate of the line)
Iin_X2	INT: (X2-coordinate of the line)
Iby_ATTR	BYTE: (color code of the line)
Iby_BORDER	BYTE: (type of line)
IN_OUT Xus_TN_SCREEN	Us_TN_SCREEN
	The module TN_SC_LINE is used to draw horizontal and vertical lines. By means of the X1/Y1 and X2/Y2 coordinates defines the beginning and the end of the line. The line type is passed by Iin_BORDER and the color code with Iby_ATTR. If when drawing a line and another line of this type cut, automatically the appropriate crossing sign is used.
Border types	
	1 = line with single line
	2 = line with double line
	> 2 = line is drawn with the specified character in Iin_BORDER

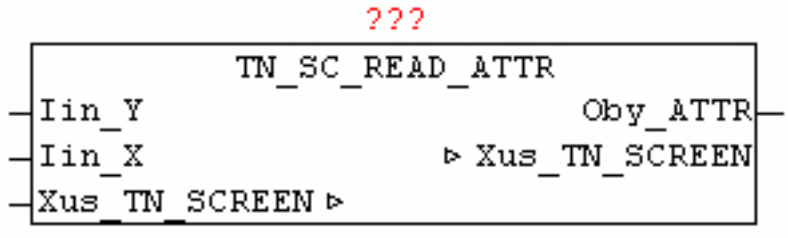


Example:

Example:
Horizontal line: type single-line
Vertical line: Type Double-Line
Horizontal and vertical lines crossed: Single-Line Type
Horizontal and vertical lines crossed: Type Double-Line
Horizontal line: type character (X)

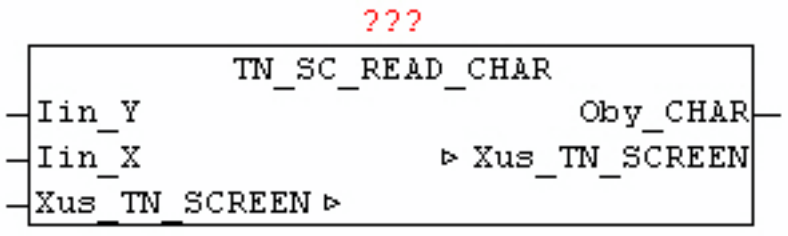
5.0.15 TN_SC_READ_ATTR

Type	Function module
INPUT Iin_Y	INT: (Y coordinate)
Iin_X	INT: (X coordinate)
OUTPUT Oby_ATTR	BYTE: (color information at position X / Y)
IN_OUT Xus_TN_SCREEN	Us_TN_SCREEN
	The block TN_SC_READ_ATTR is used to read the current color of the character at the specified location X / Y.



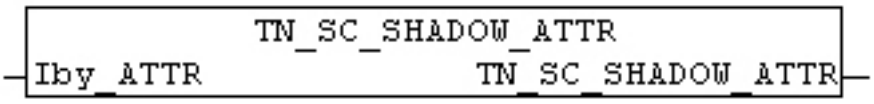
5.0.16 TN_SC_READ_CHAR

Type	Function module
INPUT Iin_Y	INT: (Y coordinate)
Iin_X	INT: (X coordinate)
OUTPUT Oby_CHAR	BYTE: (character at position X / Y)
IN_OUT Xus_TN_SCREEN	Us_TN_SCREEN
	The module TN_SC_READ_CHAR is used to read the current character at the specified location X / Y.



