OSCAT Network Dokumentation

Franz Höpfinger

Inhaltsverzeichnis

OSCAT Network Dokumentation

Willkommen zur Dokumentation der OSCAT Network Library! (Dokumentation als PDF herunterladen)

Diese Bibliothek enthält Funktionsbausteine für Netzwerkkommunikation, Protokolle und Internet-Dienste.

1.1 Kategorien

- **NETWORK** - Netzwerkprotokolle und Kommunikation
- **NET_VAR** - Netzwerkvariablen
- **OTHER** - Sonstige Funktionen
- **TELNET_VISION** - Telnet und Visualisierung
- **WEATHER** - Wetterdienste und Datenauswertung

1.2 Lizenz

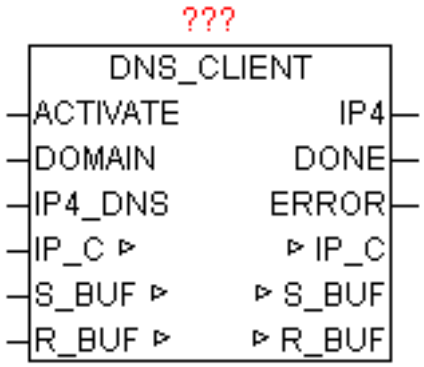
Diese Dokumentation steht unter der Eclipse Public License 2.0.

NETWORK

2.0.1 DNS_CLIENT

Type	Funktionsbaustein	
IN_OUT IP_C	IP_C (Parametrierungsdaten)	
S_BUF	NETWORK_BUFFER (Sendedaten)	
R_BUF	NETWORK_BUFFER (Empfangsdaten)	
INPUT ACTIVATE	BOOL	(Abfrage starten durch positive Flanke)
DOMAIN	STRING	(Domain Name oder IP als String)
IP4_DNS	DWORD (IPv4 Adresse des DNS-Server)	
OUTPUT IP4	DWORD (IPv4 Adresse der angefragten Domain)	
DONE	BOOL (IP der Domain erfolgreich ermittelt)	
ERROR	DWORD (Fehlercode)	
	DNS_CLIENT ermittelt aus den übergebenen qualifizierten DOMAIN Namen die zugehörige IPv4 Adresse z.B. „www.oscat.de“ . Dazu wird eine DNS-Abfrage über den parametrierten DOMAIN Namen bei einen DNS-Server gemacht. Bei positiver Flanke von ACTIVATE wird die angegebene DOMAIN zwischen gespeichert, so dass diese nicht länger vorhanden sein muss. Sollte die Abfrage mehrere IP-Adressen liefern, so wird immer die mit dem höchsten Wert von TTL (Time To Live) benutzt. Als IP4_DNS kann ein beliebiger öffentlicher DNS-Server verwendet werden. Wenn die SPS hinter einen DSL-Router sitzt, kann dieser über seine Gateway-Adresse dieser als DNS-Server verwendet werden. Was letztendlich auch bei wiederkehrenden Abfragen zu schnelleren Antwortzeiten führt, da diese im Router-Cache verwaltet werden. Bei positiven Ergebnis DONE = TRUE enthält IP4 die gesuchte IP-Adresse, bis zum Start der nächsten Abfrage durch positive Flanke von ACTIVATE. Wird im DOMAIN Name eine gültige IPv4 Adresse erkannt,	

	wird logischerweise keine DNS-Abfrage mehr durchgeführt und gleich in konvertierter Form wieder bei IPv4 ausgegeben und DONE auf TRUE gesetzt. ERROR liefert im Fehlerfall die genaue Ursache.	
Fehlercodes		

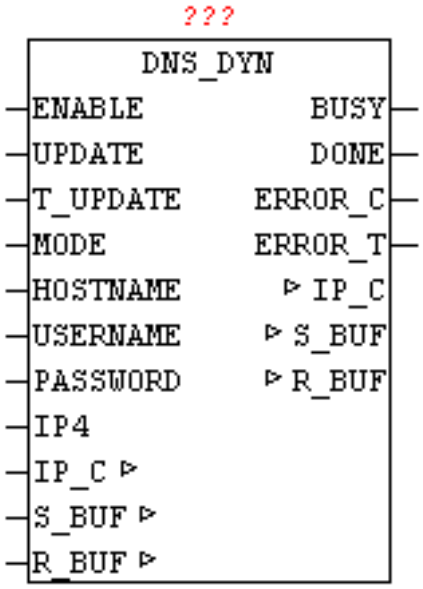


Wert	Ursprung	Beschreibung
B3	B2	B1
nn	nn	nn
xx	xx	xx
xx	xx	xx
xx	xx	xx
xx	xx	xx
xx	xx	xx
xx	xx	xx
xx	xx	xx
xx	xx	xx
xx	xx	xx
xx	xx	xx
xx	xx	xx
xx	xx	xx

2.0.2 DNS_DYN

Type	Funktionsbaustein
INPUT ENABLE	BOOL (Freigabe des Bausteins)
UPDATE	BOOL (Startet neue DNS-Registrierung sofort)
T_UPDATE	TIME (Wartezeit für neue DNS-Registrierung)
MODE	BYTE (Auswahl des DynDNS-Providers)
HOSTNAME	STRING(30) (eigener Domain-Name)

USERNAME	STRING(20) (Name für Registrierung)
PASSWORD	STRING(20) (Passwort für Registrierung)
IP4	DWORD (Optionale Angabe der IP-Adresse)
OUTPUT BUSY	BOOL (Baustein ist aktiv, Abfrage wird durchgeführt)
DONE	BOOL (DNS-Registrierung erfolgreich durchgeführt)
ERROR_C	DWORD (Fehlercode)
ERROR_T	BYTE (Fehlertype)
IN_OUT IP_C	IP_C (Parametrierungsdaten)
S_BUF	NETWORK_BUFFER (Sendedaten)
R_BUF	NETWORK_BUFFER (Empfangsdaten)
	Mittels DNS_DYN können dynamische IP-Adressen als Domain-Name registriert werden. Bei vielen Internet-Providern wird bei Einwahl ins Internet eine dynamische IP-Adresse vergeben. Um von anderen Internet-Teilnehmern trotzdem auffindbar und erreichbar zu sein, ist eine der Möglichkeiten, das man seine aktuelle IP-Adresse mittels Dyn-DNS immer aktualisiert. Das Verfahren ist leider nicht standardisiert, somit ist für jeden Dyn-DNS Provider ein spezielle Lösung zu erstellen. Der Baustein kann in Verbindung mit DynDNS.org und Selfhost.de eingesetzt werden. Diese Provider bieten neben kostenpflichtigen auch kostenlose Dyn-DNS Dienste an.
	Wird ENABLE auf TRUE gesetzt, dann wird der Baustein aktiv. Mittels einer positiven Flanke auf UPDATE kann jederzeit eine Aktualisierung gestartet werden. Wenn bei T_UPDATE eine Zeit angegeben ist, so wird immer nach Ablauf dieser Zeit auch eine Aktualisierung durchgeführt.
	Achtung , die meisten DynDNS Provider bewerten eine zu häufiges bzw. unnötiges Updaten als Angriff, und blockieren eventuell den Account für eine gewisse Zeit.
	Die Zeit T_UPDATE sollte nicht unter einer Stunde eingestellt sein. Wird der Parameter T_UPDATE nicht beschalten so wird als Updatezeit 1 Stunde angenommen. Wird kein Updaten über Zeit benötigt, dann sollte T#0ms übergeben werden.
	Der Parameter MODE ermöglicht die Auswahl der DynDNS-Providers
	(0 = DynDNS.org , 1 = SELFHOST.DE)
	Der eigene Domainname muss bei HOSTNAME übergeben werden. Zur Sicherheit muss man beim DynDNS-Provider mittels USERNAME und PASSWORD die Authorisierungs-Daten angeben. Wird der Parameter IP4 nicht belegt, so wird vom DynDNS Provider automatisch die aktuelle WAN-IP mit der die Aktualisierung durchgeführt wird, als aktuelle Registrierung-IP angenommen. Durch Angabe von eine IP-Adresse kann aber auch eine individuelle IP-Adresse angegeben werden.
	Bei fehlerfreier Durchführung wird DONE = TRUE , ansonsten wird an ERROR_C und ERROR_T der Fehlercode und der Fehlertype ausgegeben.(Siehe Fehlercodes).
ERROR_T	

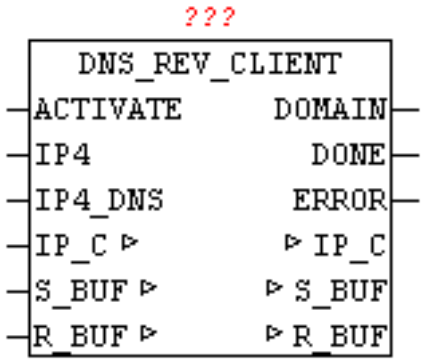


Wert	Eigenschaften
1	Die genaue Bedeutung von ERROR_C ist beim Baustein DNS_CLIENT nachzulesen
2	Die genaue Bedeutung von ERROR_C ist beim Baustein HTTP_GET nachzulesen
3	Der DynDNS Provider hat die Registrierung abgelehnt

2.0.3 DNS_REV_CLIENT

Type	Funktionsbaustein	
IN_OUT IP_C	IP_C (Parametrierungsdaten)	
S_BUF	NETWORK_BUFFER (Sendedaten)	
R_BUF	NETWORK_BUFFER (Empfangsdaten)	
INPUT ACTIVATE	BOOL (Abfrage starten durch positive Flanke)	
DOMAIN	STRING	(Domain Name oder IP als String)
IP4_DNS	DWORD (IPv4 Adresse des DNS-Server)	
OUTPUT IP4	DWORD (IPv4 Adresse der angefragten Domain)	
DONE	BOOL (IP der Domain erfolgreich ermittelt)	
ERROR	DWORD (Fehlercode)	
	DNS_REV_CLIENT ermittelt aus den übergebenen IP-Adresse den offiziell registrierten DOMAIN Namen . Dazu wird eine Reverse DNS-Abfrage über die parametrierte IP-Adresse bei einen DNS-Server gemacht. Bei positiver Flanke von ACTIVATE wird die angegebene IP zwischen gespeichert, so dass diese nicht	

	länger vorhanden sein muss. Sollte die Abfrage mehrere Ergebnisse liefern, so wird immer der letzte Datensatz herangezogen. Als IP4_DNS kann ein beliebiger öffentlicher DNS-Server verwendet werden. Wenn die SPS hinter einen DSL-Router sitzt, kann dieser über seine Gateway-Adresse dieser als DNS-Server verwendet werden. Was letztendlich auch bei wiederkehrenden Abfragen zu schnelleren Antwortzeiten führt, da diese im Router-Cache verwaltet werden. Bei positiven Ergebnis DONE = TRUE enthält DOMAIN den offiziell registrierten Domain-Namen, bis zum Start der nächsten Abfrage durch positive Flanke von ACTIVATE. ERROR liefert im Fehlerfall die genaue Ursache. (Siehe Fehlercodes).	
Fehlercodes		

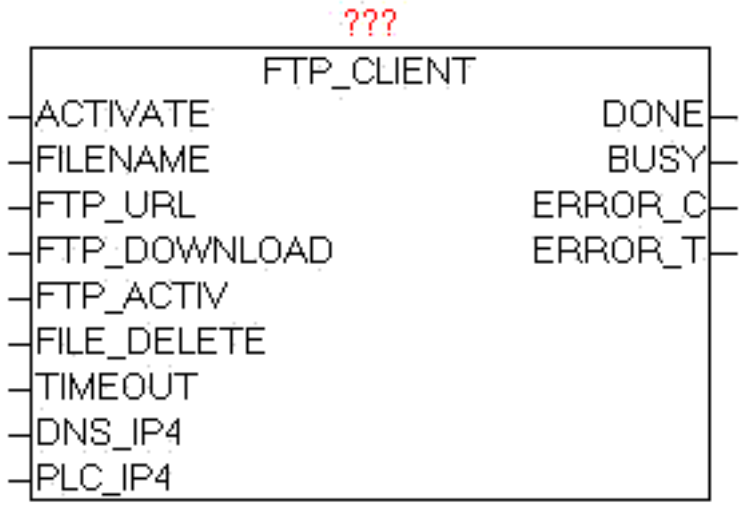


Wert	Ursprung	Beschreibung
B3	B2	B1
nn	nn	nn
xx	xx	xx
xx	xx	xx
xx	xx	xx
xx	xx	xx
xx	xx	xx
xx	xx	xx
xx	xx	xx
xx	xx	xx
xx	xx	xx
xx	xx	xx
xx	xx	xx

2.0.4 FTP_CLIENT

Type Funktionsbaustein	
INPUT ACTIVATE	BOOL (positive Flanke startet die Abfrage)
FILENAME	STRING (Dateipfad / Dateiname)
FTP_URL	STRING(STRING_LENGTH) (FTP Zugriffspfad)
FTP_DOWNLOAD	BOOL (UPLOAD = 0 / DOWNLOAD = 1)
FTP_ACTIV	BOOL (PASSIV = 0 / ACTIV = 1)
FILE_DELETE	BOOL (Datei nach Übertragung löschen)
TIMEOUT	TIME (Zeitüberwachung)
DNS_IP4	DWORD (IP4-Adresse des DNS-Server)
PLC_IP4	DWORD (IP4-Adresse der eigenen Steuerung)
OUTPUT DONE	BOOL (Transfer ohne Fehler beendet)
BUSY	BOOL (Transfer ist aktiv)
ERROR_C	DWORD (Fehlercode)
ERROR_T	BYTE (Fehlertype)
Der Baustein FTP_CLIENT dient zum übertragen von Dateien von der SPS zu einen FTP-Server, und zum übertragen vom FTP-Server zur SPS. Mittels positiver Flanke bei ACTIVATE wird der Übertragungsvorgang gestartet. Bei FTP_DOWNLOAD kann die Übertragungsrichtung vorgegeben werden. Der Parameter FTP_URL enthält den Namen des FTP-Server und optional den Benutzernamen und das Passwort, einen Zugriffpfad und eine zusätzliche Portnummer für den Datenkanal.Wird kein Benutzername bzw. Passwort übergeben, so versucht der Baustein sich automatisch als „Anonymous“ anzumelden. Der Parameter FTP_ACTIV bestimmt ob der FTP-Server im AKTIV oder PASSIV Modus betrieben wird. Im ACTIV Modus versucht der FTP-Server den Datenkanal zur Steuerung aufzubauen, dabei kann es jedoch durch Sicherheitssoftware, Firewall etc. zu Problemen kommen , da diese die Verbindungswunsch blockieren könnten. Hierzu müsste in der Firewall eine dementsprechende Ausnahmeregel definiert werden. Beim PASSIV Modus wird dieses Problem entschärft, da die Steuerung die Verbindung aufbaut, und somit problemlos die Firewall passieren kann. Der Steuerkanal wird immer über PORT 20 aufgebaut, und der Datenkanal standardmäßig über PORT21, dies ist aber wiederum abhängig ob ACTIV oder PASSIV Mode genutzt wird, bzw. optional eine PORT-Nummer in der FTP-	Speicher auf SPS, FTP-Speicher und der Faktor Übertragungszeit. Bei DNS_IP4 muss die IP-Adresse des DNS-Servers angegeben werden, wenn in der FTP-URL ein DNS-Name angegeben wird, alternativ kann in der FTP-URL auch eine IP-Adresse eingetragen werden. Bei Parameter PLC_IP4 muss die eigene IP-Adresse angegeben werden. Sollten bei der Übertragung Fehler auftreten werden diese bei ERROR_C und ERROR_T ausgegeben. Solange die Übertragung läuft ist BUSY = TRUE, und nach fehlerfreien Abschluss des Vorgangs wird DONE = TRUE. Sobald ein neuer Übertragungsvorgang gestartet wird, werden DONE,ERROR_T und ERROR_C rückgesetzt.

URL angegeben wurde. Mit dem Parameter FILE_DELETE kann bestimmt werden, ob die Quell-Datei nach erfolgreicher Übertragung gelöscht werden soll. Dies funktioniert auf FTP als auch auf der Steuerungsseite. Bei Vorgabe von FTP-Verzeichnissen ist das Verhalten FTP-Server abhängig, ob hierbei diese schon existieren müssen, oder automatisch angelegt werden. Im Normalfall sollten diese schon vorhanden sein. Die Größe der Dateien ist an sich unbegrenzt, jedoch gibt es hier praktische Limits	
	Der Baustein hat den IP_CONTROL integriert und muss somit nicht mehr extern mit diesen verknüpft werden, so wie dies normalerweise notwendig wäre.
Grundlagen	http://de.wikipedia.org/wiki/File_Transfer_Protocol
URL-Beispiele	
ftp	//benutzername:password@servername:portnummer/verzeichnis/
ftp	//benutzername:password@servername
ftp	//benutzername:password@servername/verzeichnis/
ftp	//servername
ftp	//benutzername:password@192.168.1.1/verzeichnis/
ftp	//192.168.1.1
ERROR_T	



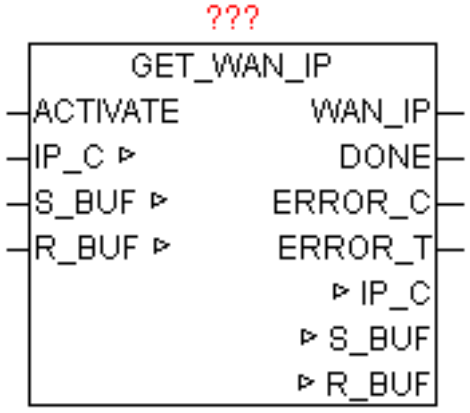
Wert	Eigenschaften
1	Störung: DNS_CLIENTDie genaue Bedeutung von ERROR_C ist beim Baustein DNS_CLIENT nachzulesen

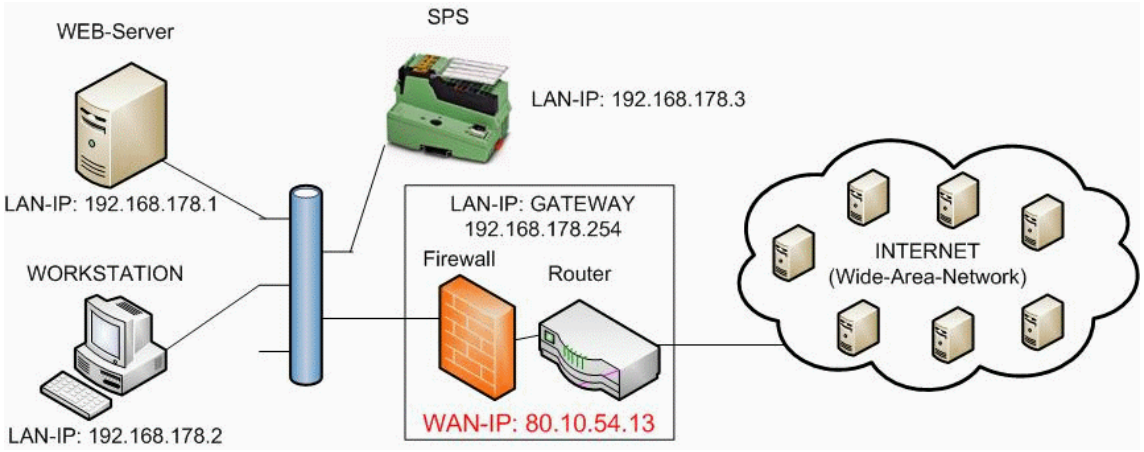
2	Störung: FTP SteuerkanalDie genaue Bedeutung von ERROR_C ist beim Baustein IP_CONTROL nachzulesen
3	Störung: FTP DatenkanalDie genaue Bedeutung von ERROR_C ist beim Baustein IP_CONTROL nachzulesen
4	Störung: FILE_SERVERDie genaue Bedeutung von ERROR_C ist beim Baustein FILE_SERVER nachzulesen
5	Störung: ABLAUF – TIMEOUTERROR_C enthält im linken WORD die Ablauf-Schrittnummer, und im rechten WORD den zuletzt vom FTP-Server empfangenen Response-Code.Achtung der Parameter muss zuerst als HEX-Wert betrachtet, in zwei WORDS geteilt werden,und dann als Dezimalzahl betrachtet werden. Beispiel:ERROR_T = 5ERROR_C = 0x0028_00DCAblauf-Schrittnummer 0x0028 = 40Response-Code 0x00DC = 220

2.0.5 Type Funktionsbaustein:

INPUT ACTIVATE	BOOL (Freigabe zur Abfrage)
OUTPUT	WAN_IP: DWORD (Wide-Area-Network-Adresse)
DONE	BOOL (Abfrage ohne Fehler beendet)
ERROR_C	DWORD (Fehlercode)
ERROR_T	BYTE (Fehlertype)
	Der Baustein stellt die IP-Adresse fest, die der Internet-Routers im Wide-Area-Network (Internet) benutzt. Diese IP-Adresse ist z.B. notwendig um DynDNS Anmeldungen durchführen zu können. Mit einer positiven Flanke von ACTIVATE wird die Abfrage gestartet. Nach erfolgreich beendeter Abfrage wird DONE=TRUE ausgegeben, und bei Parameter WAN_IP wird die aktuelle WAN-IP Adresse ausgegeben. Sollte bei der Abfrage ein Fehler auftreten so wird dieser unter ERROR_C gemeldet in Kombination mit ERROR_T.
ERROR_T	

IN_OUT IP_C : Datenstruktur ‚IP_CONTROL' (Parametrierungsdaten)
S_BUF: Datenstruktur ‚NETWORK_BUFFER' (Sendedaten)
R_BUF: Datenstruktur ‚NETWORK_BUFFER' (Empfangsdaten)





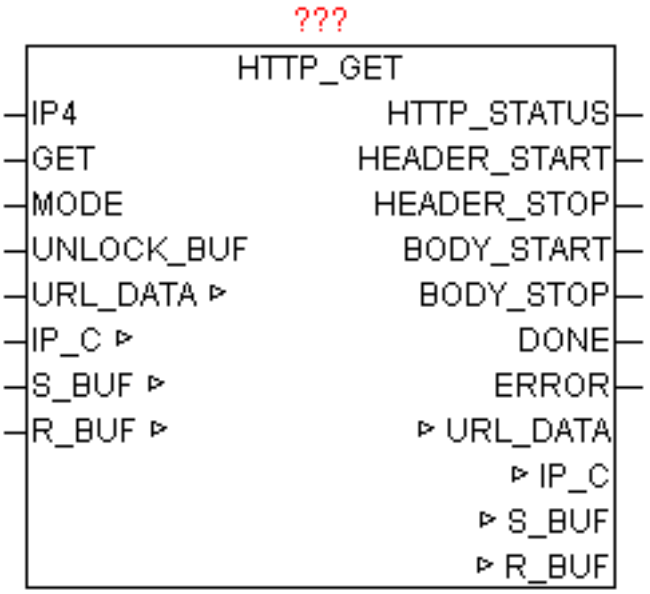
Wert	Eigenschaften
1	Die genaue Bedeutung von ERROR_C ist beim Baustein DNS_CLIENT nachzulesen
2	Die genaue Bedeutung von ERROR_C ist beim Baustein HTTP_GET nachzulesen

2.0.6 HTTP_GET

Type Funktionsbaustein	
IN_OUT URL_DATA	URL (Daten von STRING_TO_URL)
IP_C	IP_C (Parametrierungsdaten)
S_BUF	NETWORK_BUFFER (Sendedaten)
R_BUF	NETWORK_BUFFER (Empfangsdaten)
INPUT IP4	DWORD (IP-Adresse des HTTP-Servers)
GET	BOOL (Startet die HTTP Abfrage)
MODE	BYTE (Version der HTTP-GET Abfrage)
UNLOCK_BUF	BOOL (Freigabe des Empfangs-Datenbuffers)
OUTPUT HTTP_STATUS	STRING (ermittelter HTTP Statuscode)
HTTP_START	UINT (Start-Position des Message-Headers)
HTTP_STOP	UINT (Stopp-Position des Message-Headers)
BODY_START	UINT (Start-Position des Message-Body)
BODY_STOP	UINT (Stopp-Position des Message-Body)
DONE	BOOL (Aufgabe ohne Fehler durchgeführt)
ERROR	DWORD (Fehlercode)
	HTTP_GET führt nach positiver Flanke von GET ein GET-Kommando auf einem HTTP-Server durch. Mittels MODE kann die HTTP Protokollversion vorgegeben werden. Die gewünschte URL (Web-Link) muss vorab in der URL_DATA Struktur fertig aufbereitet vorliegen. Die vollständige URL sollte darum vor dem Bausteinaufruf mittels „STRING_TO_URL“ aufbereitet werden.