5.0.17 TN_SC_SHADOW_ATTR

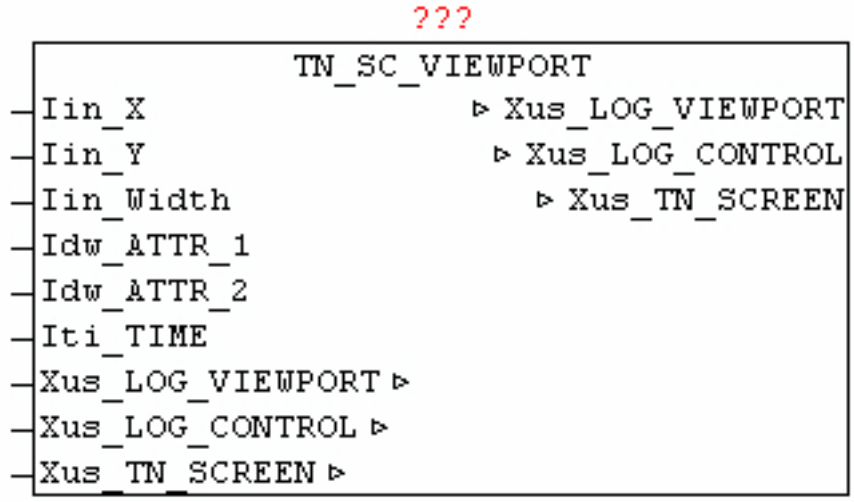
Type Function	BYTE
INPUT Iby_ATTR	BYTE: (Color Information)
	The block TN_SC_SHADOW_ATTR converts a light color to a dark color.



5.0.18 TN_SC_VIEWPORT

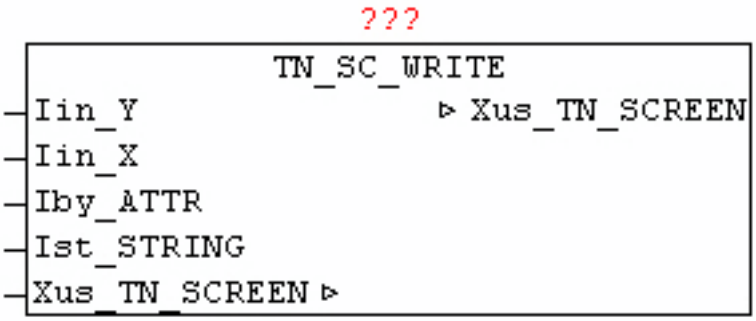
Type	Function module
INPUT Iin_Y	INT: (Y coordinate)
Iin_X	INT: (X coordinate)
Iin_Width	INT: (width of the window - the number of characters)
Idw_ATTR_1	DWORD: (color 1,2,3 and 4)
Idw_ATTR_2	DWORD: (color 5,6,7 and 8)

Iti_TIME	TIME: (update time)
IN_OUT Xus_LOG_VIEWPORT	LOG_VIEWPORT
Xus_LOG_CONTROL	LOG_CONTROL
Xus_TN_SCREEN	us_TN_SCREEN
	The module TN_SC_VIEWPORT is used to display messages from the data structure LOG_CONTROL within a rectangular area on the screen. The desired messages are processed before using with Block LOG_VIEWPORT, and if necessary, with Xus_LOG_VIEWPORT.UPDATE an update is triggered. Means Iin_X and Iin_Y defines the upper-left corner of the window, and with Iin_Width the width if of the viewing window is defined. The number of rows to be displayed is determined by Xus_LOG_VIEWPORT.COUNT. The color information is stored in Xus_LOG_CONTROL.MSG_OPTION [x] per message. It is converted to the configured color codes from Idw_ATTR_1 and Idw_ATTR2 automatically, so the colors in the presentation can always be adjusted individually. The messages are always automatically reduced to the width of the window or cut off.



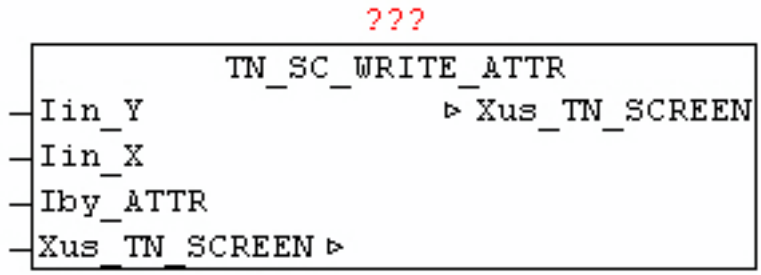
5.0.19 TN_SC_WRITE

Type	Function module
INPUT Iin_Y	INT (Y coordinate)
[fzy] Iin_X	INT: (X coordinate)
Iby_ATTR	BYTE: (color code - font color)
Ist_STRING	STRING (text)
IN_OUT Xus_TN_SCREEN	Us_TN_SCREEN
	The module TN_SC_WRITE passes the text Ist_STRING at the coordinates Iin_Y, Iin_Y and the color of Iby_ATTR.
	Is specified color code = 0, then the string is displayed without change the existing old color information at he respective character positions.



5.0.20 TN_SC_WRITE_ATTR

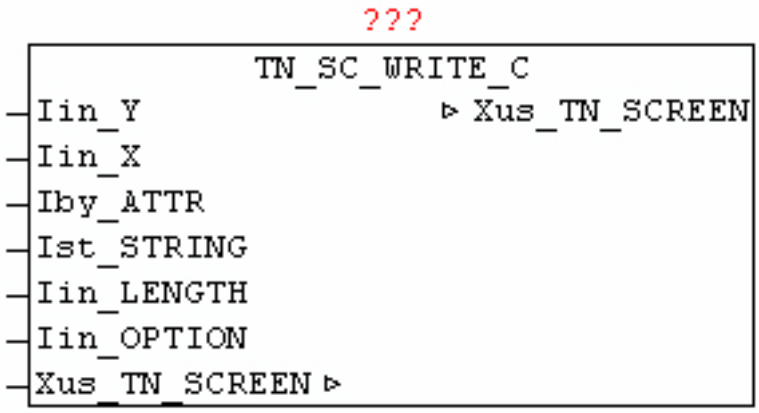
Type	Function module
INPUT Iin_Y	INT (Y coordinate)
[fzy] Iin_X	INT: (X coordinate)
Iby_ATTR	BYTE: (color code)
IN_OUT Xus_TN_SCREEN	Us_TN_SCREEN
	The module TN_SC_WRITE_ATTR changes at the given coordinates Iin_Y, Iin_Y the colorcode to change without changing the existing character at that position.



5.0.21 TN_SC_WRITE_C

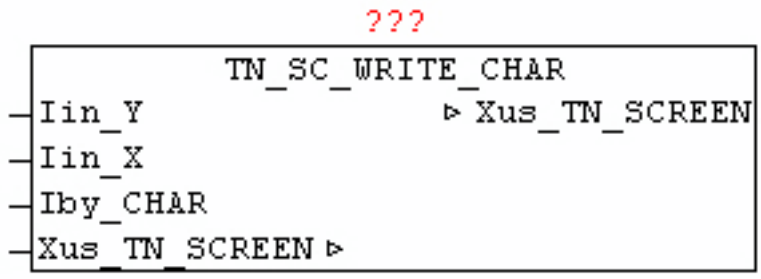
Type	Function module
INPUT Iin_Y	INT (Y coordinate)
[fzy] Iin_X	INT: (X coordinate)
Iby_ATTR	BYTE: (color code)
Ist_STRING	STRING: (text)
Iin_LENGTH	INT: (text will be adjusted to this length)
Iin_OPTION	INT: (option-length adaptation of the text)
IN_OUT Xus_TN_SCREEN	Us_TN_SCREEN
	The module TN_SC_WRITE_C is at the given coordinates Iin_Y, Iin_Y Ist_STRING the text with the color of Iby_ATTR. The text is adapted before output on the length Iin_LENGTH, and by Iin_OPTION, the text position is determined.