	Nach erfolgreicher Abfrage wird DONE = TRUE ausgegeben, und die Parameter HTTP_START und HTTP_STOP zeigen auf den Datenbereich in dem die Message-Header Daten zur weiteren Bearbeitung und Auswertung vorzufinden sind. Im Normalfall ist auch ein Message-Body vorhanden, der wiederum mittels BODY_START und BODY_STOP übermittelt wird. Weiters wird auf HTTP_STATUS der zurückgemeldete HTTP-Statuscode als String ausgegeben. Eine der Schwierigkeiten beim HTTP Empfang ist das erkennen vom Ende des Datenstream. Dabei werden vom Baustein mehrere Strategien verfolgt. Bei Anwendung von HTTP/1.0 wird das Ende des Datenempfangs über einen Verbindungsabbau seitens des Host erkannt. Weiters wird immer geprüft ob sich im Header die Info „Content-Length“ befindet, damit kann dann eindeutig erkannt werden ob alle Daten empfangen wurden. Trifft keines der vorigen Varianten zu, so wird über einen einfachen Receive-Timeout Error das Ende der Datenübertragung festgestellt und beendet. Einziger Nachteil dabei ist, das dies je nach eingestellte Timeout Zeit mitunter länger dauert als gewünscht. Darum ist es nicht schlecht wenn eine vernünftiger Timeout-Wert am IP_CONTROL gewählt wird. ERROR liefert im Fehlerfall die genaue Ursache (Siehe Baustein IP_CONTROL).
Das „Chunked transfer encoding“ wird vom Baustein nicht unterstützt (https	//en.wikipedia.org/ wiki/Chunked_transfer_encoding)
HTTP-Authentifizierung	
	Der Baustein unterstützt die Authentifizierung über Basic Authentication
wie z.B. benutzername	password@serveradresse/login/main.php
	SSL/TLS Verschlüsselung
	SSL steht für Secure Socket Layer
	TLS bedeutet Transport Layer Security
	SSL
	Die Kommunikation zwischen Client und Server beginnt sofort verschlüsselt. Der meistens benutzte Standardport ist Port 443
	Hinweis !
	Die SSL/TLS Verschlüsselung wird aktuell nur für Steuerungen von Fa. Phoenix Contact unterstützt. Um diese Funktionalität nutzen zu können muss die SPS eine Firmware haben die SSL/TLS unterstützt.
	Für die Aktivierung muss dieURL Adresse mit HTTPS beginnen
	z.B.

https	//benutzername:password@serveradresse/login/main.php
Https	//serveradresse/login/main.php
HTTPS	//serveradresse.at
Folgende MODE können verwendet werden	

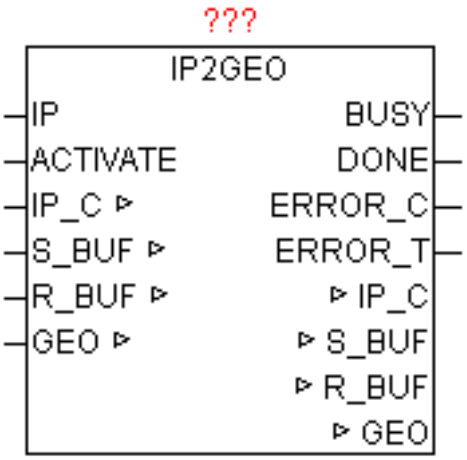


Mode	Protokollversion	Eigenschaften
0	HTTP/1.0	Der Host beendet selbstständig nach Übertragung der Daten die TCP-Verbindung
1	HTTP/1.0	Durch Anwendung von „Connection: Keep-Alive“ wird der Host angewiesen eine persistente Verbindung zu verwenden. Der Client sollte nach Ende der Aktivitäten die Verbindung wieder abbauen.
2	HTTP/1.1	Der Host verwendet eine persistente Verbindung und muss von Client beendet werden.
3	HTTP/1.1	Durch Anwendung von „Connection: Close“ wird der Host angewiesen nach Übertragung der Daten die TCP-Verbindung zu beenden.

2.0.7 IP2GEO

Type Funktionsbaustein	
IN_OUT IP_C	IP_C (Parametrierungsdaten)

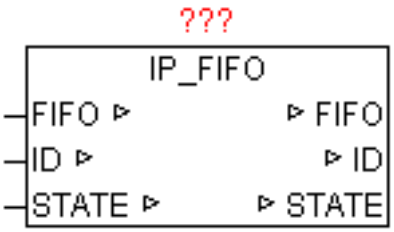
S_BUF	NETWORK_BUFFER (Sendedaten)
R_BUF	NETWORK_BUFFER (Empfangsdaten)
GEO	IP2GEO (Geographische Daten)
INPUT ACTIVATE	BOOL (Freigabe zur Abfrage)
OUTPUT BUSY	BOOL (Abfrage ist aktiv)
DONE	BOOL (Abfrage ohne Fehler beendet)
ERROR_C	DWORD (Fehlercode)
ERROR_T	BYTE (Fehlertype)
	Der Baustein liefert aufgrund der Wide-Area-Network IP-Adresse die Geographischen Informationen des Internet-Zuganges. Da die WAN-IP-Adressen weltweit registriert sind, lässt sich somit annähernd die Geographische Position der SPS bestimmen. Sollte der Zugang über einen Proxy-Server laufen, so wird die Geographische Position von diesen ermittelt und nicht von der SPS. Normalerweise ist der aber in unmittelbarer Nähe der SPS, und somit die Abweichung nicht relevant. Somit ergeben sich ermittelte Positionen die nur ein paar Kilometer von der wirklichen Position abweichen, und relativ genau sind.
	Wird am Parameter „IP“ keine IP-Adresse angegeben, so wird automatisch die aktuelle WAN-IP herangezogen, andernfalls werden die Informationen der parametrierten IP geliefert. Mit einer positiven Flanke von ACTIVATE wird die Abfrage gestartet. Solange die Abfrage nicht beendet ist, wird BUSY=TRUE ausgegeben. Nach erfolgreich beendeter Abfrage wird DONE=TRUE ausgegeben, und bei Parameter WAN_IP wird die aktuelle WAN-IP Adresse ausgegeben. Sollte bei der Abfrage ein Fehler auftreten so wird dieser unter ERROR_C gemeldet in Kombination mit ERROR_T.
ERROR_T	
Der „COUNTRY_CODE ist nach „ISO 3166 Country-Code ALPHA-2“ kodiert. http	//www.iso.org/iso/english_country_names_and_code_elements http://de.wikipedia.org/wiki/ISO-3166-1-Kodierliste
Der „REGION_CODE“ ist nach „FIPS Region Code“ kodiert. http	//en.wikipedia.org/wiki/List_of_FIPS_region_codes



Wert	Eigenschaften
1	Die genaue Bedeutung von ERROR_C ist beim Baustein DNS_CLIENT nachzulesen
2	Die genaue Bedeutung von ERROR_C ist beim Baustein HTTP_GET nachzulesen

2.0.8 IP_FIFO

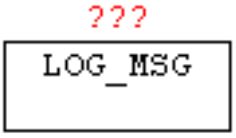
Type Funktionsbaustein	
IN_OUT FIFO	IP_FIFO_DATA (IP-FIFO Verwaltungsdaten)
ID	BYTE (aktuelle vom IP_FIFO-Baustein vergebene ID)
STATE	BYTE (Steuerbefehle und Statusmeldungen)
	Dieser Baustein dient in Kombination mit IP_CONTROL zur Ressourcen-Verwaltung. Damit ist es möglich das Client-Applikation die alleinigen Zugriffsrechte anfordern und auch wieder abgeben können. Durch das FIFO wird gewährleistet das jeder Teilnehmer gleich oft den Ressourcen-Zugriff zugeteilt bekommt.
	Bei ersten Aufruf des Bausteins wird automatisch eine neue eindeutige Applikation-ID vergeben, mittels der dann die Verwaltung im FIFO durchgeführt wird. Der Parameter STATE wird von der Applikation als auch vom IP_FIFO Baustein verändert. Jede Applikation kann sich in der Standardeinstellung nur einmal im FIFO eintragen.
STATE	
Ablauf	
	1. Applikation setzt STATE auf 1
	2. Zugriffsrechte werden angefordert, während dessen ist STATE=2
	3. wenn Ressource frei ist, und Zugriffsrechte vorhanden sind, dann ist STATE=3
	4. Wenn die Applikation die Ressource bzw. den Zugriff nicht mehr benötigt,wird von der Applikation STATE auf 4 gesetzt. Danach gibt IP_FIFO die Ressource wieder frei und setzt STATE auf 5.
	5. Vorgang wiederholt sich (gleicher oder anderer Applikation)
	Anwendungsbeispiel ist beim Baustein IP_CONTROL zu finden !



Wert	Zustandsmeldung
0	Keine Aktion
1	Zugriffsrecht anfordern
2	Zugriffsrecht anfordern wurde in FIFO übernommen
3	Zugriffsrecht erhalten (Ressourcen-Zugriff erlaubt)
4	Zugriffsrecht abgeben
5	Zugriffsrecht wurde aus FIFO wieder entfernt

2.0.9 LOG_MSG

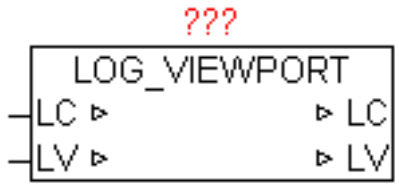
Type Funktionsbaustein	
IN_OUT LOG_CL	LOG_CONTROL (LOG-Daten)
	Mit LOG_MSG können Meldungen (STRINGS) in einen RING-BUFFER abgelegt werden. Die Meldungen können mit Zusatzparameter versehen werden wie Frontcolor und Backcolor für die Ausgabe auf TELNET sowie ein Eintragsfilter durch vorgabe eines Nachrichten-Level. Ist der Level der Nachricht grösser als der Vorgabe-Log-Level , so wird diese Nachricht verworfen. Weiters kann mit Enable auch das Logging allgemein deaktiviert werden. Somit ist es kein Problem viele Meldungen pro SPS-Zyklus zu archivieren. Der Nachrichtenbuffer kann z.B. mit dem Baustein TELNET_LOG auf einen TELNET-CLIENT ausgegeben werden. Details zur Schnittstelle sind in der nachfolgenden Tabelle ersichtlich.
	Wenn aus verschiedenen Bausteininstanzen Meldungen in den selben LOG_BUFFER eingetragen werden sollen, so muss die „LOG_CL“ Datenstruktur GLOBAL angelegt werden.



2.0.10 LOG_VIEWPORT

Type	Funktionsbaustein
IN_OUT LC	LOG_CONTROL
LV	us_LOG_VIEWPORT

Der Baustein LOG_VIEWPORT dient zum erzeugen einer Indexliste der LOG_CONTROL Meldungen die sich aktuell in der virtuellen Ansicht befinden. Um sich innerhalb der Meldungsliste zu bewegen (scrollen) kann über LV.MOVE_TO_X ein Scroll-offset angegeben werden. Ein positiver Wert scrollt in richtig neuere Meldungen und ein negativer Wert in richtig der älteren Meldungen. Die Anzahl der Zeilen der Meldungsliste der virtuellen Ansicht wird über LV.COUNT vorgegeben. Die in der aktuellen virtuellen Ansicht sich befindenden Meldungen werden in LV.LINE_ARRAY[x] abgelegt, und stehen für die weitere Verarbeitung zu Verfügung. Eine Veränderung der Meldungsliste wird immer mit LV.UPDATE	= TRUE signalisiert, und muss vom Anwender wieder rückgesetzt werden.
	Folgende LV.MOVE_TO_X Werte erzeugen ein spezielles Verhalten.
	+30000 = älteste Meldungen anzeigen (Anfang des Ringbuffer)
	+30001 = neueste Meldungen anzeigen (Ende des Ringbuffer)
	+30002 = eine ganze Seite in Richtig neuere Meldungen scrollen
	+30003 = eine ganze Seite in Richtig ältere Meldungen scrollen

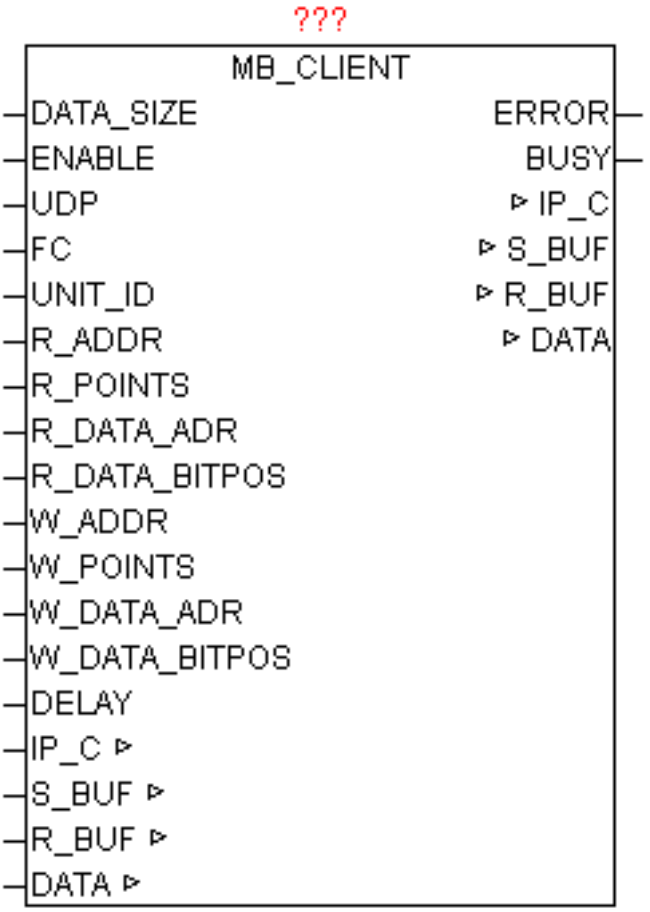


2.0.11 MB_CLIENT (OPEN-MODBUS)

Type Funktionsbaustein	
IN_OUT IP_C	IP_C (Parametrierungsdaten)
S_BUF	NETWORK_BUFFER_SHORT (Sendedaten)
R_BUF	NETWORK_BUFFER_SHORT (Empfangsdaten)
DATA	ARRAY [0..255] OF WORD (MODBUS-Register)
INPUT DATA_SIZE	INT (Anzahl der Datenwörter in Struktur MB_DATA)
ENABLE	BOOL (Freigabe)
UDP	BOOL (Vorwahl TCP / UDP, TRUE = UDP)
FC	INT (Funktionsnummer)

UNIT_ID	BYTE (Geräte-Identifikationsnummer)
R_ADDR	INT (Lesebefehl: MODBUS Datenpunkt Adresse)
R_POINTS	INT (Lesebefehl: MODBUS Anzahl der Datenpunkte)
R_DATA_ADR	INT (Lesebefehl: DATA Datenpunkt Adresse)
R_DATA_BITPOS	INT (Lesebefehl: DATA Datenpunkt Bitpos.)
W_ADDR	INT (Lesebefehl: MODBUS Datenpunkt Adresse)
W_POINTS	INT (Lesebefehl: MODBUS Anzahl der Datenpunkte)
W_DATA_ADR	INT (Lesebefehl: DATA Datenpunkt Adresse)
W_DATA_BITPOS	INT (Lesebefehl: DATA Datenpunkt Bitpos.)
DELAY	TIME (Wiederholungszeit)
OUTPUT ERROR	DWORD (Fehlercode)
BUSY	BOOL (Baustein ist aktiv)
	Der Baustein ermöglicht den Zugriff auf Ethernet-Geräte die MODBUS-TCP bzw. MODBUS-UDP unterstützten, bzw. MODBUS RS232/485 Geräte die über Ethernet-Modbus-Gateway angebunden sind. Es werden Kommandos aus den Klassen 0,1,2 unterstützt. Die Parameter IP_C, S_BUF, R_BUF bilden hierbei die Schnittstelle zum IP_CONTROL Baustein der hier zur Abwicklung bzw. Koordinierung benutzt wird. Die gewünschte IP-Adresse und Port-Nummer (Standard für MODBUS ist 502) muss am IP_CONTROL zentral angegeben werden. Die DATA-Struktur ist als WORD-Array ausgeführt und beinhaltet die MODBUS-Daten für Lesen und Schreiben. Mittels DATA_SIZE wird die Größe des WORD_ARRAY angegeben. Durch ENABLE wird der Baustein freigegeben, und bei Wegnahme der Freigabe wird eine eventuell aktive Abfrage noch beendet. Bei Geräten die MODBUS-UDP unterstützen kann mittels UDP=TRUE dieser Modus aktiviert werden. Der Parameter UNIT_ID muss nur bei Nutzung eines Ethernet-Modbus-Gateway angegeben werden. Die gewünschte Funktion wird mittels FC angegeben (Siehe Funktionscode-Tabelle). Je nach Funktion müssen die R_xxx und W_xxx Parameter mit Daten versorgt werden. Durch Angabe DELAY kann die die Wiederholungszeit vorgegeben werden. Wird keine Zeit angeben so wird versucht so oft wie möglich das Kommando auszuführen. Durch die integrierte Zugriffsverwaltung ist automatisch sichergestellt, das andere Baustein-Instanzen auch an die Reihe kommen. Eine negative Befehlsausführung wird mittels ERROR gemeldet (Siehe ERROR-Tabelle).Wenn der Baustein aktiv eine Abfrage durchführt, dann wird während dieser Zeit BUSY=TRUE ausgegeben.

Unterstützte Funktionscodes und verwendete Parameter	
ERROR	



Funktionscode	Bit Zugriff	16 Bit Zugriff (Register)	Gruppe	Funktion Beschreibung	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	Read In-	x	x	x					

			Register	put Register							
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	

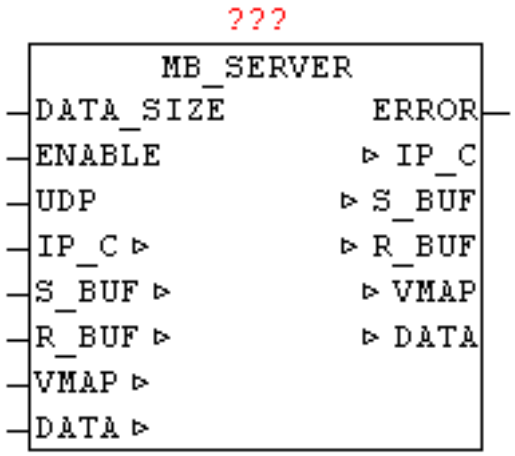
Wert	Ursprung	Beschreibung
B3	B2	B1
nn	nn	nn
xx	xx	xx
xx	xx	xx
xx	xx	xx
xx	xx	xx
xx	xx	xx

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

2.0.12 MB_SERVER (OPEN-MODBUS)

Type Funktionsbaustein	
IN_OUT IP_C	IP_C (Parametrierungsdaten)
S_BUF	NETWORK_BUFFER_SHORT (Sendedaten)
R_BUF	NETWORK_BUFFER_SHORT (Empfangsdaten)
VMAP	ARRAY [1..10] OF VMAP_DATA (Virtuelle Adresstabellen)
DATA	ARRAY [0..255] OF WORD (MODBUS-Register)
INPUT DATA_SIZE	INT (Anzahl der Datenwörter in DATA)
ENABLE	BOOL (Freigabe)
UDP	BOOL (Vorwahl TCP / UDP, TRUE = UDP)
OUTPUT ERROR	DWORD (Fehlercode)
	Der Baustein ermöglicht den Zugriff von Extern auf lokale MODBUS-Datentabellen über Ethernet . Es werden Kommandos der Klassen 0,1,2 unterstützt. Die Parameter IP_C, S_BUF, R_BUF bilden hierbei die Schnittstelle zum IP_CONTROL Baustein der hier zur Abwicklung bzw. Koordinierung benutzt wird. Die gewünschte Port-Nummer (Standard für MODBUS ist 502) muss am IP_CONTROL zentral angegeben werden. Die IP-Adresse wird am IP_CONTROL nicht benötigt, da dieser hier im Modus PASSIV agiert. Die DATA-Struktur ist als WORD-Array ausgeführt und beinhaltet die MODBUS-Daten. Mittels DATA_SIZE wird die Größe von DATA angegeben. Durch ENABLE wird der Baustein freigegeben, und bei Wegnahme der Freigabe wird eine eventuell aktive Abfrage noch beendet. Bei Geräten die MODBUS-UDP unterstützen kann mittels UDP=TRUE dieser Modus aktiviert werden. Eine negative Befehlsausführung wird mittels ERROR gemeldet (Siehe ERROR-Tabelle).
	Mittels Einträgen in der Datenstruktur VMAP können virtuelle Datenbereiche angelegt werden, und der Zugriff für bestimmte Funktionscodes und Datenbereiche parametriert werden. Somit ist es sehr einfach möglich virtuelle Adressbereiche in einen zusammenhängenden Datenblock (DATA) abzubilden, oder Datenbereiche mit einen

	Schreibschutz zu versehen, oder Bereiche die mit Ausgangsperipherie gekoppelt sind mit einer Watchdog zu versehen.
	Die Handhabung der VMAP-Daten ist beim Baustein MB_VMAP näher beschrieben.
ERROR	
Unterstützte Funktionscodes und verwendete Parameter	



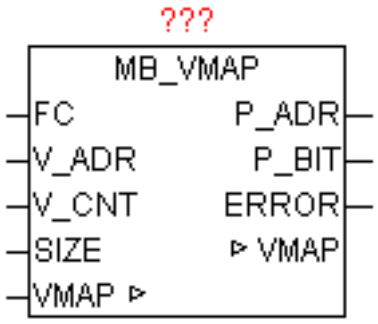
Wert	Ursprung	Beschreibung
B3	B2	B1
nn	nn	nn
xx	xx	xx
xx	xx	xx
xx	xx	xx
xx	xx	xx

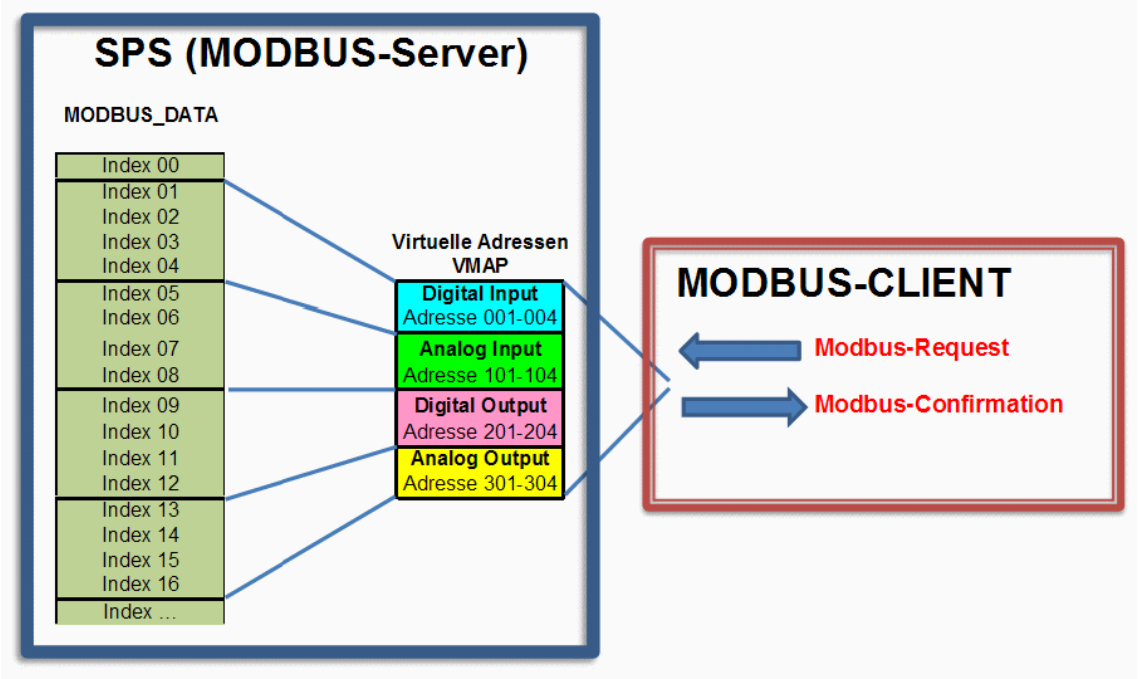
Funktionscode	Bit Zugriff	16 Bit Zugriff (Register)	Gruppe	Funktion Beschreibung
1	x		Coils	Read Coils
2	x		Input Discrete	Read Discrete Inputs
3		x	Holding Register	Read Holding Registers
4		x	Input Register	Read Input Register
5	x		Coils	Write Single Coil
6		x	Holding Register	Write Single Register
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 Funktionsbaustein	
IN_OUT VMAP	ARRAY [1..10] OF VMAP_DATA (VIRTUAL_MAP Daten)
INPUT FC	INT (Funktionsnummer)
V_ADR	INT (Virtuelle Adressbereich: Startadresse)
V_CNT	INT (Virtuelle Adressbereich: Anzahl der Datenpunkte)
SIZE	INT (Anzahl der MODBUS-Register in Struktur DATA)
OUTPUT	P_ADR: INT (Realer Adressbereich: Startadresse)
P_BIT	INT (Realer Adressbereich: Bitposition)
ERROR	DWORD (Fehlercode)
	Der Baustein ermöglicht die Umrechnung von Virtuellen Adressen auf einen Realen Adressbereich im der MODBUS-DATA Struktur. Dazu werden im VMAP-Datenarray Virtuelle Adressbereiche definiert. Wird der Baustein aufgerufen und dabei festgestellt das in den VMAP-Daten nichts eingetragen ist, wird automatisch ein Block angelegt, der den vollen Zugriff auf die ganzen MODBUS-Daten ermöglicht. In jedem Adressblock wird ein auch Watchdog-Timer verwaltet, der bei jedem Zugriff über diesem Block den Timer auf Null setzt. Somit kann einfach durch vergleichen des TIME_OUT Wertes mit einem Abschaltwert auf Kommunikationsstörungen (keine Aktualisierung) reagiert werden.
	Mittels dem Parameter FC wird erkannt um welchen Funktionscode es sich handelt, und ob dabei Register (16Bit) oder einzelne Bits behandelt werden müssen. Die Bitnummer entspricht dem Funktionscode. Das heißt das z.B. BIT5=1 in FC den Funktionscode 5 (Write Single Coil) aktiviert. Durch V_ADR wird die virtuelle Startadresse angegeben (Bei 16Bit Befehlen ist dies eine Register-Adresse und bei Bit Befehlen eine absolute Bitnummer innerhalb eines definierten Blocks. Der Parameter V_CNT definiert die Anzahl der Datenpunkte (Einheit 16 Bit oder Bit je nach Funktionscode). Die Gesamtgröße des MODBUS_ARRAY wird über SIZE (Anzahl WORDS) vorgegeben. Anhand dieser Parameter durchsucht der Baustein die VMAP Datentabelle nach einen passenden Datenblock, und gibt vom ersten korrekten Datenblock die P_ADR als Ergebnis zurück. Der Wert entspricht den realen Index für MODBUS_DATA Array. Bei einem Funktionscode mit Bit Zugriff wird noch zusätzlich die Bit-Position innerhalb P_ADR mit ausgegeben.