	Iin_OPTION
0 = right fill with spaces	eg 'TEST '
1 = left fill with blanks	eg ' TEST '
	2 = center and fill with blanks eg ' TEST '



5.0.22 TN_SC_WRITE_CHAR

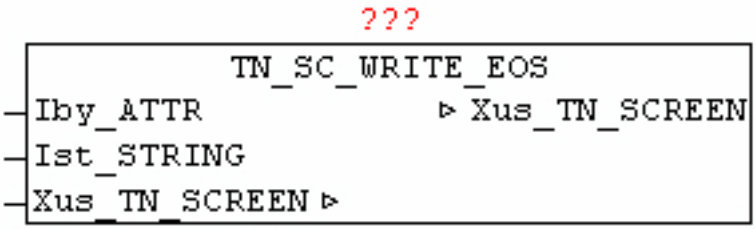
Type	Function module
INPUT Iin_Y	INT: (Y coordinate)
Iin_X	INT: (X coordinate)
OUTPUT Iby_CHAR	BYTE: (sign)
IN_OUT Xus_TN_SCREEN	Us_TN_SCREEN
	The module TN_SC_WRITE_CHAR passes the character Iby_CHAR at the given coordinates Iin_Y, Iin_X, and does not change the color information at the specified position.



5.0.23 TN_SC_WRITE_EOS

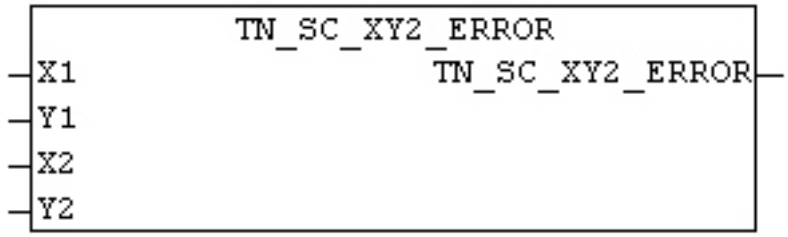
Type	Function module
INPUT Iby_ATTR	BYTE: (color code - font color)
Ist_STRING	STRING (text)
IN_OUT Xus_TN_SCREEN	Us_TN_SCREEN

	The module TN_SC_WRITE_EOS passes at the end position of the last, with TN_SC_WRITE or TN_SC_WRITE_EOS issued text, the text Ist_STRING with the color of Iby_ATTR.
	This allows to continuous passes texts without the need to always pass the new coordinates.



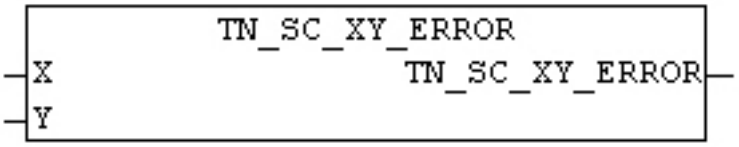
5.0.24 TN_SC_XY2_ERROR

Type Function	BOOL
INPUT	X1: INT: (X1 coordinate of the area)
Y1	INT: (Y1 coordinate of the area)
X2	INT: (X2-coordinate of the area)
Y2	INT: (Y2 coordinate of the area)
	The module TN_SC_XY2_ERROR checks whether the specified coordinates are within the screen area. The area may not cross off the screen. If the check fails, as result is passes TRUE.



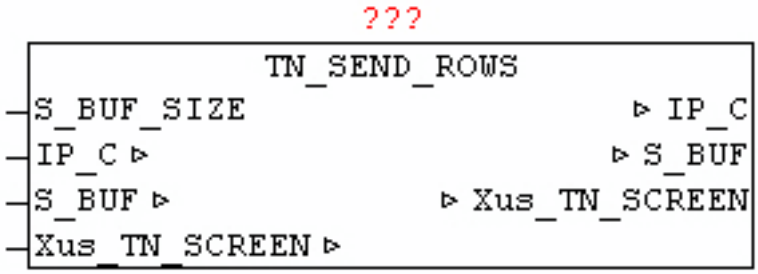
5.0.25 TN_SC_XY_ERROR

Type Function	BOOL
INPUT	X: INT: (X coordinate)
Y	INT: (Y coordinate)
	The module TN_SC_XY_ERROR checks whether the specified coordinate is within the screen area. If the check fails, as result is passes TRUE.