	Ein möglicherweise auftretender Fehler bei der Auswertung wird beim Parameter „Error“ gemeldet (Siehe Error-Tabelle). Der Watchdog-Timer wird bei jeden Zugriff mit einem Funktionscode aus der Gruppe der schreibenden Kommandos zurückgestellt.
Wird keine Sonderbehandlung gewünscht so sind in VMAP keinerlei Einstellungen notwendig, und das MODBUS_ARRAY wird dann 1	1 beim Zugriff abgebildet.
ERROR	
	! Hinweis zur Sonderbehandlung des Funktionscode 23 !
	Der Modbus-Funktionscode 23 ist ein kombinierter Befehl, weil er aus zwei Aktionen besteht. Zuerst werden Register geschrieben, und danach Register gelesen. Wird festgestellt das Schreib oder Lese-Parameter nicht zulässig sind, so wird keine der beiden Aktionen durchgeführt.
	Damit über VMAP auch zwischen dem Lesen und Schreiben unterschieden werden kann, wird der Lesebefehl in VMAP bei FC 23 als BIT23 (Read/ Write Multiple Register) , und der Schreibbefehl hingegen über BIT16 (Write Multiple Register) geprüft.





Beispiel:

Beispielkonfiguration
(* Virtueller Block 1 *)
VMAP[1].FC := DWORD#2#00000000_10000000_00000000_00011100); (FC 2,3,4,23)
VMAP[1].V_ADR := 1; (* Virtueller Adressbereich: Startadresse *)
VMAP[1].V_SIZE := 4; (* Virtueller Adressbereich: Anzahl der WORD *)
VMAP[1].P_ADR := 1; (* Realer Adressbereich: Startadresse *)
(* Virtueller Block 2 *)
VMAP[2].FC := DWORD#2#00000000_10000000_00000000_00011000); (FC 3,4,23)
VMAP[2].V_ADR := 101;(* Virtueller Adressbereich: Startadresse *)
VMAP[2].V_SIZE := 4; (* Virtueller Adressbereich: Anzahl der WORD *)
VMAP[2].P_ADR := 5; (* Realer Adressbereich: Startadresse *)
(* Virtueller Block 3 *)
VMAP[3].FC := DWORD#2#00000000_11000001_10000000_01111010);(FC1,3-6,15-16,23)
VMAP[3].V_ADR := 201;(* Virtueller Adressbereich: Startadresse *)
VMAP[3].V_SIZE := 4; (* Virtueller Adressbereich: Anzahl der WORD *)
VMAP[3].P_ADR := 9; (* Realer Adressbereich: Startadresse *)
(* Virtueller Block 4 *)
VMAP[4].FC := DWORD#2#00000000_11000001_00000000_01011000); (FC 3,4,6,16,23)
VMAP[4].V_ADR := 301;(* Virtueller Adressbereich: Startadresse *)
VMAP[4].V_SIZE := 4; (* Virtueller Adressbereich: Anzahl der WORD *)
VMAP[4].P_ADR := 12; (* Realer Adressbereich: Startadresse *)
Die Konfiguration ergibt folgende Zugriffs-Matrix:

Wert	Beschreibung
0	Keine Fehler
1	Ungültiger Funktionscode
2	Ungültige Datenadresse

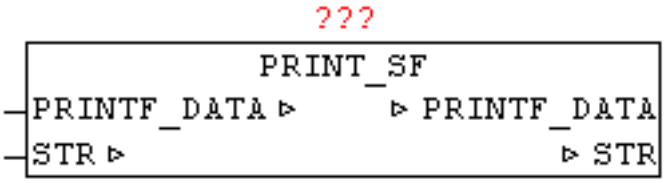
Funktion Beschreibung	Funktionscode	Bit Zugriff	16 Bit Zugriff (Register)	Lesen / Schreiben	Digital Input	Analog Input	Digital Output	Analog Output

Read Coils	1	x		Lesen			x	
Read Discrete Inputs	2	x		Lesen	x			
Read Holding Registers	3		x	Lesen	x	x	x	x
Read Input Register	4		x	Lesen	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	Lesen / Schreiben	x	x	x	x

2.0.14 PRINT_SF

Type Funktionsbaustein		
IN_OUT PRINTF_DATA	ARRAY[1..11] OF STRING(STRING_LENGTH)	(Parameterdaten)
STR	STRING(STRING_LENGTH) (Ergebnisstring)	
	Mit PRINT_SF kann ein STRING mit Teilstrings dynamisch ergänzt werden. Die Position der einzufügenden Teilstrings wird mittels , , Tilde Zeichen angegeben und die nachfolgenden Nummer definiert die Parameternummer. , 1‘ bis , 9‘ werden somit automatisch verarbeitet. Wird beim einfügen des Teilstrings die maximale Anzahl an Zeichen erreicht so wird anstatt des Teilstring nur mit ‚..‘ eingefügt.	
	VAR	
LITER	REAL := 545.4;	

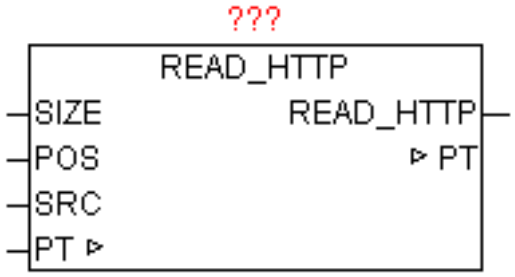
FUELLZEIT	INT := 25;	
NAME	STRING := ‚Tankinhalt‘;	
PARA	ARRAY[1..11] OF STRING(STRING_LENGTH);	
PS	PRINT_SF;	
	END_VAR	
PARA[1]	= REAL_TO_STRING(Liter); (* Parameter 1: in String wandeln *)	
PARA[2]	= INT_TO_STRING(Fuellzeit); (* Parameter 2: in String wandeln *)	
PARA[3]	= NAME; (* Parameter 3: *)	
PS.STR	= , 3: 1 Liter, Füllzeit: 2 Min.‘; (* Textausgabe-Maske *)	
PS.PRINTF_DATA	= PARA; (* Parameter Datenstruktur übergeben *)	
	PS(); (* Baustein ausführen *)	
	Der String PS.STR hat danach folgenden Inhalt	
’Tankinhalt	545.4 Liter, Füllzeit: 25 Min.’	



2.0.15 READ_HTTP

Type Funktionsbaustein		
INPUT SIZE	UINT	(Größe des Puffers)
POS	INT	(Position ab der gesucht wird)
SRC	STRING	(Suchstring)
IN_OUT PT	POINTER	(Adresse des Puffers)
OUTPUT	VALUE (Parameter der Header-Information)	
	Nach einem erfolgreichem HTTP-GET Request sind immer ein HTTP-Header (Message Header) und ein Nachrichtenkörper (Message Body) im Buffer vorhanden. Im HTTP-Header sind diverse Informationen zur angefragten HTTP-Seite abgelegt. Der nachfolgende Nachrichtenkörper beinhaltet die eigentlichen angefragten Daten. Mittels READ_HTTP können die HTTP-Header-Informationen ausgewertet werden. Der Baustein durch-	

	sucht ein beliebiges Array of Byte auf den Inhalt einer Zeichenkette und wertet danach den nachfolgenden Parameter aus, und liefert diesen String als Ergebnis zurück. Die Daten im Buffer werden automatisch auf Großbuchstaben konvertiert, daher müssen auch alle Suchstring bei SRC in Großbuchstaben übergeben werden. Mit POS kann an beliebiger Position mit der Suche begonnen werden. Das erste Element im Array hat die Positionsnummer 1.	



Beispiel:
Beispiel einer HTTP-Response (Header-Informationen):
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

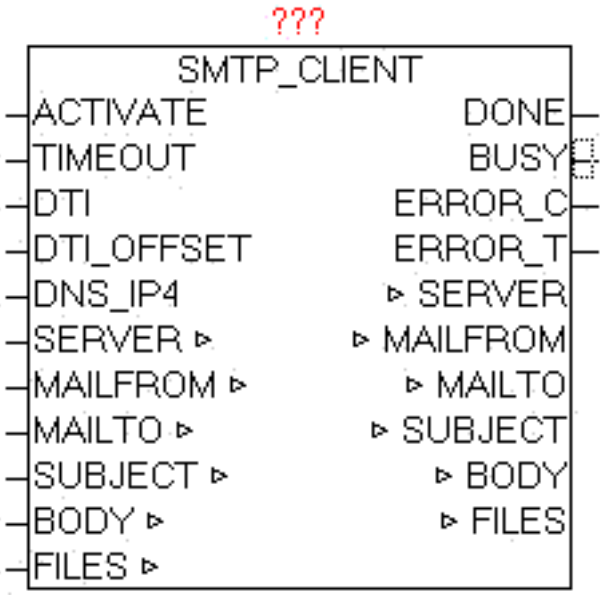
Wenn SRC keinen Suchbegriff beinhaltet, wird automatisch die HTTP Version und der HTTP Statuscode im Buffer besucht und ausgewertet. Als Ergebnis wird nach obigen Beispiel „1.0 200 OK“ zurückgegeben. Ist in SRC ein Suchbegriff enthalten wird diese Header-Information im Buffer gesucht und der Wert als String zurückgegeben z.B. ‚Content-Length‘ = „2165“.

2.0.16 SMTP_CLIENT

Type Funktionsbaustein	
IN_OUT SERVER	STRING (URL des SMTP-Server)
MAILFROM	STRING (Absenderadresse)
MAILTO	STRING(STRING_LENGTH) (Empfängeradresse)
SUBJECT	STRING (Betreff-Text)
BODY	STRING(STRING_LENGTH) (Email-Inhalt)
FILES	STRING(STRING_LENGTH) (Dateien anhänge)
INPUT ACTIVATE	BOOL (positive Flanke startet die Abfrage)
TIMEOUT	TIME (Zeitüberwachung)
DTI	DT (aktuelles Datum-Uhrzeit)
DTI_OFFSET	INT (Zeitzonen Offset zu UTC)
DNS_IP4	DWORD (IP4-Adresse des DNS-Server)

OUTPUT DONE	BOOL (Transfer ohne Fehler beendet)
BUSY	BOOL (Transfer ist aktiv)
ERROR_C	DWORD (Fehlercode)
ERROR_T	BYTE (Fehlertype)
	Der Baustein SMTP_CLIENT dient zum versenden von klassischen Emails.
Folgenden Funktionen werden unterstützt	
	SMTP-Protokoll
	Extended-SMTP-Protokoll
	Versenden von Betreffzeile, und Inhaltstext
Angabe von Email-Absenderadresse (From	) inklusive „Angezeigter Name“
Angabe von Empfänger(n) (To	)
Angabe von CarbonCopy-Empfänger(n) (Cc	)
Angabe von Blindcopy-Empfänger(n) (Bc	)
	Versenden von Datei(en) als Anhang
Authentifizierungs-Verfahren	OHNE,PLAIN,LOGIN,CRAM-MD5
	Angabe der PORT-Nummer
	Mittels positiver Flanke bei ACTIVATE wird der Übertragungsvorgang gestartet.Der Parameter SERVER enthält den Namen des SMTP-Server und optional den Benutzernamen und das Passwort und eine Port-Nummer. Wird kein Benutzername bzw. Passwort übergeben, so wird nach Standard SMTP vorgegangen.
Bei Angabe von Benutzer und Passwort wird Extend-SMTP benutzt, und automatisch das möglichst sicherste Authentifizierungs-Verfahren angewendet. Bei Parameter MAIL-FROM wird die Absenderadresse angeben	
	z.B. oscat@gmx.net
	optional kann ein zusätzlicher „Angezeigter Name“ hinzugefügt werden. Dieser wird vom Email-Client automatisch anstatt der echten Absenderadresse angezeigt. Damit kann immer ein leicht erkennbarer Name angewendet werden.
	z.B. oscat@gmx.net;Station_01
	Der Email-Client zeigt als Absender dann „Station_01“ an. Somit können mehrere Teilnehmer die gleiche Email-Adresse benutzen, jedoch eine eigenen „Alias“ Kennung mitsenden.
	Bei Parameter MAILTO können To,Cc,Bc angeben werden. Die verschiedene Empfängergruppen werden mittels ‚#‘ als Trennzeichen als Liste angegeben. Mehrere Adressen innerhalb der selben Gruppe werden mit dem Trennzeichen ‚;‘

	unterteilt. Es können von jeder Gruppe beliebig viele Empfänger vorgegeben werden, einzige Beschränkung ist die Länge des MAILTO-Strings.
	To;To..#Cc;Cc...#Bc;Bc...



Beispiel:

Beispiele.

o1@gmx.net;o2@gmx.net#o1@gmx.net#o2@gmx.net

definiert zwei To-Adressen, eine Cc-Adresse und eine Bc-Adresse

##o2@gmx.net

definiert nur eine Bc-Adresse

Mit SUBJECT kann ein Betreff-Text vorgeben werden, sowie bei BODY ein Email Inhalt als Text. Der aktuelle Date/Time Wert muss bei DTI , und bei DTI_OFFSET der Korrekturwert als Offset in Minuten zur UTC (Weltzeit) angegeben werden. Wenn bei DTI die Weltzeit UTC übergeben wird, muss bei DTI_OFFSET 0 übergeben werden.

Es können Dateien als Anhang versendet werden. Die Dateien müssen bei Parameter FILES in Listenform übergeben werden. Es können beliebig viele Dateien vorgegeben werden, einzige Beschränkung ist die Länge des FILES-Strings, und der Speicherplatz des Email-Postfachs (in der Praxis 5-30 Megabyte).

Durch eine zusätzliche optionale Angabe von '#DEL# kann nach erfolgreicher Übertragung der Dateien per Email, das Löschen der Dateien auf der Steuerung ausgelöst werden.

z.B.

FILES: ‚log1.csv;log2.csv;#DEL#'

Die beiden Dateien werden nach erfolgreicher Übertragung gelöscht.

Die Überwachungszeit kann bei Parameter TIMEOUT vorgegeben werden. Bei DNS_IP4 muss die IP-Adresse des DNS-Servers angegeben werden, wenn bei SERVER ein DNS-Name angegeben wird.Sollten bei der Übertragung Fehler auftreten, werden diese bei ERROR_C und ERROR_T ausgegeben. Solange die Übertragung läuft ist BUSY = TRUE, und nach fehlerfreien Abschluss des Vorgangs wird DONE =

TRUE. Sobald ein neuer Übertragungsvorgang gestartet wird, werden DONE,ERROR_T und ERROR_C rückgesetzt.

Der Baustein hat den IP_CONTROL integriert und muss somit nicht mehr extern mit diesen verknüpft werden, so wie dies normalerweise notwendig wäre.

SSL/TLS Verschlüsselung

SSL steht für Secure Socket Layer

TLS bedeutet Transport Layer Security

STARTTLS

Die Kommunikation zwischen client und Server beginnt unverschlüsselt Wenn der Server und der Client SSL/TLS beherschen wird noch vor der Authorisierung auf Verschlüsselung der Daten umgeschaltet. Der meistens benutzte Standardport ist Port 587

SSL

Die Kommunikation zwischen Client und Server beginnt sofort verschlüsselt. Der meistens benutzte Standardport ist Port 465

Hinweis !

Die SSL/TLS Verschlüsselung wird aktuell nur für Steuerungen von Fa. Phoenix Contact unterstützt. Um diese Funktionalität nutzen zu können muss die SPS eine Firmware haben die SSL/TLS unterstützt.

Um die Datenverschlüsselung anzuwenden muss lediglich bei der Server-URL als Protokoll „SSL:

GMAIL Kompatibilität

Da Gmail immer mit maximaler Sicherheit arbeitet und ältere SSL/TLS Protokolle mitunter nicht mehr unterstützt kann es notwendig sein das für den SMTP_CLIENT Baustein Einstellungen am Gmail-Konto gemacht werden müssen. Ein spezielle Hinweis darauf ist der SMTP Fehlercode 534 bei GMail

Einige Apps und Geräte nutzen weniger sichere Anmeldetechnologien. Dadurch wird Ihr Konto angreifbarer. Sie können den Zugriff für diese Apps deaktivieren (empfohlen) oder den Zugriff aktivieren, wenn Sie die Apps trotz des Risikos verwenden möchten.

https://support.google.com/accounts/answer/6010255?hl=de

Link für Einstellungsänderung

https://www.google.com/settings/security/lesssecureapps

Zugriff für weniger sichere Apps -> Aktivieren

SERVER: URL-Beispiele:

smtp_server

benutzername:password@smtp_server

benutzername:password@smtp_server:portnummer

TLS:

SSL:

Grundlagen:

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

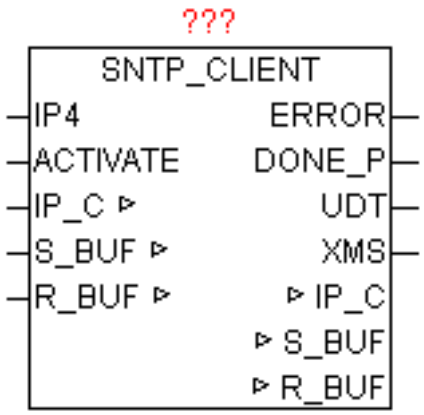
http://de.wikipedia.org/wiki/Simple_Mail_Transfer_Protocol

ERROR_T:

Wert	Eigenschaften
1	Störung: DNS_CLIENTDie genaue Bedeutung von ERROR_C ist beim Baustein DNS_CLIENT nachzulesen
2	Störung: SMTP SteuerkanalDie genaue Bedeutung von ERROR_C ist beim Baustein IP_CONTROL nachzulesen
4	Störung: FILE_SERVERDie genaue Bedeutung von ERROR_C ist beim Baustein FILE_SERVER nachzulesen
5	Störung: ABLAUF – TIMEOUTERROR_C enthält im linken WORD die Ablauf-Schrittnummer, und im rechten WORD den zuletzt vom SMTP-Server empfangenen Response-Code.Achtung der Parameter muss zuerst als HEX-Wert betrachtet, in zwei WORDS geteilt werden,und dann als Dezimalzahl betrachtet werden. Beispiel:ERROR_T = 5ERROR_C = 0x0028_00FAAblauf-Schrittnummer 0x0028 = 40Response-Code 0x00DC = 250

2.0.17 SNTP_CLIENT

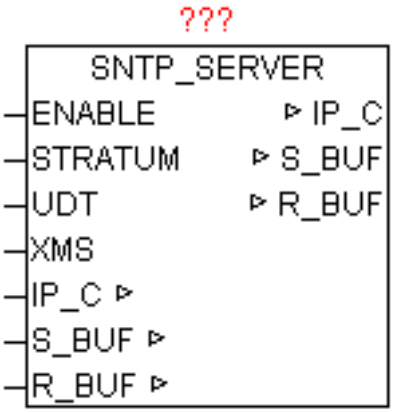
Type Funktionsbaustein	
IN_OUT IP_C	IP_C (Parametrierungsdaten)
S_BUF	NETWORK_BUFFER (Sendedaten)
R_BUF	NETWORK_BUFFER (Empfangsdaten)
INPUT IP4	DWORD (IP-Adresse des SNTP-Servers)
ACTIVATE	BOOL (Startet die Abfrage)
OUTPUT ERROR	DWORD (Fehlercode)
DONE_P	BOOL (positiver Flanke Fertig ohne Fehler)
UDT	DT (Datums und Zeit Ausgang als Weltzeit)
XMS	INT (Millisekunden der Weltzeit UDT)
	Der SNTP_CLIENT dient zum synchronisieren der lokalen Uhrzeit mit einem SNTP-Server. Dazu wird das Simple-Network-Time-Protokoll verwendet, das speziell entwickelt wurde , um eine zuverlässige Zeitangabe über Netzwerke mit variabler Paketlaufzeit zu ermöglichen. Das SNTP ist Datentechnisch völlig identisch mit NTP, das heißt hier bestehen keinerlei Unterschiede. Somit können sämtliche gekannte SNTP und NTP Server ob im lokalen Netzwerk als auch über das Internet genutzt werden. Bei IP4 wird die IP-Adresse eines SNTP/NTP Server angegeben. Eine positive Flanke bei ACTIVATE startet die Abfrage. Die vergangene Zeit zwischen Senden und dem Empfang der Zeit wird gemessen und daraus wird eine Zeitkorrektur errechnet. Danach wird die empfangene Zeit wird um diesen Wert korrigiert. Bei erfolgreicher Beendigung gibt DONE_P eine positiven Flanke aus, und die aktuelle Zeit wird bei UDT ausgegeben. Weiters werden auf XMS noch die zugehörigen Sekundenbruchteile als Millisekunden ausgegeben. Die Werte von UDT und XMS sind nur bei DONE_P = TRUE gültig, da es sich um einen statischen Uhrzeitwert handelt, und dienen nur zum Impuls gesteuerten Uhrzeitstellen. ERROR liefert im Fehlerfall die genaue Ursache (Siehe Baustein IP_CONTROL).

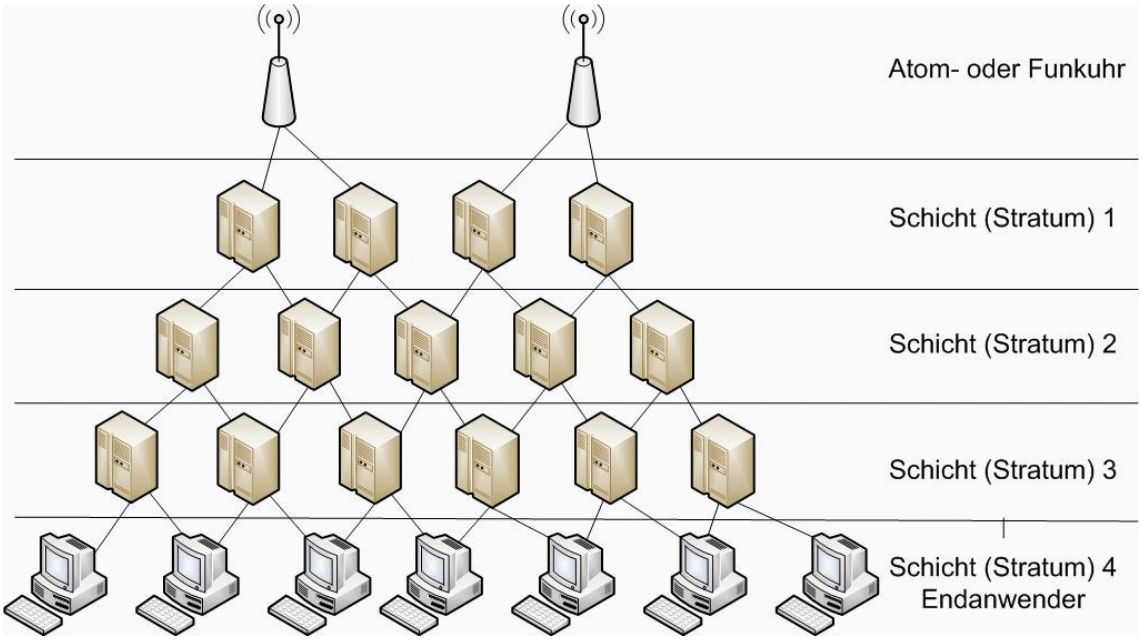


2.0.18 SNTP_SERVER

Type Funktionsbaustein		
IN_OUT IP_C	IP_C (Parametrierungsdaten)	
S_BUF	NETWORK_BUFFER (Sendedaten)	

R_BUF	NETWORK_BUFFER (Empfangsdaten)	
INPUT ENABLE	BOOL (Startet die SNTP-Server)	
STRATUM	BYTE (Angabe der hierarchische Ebene bzw.	Genauigkeit)
UDT	DT (Datums und Zeit Eingang als Weltzeit)	
XMS	INT (Millisekunden der Weltzeit UDT)	
	Der Baustein stellt die Funktionalität eines SNTP (NTP) Servers zu Verfügung. Bei ENABLE = TRUE meldet sich der Baustein bei IP_CONTROL an und wartet auf Freigabe der Ressource, sollte diese von anderen Teilnehmern momentan noch belegt sein. Danach wartet der Baustein auf Anfragen von anderen SNTP-Clients und beantwortet diese mit der aktuellen Uhrzeit von UDT und XMS. Solange ENABLE = TRUE ist, wird der Zugriff auf diese Ethernet-Ressource dauerhaft für andere Teilnehmer gesperrt (Exklusiv-Zugriff - wegen Passiv UDP Modus). SNTP nutzt ein hierarchisches System verschiedener Strata. Als Stratum 0 bezeichnet man das exakte Zeitnormal. Die unmittelbar mit ihr gekoppelten Systeme, beispielsweise NTP, GPS oder DCF77 Zeitsignale heißen Stratum 1. Jede weitere abhängige Einheit verursacht einen zusätzlichen Zeitversatz von 10-100ms und erhält bei der Bezeichnung eine höhere Nummer (Stratum 2, Stratum 3 bis 255). Wird kein STRATUM am Baustein angegeben wird STRATUM=1 als Standard verwendet.	
	Wenn ein SNTP-Client selber eine Uhrzeit mit höheren Stratum als eine SNTP-Server hat, wird die Uhrzeit von diesen mitunter abgelehnt, da diese ungenauer ist ,als die eigene Referenz. Darum ist es wichtig ein logisch richtiges STRATUM vorzugeben. Der Baustein SNTP_CLIENT ignoriert jedoch absichtlich das STRATUM und synchronisiert sich auf jeden Fall mit dem SNTP-Server, da ziemlich jeder SNTP-Server eine genauere Uhrzeit besitzt , als eine SPS.	

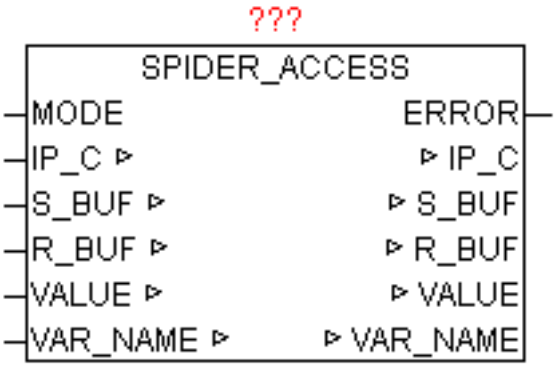




2.0.19 SPIDER_ACCESS

Type Funktionsbaustein	
IN_OUT IP_C	IP_C (Parametrierungsdaten)
S_BUF	NETWORK_BUFFER (Sendedaten)
R_BUF	NETWORK_BUFFER (Empfangsdaten)
VALUE	STRING (Wert der Variable)
NAME	STRING(40) (Variablename)
INPUT MODE	BYTE (Betriebsmodus: 1= lesen / 2=schreiben)
ERROR ERROR	DWORD (Fehlercode)
ERROR	
Mit SPIDER_ACCESS können von Steuerungen die Visualisierungen über Webserver auf Basis „SpiderControl“ von Fa. IniNet Solution GmbH integriert haben, Variablen gelesen und geschrieben werden. Für folgende Steuerungen gibt es diese Webserver Integration	
	Simatic S7 200/300/400, SAIA-Burgess PCD, Wago (750-841), Beckhoff (CX Reihe), Phoenix Contact (ILC Reihe), Selectron, Berthel, Tbox, Beck IPC
	Im SPS-Programm der Zielsteuerung müssen die gewünschten Variablen für den Webzugriff freigegeben sein. Da die Kommunikation mittels HTTP (Port 80) durchgeführt wird, kann auch über Firewalls hinweg der Datenaustausch problemlos erfolgen. Es können Globale als auch Instanz-Variablen verarbeitet werden.

Format der Variablen	
	Bei einer Globalen Variablen muss nur der normale Variablenname angeben werden. Eine Instanz-Variable muss folgend angegeben werden: „instanzname.Variablenname“
Modus	Lesen
	Wird der Parameter MODE auf „1“ gesetzt und bei „NAME“ der Variablenname angegeben, so wird zyklisch eine HTTP Anfrage an den Webserver (SPS) durchgeführt, und das gemeldete Ergebnis bei „VALUE“ als STRING ausgegeben.
Modus	Schreiben
	Wird der Parameter MODE auf „2“ gesetzt und bei „VALUE“ der Variablenwert und bei „NAME“ der Variablenname als STRING angegeben, so wird zyklisch eine HTTP Anfrage an den Webserver (SPS) durchgeführt
	Der Modus bzw. der Variablenname kann im zyklischen Betrieb jederzeit geändert werden. Sollten mehrere Variablen verarbeitet werden müssen, so müssen lediglich dementsprechend viele Bausteininstanzen aufgerufen werden.

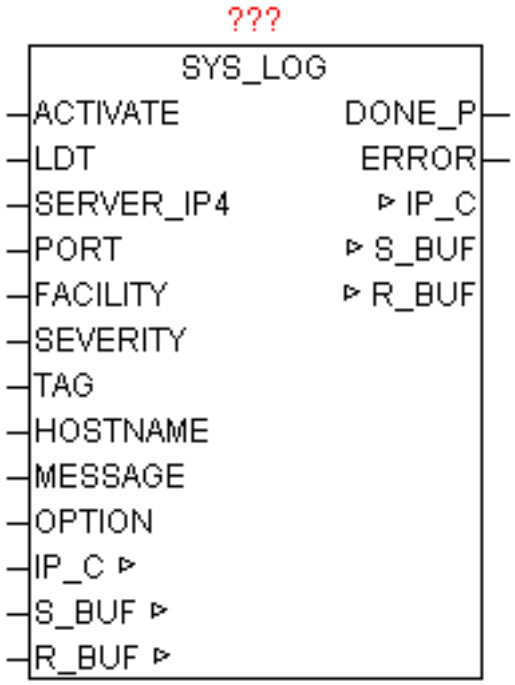


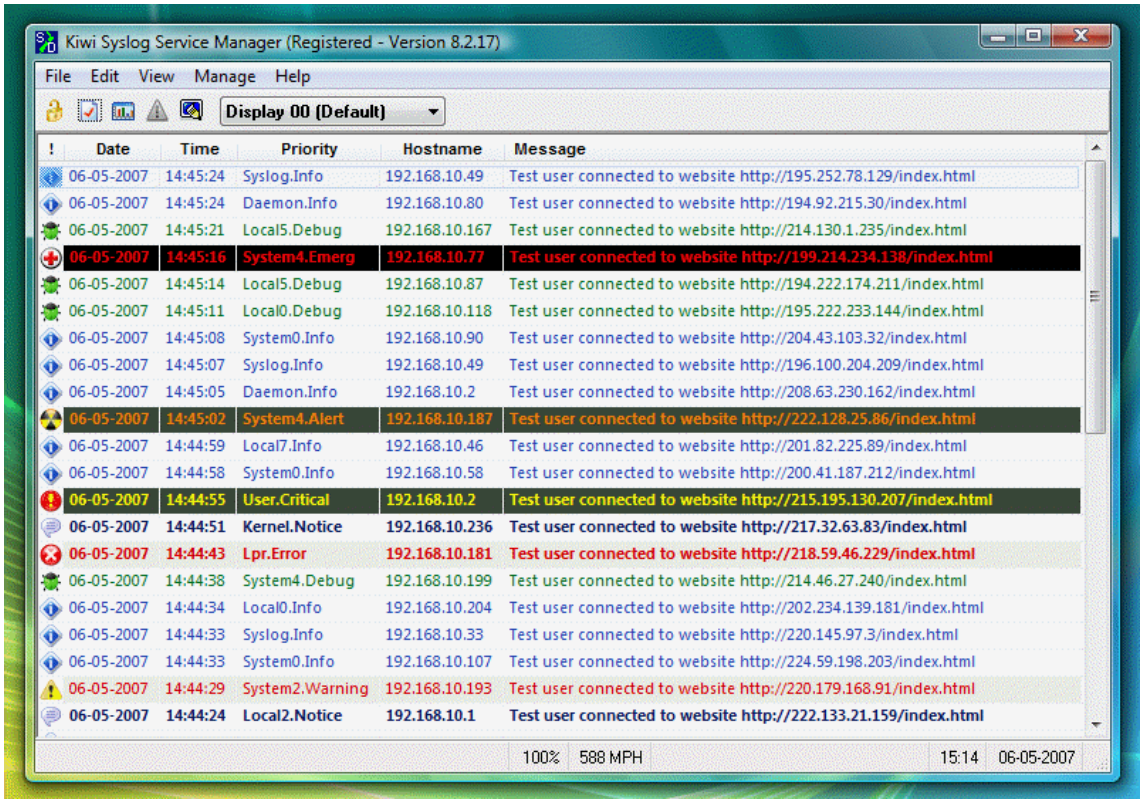
Wert	Eigenschaften
1	Beim Schreiben von Variablenwerte ist ein Fehler aufgetreten
> 1	Die genaue Bedeutung von ERROR ist beim Baustein HTTP_GET nachzulesen

2.0.20 SYS_LOG

Type Funktionsbaustein	
IN_OUT IP_C	IP_C (Parametrierungsdaten)
S_BUF	NETWORK_BUFFER (Sendedaten)
R_BUF	NETWORK_BUFFER (Empfangsdaten)
INPUT ACTIVATE	BOOL (positive Flanke startet die Abfrage)
LDT	DT (Lokalzeit)

SERVER_IP4	DWORD (IP-Adresse des SYS-LOG Servers)
PORT	WORD (Port-Nummer des SYS-LOG Servers)
FACILITY	BYTE (spezifiziert den Dienst oder die Komponente)
SEVERITY	BYTE (Klassifizierung des Schweregrades)
TAG	STRING(32) (Prozessname , ID etc.)
HOSTNAME	STRING (Name oder IP-Adresse des Senders)
MESSAGE	STRING(STRING_LENGTH) (Nachrichtentext)
OPTION	BYTE (diverse)
OUTPUT DONE	BOOL (Abfrage ohne Fehler beendet)
ERROR	DWORD (Fehlercode)
	SYSLOG ist ein Standard zur Übermittlung von Meldungen in einem IP-Rechner-Netzwerk. Das Protokoll ist dabei sehr einfach aufgebaut – der Client sendet eine kurze Textnachricht an den SYSLOG-Empfänger. Der Empfänger wird auch als „syslog daemon“ oder „syslog server“ bezeichnet. Die Meldungen werden mittels UDP-Port 514 oder über TCP-Port 1468 gesendet und enthalten die Nachricht im Klartext. SYSLOG wird typischerweise für Computersystem-Management und Sicherheits-Überwachung benutzt. Damit ermöglicht es die leichte Integration von verschiedensten LOG-Quellen auf einen zentralen SYSLOG-Server. Die Server-Software gibt es für alle Plattformen mitunter auch als Free / Shareware. Unix bzw. Linux-Systeme haben einen SYSLOG-SERVER schon integriert. Durch eine positive Flanke von ACTIVATE wird aus den Parametern LDT, FACILITY, SEVERITY, TAG, HOSTNAME,MESSAGE eine SYSLOG-Nachricht generiert und an die SERVER_IP4 Adresse gesendet. Mittels OPTION können noch diverse Eigenschaften gesteuert werden (Siehe Tabelle OPTION). Nach erfolgreichen Versenden wird DONE=TRUE, ansonsten wird bei ERROR die konkrete Fehlermeldung ausgegeben (Siehe ERROR von Baustein IP_CONTROL).
	FACILITY,SEVERITY,TIMESTAMP,HOSTNAME,TAG,MESSAGE





Beispiel:

Beispiel:

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

Folgende OPTION können verwendet werden

Folgende Severity sind als Standard definiert:

Folgende Facility sind als Standard definiert:

Für allgemeine syslog-Nachrichten sind die Facility-Werte 16-23 vorgesehen (local0 bis local7). Es ist aber durchaus zulässig, auch die vordefinierten Werte 0 bis 15 für eigene Zwecke zu verwenden.

Mit Hilfe von Facility und Severity kann später auf den SYSLOG-Server (Datenbank) sehr leicht nach bestimmten Meldungen gefiltert werden, wie beispielsweise: „Erfasse alle Mailserver-Nachrichten vom Schweregrad error“.

Beispiel (Screenshot) eines SYSLOG-Servers für Windows

BIT	Funktion
0	FALSE = mit Facility,Severity-CodeTRUE = ohne Facility,Severity-Code
1	FALSE = mit RFC HeaderTRUE = ohne RFC Header (nur die MESSAGE alleine wird versendet)
2	FALSE = mit CR,LF am EndeTRUE = ohne CR,LF am Ende
3	FALSE = UDP ModusTRUE = TCP Modus

Severity	Beschreibung
0	Emergency
1	Alert
2	Critical
3	Error
4	Warning
5	Notice

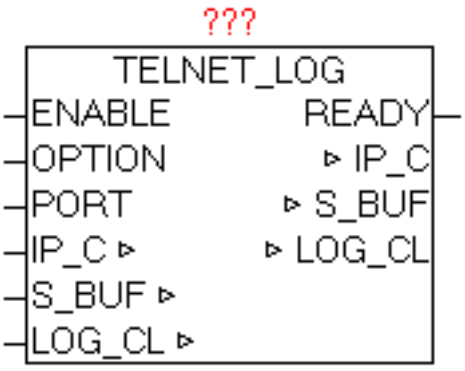
6	Informational
7	Debug

Facility	Beschreibung
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 Funktionsbaustein	
IN_OUT IP_C	IP_C (Parametrierungsdaten)
S_BUF	NETWORK_BUFFER (Sendedaten)
LOG_CL	LOG_CONTROL (LOG-Daten)
INPUT S_BUF_SIZE	UINT (Größe des S_BUF)
ENABLE	BOOL (TELNET Server freigeben)

OPTION	BYTE (Sende-Optionen)
PORT	WORD (Port Nummer)
OUTPUT READY	BOOL (TELNET Client hat Verbindung aufgebaut)
	TELNET_LOG wird benutzt um alle Meldungen die sich im LOG_CONTROL Ring-buffer befinden über TELNET auszugeben. Durch „ENABLE“ kann der Baustein aktiviert werden. Bei Parameter PORT kann die gewünschte Port-Nummer vorgegeben werden, wird diese Parameter nicht belegt so wird standardmäßig Port 23 verwendet.
	Mittels Parameter OPTION können verschiedene Optionen gewählt werden (Siehe Tabelle OPTION). Wird der Parameter OPTION nicht beschaltet, so wird folgende Einstellung standardmäßig verwendet.
	OPTION = BYTE#2#1000_1100;
	Sobald sich ein ein TELNET-Client mit dem Baustein verbindet, wird dies über Parameter „READY“ angezeigt. Daraufhin werden automatisch alle Meldungen an TELNET ausgegeben. Sobald im weiteren Verlauf neue Meldungen in LOG_CONTROL auflaufen werden diese immer wieder automatisch auch ausgegeben. Bei einen erneuten Verbindung ab/aufbau werden wieder alles Meldungen erneut ausgegeben. Die meisten TELNET-Clients bieten auch die Möglichkeit den Datenstrom in eine Datei umzuleiten, um hier eine langfristige Datenarchivierung zu ermöglichen.
OPTION	



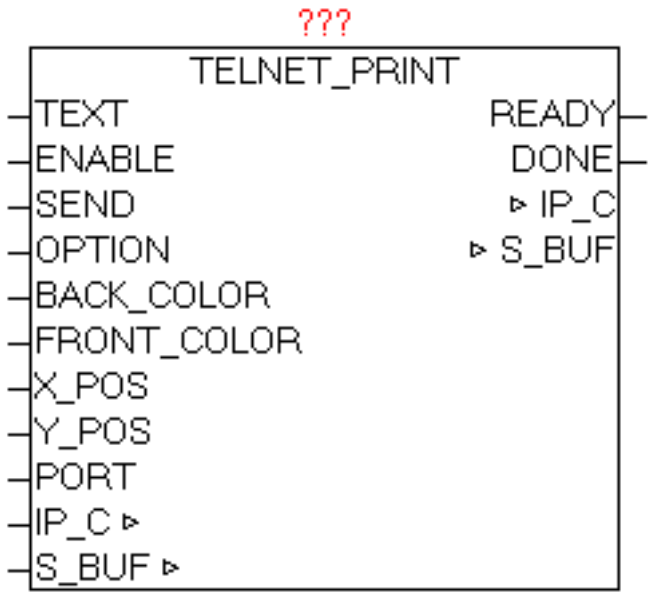
BIT	Funktion	Beschreibung
0	SCREEN_INIT	Nach dem Verbindungsaufbau mit der TELNET-Konsole wird der gesamte Bildschirm gelöscht. Wenn die OPTION COLOR aktiviert ist, wird der Bildschirm mit BACK_COLOR gelöscht.
1	AUTOWRAP	Bei AUTOWRAP=1 wird der Schreib-Cursor bei erreichen des Zeilenendes automatisch auf eine nächste Zeile gesetzt. Wenn bei der Textausgabe die X,Y Positionen immer mit angegeben werden , ist es besser wenn AUTOWRAP=0 ist.
2	COLOR	Aktiviert die den Farbe-Modus , dabei werden BACK_COLOR und FRONT_COLOR bei der Ausgabe angewandt.
3	NEW_LINE	Bei NEW_LINE=1 wird automatisch am Ende des Textes ein Carriage Return und Line-Feed angehängt. So dass die nächste Textausgabe in einer neuen Zeile beginnt. Dies ist aber nur dann Sinnvoll, wenn keine X_POS und Y_POS vorgegeben werden.

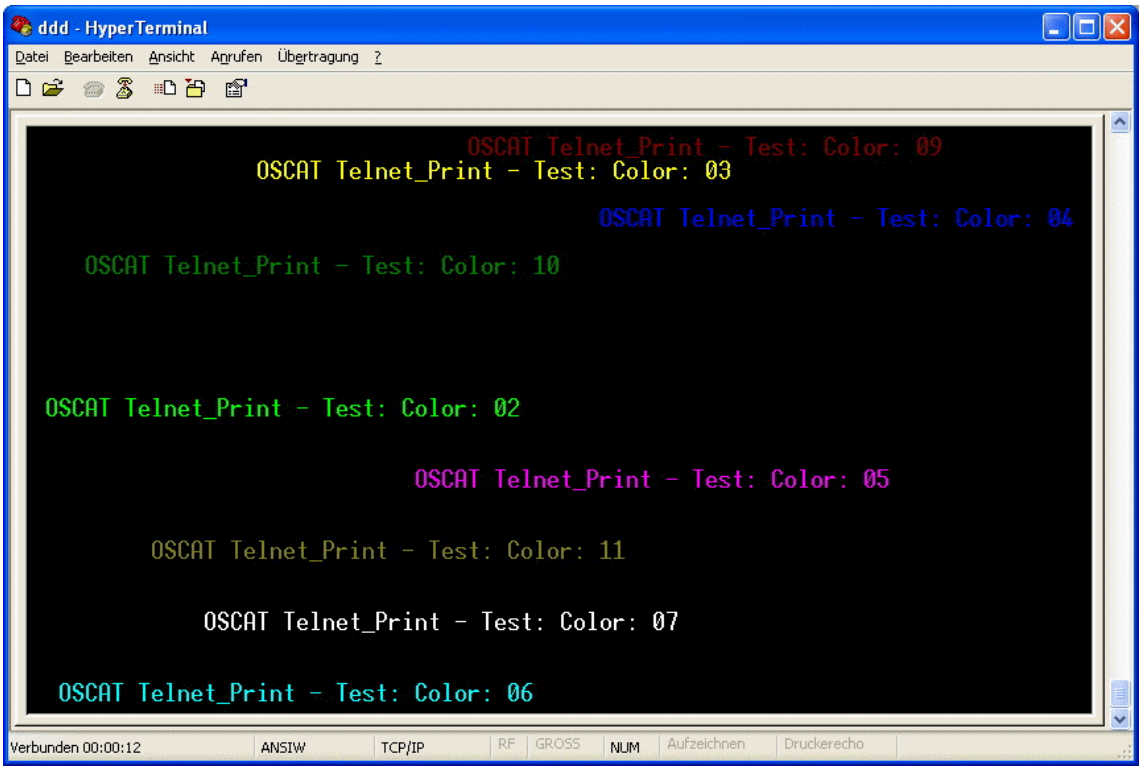
4	RESERVE	
5	RESERVE	
6	RESERVE	
7	NO_BUF_FLUSH	Verhindert das die Daten im Buffer sofort gesendet werden. Nur wenn der Buffer komplett gefüllt ist, oder diese Option deaktiviert ist, werden die Daten versendet. Ermöglicht das schnelle Senden von vielen Texten im selben Zyklus

2.0.22 TELNET_PRINT

Type Funktionsbaustein	
IN_OUT IP_C	IP_C (Parametrierungsdaten)
S_BUF	NETWORK_BUFFER (Sendedaten)
INPUT TEXT	STRING(STRING_LENGTH) (Ausgabe Text)
S_BUF_SIZE	UINT (Größe des S_BUF-Puffers)
ENABLE	BOOL (Freigabe Kommunikation)
SEND	BOOL (positive Flanke - Sendeanstoß)
OPTION	BYTE (Sende-Optionen)
BACK_COLOR	BYTE (Hintergrundfarbe)
FRONT_COLOR	BYTE (Vordergrundfarbe)
X_POS	BYTE (X-Koordinate der Schreibposition)
Y_POS	BYTE (Y-Koordinate der Schreibposition)
PORT	WORD (Port-Nummer)
OUTPUT	READY: BOOL (Baustein bereit)
DONE	BOOL (positive Flanke - Senden beendet)
	Der Baustein ermöglicht die einfache Ausgabe von Texten an eine TELNET-Konsole. Beim Parameter TEXT wird der gewünschte String übergeben. Um den Baustein für die Kommunikation freizuschalten muss ENABLE=1 gesetzt werden, damit erfolgt die Anmeldung beim IP_CONTROL. Bei Parameter PORT wird die gewünschte Port-Nummer vorgegeben, wird der Parameter nicht beschalten , wo wird der Standard-Telnet-Port 23 verwendet. Mittels BACK_COLOR und FRONT_COLOR können die gewünschten Farben vorgegeben werden, vorausgesetzt die Funktion ist Parameter OPTION aktiviert. Die Parameter X_POS und Y_POS geben die gewünschte Koordinate der TEXT Ausgabe an. Wird bei X_POS und Y_POS der WERT „0“ angegeben, so ist die Textpositionierung inaktiv, und die Texte werden immer an der aktuellen Schreib-Cursor Position angehängt. Die Standard Telnet-Konsole erlaubt eine X_POS (Horizontal) von 1 bis 80 und eine Y_POS (Vertikal) von 1 bis 25. Das Verhalten hier kann wiederum mittels OPTION beeinflusst werden (Autowrap, Carriage-Return, Line-Feed, Buf_Flush etc..). Wenn eine große Menge an Texten auf einmal ausgegeben werden muss, so kann eine Bufferung aktiviert werden, sodass die Daten erst geschrieben werden wenn entweder der Buffer voll ist (dies wird vom Baustein selbstständig veranlasst), oder dies durch den geänderten OPTION Para-

	meter signalisiert wird. Durch SEND=1 werden die Daten in den Buffer geschrieben. Die Parameter dürfen erst wieder verändert werden wenn READY=1 ist, und mittels DONE wird die erfolgte Datenübernahme als positive Flanke angezeigt.
OPTION	
FRONT_COLOR	
BACK_COLOR	