5.0.26 TN_SEND_ROWS

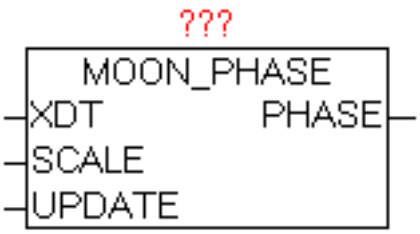
Type	Function module
INPUT S_BUF_SIZE	UINT (number of bytes in S_BUF.BUFFER)
IN_OUT IP_C	IP_CONTROL (Connection data)
S_BUF	NETWORK_BUFFER (transmit data)
Xus_TN_SCREEN	us_TN_SCREEN
	The module TN_SEND_ROWS is used to automatically update the graphical changes to the Telnet screen, by send the modified lines to the Telnet client.
	If you change the Telnet screen a color or a character in a line, this line is always automatically selected for update. The module checks if marked at Xus_TN_SCREEN.bya_Line_Update [0..23] one or more lines, and generates an ANSI-code byte-stream which is sent to the Telnet client. Furthermore, when Xus_TN_SCREEN.bo_Clear_Screen = TRUE a clear screen is triggered. Upon detection of a new Telnet client connection automatically all the rows are marked for update, so that the whole screen content is rendered. If the required amount of data greater than S_BUF.BUFFER the data is automatically output in blocks.



6. WEATHER

6.0.1 MOON_PHASE

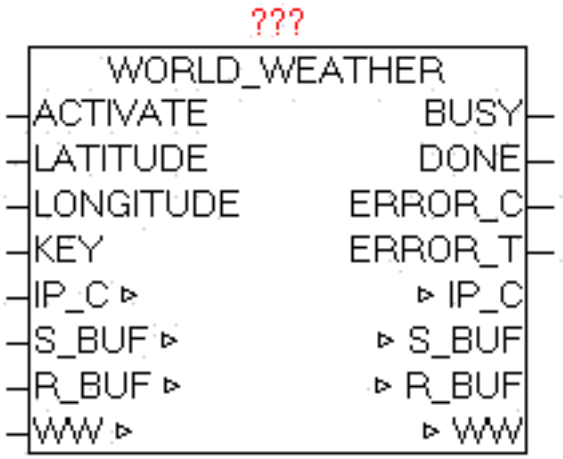
Type Function module	
INPUT XDT	DT (date / time)
SCALE	BYTE (scaling factor)
UPDATE	TIME (update time)
OUTPUT PHASE	BYTE (Scaled value of the lunar phase)
	The module MOON_PHASE is used to calculate the moon phase pf the specified date. At parameter XDT the current date and time is passed, and always recalculated after delay of the time parameter “UPDATE”. The default value for UPDATE is 1 hour and the scaling factor is 12.
A moon phase takes about 29.53 days, and goes through the typical conditions of this new moon to full moon (resp. increasing and decreasing moon). This cycle can be scaled by SCALE to a desired value between 0 and 255. Example	if 100 is given, the moon phase is displayed as a percentage.
	The real length of a single-moon period, is subject to relatively large variations, and thjs is not included in the calculation method used. Thus, you can identify deviations from a few hours. The viewing location (geo-location) is a virtual point in the center of the earth.
	If the moon phase is visualized using graphics, a scaling factor of 12 is used in order to get to the steps 0-11
	See Chapter visualization - Moon Graphics
http	//de.wikipedia.org/wiki/Mondphase



6.0.2 WORLD_WEATHER

Type Function module	
IN_OUT IP_C	IP_C (parameterization)
S_BUF	NETWORK_BUFFER (Transmit data)
R_BUF	NETWORK_BUFFER (Receive data)
WW	WORLD_WEATHER_DATA (Weather data)
INPUT ACTIVATE	BOOL (positive edge starts the query)
LATITUDE	REAL (latitude of the reference location)
LONGITUDE	REAL (longitude of the reference location)
KEY	STRING (30) (API-Key)
OUTPUT BUSY	BOOL (Query is active)
DONE	BOOL (Query completed without errors)
ERROR_C	DWORD (Error code)
ERROR_T	BYTE (error type)
The module loads the current weather data for the specified location of http	//worldweather.com down, analyzes the data and stores the essential data processed in the WORLD_WEATHER_DATA data structure.
	Following values are stored from the current day.
	Observation time (UTC) Temperature (°C), Unique Weather code Weather description text, wind speed in miles per hour, wind speed in kilometer per hour, wind direction in degree, 16-point wind direction compass, precipitation amount in millimeter, Humidity (%), Visibility (km) Atmospheric pressure in milibars , Cloud cover (%)
	From the current day and the next four days the following values are stored.
	Date For which the weather is forecasted, Day and night temperature in °C (Celsius) and °F (Fahrenheit) Wind speed in mph (miles per hour) and kmph (kilometers per hour) 16-point compass wind direction, A unique weather condition code; Weather description text , Precipitation Amount (millimetre)
	With a positive edge of ACTIVATE, the query started and process a DNS query with the following HTTP-GET. After successful receiving all data elements are processed and if necessary stored in the data structure in converted form. By the parameters of latitude and longitude the exact place (geographical position) of the weather is indicated. While the query is active, BUSY = TRUE is passed. After successful completion of the query DONE = TRUE is shown. If an error occurs during the query it is reported in ERROR_C in combination with ERROR_T.
ERROR_T	

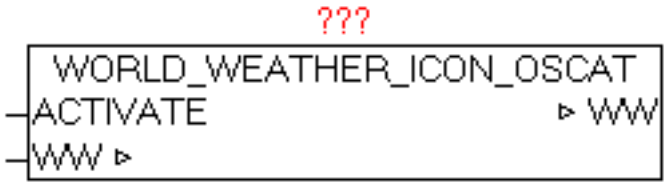
Creating a new API KEY	
Use your Internet browser and call the page http	//www.worldweatheronline.com, call the “Free sign up” registration dialog, and fill out the required fields. After registering, an email is sent, in turn, has to be confirmed, and subsequently secon ad e-mail is sent with the personal API key. This API Key must be passed to the module-KEY API parameters.
Determine Latitude and longitude of a specific place	
Use your Internet browser to access http	//www.mygeoposition.com/ page, and enter the name of the desired location and search the location using “calculate the spatial data” . Then the desired location is shown on the map, including the latitude and longitude needed in decimal notation. The determined position has to be passed to the block parameters, latitude and longitude.



Value	Properties
1	The exact meaning of ERROR_C can be read at module DNS_CLIENT
2	The exact meaning of ERROR_C can be read at module HTTP_GET

6.0.3 WORLD_WEATHER_ICON_OSCAT

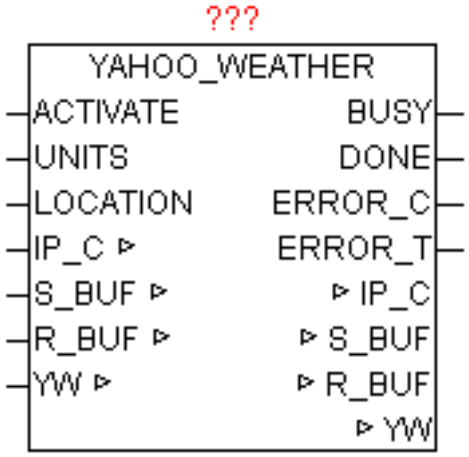
Type Function module	
IN_OUT WW	WORLD_WEATHER_DATA (Weather data)
INPUT ACTIVATE	BOOL (positive edge starts the query)
	The module replaces the original vendor-specific numbers on the weather icons by OSCAT standard icon numbers. Following a positive edge at ACTIVATE the elements (icon numbers) in the WORLD_WEATHER_DATA data structure is replaced. After querying the weather data using WORLD_WEATHER this module should be called subsequently. It is simply the parameter DONE from the module WORLD_WEATHER that is interconnected with ACTIVATE.
The following elements will be adapted	
	WW.WORLD_WEATHER_CUR.WEATHER_ICON
	WW.WORLD_WEATHER_DAY[0..4].WEATHER_ICON