BIT	Funktion	Beschreibung

0	SCREEN_INIT	Nach dem Verbindungsaufbau mit der TELNET-Konsole wird der gesamte Bildschirm gelöscht. Wenn die OPTION COLOR aktiviert ist, wird der Bildschirm mit BACK_COLOR gelöscht.
1	AUTOWRAP	Bei AUTOWRAP=1 wird der Schreib-Cursor bei erreichen des Zeilenendes automatisch auf eine nächste Zeile gesetzt. Wenn bei der Textausgabe die X,Y Positionen immer mit angegeben werden , ist es besser wenn AUTOWRAP=0 ist.
2	COLOR	Aktiviert die den Farbe-Modus , dabei werden BACK_COLOR und FRONT_COLOR bei der Ausgabe angewandt.
3	NEW_LINE	Bei NEW_LINE=1 wird automatisch am Ende des Textes ein Carriage Return und Line-Feed angehängt. So dass die nächste Textausgabe in einer neuen Zeile beginnt. Dies ist aber nur dann Sinnvoll, wenn keine X_POS und Y_POS vorgegeben werden.
4	RESERVE	
5	RESERVE	
6	RESERVE	
7	NO_BUF_FLUSH	Verhindert das die Daten im Buffer sofort gesendet werden. Nur wenn der Buffer komplett gefüllt ist, oder diese Option deaktiviert ist, werden die Daten versendet. Ermöglicht das schnelle Senden von vielen Texten im selben Zyklus

Byte	Farbe	Byte	Farbe
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	Farbe
0	Black
1	Red

2	Green
3	Brown
4	Blue
5	Purple / Magenta
6	Cyan
7	Gray

2.0.23 XML_READER

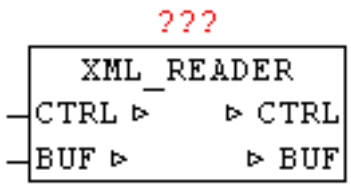
Type Funktionsbaustein		
IN_OUT CTRL	XML_CONTROL	
	(Steuer und Statusdaten)	
BUF	NETWORK_BUFFER (Empfangsdaten)	
	Mittels XML_READER ist es möglich so genannte ‚Well-formed' XML-Dokumente zu parsen. Hierbei werden nicht wie bei Hochsprachen üblich, die ganze XML-Daten eingelesen und als Datenstruktur im Speicher abgelegt, sonder es wird eine sehr Ressourcen schonende Variante angewandt. Der XML_READER liest XML-Daten als sequentiellen Datenstrom aus dem Buffer und meldet die im COMMAND definierte Elemente-Typen automatisch zurück.	
	Bei XML wird strikt zwischen Groß- und Kleinschreibung unterschieden. Ein XML-gerechtes Dokument besteht aus Elementen, Attributen, ihren Wertzuweisungen, und dem Inhalt der Elemente, der aus Text oder aus untergeordneten Elementen bestehen kann, die ihrerseits wieder Attribute mit Wertzuweisungen und Inhalt haben können. Es gibt Elemente mit und ohne Attribute, Elemente, innerhalb deren viele andere Elemente vorkommen können, und solche, innerhalb deren nur Text vorkommen kann, und sogar leere Elemente, die keinen Inhalt haben können. Die Struktur, die aus diesen Bestandteilen	

	und ihren Grundregeln entsteht, lässt sich als Baumstruktur begreifen. Elemente bestehen immer aus Tags und End-Tags. Attribute sind zusätzliche Informationen zu Elementen. Es sind auch Kommentar-Elemente erlaubt , jedoch dürfen sich diese beim XML_READER nicht zwischen Start- und End-Tags von Elementen befinden. Die möglichen DTD – Document Type Definition werden nur als DTD-Element gemeldet, jedoch nicht weiter vom XML_READER inhaltlich ausgewertet und angewandt. Mit einem CDATA-Abschnitt wird einem Parser mitgeteilt, dass kein Markup folgt, sondern normaler Text, der mittels Start-End-Block zurück gemeldet.	
	Vor dem ersten Aufruf des XML_READER müssen ein paar Parameter in der CTRL Datenstruktur initialisiert werden. Mittels CTRL.START_POS und CTRL.STOP_POS wird der Beginn und das Ende der XML-Daten im Buffer definiert. Mit CTRL.COMMAND kann einerseits ein Initialisierung (BIT15=TRUE) angestoßen werden, und mit Bit 0-14 kann definiert werden welche Element/Datentypen zurückgemeldet werden sollen. Dabei entsprechen die Typ-Codes der nachfolgenden Tabelle genau der Bit-Nummer die im CTRL.COMMAND dann jeweils auf TRUE gesetzt werden müssen.	
	Es wird immer versucht den Text von Element, Attribute, Value und Path in gesamter Länge an zugehörigen STRINGS zu übergeben. Bei STRINGS größer 255 Zeichen werden diese linksbündig abgeschnitten, jedoch mittels Block-Start und Block-End Parameter sehr wohl zurückgemeldet, so dass diese nachträglich trotzdem komplett ausgewertet werden können. Die BLOCK-START/STOP Index werden aber immer parallel	

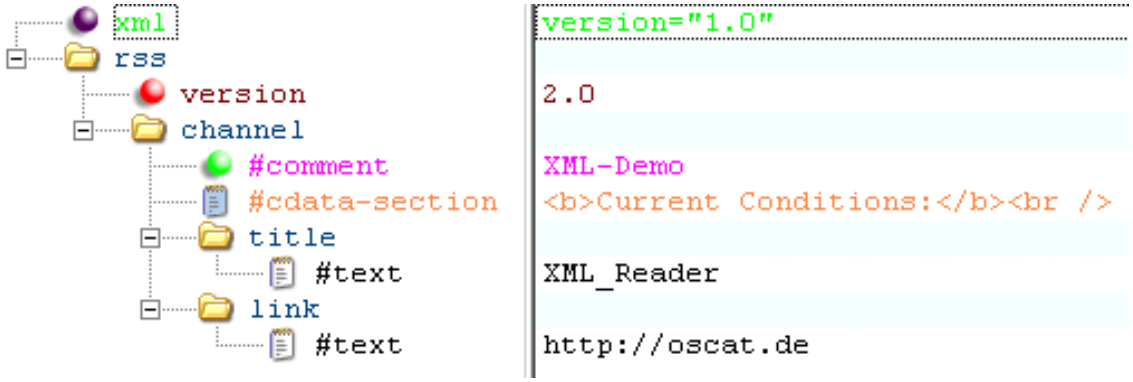
	zu den STRINGS mit übergeben. Wird der PATH-STRING größer 255 Zeichen so wird die PATH-Verfolgung deaktiviert, sodass nur noch „OVERFLOW“ als Text eingetragen ist.	
	Da bei sehr großen und komplexen XML-Daten nicht klar ist, wie lange es dauert bis der Baustein Daten zum Rückmelden findet, ist eine WATCHDOG Funktion integriert. Hiermit kann eine maximale Bearbeitungszeit parametriert werden. Beim Überschreiten der Zeit wird der Baustein-Aufruf automatisch abgebrochen, und im nächsten Zyklus wieder an gleicher Stelle fortgesetzt Dabei wird der Typ-Code 98 zurückgemeldet.	
	Folgende Typ-Kennungen sind definiert.	
	XML-Beispiel	
	Flache Darstellung	
<![CDATA[CurrentConditions	]]>XML_Reader http://oscat.de	
	Darstellung der Ebenen (ohne Processing-Instruction)	
	Darstellung als Baumansicht mit Elementtypen	
Legende		
Anwendungs-Beispiel		
	CASE STATE OF	
00		
STATE	= 10;	
CTRL.START_POS	= HTTP_GET.BODY_START; (Index des ersten Zeichen *)	
CTRL.STOP_POS	= HTTP_GET.BODY_STOP; (Index des letzten Zeichen *)	
CTRL.COMMAND	= WORD#2#11111111_11111111; (* Init + alle Elemente melden *)	
10		
	(* XML Daten seriell lesen *)	
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 – keine weiteren Elemente vorhanden *)	
	ELSIF CTRL.TYP < 98 THEN (* bei Timeout (Code 98) nichts machen *)	
	(* Auswertung der XML-Elemente über CTRL-Datenstruktur *)	
	END_IF;	
20		
	(* sonstiges..... *)	
	END_CASE;	
	Folgende Information werden dabei über die CTRL-Datenstruktur ausgegeben	
–––– erster Durchlauf –––– COUNT	1 TYPE:	5 (OPEN ELEMENT - PROCESSING INSTRUCTION) LEVEL: 1 ELEMENT: ‚xml‘ PATH: ‚/xml‘ –––– nächster Durchlauf –––– COUNT: 2 TYPE: 4 (ATTRIBUTE) LEVEL: 1 ELEMENT: ‚xml‘ ATTRIBUTE: ‚version‘ VALUE: ‚1.0‘ PATH: ‚/xml‘ –––– nächster Durchlauf –––– COUNT: 3 TYPE: 2 (CLOSE ELEMENT) LEVEL: 0 ELEMENT: ‚xml‘ PATH: ’’ –––– nächster Durchlauf –––– COUNT: 4 TYPE: 1 (OPEN ELEMENT - Standard) LEVEL: 1 ELEMENT: ‚rss‘ PATH: ‚/rss‘ –––– nächster Durchlauf –––– COUNT: 5 TYPE: 4 (ATTRIBUTE) LEVEL: 1 ELEMENT: ‚rss‘ ATTRIBUTE: ‚version‘ VALUE: ‚2.0‘ PATH: ‚/rss‘ –––– nächster Durchlauf –––– COUNT: 6 TYPE: 1 (OPEN ELEMENT - Standard) LEVEL: 2 ELEMENT: ‚channel‘ PATH: ‚/rss/channel‘ –––– nächster Durchlauf –––– COUNT: 7 TYPE: 13 (COMMENT-ELEMENT) LEVEL: 2 VALUE: ’ XML-Demo ’ PATH: ‚/rss/channel‘ –––– nächster Durchlauf –––– COUNT: 8 TYPE: 12 (CDATA) LEVEL: 2 VALUE: ‚Cur-

		rent Conditions:‘ PATH: ,/rss/channel‘ ---- nächster Durchlauf ---- COUNT: 9 TYPE: 1 (OPEN ELEMENT - Standard) LEVEL: 3 ELEMENT: ,title‘ PATH: ,/rss/channel/title‘ ---- nächster Durchlauf ---- COUNT: 10 TYPE: 3 (TEXT) LEVEL: 3 ELEMENT: ,title‘ VALUE: ’ XML_Reader’ PATH: ,/rss/channel/title‘ ---- nächster Durchlauf ---- COUNT: 11 TYPE: 2 (CLOSE ELEMENT) LEVEL: 2 ELEMENT: ,title‘ PATH: ,/rss/channel‘ ---- nächster Durchlauf ---- COUNT: 12 TYPE: 1 (OPEN ELEMENT - Standard) LEVEL: 3 ELEMENT: ,link‘ PATH: ,/rss/channel/link‘ ---- nächster Durchlauf ---- COUNT: 13 TYPE: 3 (TEXT) LEVEL: 3 ELEMENT: ,link‘ VALUE: ,http://oscat.de‘ PATH: ,/rss/channel/link‘ ---- nächster Durchlauf ---- COUNT: 14 TYPE: 2 (CLOSE ELEMENT) LEVEL: 2 ELEMENT: ,link‘ PATH: ,/rss/channel‘ ---- nächster Durchlauf ---- COUNT: 15 TYPE: 2 (CLOSE ELEMENT) LEVEL: 1 ELEMENT: ,channel‘ PATH: ,/rss‘ ---- nächster Durchlauf ---- COUNT: 16 TYPE: 2 (CLOSE ELEMENT) LEVEL: 0 ELEMENT: ,rss‘ PATH: ’’ ---- nächster Durchlauf ---- 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>
```

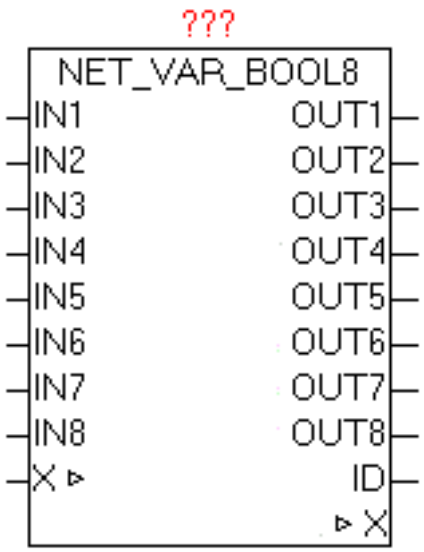


Typ (Code)	Datentyp	Beschreibung
00	unbekannt	Undefiniertes Element gefunden
01	TAG (Element)	Beginn – des ElementDatenzeiger für Element über BLOCK1
02	END-TAG (Element)	Ende – des Element
03	TEXT	Inhalt eines ElementDatenzeiger für Value über BLOCK1
04	ATTRIBUTE	Attribute eines ElementDatenzeiger für Attribute über BLOCK1Datenzeiger für Value über BLOCK2
05	TAG (Processing Instruction)	Anweisungen für die VerarbeitungDatenzeiger für Element über BLOCK1
12	CDATA	Inhaltlich nicht analysierter TEXTDatenzeiger für Value über BLOCK1
13	COMMENT	KommentarzeilenDatenzeiger für Value über BLOCK1
14	DTD	Dokumenten Typ DeklarationDatenzeiger für Value über BLOCK1
98	WATCHDOG	Maximale Durchlaufzeit erreicht- Abbruch
99	ENDE	Keine weiteren Elemente vorhanden

NET VAR

3.0.1 NET_VAR_BOOL8

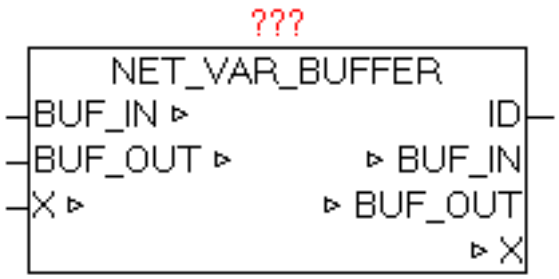
Type Funktionsbaustein	
IN_OUT X	NET_VAR_DATA (NET_VAR Datenstruktur)
INPUT IN1..8	BOOL (Signaleingänge)
OUTPUT OUT1..8	BOOL (Signalausgänge)
ID	BYTE (Identifikationsnummer)
	Der Baustein NET_VAR_BOOL8 dient zum bidirektionalen Übertragen von 8 binären Signalen vom Master zum Slave und umgekehrt. Die Signale IN1..8 werden erfasst und an der anderen Seite (Steuerung) am gleichen Baustein an der gleichen Position als OUT1..8 wieder ausgegeben.
	Gleichzeitig werden die an der Gegenseite (andere Steuerung) übergebenen Eingangsdaten hier als OUT1..8 wieder ausgegeben.
	Parameter ID zeigt die aktuelle Identifikationsnummer der Bausteininstanz. Ist die Konfiguration des Master und des Slave Programmes unterschiedlich (fehlerhaft) wird diese ID-Nummer als Fehlerort beim Baustein NET_VAR_CONTROL ausgegeben.



3.0.2 NET_VAR_BUFFER

Type Funktionsbaustein	
IN_OUT X	NET_VAR_DATA (NET_VAR Datenstruktur)
BUF_IN	ARRAY[1..64] OF BYTE (Eingang-Datenpuffer)

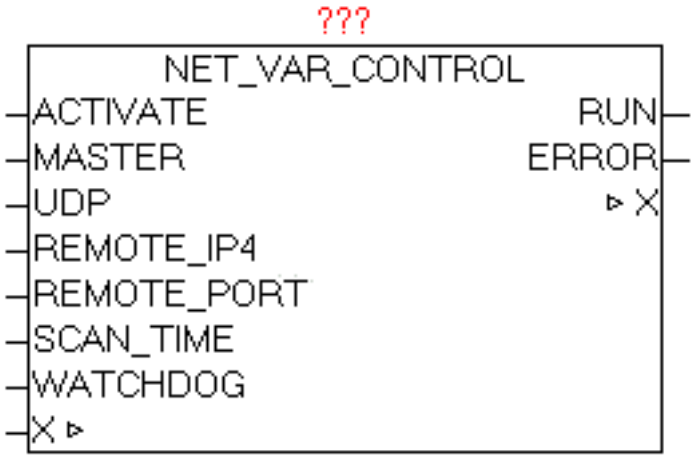
BUF_OUT	ARRAY[1..64] OF BYTE (Ausgang-Datenpuffer)
OUTPUT ID	BYTE (Identifikationsnummer)
	Der Baustein NET_VAR_BUFFER dient zum bidirektionalen Übertragen von 64 Bytes vom Master zum Slave und umgekehrt. Die Daten von BUF_IN werden erfasst und an der anderen Seite (andere Steuerung) am gleichen Baustein an der gleichen Position als BUF_OUT wieder ausgegeben.
	Gleichzeitig werden die an der Gegenseite (andere Steuerung) übergebenen Eingangsdaten hier als BUF_OUT wieder ausgegeben.
	Parameter ID zeigt die aktuelle Identifikationsnummer der Bausteininstanz. Ist die Konfiguration des Master und des Slave Programmes unterschiedlich (fehlerhaft) wird diese ID-Nummer als Fehlerort beim Baustein NET_VAR_CONTROL ausgegeben.



3.0.3 NET_VAR_CONTROL

Type Funktionsbaustein	
IN_OUT X	NET_VAR_DATA (NET_VAR Datenstruktur)
INPUT ACTIVATE	BOOL (Aktiviert den Datenaustausch)
MASTER	BOOL (FALSE=SLAVE / TRUE = MASTER)
UDP	BOOL (FALSE=TCP / TRUE = UDP)
REMOTE_IP4	DWORD (IP4-Adresse der anderen SPS)
REMOTE_PORT	WORD (PORT-Nummer der anderen SPS)
SCAN_TIME	TIME (Aktualisierungszeit)
WATCHDOG	TIME (Überwachungszeit)
OUTPUT RUN	BOOL (Datenaustausch aktiv – kein Fehler)
ERROR	DWORD ((Fehlercode)
	Der Baustein NET_VAR_CONTROL koordiniert den Datenaustausch zwischen den beiden Steuerungen und den Satellitenbausteinen NET_VAR_*. Mit ACTIVATE = TRUE kann der Datenaustausch freigegeben werden. Der Baustein muss auf beiden Steuerungen aufgerufen werden, wobei der Parameter MASTER einmal mit TRUE und einmal mit FALSE belegt werden muss. Damit wird bestimmt welche Seite den aktiven Verbindungsaufbau vornimmt. Mit UDP (FALSE/TRUE) kann vorgeben werden, ob eine UDP oder TCP-Verbindung benutzt wird. Die IP-Adresse der Gegenseite muss bei REMOTE-IP4 vorgeben werden,

	und alternativ auch die Port-Adresse (Standard-Port ist 10000). Die SCAN-TIME bestimmt in welchen Zeitabstand eine Datenaktualisierung durchgeführt wird (Standardwert ist T#1s). Über WATCHDOG wird die Überwachungszeit vorgegeben (Standardwert ist T#2s).Bei funktionierenden Datenaustausch wird Parameter RUN = TRUE. Ist der Datenaustausch länger als die Watchdog Zeit nicht möglich , wird RUN = FALSE und ein Fehler ausgegeben. Der Fehler muss nicht quittiert werden, da der Baustein selbstständig versucht den Datenaustausch wiederherzustellen. Sobald kein Fehler mehr vorhanden ist, wird RUN = TRUE und der Fehlercode wird wieder gelöscht.
ERROR	(als HEX-Wert betrachten !)

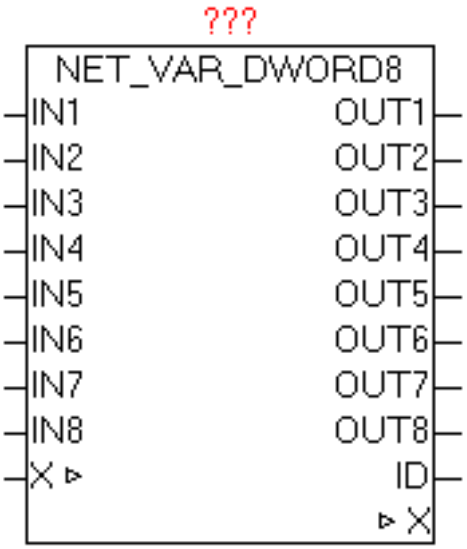


DWORD	Meldungstype	Beschreibung
B3	B2	B1
XX	..	..
..	XX	..
..	..	XX
..	..	..

3.0.4 NET_VAR_DWORD8

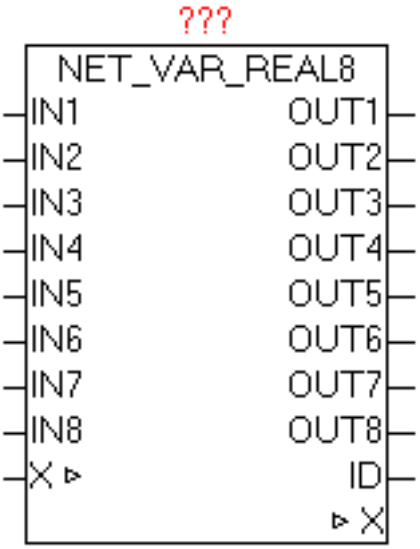
Type Funktionsbaustein	
IN_OUT X	NET_VAR_DATA (NET_VAR Datenstruktur)
INPUT IN1..8	DWORD (Eingangs DWORD)
OUTPUT OUT1..8	DWORD (Ausgangs DWORD)
ID	BYTE (Identifikationsnummer)
	Der Baustein NET_VAR_DWORD8 dient zum bidirektionalen Übertragen von acht DWORD vom Master zum Slave und umgekehrt. Die DWORD IN1..8 werden erfasst und an der anderen Seite (Steuerung) am gleichen Baustein an der gleichen Position als OUT1..8 wieder ausgegeben.

	Gleichzeitig werden die an der Gegenseite (andere Steuerung) übergebenen Eingangs-Datenwords hier als OUT1..8 wieder ausgegeben.
	Parameter ID zeigt die aktuelle Identifikationsnummer der Bausteininstanz. Ist die Konfiguration des Master und des Slave Programmes unterschiedlich (fehlerhaft) wird diese ID-Nummer als Fehlerort beim Baustein NET_VAR_CONTROL ausgegeben.



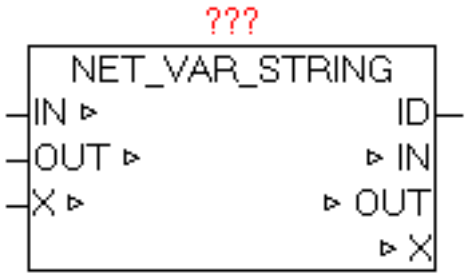
3.0.5 NET_VAR_REAL8

Type Funktionsbaustein	
IN_OUT X	NET_VAR_DATA (NET_VAR Datenstruktur)
INPUT IN1..8	REAL (Eingangwert)
OUTPUT OUT1..8	REAL (Ausgangswert)
ID	BYTE (Identifikationsnummer)
	Der Baustein NET_VAR_REAL8 dient zum bidirektionalen Übertragen von acht REAL-Werten vom Master zum Slave und umgekehrt. Die REAL-Werte IN1..8 werden erfasst und an der anderen Seite (Steuerung) am gleichen Baustein an der gleichen Position als OUT1..8 wieder ausgegeben.
	Gleichzeitig werden die an der Gegenseite (andere Steuerung) übergebenen Eingangs-REAL-Werte hier als OUT1..8 wieder ausgegeben.
	Parameter ID zeigt die aktuelle Identifikationsnummer der Bausteininstanz. Ist die Konfiguration des Master und des Slave Programmes unterschiedlich (fehlerhaft) wird diese ID-Nummer als Fehlerort beim Baustein NET_VAR_CONTROL ausgegeben.



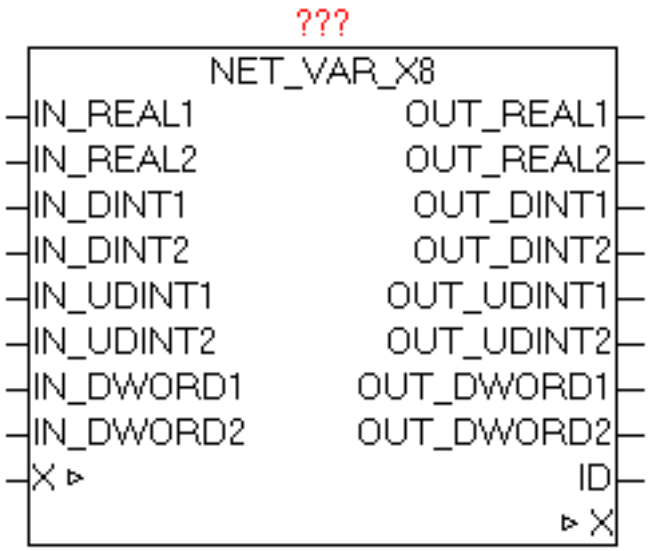
3.0.6 NET_VAR_STRING

Type Funktionsbaustein	
IN_OUT X	NET_VAR_DATA (NET_VAR Datenstruktur)
IN	STRING(STRING_LENGTH) (Eingangs-String)
OUT	STRING(STRING_LENGTH) (Ausgangs-String)
OUTPUT ID	BYTE (Identifikationsnummer)
	Der Baustein NET_VAR_STRING dient zum bidirektionalen Übertragen von einem STRING vom Master zum Slave und umgekehrt. Der STRING bei Parameter IN wird erfasst und an der anderen Seite (Steuerung) am gleichen Baustein an der gleichen Position als OUT Parameter wieder ausgegeben.
	Gleichzeitig wird der an der Gegenseite (andere Steuerung) übergebenen Eingangs-STRING-Wert hier als OUT wieder ausgegeben.
	Parameter ID zeigt die aktuelle Identifikationsnummer der Bausteininstanz. Ist die Konfiguration des Master und des Slave Programmes unterschiedlich (fehlerhaft) wird diese ID-Nummer als Fehlerort beim Baustein NET_VAR_CONTROL ausgegeben.



3.0.7 NET_VAR_X8

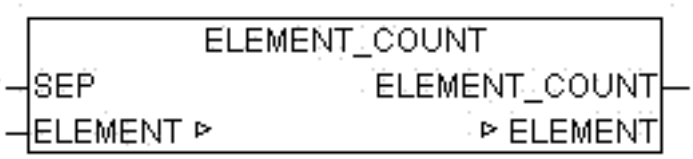
Type Funktionsbaustein	
IN_OUT X	NET_VAR_DATA (NET_VAR Datenstruktur)
INPUT IN_REAL1	REAL (Eingangwert)
IN_REAL2	REAL (Eingangwert)
IN_DINT1	DINT (Eingangwert)
IN_DINT2	DINT (Eingangwert)
IN_UDINT1	DINT (Eingangwert)
IN_UDINT2	DINT (Eingangwert)
IN_DWORD1	DINT (Eingangwert)
IN_DWORD2	DINT (Eingangwert)
OUTPUT OUT_REAL1	REAL (Ausgangwert)
OUT_REAL2	REAL (Ausgangwert)
OUT_DINT1	DINT (Ausgangwert)
OUT_DINT2	DINT (Ausgangwert)
OUT_UDINT1	DINT (Ausgangwert)
OUT_UDINT2	DINT (Ausgangwert)
OUT_DWORD1	DINT (Ausgangwert)
OUT_DWORD2	DINT (Ausgangwert)
ID	BYTE (Identifikationsnummer)
	Der Baustein NET_VAR_X8 dient zum bidirektionalen Übertragen von je zwei REAL,DINT,UDINT,DWORD Werten vom Master zum Slave und umgekehrt. Die Eingangswerte werden erfasst und an der anderen Seite (Steuerung) am gleichen Baustein an der gleichen Position wieder ausgegeben.
	Gleichzeitig werden die an der Gegenseite (andere Steuerung) übergebenen Eingangswerte hier wieder ausgegeben.
	Parameter ID zeigt die aktuelle Identifikationsnummer der Bausteininstanz. Ist die Konfiguration des Master und des Slave Programmes unterschiedlich (fehlerhaft) wird diese ID-Nummer als Fehlerort beim Baustein NET_VAR_CONTROL ausgegeben.


???
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

OTHER

4.0.1 ELEMENT_COUNT

Type Funktion	INT
Input SEP	BYTE (Separationszeichen der Elemente)
I/O ELEMENT	STRING(ELEMENT_LENGTH) (Eingangsliste)
Output	INT (Anzahl der Elemente in der Liste)
	ELEMENT_COUNT ermittelt die Anzahl der Elemente einer Liste.
	Ist der Parameter ELEMENT ein leerer String so wird als Ergebnis 0 ausgegeben. Befindet sich mindestens ein Zeichen in ELEMENT wird dies als einzelnes Element bewertet und als ELEMENT_COUNT = 1 ausgegeben.



Beispiel:
Beispiele:
ELEMENT_COUNT(‚0,1,2,3‘,44) = 4
ELEMENT_COUNT(’’,44) = 0
ELEMENT_COUNT(‚x‘,44) = 1

4.0.2 ELEMENT_GET

Type Funktion	STRING(ELEMENT_LENGTH)
Input SEP	BYTE (Separationszeichen der Elemente)
POS	INT (Position des Elements)
I/O ELEMENT	STRING(ELEMENT_LENGTH) (Eingangsliste)
Output	STRING (Ausgangsstring)
	ELEMENT_GET lieferte das Element an der Stelle POS aus einer Elementeinliste. Die Liste besteht aus Zeichenketten die mit dem Separationszeichen SEP getrennt sind. Das erste Element der Liste hat die Position 0.

Beispiel:
Beispiele:
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

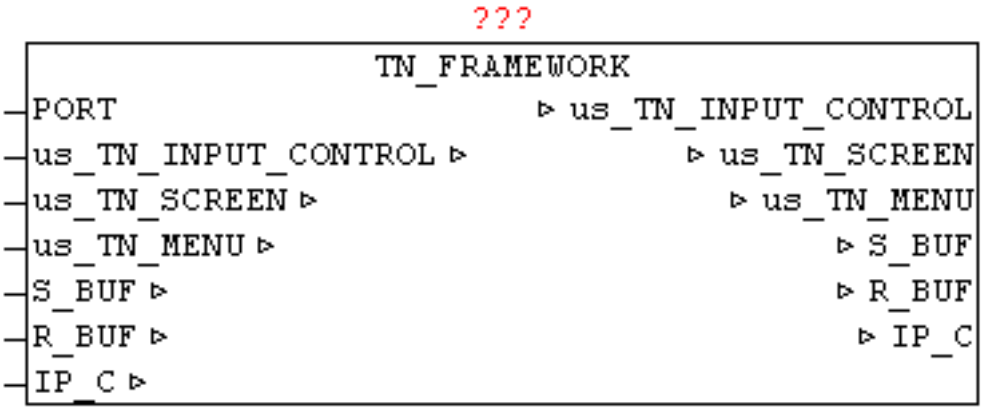
Type Funktion	DWORD
Input IN	BOOL (wenn TRUE liefert der Baustein das Release Datum)
Output	(Version der Bibliothek)
	NETWORK_VERSION gibt wenn IN = FALSE die aktuelle Versionsnummer als DWORD zurück. Wird IN auf TRUE gesetzt so wird das Release Datum der aktuellen Version als DWORD zurückgegeben.

Beispiel:
Beispiel: NETWORK_VERSION(FALSE) = 111 für Version 1.11
DWORD_TO_DATE(NETWORK_VERSION(TRUE)) = 2011-2-3

TELNET VISION

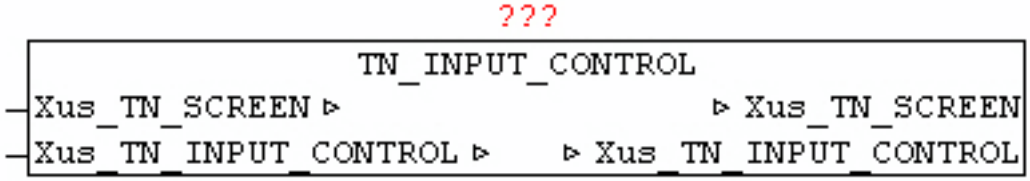
5.0.1 TN_FRAMEWORK

Type	Funktionsbaustein
IN PORT	WORD (optionale PORT-Nummer)
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 (Sendedaten)
R_BUF	NETWORK_BUFFER (Empfangsdaten)
IP_C	IP_CONTROL (Parametrierungsdaten)
	Der Baustein TN_FRAMEWORK ist eine Rahmenstruktur die ein fertiges Laufzeitmodell für TELNET-Vision zur Verfügung stellt.
	Der Parameter PORT ermöglicht die optionale Vorgabe der PORT Nummer. Wird der Parameter nicht vorgegeben, so wird der Standard Port 23 benutzt. Alle anderen Parameter sind reine Datenstrukturen die für die interne Funktion des Framework notwendig sind.
	Folgende Aufgaben und Funktionen werden abgehandelt.
	Verbindungsaufbau und Abbau mit Telnet-Client
	Daten senden und Empfangen
	Datenstrukturen für grafische Funktionen
	Datenstrukturen für INPUT_CONTROL Elemente
	Automatisches Intelligentes Updaten des Telnet-Anzeige
	Menu-Bar Darstellung
	Direkter Zugang zu allen Datenstrukturen für Anwender-programm



5.0.2 TN_INPUT_CONTROL

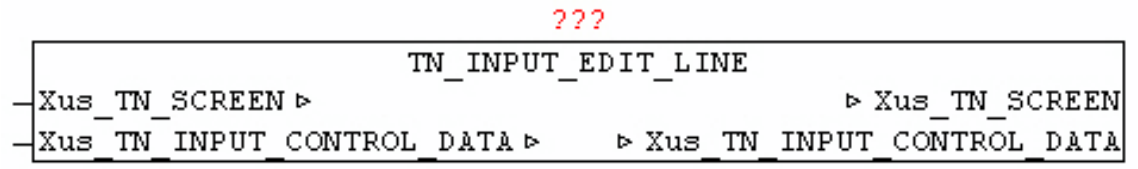
Type	Funktionsbaustein
IN_OUT Xus_TN_SCREEN	us_TN_SCREEN
Xus_TN_INPUT_CONTROL	us_TN_INPUT_CONTROL
	Der Baustein TN_INPUT_CONTROL dient zum verwalten der INPUT_CONTROL Elemente. Wird Xus_TN_INPUT_CONTROL.bo_Reset_Fokus = TRUE dann wird bei allen Elementen der Fokus deaktiviert und das erste Element bekommt den Fokus zugeteilt. Mit den Tasten Cursor Auf/Ab und Tabulator können die einzelnen Elemente angewählt bzw. gewechselt werden. Das aktuelle Element verliert dabei den Fokus und das nächstfolgende Element bekommt danach den Eingabe-Fokus neu zugeteilt. Bei dem Fokus Wechsel der Elemente wird immer automatisch das neu zeichnen der jeweiligen Elemente ausgelöst. Der Grafik / Blink-Cursor wird dabei immer je aktiven Element positioniert und dargestellt. Dabei wird auch immer automatisch der ToolTip Text ausgegeben und aktualisiert, soweit dieser parametriert wurde.
	Es werden folgende Elemente unterstützt.
	TN_INPUT_EDIT_LINE
	TN_INPUT_SELECT_TEXT
	TN_INPUT_SELECT_POPUP



5.0.3 TN_INPUT_EDIT_LINE

Type	Funktionsbaustein	

IN_OUT Xus_TN_SCREEN	us_TN_SCREEN	
Xus_TN_INPUT_CONTROL	us_TN_INPUT_CONTROL	
	Der Baustein TN_INPUT_EDIT_LINE dient zum verwalten einer Eingabezeile. Dazu muss *.in_Type = 1 gesetzt werden.	
	Das Element wird an *.in_X und *.in_Y dargestellt.Jede Eingabezeile kann auch mit einem Titletext versehen werden. Mit *.in_Title_X_Offset und *.in_Title_Y_Offset wird die Position relativ zu den Element-Koordinaten angegeben. Die Farbe kann mit *.by_Title_Attr bestimmt werden, sowie der Text durch *.st_Title_String. Soll ein ToolTip zum Element angezeigt werden so muss bei *.st_Input_ToolTip der Text angegeben werden.	
	Besitzt das Element den Fokus , kann mittels der Tasten Cursor Links/Rechts innerhalb der Zeile der Blink-Cursor verschoben werden. Mit der Backspace Taste können eingegebene Zeichen wieder gelöscht werden. Durch Betätigen der Eingabe/Return Taste wird der Eingabetext bei *.st_Input_String ausgegeben und *.bo_Input_Entered = TRUE. Das Eingabe-Flag muss nach Entgegennahme vom Anwender rückgesetzt werden. Mittels *.bo_Input_Hidden = TRUE wird die verdeckte Eingabe aktiviert, dadurch werden alle eingegebenen Zeichen mit ”* dargestellt.	
	Mittels *.st_Input_Mask wird bestimmt an welcher Position und wie viele Zeichen eingegeben werden können. An jeder Position an der sich ein Leerzeichen befindet können eingaben gemacht werden. Bei der Initialisierung muss einmalig *.st_Input_Mask nach *.st_Input_Data kopiert werden.	