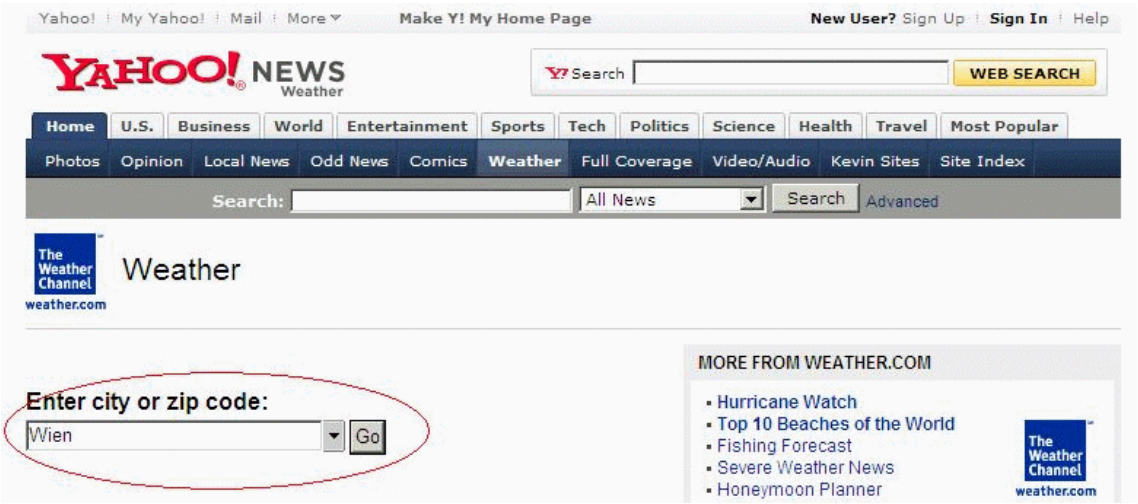


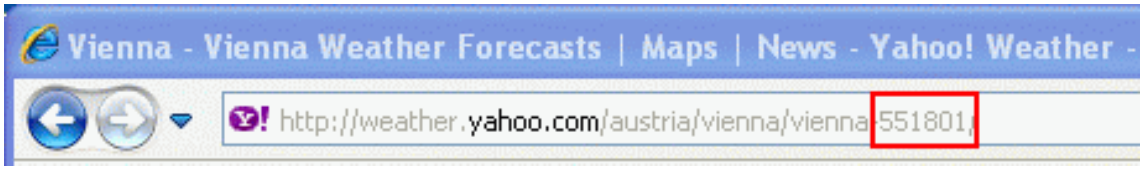
6.0.4 YAHOO_WEATHER

Type Function module	
IN_OUT IP_C	IP_C (parameterization)
S_BUF	NETWORK_BUFFER (Transmit data)
R_BUF	NETWORK_BUFFER (Receive data)
YW	YAHOO_WEATHER (weather data)
INPUT ACTIVATE	BOOL (positive edge starts the query)
UNITS	BOOL (FALSE = Celsius, TRUE = Fahrenheit)
LOCATION	STRING (20) (location specified by LOCATION-ID)
OUTPUT BUSY	BOOL (Query is active)
DONE	BOOL (Query completed without errors)
ERROR_C	DWORD (Error code)
ERROR_T	BYTE (error type)
The module loads the current weather data for the specified location using an RSS feed (XML data structure) of http	//weather.yahooapis.com down, analyzes the XML data and provides the essential data processed from the YAHOO_WEATHER data structure. With a positive edge of ACTIVATE, the query started and process a DNS query with the following HTTP-GET. After successful receipt of

	data by XML_READER all elements are processed and if necessary stored in the data structure in converted form. With UNITS may still be selected between Fahrenheit and Celsius as a unit. By specifying the precise LOCATION_ID the location of the weather is indicated. While the query is active, BUSY = TRUE is passed. After successful completion of the query DONE = TRUE is shown. If occur in the query, then this error is reported under ERROR_C in combination with ERROR_T.
ERROR_T	
Find the Location ID of a specific place	
Use your Internet browser the page http	//weather.yahoo.com/ and in the field: “Enter city or zip code” and enter the name of the desired location and search.
	After being selected in the browser window displays the current weather information of the specified location. In the URL (web link) line is now the location ID can be seen.
	Thus, the desired settlement “Wien (Vienna)” returns the Location ID “551801”. This code must be passed on the module as parameters.







Example:

Example of an RSS feed:
<rss version="2.0" xmlns:yweather="http://weather.yahooapis.com/ns/rss/1.0"
xmlns:geo="http://www.w3.org/2003/01/geo/wgs84_pos#">
*http://weather.yahoo.com/forecast/94089_f.html
Yahoo! Weather for Sunnyvale, CA
en-us
Tue, 29 Nov 2005 3:56 pm PST
60
<yweather:location city="Sunnyvale" region="CA" country="US"></yweather:location>
<yweather:units temperature="F" distance="mi" pressure="in" speed="mph"></yweather:units>
<yweather:wind chill="57" direction="350" speed="7"></yweather:wind>
<yweather:atmosphere humidity="93" visibility="1609" pressure="30.12"rising="0"></
yweather:atmosphere>
<yweather:astronomy sunrise="7:02 am" sunset="4:51 pm"></yweather:astronomy>
142
18
http://us.i1.yimg.com/us.yimg.com/i/us/nws/th/main_142b.gif
geo:lat37.39</geo:lat>
geo:long–122.03</geo:long>
Sunnyvale__CA/*
http://weather.yahoo.com/ forecast/94089_f.html
Tue, 29 Nov 2005 3:56 pm PST
<yweather:condition text="Mostly Cloudy" code="26" temp="57" date="Tue, 29 Nov 2005 3:56
pm PST"></yweather:condition>
<![CDATA[

Current Conditions:
Mostly Cloudy, 57 F
Forecast:
Tue - Mostly Cloudy. High: 62 Low: 45
Wed - Mostly Cloudy. High: 60 Low: 52
Thu - Rain. High: 61 Low: 46
Full Forecast at Yahoo! Weather
(provided by The Weather Channel)]]>
<yweather:forecast day="Tue" date="29 Nov 2005" low="45" high="62" text="Mostly Cloudy"
code="27"></yweather:forecast>
<yweather:forecast day="Wed" date="30 Nov 2005" low="52" high="60" text="Mostly Cloudy"
code="28"></yweather:forecast>
94089_2005_11_29_15_56_PST
The XML data the required elements are processed and stored in the YAHOO_WEATHER data structure.

Value	Properties
1	The exact meaning of ERROR_C can be read at module DNS_CLIENT
2	The exact meaning of ERROR_C can be read at module HTTP_GET

6.0.5 YAHOO_WEATHER_ICON_OSCAT

Type Function module	
IN_OUT YW	YAHOO_WEATHER_DATA (Weather data)
INPUT ACTIVATE	BOOL (positive edge starts the query)
	The module replaces the original vendor-specific numbers on the weather icons by OSCAT standard icon numbers. Following a positive edge at ACTIVATE the elements (icon numbers) in the YAHOO_WEATHER_DATA data structure is replaced. After querying the weather data using YAHOO_WEATHER this module should be called subsequently. It is simply the parameter DONE from the module YAHOO_WEATHER that is interconnected with ACTIVATE.
The following elements will be adapted	
	YW.CUR_CONDITIONS_ICON
	YW.FORCAST_TODAY_ICON
	YW.FORCAST_TOMORROW_ICON