	Ist *.bo_Input_Only_Num = TRUE werden nur nummerische Tasten akzeptiert und übernommen.	
***.in_Type**	= INT#01; *.in_Y := INT#16; *.in_X := INT#09; *.by_Attr_mF := BYTE#16#72; (* Weiss, Grün *) *.by_Attr_oF := BYTE#16#74; (* Weiss, Blau *) *.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 := 'Eingabezeile aktiv	SCROLL F1/F2/F3/F4

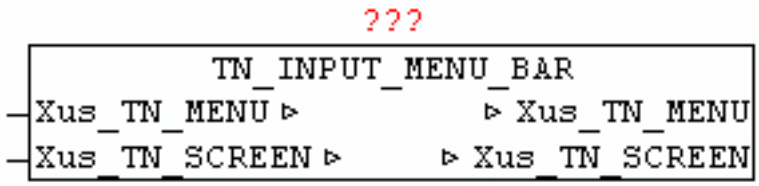


Beispiel:
Beispiel:
Ergibt folgende Ausgabe:

5.0.4 TN_INPUT_MENU_BAR

Type	Funktionsbaustein
IN_OUT Xus_TN_SCREEN	us_TN_SCREEN
Xus_TN_MENU	us_TN_MENU
	Der Baustein TN_INPUT_MENU_BAR dient zum verwalten und anzeigen der Menu_Bar. Das Element wird an *.in_X und *.in_Y dargestellt. Die Menu-Elemente sind als verschachelte Elemente innerhalb *.st_MENU_TEXT hinterlegt. Dabei werden zwei verschiedene Trennzeichen verwendet. Ein ‚$' trennt die verschiedenen Menu-Listen, und jede Menu-Liste wird wiederum mittels ‚#' in einzelne Menu-Elemente unterteilt. Die erste Menu-Liste stellt die eigentliche Menu-Bar dar, das heißt daraus ergibt sich die Anzahl der Sub-Menu bzw. die Überschriften der Elemente. Danach folgen alle Sub-Menu-Listen immer getrennt mit ‚%'. Um einzelne Sub-Menu Elemente voneinader abzugrenzen bzw. mit einer Trennlinie zu versehen, muss als Text des Menu-Elementes ein ‚-' angegeben werden.
**Durch betätigen er Escape-Taste wird die Menu-Bar aktiviert und das jeweilige Sub-Menu wird mit Hilfe des Bausteins TN_INPUT_MENU_POPUP dargestellt. Innerhalb des Sub-Menu kann mit Cursor oben/unten navigiert werden. Wird Cursor Links/Rechts betätigt so wird automatisch zum nächsten Haupt-Menu gewechselt und das zugehörige Sub-Menu angezeigt. Wird ein Sub-Menu-Element mit Enter/Return Taste bestätigt, so wird bei* .in_Menu_Selected die Nummer des gewählten Menu-Punktes ausgegeben. Die Berechnung der Menu-Punktnummer erfolgt folgend	Hauptmenu-Index * 10 + Submenu-Index. Der Eintrag in *.in_Menu_Selected muss vom Anwender nach Entgegennahme wieder auf 0 gesetzt werden.
	Somit sind maximal 9 Hauptmenu-Items und je 9 Submenu-Items ausführbar. Mittels Escape-Taste

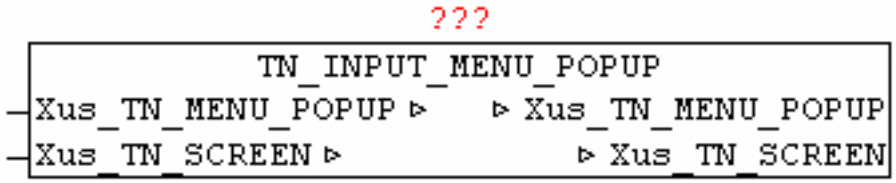
	kann jederzeit das Menu wieder ausgeblendet werden.
	Ein aktives Menu sichert automatisch den Hintergrund bevor es gezeichnet wird, und restauriert den Hintergrund wieder nach Beendigung.
	Solange eine Menu-Anzeige aktiv ist, dürfen vom Anwenderprogramm aber keinerlei grafisches Veränderungen gemacht werden. Dies kann mittels TN_SCREEN.bo_Menue_Bar_Dialog = TRUE überprüft werden.
***.in_X**	= INT#00;
***.in_Y**	= INT#00;
***.by_Attr_mF**	= BYTE#16#33; (* yellow + brown *)
***.by_Attr_oF**	= BYTE#16#0F; (* schwarz + grau *)
***.st_MENU_TEXT**	= ‚Datei#Bearbeiten#Ansicht#Ende‘;
***.st_MENU_TEXT**	= CONCAT(*.st_MENU_TEXT,
***.st_MENU_TEXT**	= CONCAT(*.st_MENU_TEXT,
***.bo_Create**	= TRUE;



Beispiel:
Beispiel:
‚%oeffnen#-#speichern#beenden%loeschen#-#einfuegen#-#kopieren‘);
‚%alles#detail#kopieren%Logout‘);
Ergibt folgende Ausgabe:

5.0.5 TN_INPUT_MENU_POPUP

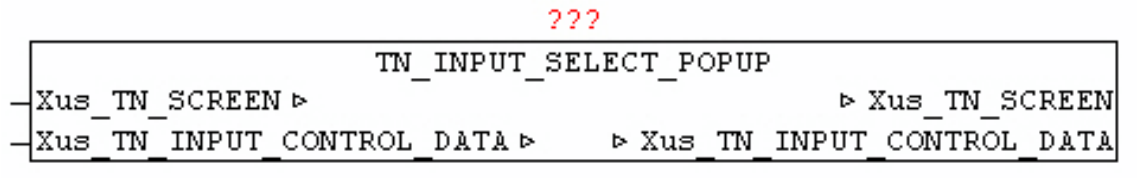
Type	Funktionsbaustein
IN_OUT Xus_TN_SCREEN	us_TN_SCREEN
Xus_TN_MENU	us_TN_MENU
	Der Baustein TN_INPUT_MENU_POPUP dient zum verwalten und anzeigen der Menu_Bar Submenu und für der Darstellung von TN_INPUT_SELECT_POPUP Elementen. Das Element wird an *.in_X und *.in_Y dargestellt. Die Menu-Elemente sind als Elemente innerhalb *.st_Menu_Text hinterlegt. Dabei werden die einzelnen Element mittels ‚#‘ voneinander getrennt. Um einzelne Sub-Menu Elemente voneinander abzugrenzen bzw. mit einer Trennlinie zu versehen, muss als Text des Menu-Elementes ein ‚-‘ angegeben werden.
	Innerhalb des Sub-Menu kann mit Cursor oben/unten navigiert werden. Wird ein Sub-Menu-Element mit Enter/Return Taste bestätigt, so wird bei *.in_Menu_Selected die Nummer des gewählten Menu-Punktes ausgegeben.
	Ein aktives Menu-Popup sichert automatisch den Hintergrund bevor es gezeichnet wird, und restauriert den Hintergrund wieder nach Beendigung.
	Solange eine Menu-Anzeige aktiv ist, dürfen vom Anwenderprogramm aber keinerlei grafisches Veränderungen gemacht werden. Dies kann mittels TN_SCREEN.bo_Menue_Bar_Dialog = TRUE bzw. TN_SCREEN.bo_Modal_Dialog = TRUE überprüft werden.
	Der Baustein wird primär vom TN_INPUT_MENU_BAR und TN_INPUT_SELECT_POPUP intern benutzt, und muss nicht direkt vom Anwender ausgeführt werden.

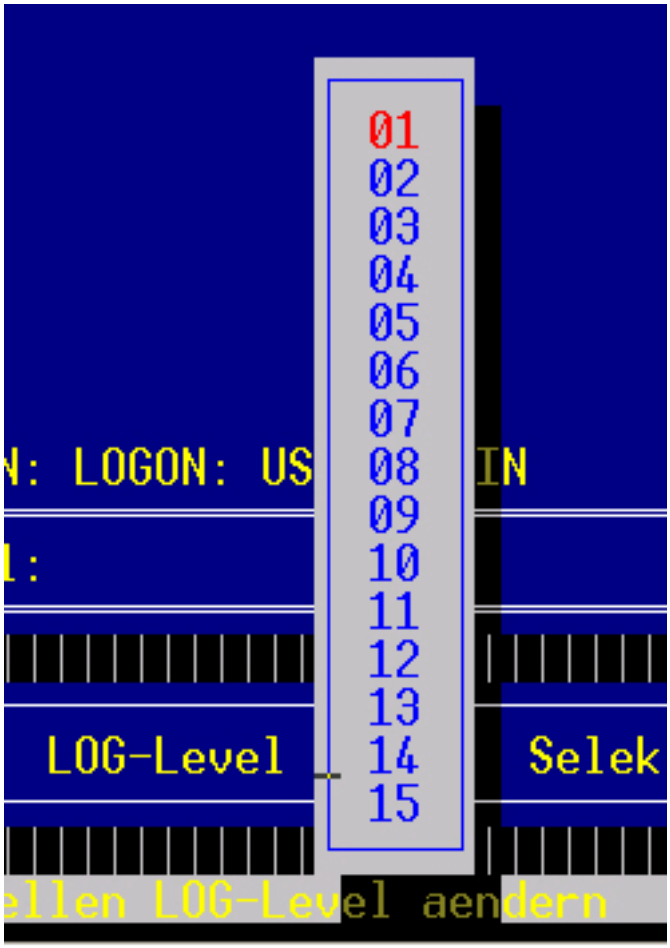


5.0.6 TN_INPUT_SELECT_POPUP

Type	Funktionsbaustein	
IN_OUT Xus_TN_SCREEN	us_TN_SCREEN	
Xus_TN_INPUT_CONTROL	us_TN_INPUT_CONTROL	
	Der Baustein TN_INPUT_SELECT_POPUP dient zum verwalten einer Aus-	

	wahl von Texten, durch anzeige eines Popup-Dialogs. Dazu muss *.in_Type = 3 gesetzt werden.	
	Das Element wird an *.in_X und *.in_Y dargestellt.Jede Eingabezeile kann auch mit einem Titletext versehen werden. Mit *.in_Title_X_Offset und *.in_Title_Y_Offset wird die Position relativ zu den Element-Koordinaten angegeben. Die Farbe kann mit *.by_Title_Attr bestimmt werden, sowie der Text durch *.st_Title_String. Soll ein ToolTip zum Element angezeigt werden so muss bei *.st_Input_ToolTip der Text angegeben werden.	
	Die Auswahltexte werden über *.st_Input_Data übergeben. Die Text-Element müssen durch das Zeichen ‚#' von einander getrennt werden.	
	Besitzt das Element den Fokus , kann mittels der Enter/Return Taste der Auswahl-Dialog aktiviert werden.	
	Mittels Cursor Hoch/Runter kann zwischen den einzelnem Elementen gewechselt werden. Wird der Anfang bzw. das Ende der Liste erreicht so wird an der gegenüberliegenden Seite fortgesetzt	
	Die Text-Element wird dabei mittels *.st_Input_Mask verknüpft, das heißt damit kann die Ausgabe-Textlänge nachträglich beeinflusst werden.	
	Durch Betätigen der Eingabe/ Return Taste wird der Text des gewählten Elementes bei *.st_Input_String ausgegeben und *.bo_Input_Entered = TRUE. Das Eingabe-Flag muss nach Entgegennahme vom Anwender rückgesetzt werden.	
	Eine aktive Auswahl (Auswahl-Dialog) kann jederzeit mit der Escape-Taste abgebrochen werden.	
***.in_Type**	= 03; *.in_Y := 20; *.in_X := 18; *.by_Attr_mF := 16#17; *.by_Attr_oF := 16#47; *.st_Input_ToolTip :=	' aktuellen LOG-Level aendern



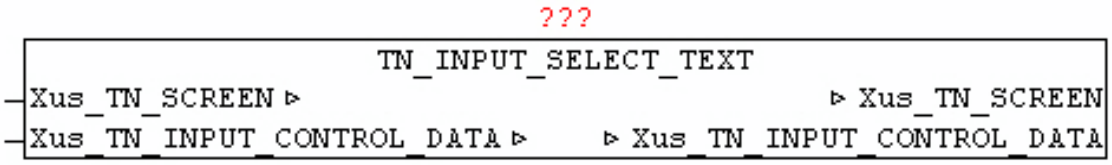


Beispiel:
Beispiel:
Ergibt folgende Ausgabe:

5.0.7 TN_INPUT_SELECT_TEXT

Type	Funktionsbaustein	
IN_OUT Xus_TN_SCREEN	us_TN_SCREEN	
Xus_TN_INPUT_CONTROL	us_TN_INPUT_CONTROL	
	Der Baustein TN_INPUT_SELECT_TEXT dient zum verwalten einer Auswahl von Texten. Dazu muss *.in_Type = 2 gesetzt werden.	
	Das Element wird an *.in_X und *.in_Y dargestellt.Jede Eingabezeile kann auch mit einem Titletext versehen werden. Mit *.in_Title_X_Offset und *.in_Title_Y_Offset wird die Position relativ zu den Element-	

	Koordinaten angegeben. Die Farbe kann mit *.by_Title_Attr bestimmt werden, sowie der Text durch *.st_Title_String. Soll ein ToolTip zum Element angezeigt werden so muss bei *.st_Input_ToolTip der Text angegeben werden.	
	Die Auswahltexte werden über *.st_Input_Data übergeben. Die Text-Element müssen durch das Zeichen ‚#' von einander getrennt werden.	
	Besitzt das Element den Fokus , kann mittels der Leertaste (Space) zwischen den einzelnen Texten gewechselt werden. Das ausgewählte Text-Element wird dabei mittels *.st_Input_Mask verknüpft, das heißt damit kann die Ausgabe-Textlänge nachträglich beeinflusst werden.	
	Durch Betätigen der Eingabe/Return Taste wird der Eingabetext bei *.st_Input_String ausgegeben und *.bo_Input_Entered = TRUE. Das Eingabe-Flag muss nach Entgegennahme vom Anwender rückgesetzt werden.	
***.in_Type**	= 2; *.in_Y := 20; *.in_X := 58; *.by_Attr_mF := 16#17; *.by_Attr_oF := 16#47; *.st_Input_ToolTip := ' Selektion-Text aktiv	press space to select

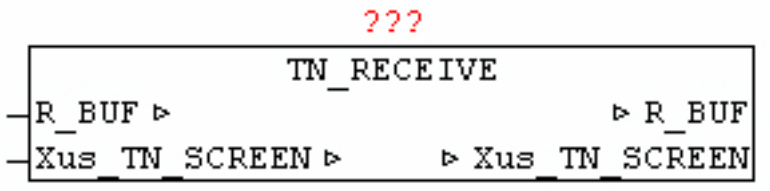


Beispiel:
Beispiel:
Ergibt folgende Ausgabe:

5.0.8 TN_RECEIVE

Type	Funktionsbaustein	
IN_OUT Xus_TN_SCREEN	us_TN_SCREEN	

R_BUF	NETWORK_BUFFER	(Telnet Empfangsbuffer)
	Der Baustein TN_RECEIVE empfängt die Eingabedaten vom Telnet-Client und wertet die Tastencodes aus.	
	Liegt der Tastencode im Bereich 32-126 so wird dieser als ASCII-Code unter Xus_TN_SCREEN.by_Input_ASCII_Code abgelegt. Zusätzlich wird Xus_TN_SCREEN.bo_Input_ASCII_IsNum = TRUE wenn dieser eine Ziffer zwischen 0 und 9 entspricht.	
	Ist der Tastencode einer der folgenden Extended-Codes dann wird dieser unter Xus_TN_SCREEN.by_Input_Exten_Code abgelegt.	

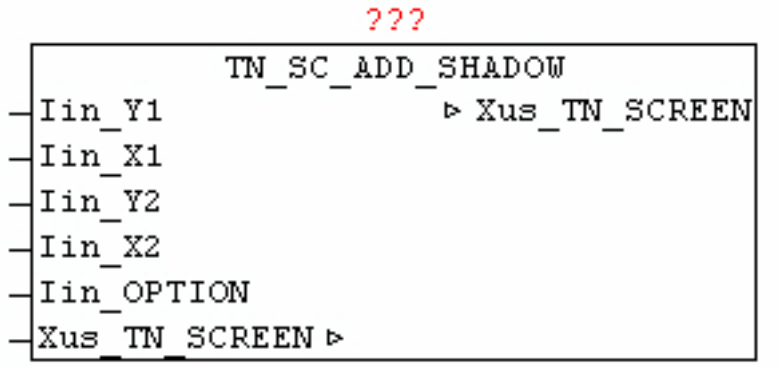


Exten_code	Tastenbezeichnung
65	Cursor oben
66	Cursor unten
67	Cursor rechts
68	Cursor links
72	Pos1
75	Ende
80	F1
81	F2
82	F3
83	F4
8	Backspace
9	Tabulator
13	Return (Enter)
27	Escape

5.0.9 TN_SC_ADD_SHADOW

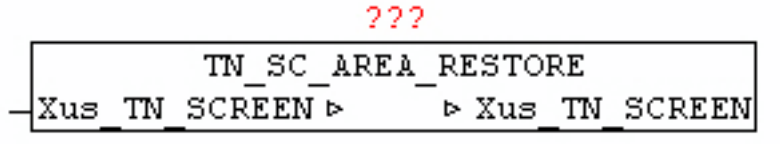
Type	Funktionsbaustein
INPUT	Iin_Y1 : INT : (Y1-Koordinate der Fläche)
Iin_X1	INT : (X1-Koordinate der Fläche)
Iin_Y2	INT : (Y2-Koordinate der Fläche)

Iin_X2	INT : (X2-Koordinate der Fläche)
Iin_OPTION	INT : (Art des Schattens)
IN_OUT Xus_TN_SCREEN	us_TN_SCREEN
	Der Baustein TN_SC_ADD_SHADOW ermöglicht das hinzufügen von optischen Schatten zu Rechteckigen Zeichenelementen. Durch Angabe einer rechteckigen Fläche mittels der Parameter X1,Y1 und X2,Y2 wird der Grundrahmen definiert, zu dem rechts und unten farblich abgedunkelter Linien gezeichnet werden. Die Schattenkoordinaten X1,Y1 und X2,Y2 werden immer +1 zum eigentlichen Grundelement angegeben. Mittels OPTION kann zwischen zwei Schattenvarianten gewählt werden. Wenn OPITION = 0 dann wird der Schatten durch reine Farbanpassung (Abdunkelung der Zeichen) erreicht. Wird eine OPTION > 0 angegeben, werden im Bereich des Schattens alle Zeichen durch schwarze ausgefüllte Zeichen ersetzt.



5.0.10 TN_SC_AREA_RESTORE

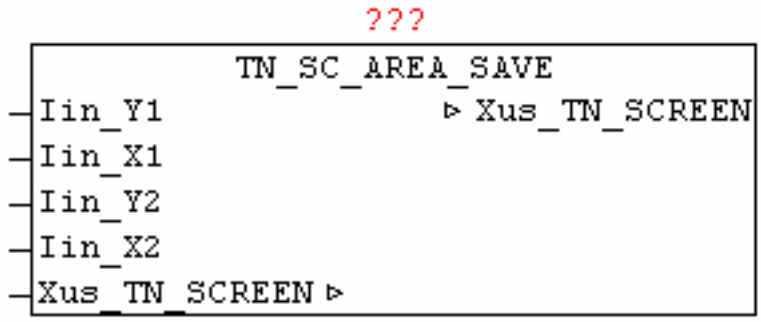
Type	Funktionsbaustein
IN_OUT Xus_TN_SCREEN	us_TN_SCREEN
	Der Baustein TN_SC_AREA_RESTORE ermöglicht das wiederherstellen des zuvor gesicherten Bildschirmbereiches.Die im Datenbereich Xus_TN_SCREEN.bya_BACKUP[x] abgelegten Bildschirmdaten werden mit Hilfe der hinterlegten Koordinaten wiederhergestellt. Dies wird vorwiegend nach dem Aufruf vom Baustein MENU-BAR und MENU-POPUP durchgeführt, um den veränderten Bildschirmbereich wiederherzustellen.



5.0.11 TN_SC_AREA_SAVE

Type	Funktionsbaustein
INPUT	Iin_Y1 : INT : (Y1-Koordinate der Fläche)
Iin_X1	INT : (X1-Koordinate der Fläche)

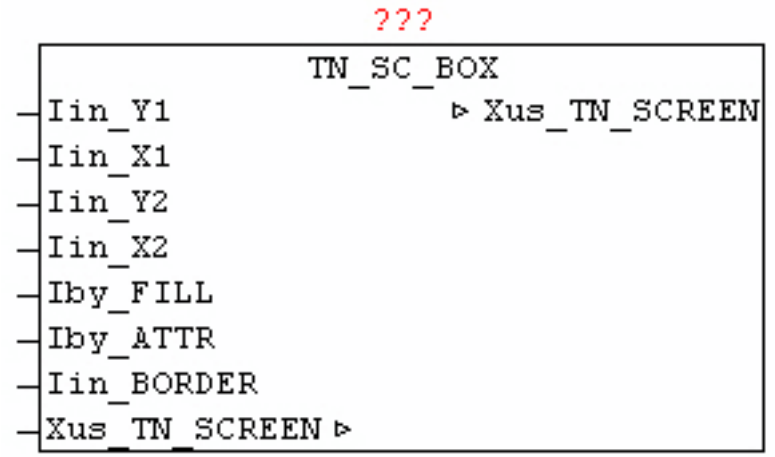
Iin_Y2	INT : (Y2-Koordinate der Fläche)
Iin_X2	INT : (X2-Koordinate der Fläche)
IN_OUT Xus_TN_SCREEN	us_TN_SCREEN
	Der Baustein TN_SC_AREA_SAVE ermöglicht das sichern von rechteckigen Flächen des Bildschirms bevor dieser durch anderen Zeichenoperationen verändert wird. Dies wird vorwiegend vor dem Aufruf vom Baustein MENU-BAR und MENU-POPUP gemacht , da diese die Elemente als Overlay-Grafik darstellen. Mittels X1,Y1 und X2,Y2 werden die Koordinaten des zu sichernden Bildschirmbereichs angegeben. Dabei werden die Daten in den Datenbereich Xus_TN_SCREEN.bya_BACKUP[x] gesichert. Es werden hierbei die Koordinaten und die eigentlichen Zeichen und Farbinformationen darin abgelegt. Der Buffer kann maximal die halbe Fläche des Bildschirms aufnehmen.

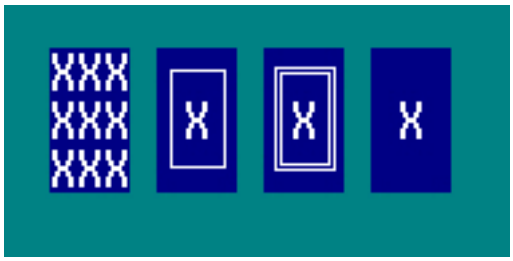


5.0.12 TN_SC_BOX

Type	Funktionsbaustein
INPUT	Iin_Y1 : INT : (Y1-Koordinate der Fläche)
Iin_X1	INT : (X1-Koordinate der Fläche)
Iin_Y2	INT : (Y2-Koordinate der Fläche)
Iin_X2	INT : (X2-Koordinate der Fläche)
Iby_FILL	BYTE : (Charakter zum füllen der Fläche)
Iby_ATTR	BYTE : (Farbcode zum füllen der Fläche)
Iby_BORDER	BYTE : (Typ der Umrahmung)
IN_OUT Xus_TN_SCREEN	us_TN_SCREEN
	Der Baustein TN_SC_BOX dient zum zeichnen eines rechteckigen Bereiches,der mit dem bei Iby_FILL angegebenen Charakter gefüllt wird. Mit Parameter Iby_ATTR kann die Füllfarbe vorgegeben werden. Der Füllbereich wird mit einer Umrandung gezeichnet, die mittels Iin_BORDER vorgegeben wird.
Border-Typen	
	0 = kein Rahmen
	1 = Rahmen mit Einzellinie
	2 = Rahmen mit Doppelllinie

	3 = Rahmen mit Leerzeichen

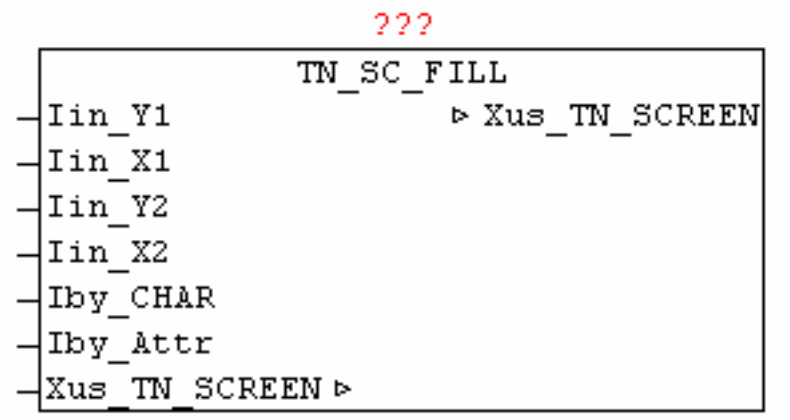




Beispiel:
Beispiel: Box mit Füllzeichen ‚X' und Farbe Weiß auf Blau
Darstellung mit Iin_BORDER Wert 0,1,2 und 3 (von Links nach Rechts)

5.0.13 TN_SC_FILL

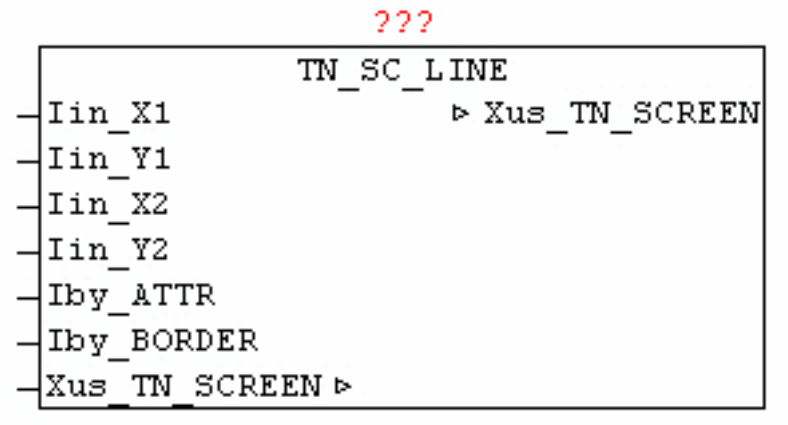
Type	Funktionsbaustein
INPUT	Iin_Y1 : INT : (Y1-Koordinate der Fläche)
Iin_X1	INT : (X1-Koordinate der Fläche)
Iin_Y2	INT : (Y2-Koordinate der Fläche)
Iin_X2	INT : (X2-Koordinate der Fläche)
Iby_CHAR	BYTE : (Charakter zum füllen der Fläche)
Iby_ATTR	BYTE : (Farbcode zum füllen der Fläche)
IN_OUT Xus_TN_SCREEN	us_TN_SCREEN
	Der Baustein TN_SC_FILL dient zum zeichnen eines rechteckigen Bereiches, der mit dem bei Iby_FILL angegebenen Charakter gefüllt wird.



Beispiel:
Beispiel: Box mit Füllzeichen ‚X' und Farbe Weiß auf Blau

5.0.14 TN_SC_LINE

Type	Funktionsbaustein
INPUT	Iin_Y1 : INT : (Y1-Koordinate der Linie)
Iin_X1	INT : (X1-Koordinate der Linie)
Iin_Y2	INT : (Y2-Koordinate der Linie)
Iin_X2	INT : (X2-Koordinate der Linie)
Iby_ATTR	BYTE : (Farbcode der Linie)
Iby_BORDER	BYTE : (Typ der Linie)
IN_OUT Xus_TN_SCREEN	us_TN_SCREEN
	Der Baustein TN_SC_LINE dient zum zeichnen von waagrechten und senkrechten Linien. Mittels der X1/Y1 und X2/Y2 Koordinaten wird der Beginn und das Ende der Linie definiert. Die Linien-Type wird mittels Iin_BORDER und der Farbcode mit Iby_ATTR übergeben. Wird beim Linien zeichnen eine andere Linie dieser Type geschnitten, so wird automatisch das passende Kreuzungszeichen benutzt.
Border-Typen	
	1 = Linie mit Einzellinie
	2 = Linie mit Doppelllinie
	>2 = Linie wird mit dem bei Iin_BORDER angegebenen Charakter gezeichnet

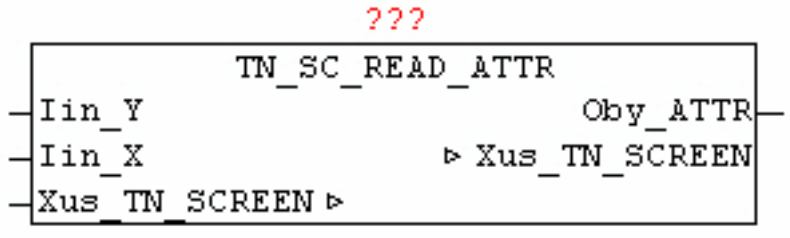


Beispiel:

Beispiel:
Horizontale Linie: Typ Single-Line
Vertikale Linie: Typ Double-Line
Horizontale und Vertikale Linie gekreuzt: Typ Single-Line
Horizontale und Vertikale Linie gekreuzt: Typ Double-Line
Horizontale Linie: Typ Charakter (X)

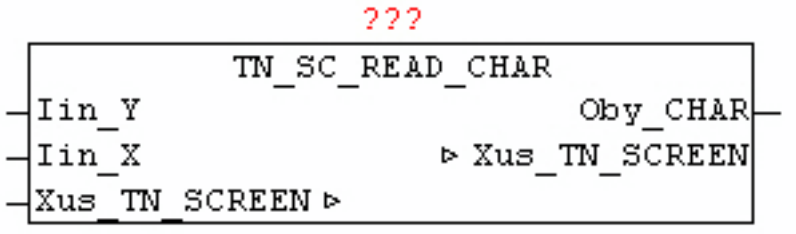
5.0.15 TN_SC_READ_ATTR

Type	Funktionsbaustein
INPUT Iin_Y	INT : (Y-Koordinate)
Iin_X	INT : (X-Koordinate)
OUTPUT Oby_ATTR	BYTE : (Farbinformation an Position X/Y)
IN_OUT Xus_TN_SCREEN	us_TN_SCREEN
	Der Baustein TN_SC_READ_ATTR dient zum auslesen der aktuellen Farbe des Charakters an der angegebenen Position X/Y.



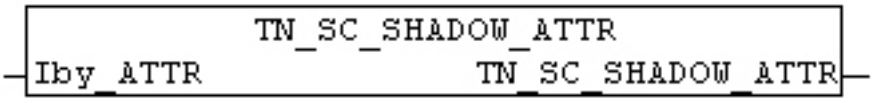
5.0.16 TN_SC_READ_CHAR

Type	Funktionsbaustein
INPUT Iin_Y	INT : (Y-Koordinate)
Iin_X	INT : (X-Koordinate)
OUTPUT Oby_CHAR	BYTE : (Zeichen an Position X/Y)
IN_OUT Xus_TN_SCREEN	us_TN_SCREEN
	Der Baustein TN_SC_READ_CHAR dient zum auslesen des aktuellen Zeichen an der angegebenen Position X/Y.



5.0.17 TN_SC_SHADOW_ATTR

Type Funktion	BYTE
INPUT Iby_ATTR	BYTE : (Farbinformation)
	Der Baustein TN_SC_SHADOW_ATTR konvertiert eine helle Farbe zu einer dunklen Farbe.

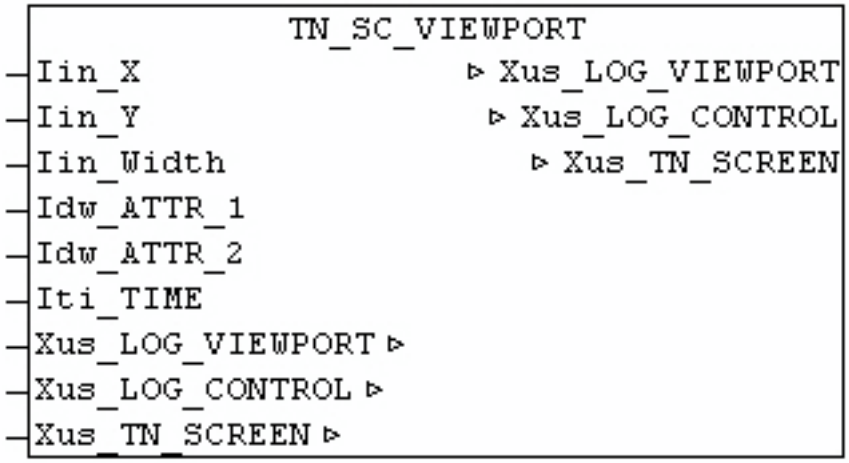


5.0.18 TN_SC_VIEWPORT

Type	Funktionsbaustein
INPUT Iin_Y	INT : (Y-Koordinate)
Iin_X	INT : (X-Koordinate)
Iin_Width	INT : (Breite des Fenster - Anzahl der Zeichen)
Idw_ATTR_1	DWORD : (Farbe 1,2,3 und 4)

Idw_ATTR_2	DWORD : (Farbe 5,6,7 und 8)
Iti_TIME	TIME : (Aktualisierungszeit)
IN_OUT Xus_LOG_VIEWPORT	LOG_VIEWPORT
Xus_LOG_CONTROL	LOG_CONTROL
Xus_TN_SCREEN	us_TN_SCREEN
	Der Baustein TN_SC_VIEWPORT dient zum anzeigen der Meldungen von der Datenstruktur LOG_CONTROL innerhalb eines rechteckigen Bereiches am Bildschirm. Die anzuzeigenden Meldungen werden zuvor mit Hilfe des Baustein LOG_VIEWPORT aufbereitet, und wenn erforderlich über Xus_LOG_VIEWPORT.UPDATE eine Aktualisierung ausgelöst. Mittels Iin_X und Iin_Y wird die linke obere Ecke des Sichtfenster, und mit Iin_Width die breite des Sichtfensters definiert. Die Anzahl der darzustellenden Zeilen wird durch Xus_LOG_VIEWPORT.COUNT vorgegeben. Die hinterlegte Farbinformation in Xus_LOG_CONTROL.MSG_OPTION[x] pro Meldung wird auf die parametrierten Farbcodes bei Idw_ATTR_1 und Idw_ATTR2 automatisch konvertiert, somit können die Farben bei der Darstellung immer individuell angepasst werden. Die Meldungen werden immer automatisch auf die breite des Sichtfenster gekürzt bzw. Abgeschnitten.

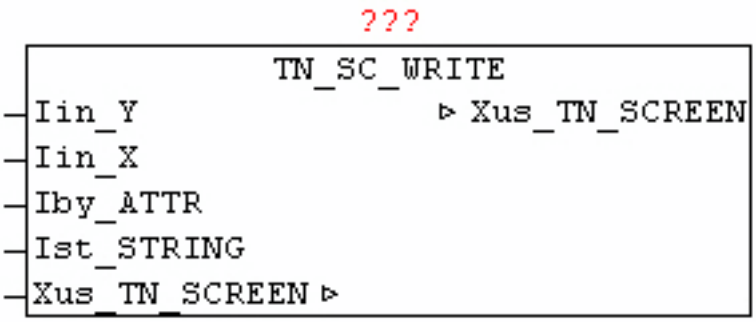
???



5.0.19 TN_SC_WRITE

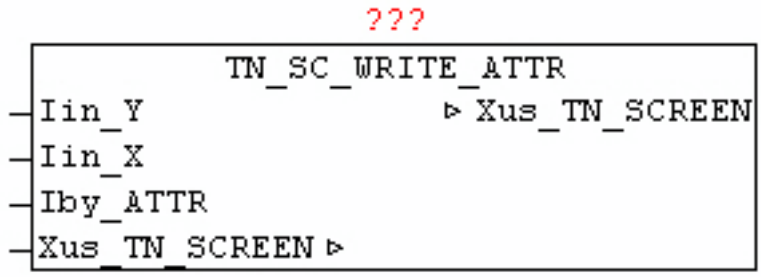
Type	Funktionsbaustein
INPUT Iin_Y	INT (Y-Koordinate)
Iin_X	INT (X-Koordinate)
Iby_ATTR	BYTE : (Farbcode - Schriftfarbe)
Ist_STRING	STRING (Text)
IN_OUT Xus_TN_SCREEN	us_TN_SCREEN
	Der Baustein TN_SC_WRITE gibt an der angegebenen Koordinate Iin_Y, Iin_Y den Text Ist_STRING mit der Farbe Iby_ATTR aus.

	Wird als Farbcode = 0 angegeben, dann wird der String ausgegeben, ohne das die vorhandenen alten Farbinformationen an den jeweiligen Zeichenpositionen verändert werden.



5.0.20 TN_SC_WRITE_ATTR

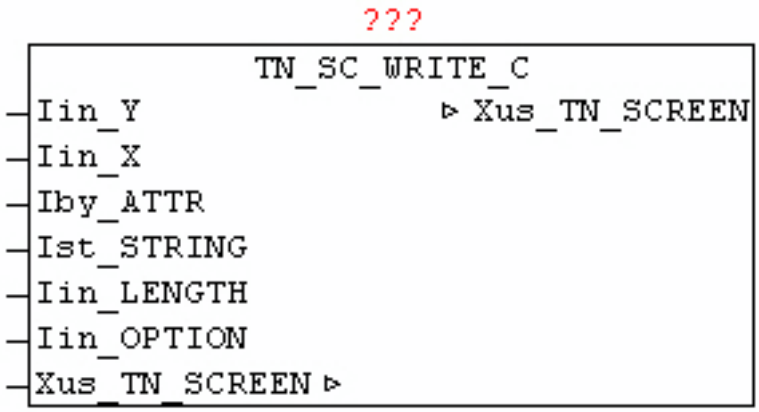
Type	Funktionsbaustein
INPUT Iin_Y	INT (Y-Koordinate)
Iin_X	INT (X-Koordinate)
Iby_ATTR	BYTE : (Farbcode)
IN_OUT Xus_TN_SCREEN	us_TN_SCREEN
	Der Baustein TN_SC_WRITE_ATTR ändert an der angegebenen Koordinate Iin_Y, Iin_Y den Farbcode ohne das vorhandene Zeichen an der Position zu verändern.



5.0.21 TN_SC_WRITE_C

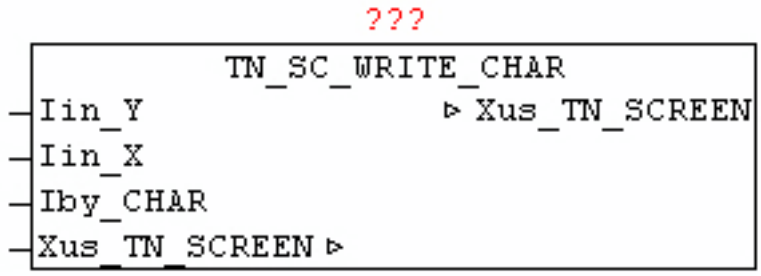
Type	Funktionsbaustein
INPUT Iin_Y	INT (Y-Koordinate)
Iin_X	INT (X-Koordinate)
Iby_ATTR	BYTE : (Farbcode)
Ist_STRING	STRING : (Text)
Iin_LENGTH	INT : (Text wird auf diese Länge angepasst)
Iin_OPTION	INT : (Option der Text-Längenanpassung)
IN_OUT Xus_TN_SCREEN	us_TN_SCREEN

	Der Baustein TN_SC_WRITE_C gibt an der angegebenen Koordinate Iin_Y, Iin_Y den Text Ist_STRING mit der Farbe Iby_ATTR aus. Der Text wird vor der Ausgabe auf die Länge Iin_LENGTH angepasst, und mittels Iin_OPTION kann die Textposition bestimmt werden.
	Iin_OPTION
0 = rechts mit Leerzeichen auffüllen	z.B. 'TEST '
1 = links mit Leerzeichen auffüllen	z.B. ' TEST'
	2 = zentrieren und mit Leerzeichen auffüllen z.B. ' TEST '



5.0.22 TN_SC_WRITE_CHAR

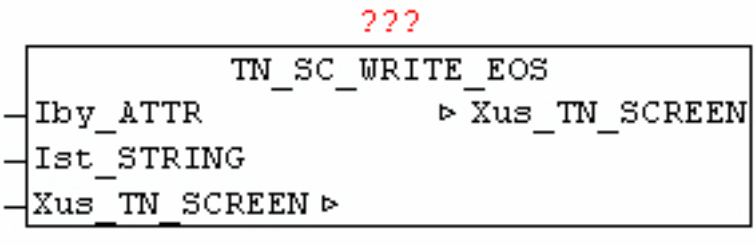
Type	Funktionsbaustein
INPUT Iin_Y	INT : (Y-Koordinate)
Iin_X	INT : (X-Koordinate)
OUTPUT Iby_CHAR	BYTE : (Zeichen)
IN_OUT Xus_TN_SCREEN	us_TN_SCREEN
	Der Baustein TN_SC_WRITE_CHAR gibt das Zeichen Iby_CHAR an der angegebenen Koordinate Iin_Y, Iin_X aus, und verändert dabei die Farbinformation an der angegebene Position nicht.



5.0.23 TN_SC_WRITE_EOS

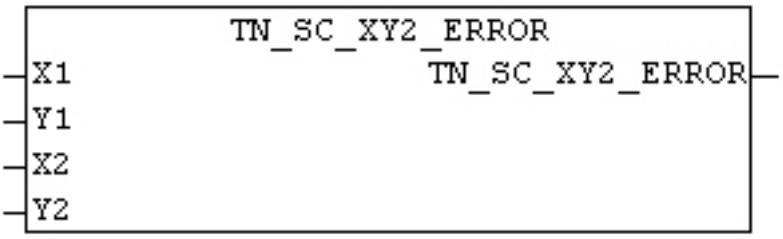
Type	Funktionsbaustein

INPUT Iby_ATTR	BYTE : (Farbcode - Schriftfarbe)
Ist_STRING	STRING (Text)
IN_OUT Xus_TN_SCREEN	us_TN_SCREEN
	Der Baustein TN_SC_WRITE_EOS gibt an der Endposition des zuletzt mit TN_SC_WRITE oder TN_SC_WRITE_EOS ausgegebenen Textes den Text Ist_STRING mit der Farbe Iby_ATTR aus.
	Damit können fortlaufend Texte ausgegeben werden, ohne das immer extra die neuen Koordinaten mit angegeben werden müssen.



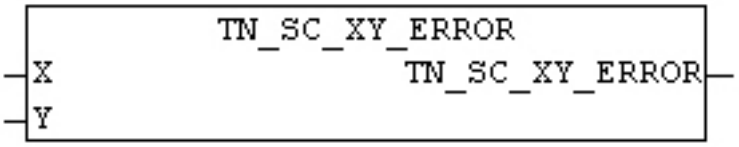
5.0.24 TN_SC_XY2_ERROR

Type Funktion	BOOL
INPUT	X1 : INT : (X1-Koordinate der Fläche)
Y1	INT : (Y1-Koordinate der Fläche)
X2	INT : (X2-Koordinate der Fläche)
Y2	INT : (Y2-Koordinate der Fläche)
	Der Baustein TN_SC_XY2_ERROR prüft ob sich die angegeben Koordinaten innerhalb des Bildschirmbereiches befindet. Die Fläche darf nicht über den Bildschirmrand hinaus gehen. Wenn die Überprüfung negativ ausfällt wird als Ergebnis TRUE ausgegeben.



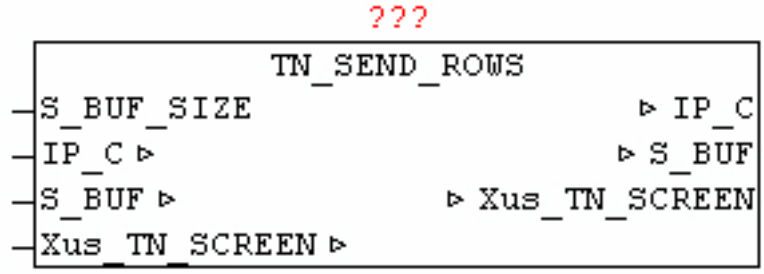
5.0.25 TN_SC_XY_ERROR

Type Funktion	BOOL
INPUT	X : INT : (X-Koordinate)
Y	INT : (Y-Koordinate)
	Der Baustein TN_SC_XY_ERROR prüft ob sich die angegebene Koordinate innerhalb des Bildschirmbereiches befindet. Wenn die Überprüfung negativ ausfällt wird als Ergebnis TRUE ausgegeben.



5.0.26 TN_SEND_ROWS

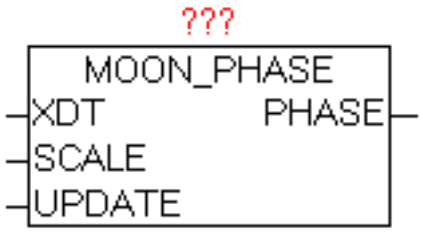
Type	Funktionsbaustein
INPUT S_BUF_SIZE	UINT (Anzahl Bytes in S_BUF.BUFFER)
IN_OUT IP_C	IP_CONTROL (Verbindungsdaten)
S_BUF	NETWORK_BUFFER (Sendedaten)
Xus_TN_SCREEN	us_TN_SCREEN
	Der Baustein TN_SEND_ROWS dient zum automatischen Updaten der grafischen Änderungen am Telnet-Screen, indem die veränderten Zeilen an den Telnet-Client versendet werden.
	Wird am Telnet-Screen eine Farbe oder ein Zeichen in einer Zeile verändert, so wird immer automatisch diese Zeile zum Update markiert. Der Baustein überprüft ob bei Xus_TN_SCREEN.bya_Line_Update[0..23] eine oder mehrere Zeilen markiert sind, und erzeugt daraus eine ANSI-Code Byte-Stream der an den Telnet-Client gesendet wird. Weiters wird bei Xus_TN_SCREEN.bo_Clear_Screen = TRUE ein Clear-Screen ausgelöst. Bei erkennen einer neuen Telnet-Client Verbindung werden automatisch alle Zeilen zum Update markiert, damit der ganze Bildschirminhalt ausgegeben wird. Ist die erforderliche Datenmenge größer als S_BUF.BUFFER werden die Daten automatisch Blockweise ausgegeben.



WEATHER

6.0.1 MOON_PHASE

Type Funktionsbaustein	
INPUT XDT	DT (Datum / Uhrzeit)
SCALE	BYTE (Skalierungsfaktor)
UPDATE	TIME (Aktualisierungszeit)
OUTPUT PHASE	BYTE (Skalierte Wert der Mondphase)
	Der Baustein MOON_PHASE dient berechnen der Mondphase zum angegeben Datum. Am Parameter XDT wird das aktuelle Datum und die Zeit übergeben, und immer nach Ablauf der Zeit bei Parameter „UPDATE“ neu berechnet. Der Standardwert für UPDATE ist 1 Stunde, und für den Skalierungsfaktor 12.
	Eine Mondphase dauert ca. 29.53 Tage, und durchläuft hierbei die typischen Zustände von Neumond bis Vollmond bzw. zunehmender und abnehmender Mond). Dieser Zyklus kann mittels SCALE auf einen gewünschten Wert zwischen 0 und 255 skaliert werden. Wird hier z.B, 100 angegeben, so wird die Mondphase als Prozentwert ausgegeben.
	Die tatsächliche Länge einer einzelnen Mond-Periode, ist verhältnismäßig großen Schwankungen unterworfen, und wird von der verwendeten Berechnungsmethode nicht berücksichtigt. Somit ergeben sich mitunter Abweichungen von einigen Stunden. Auch der Betrachtungsort (Geo-Position) ist ein virtueller Punkt im Erdmittelpunkt.
	Soll die Mondphase mittels Grafik visualisiert werden kann der Stand-Skalierungsfaktor von 12 benutzt werden, um damit die Stufen 0-11 zu erhalten
	Siehe Kapitel Visualisierung - Mond-Grafiken
http	//de.wikipedia.org/wiki/Mondphase

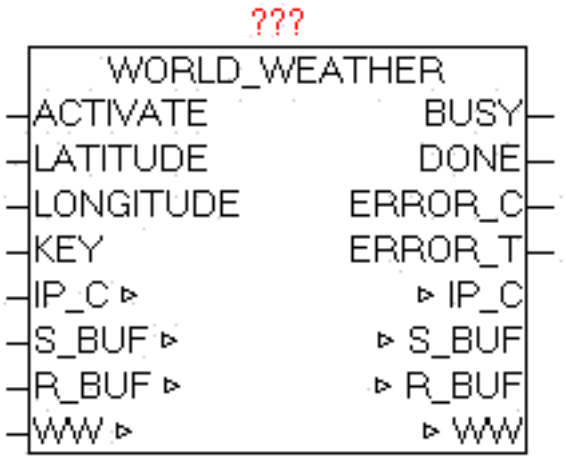


6.0.2 WORLD_WEATHER

Type Funktionsbaustein	
IN_OUT IP_C	IP_C (Parametrierungsdaten)
S_BUF	NETWORK_BUFFER (Sendedaten)

R_BUF	NETWORK_BUFFER (Empfangsdaten)
WW	WORLD_WEATHER_DATA (Wetterdaten)
INPUT ACTIVATE	BOOL (positive Flanke startet die Abfrage)
LATITUDE	REAL (Breitengrad des Bezugsortes)
LONGITUDE	REAL (Längengrad des Bezugsortes)
KEY	STRING(30) (API-Key)
OUTPUT BUSY	BOOL (Abfrage ist aktiv)
DONE	BOOL (Abfrage ohne Fehler beendet)
ERROR_C	DWORD (Fehlercode)
ERROR_T	BYTE (Fehlertype)
Der Baustein lädt die aktuellen Wetterdaten des angegebene Ortes von http	//worldweather.com herunter, analysiert die Daten und legt die wesentlichen Daten aufbereitet in der WORLD_WEATHER_DATA Datenstruktur ab.
	Vom aktuellen Tag werden folgende Werte abgelegt.
	Observation time (UTC) , Temperature (°C) , Unique Weather Code , Weather description text, Wind speed in miles per hour , Wind speed in kilometre per hour , Wind direction in degree , 16-Point wind direction compass ,Precipitation amount in millimetre , Humidity (%) , Visibility (km) ,Atmospheric pressure in milibars , Cloud cover (%)
	Vom aktuellen Tag und den darauffolgenden 4 Tagen werden folgende Werte abgelegt.
	Date for which the weather is forecasted , Day and night temperature in °C(Celcius) and °F(Fahrenheit) , Wind Speed in mph (miles per hour) and kmph (Kilometer per hour) , 16-Point compass wind direction , A unique weather condition code , Weather description text , Precipitation Amount (millimetre)
	Mit einer positiven Flanke von ACTIVATE wird die Abfrage gestartet und eine DNS Abfrage mit nachfolgender HTTP-GET durchgeführt. Nach erfolgreichen Empfang der Daten werden alle Elemente durchlaufen und wenn benötigt in der Datenstruktur in konvertierter Form abgelegt. Durch die Parameter LATITUDE und LONGITUDE wird der genaue Ort (Geografische Position) des Wetters angegeben. Während die Abfrage aktiv ist, wird BUSY=TRUE ausgegeben. Nach erfolgreich beendeter Abfrage wird DONE=TRUE ausgegeben. Sollte bei der Abfrage ein Fehler auftreten so wird dieser unter ERROR_C gemeldet in Kombination mit ERROR_T.
ERROR_T	

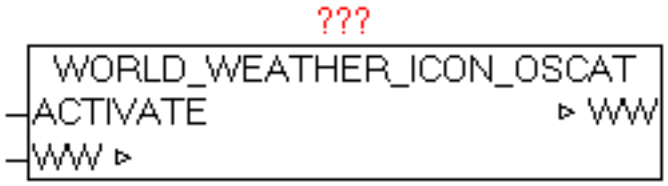
Neuen API-KEY anlegen	
Mittels Internet-Browser die Seite http	//www.worldweatheronline.com aufrufen und mittels Button „Free sign up“ den Registrierungsdialog aufrufen, und die erforderlichen Felder ausfüllen. Nach der Registrierung wird eine Email versendet, die wiederum bestätigt werden muss, und in weiterer Folge wird in einer zweiten Email der persönliche API-Key versendet. Dieser API-Key muss beim Baustein am Parameter API-KEY übergeben werden.
Breiten und Längengrad eines bestimmten Ortes bestimmen	
Mittels Internet-Browser die Seite http	//www.mygeoposition.com/ aufrufen, und im Eingabefeld den Namen des gesuchten Ortes eingeben, und mittels “Geodaten berechnen” den Ort suchen. Danach wird der gesuchte Ort auf der Karte angezeigt inklusive des benötigten Breiten und Längengrades in dezimaler Schreibweise. Die ermittelte Position gehört bei den Baustein-Parametern LATITUDE und LONGITUDE übergeben.



Wert	Eigenschaften
1	Die genaue Bedeutung von ERROR_C ist beim Baustein DNS_CLIENT nachzulesen
2	Die genaue Bedeutung von ERROR_C ist beim Baustein HTTP_GET nachzulesen

6.0.3 WORLD_WEATHER_ICON_OSCAT

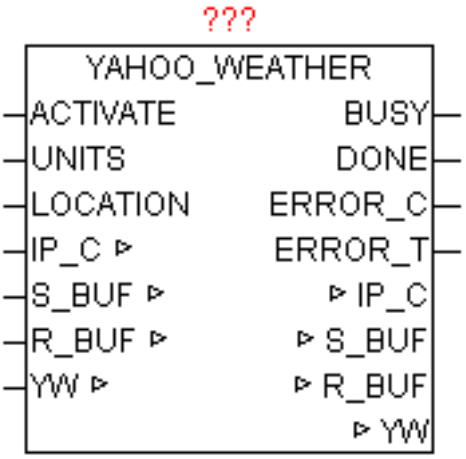
Type Funktionsbaustein	
IN_OUT WW	WORLD_WEATHER_DATA (Wetterdaten)
INPUT ACTIVATE	BOOL (positive Flanke startet die Übersetzung)
	Der Baustein ersetzt die originalen Anbieterspezifischen Wetter Icons-Nummern durch OSCAT-Standard Icon Nummern. Nach einer positiven Flanke bei ACTIVATE werden die Elemente (Icon-Nummern) in der WORLD_WEATHER_DATA Datenstruktur ersetzt. Nach erfolgter Abfrage der Wetterdaten mittels WORLD_WEATHER sollte dieser Baustein darauffolgend aufgerufen werden. Dabei kann einfach der Parameter DONE vom Baustein WORLD_WEATHER mit ACTIVATE verschalten werden.
Folgende Elemente werden angepasst	
	WW.WORLD_WEATHER_CUR.WEATHER_ICON
	WW.WORLD_WEATHER_DAY[0..4].WEATHER_ICON

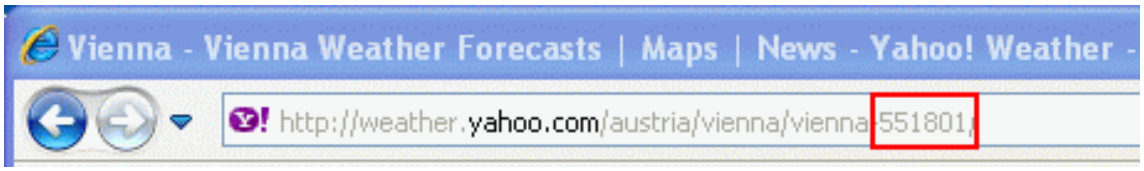


6.0.4 YAHOO_WEATHER

Type Funktionsbaustein	
IN_OUT IP_C	IP_C (Parametrierungsdaten)
S_BUF	NETWORK_BUFFER (Sendedaten)
R_BUF	NETWORK_BUFFER (Empfangsdaten)
YW	YAHOO_WEATHER (Wetterdaten)
INPUT ACTIVATE	BOOL (positive Flanke startet die Abfrage)
UNITS	BOOL (FALSE = Celsius , TRUE = Fahrenheit)
LOCATION	STRING(20) (Ortsangabe mittels LOCATION-ID)
OUTPUT BUSY	BOOL (Abfrage ist aktiv)
DONE	BOOL (Abfrage ohne Fehler beendet)
ERROR_C	DWORD (Fehlercode)
ERROR_T	BYTE (Fehlertype)
Der Baustein lädt die aktuellen Wetterdaten des angegebene Ortes mittels eines RSS feed (XML-Datenstruktur) von http	//weather.yahooapis.com herunter, analysiert die XML-Daten und legt die wesentlichen Daten aufbereitet in der YAHOO_WEATHER Datenstruktur ab. Mit einer positiven Flanke von ACTIVATE wird die Abfrage gestartet und eine DNS Abfrage mit nachfolgender HTTP-GET durchgeführt. Nach erfolgreichen Empfang der Daten werden

	mittels XML_READER alle Elemente durchlaufen und wenn benötigt in der Datenstruktur in konvertierter Form abgelegt. Mit UNITS kann noch zwischen Fahrenheit und Celsius als Einheit gewählt werden. Durch Angabe der LOCATION_ID wird der genaue Ort des Wetters angegeben. Während die Abfrage aktiv ist, wird BUSY=TRUE ausgegeben. Nach erfolgreich beendeter Abfrage wird DONE=TRUE ausgegeben. Sollte bei der Abfrage ein Fehler auftreten so wird dieser unter ERROR_C gemeldet in Kombination mit ERROR_T.
ERROR_T	
Suchen der Location-ID eines bestimmten Ortes	
Mittels Internet-Browser die Seite http	//weather.yahoo.com/ aufrufen, und im Eingabefeld “Enter city or zip code:” den Namen des gesuchten Ortes eingeben, und mittels “Go” suchen bzw. Aufrufen.
	Nach erfolgreicher Auswahl werden im Browserfenster die aktuellen Wetterdaten des angegebenen Ortes angezeigt. In der URL (Web-Link) Zeile ist nun die Location-ID ersichtlich.
	Somit ergibt der gesuchte Ort „Wien (vienna)“ die Location-ID „551801“. Dieser Code muss am Baustein als Parameter übergeben werden.
	Aus den XML-Daten werden die benötigten Elemente aufbereitet und in der YAHOO_WEATHER Datenstruktur abgelegt.

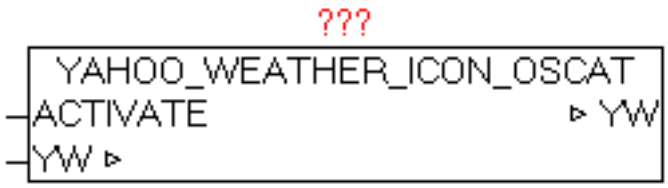




Wert	Eigenschaften
1	Die genaue Bedeutung von ERROR_C ist beim Baustein DNS_CLIENT nachzulesen
2	Die genaue Bedeutung von ERROR_C ist beim Baustein HTTP_GET nachzulesen

6.0.5 YAHOO_WEATHER_ICON_OSCAT

Type Funktionsbaustein	
IN_OUT YW	YAHOO_WEATHER_DATA (Wetterdaten)
INPUT ACTIVATE	BOOL (positive Flanke startet die Übersetzung)
	Der Baustein ersetzt die originalen Anbieterspezifischen Wetter Icons-Nummern durch OSCAT-Standard Icon Nummern. Nach einer positiven Flanke bei ACTIVATE werden die Elemente (Icon-Nummern) in der YAHOO_WEATHER_DATA Datenstruktur ersetzt. Nach erfolgter Abfrage der Wetterdaten mittels YAHOO_WEATHER sollte dieser Baustein darauffolgend aufgerufen werden. Dabei kann einfach der Parameter DONE vom Baustein YAHOO_WEATHER mit ACTIVATE verschalten werden.
Folgende Elemente werden angepasst	
	YW.CUR_CONDITIONS_ICON
	YW.FORCAST_TODAY_ICON
	YW.FORCAST_TOMORROW_ICON


???
YAHOO_WEATHER_ICON_OSCAT
ACTIVATE
YW
YW