Knihovna InternetLib

TXV 003 54.01
sedmnácté vydání
červenec 2022
změny vyhrazeny

Historie změn

Datum	Vydání	Popis změn
Říjen 2009	1	První vydání
Únor 2010	2	Doplněny změny pro verzi knihovny InternetLib 1.2
Květen 2010	3	Doplněny změny pro verzi knihovny InternetLib 1.3
Říjen 2010	4	Opravena velikost zóny pro fbHttpRequest
Prosinec 2010	5	Přidán očekávaný kód odpovědi pro fbSendToFtp (InternetLib 1.7)
Únor 2012	6	Doplněny změny pro verzi knihovny InternetLib 2.1
Září 2012	7	Doplněny změny pro verzi knihovny InternetLib 2.3
Říjen 2013	8	Doplněny změny pro verzi knihovny InternetLib 2.4
Květen 2013	9	Nahrazeno v příkladech fbNsLookUp blokem fbNsLookUpEx Zvýrazněno nutné nastavení kanálů
Září 2013	10	Doplněny chybové hlášení pro fbNsLookUpEx dle FW7.9
Únor 2014	11	Doplněny změny pro verzi knihovny InternetLib 2.9
Červenec 2014	12	Oprava popisu fbSmtp
Květen 2016	13	Doplněny změny pro verzi knihovny InternetLib 4.1
Leden 2018	14	Doplněny změny pro verzi knihovny InternetLib 4.3
Březen 2018	15	Doplněny změny pro verzi knihovny InternetLib 4.4
Prosinec 2019	16	Doplněny změny pro verzi knihovny InternetLib 5.2
Červenec 2022	17	Doplněny změny pro verzi knihovny InternetLib 5.9

OBSAH

1 Úvod....4
2 Datové typy....5
3 Konstanty....9
4 Překlad doménových jmen....9
4.1 Funkční blok fbNsLookUpEx....9
4.2 Funkční blok fbNsLookUp....12
4.3 Funkční blok fbNsLookUpByTable....15
5 Synchronizace času....18
5.1 Funkční blok fbSntp....18
6 Práce s elektronickou poštou....21
6.1 Funkční blok fbSmtp....21
6.2 Funkční blok fbSmtpDbx....26
7 Komunikace HTTP protokolem....31
7.1 Funkční blok fbHttpRequest....31
7.2 Funkční blok fbHttpRequestL....39
7.3 Funkční blok fbHttpRequestL2....41
7.4 Funkční blok fbHttpRequestL3....43
7.5 Funkční blok fbSplitUrlAddress....46
8 Komunikace FTP protokolem....47
8.1 Funkční blok fbStoreToFtp....47
8.2 Funkční blok fbRetriveFromFtp....51
9 Ověřování dostupnosti - PING....55
9.1 Funkční blok fbPingIP....55
9.2 Funkční blok fbPing....57
10 WebSocket komunikace....59
10.1 Funkční blok fbWebSocketClient....59

1 ÚVOD

Knihovna InternetLib obsahuje sadu funkcí pro práci se službami dostupnými v síti Internet. Knihovnu lze využít se systémy s centrální jednotkou řady K a L s firmware verze 4.9 a vyšší.

Obsažené funkční bloky realizují překlad doménových jmen na IP adresy, synchronizaci času s časovými servery, odesílání emailů SMTP protokolem a základní dotazy HTTP protokolu.

Knihovna využívá některé struktury, funkce a funkční bloky z knihoven FileLib (TXV 003 41) a ComLib (TXV 003 51). Pro správnou funkci musí být tyto knihovny zařazeny v projektu před knihovnou InternetLib.

Knihovna obsahuje následující funkční bloky

fbNsLookUpEx	Překlad doménového jména s podporou firmware (od verze 7.1)
fbNsLookUp	Překlad doménového jména
fbNsLookUpByTable	Překlad více doménových jmen
fbSntp	Synchronizace času
fbSmtp	Odesílání emailů
fbSmtpDbx	Odesílání emailů s přílohou z databoxu
fbSplitUrlAddress	Úprava URL odkazu pro bloky pracující s HTTP protokolem
fbHttpRequest	Komunikace protokolem HTTP
fbHttpRequestL	Komunikace protokolem HTTP s rozšířenou zónou pro metodu POST
fbHttpRequestL2	Komunikace protokolem HTTP s rozšířenou zónou pro metodu POST včetně definice typu
fbHttpRequestL3	Komunikace protokolem HTTP s rozšířenou zónou podporuje metodu PUT
fbStoreToFtp	Ukládání souborů na FTP server
fbRetriveFromFtp	Stažení souborů z FTP serveru
fbPingIP	Ověření spojení mezi PLC a hostitelem dané IP adresy
fbPing	Ověření spojení mezi PLC a hostitelem daného jména
fbWebSocketClient	Komunikace PLC se serverem WebSocket protokolem

Objednací číslo této dokumentace je TXV 003 54.01.

2 DATOVÉ TYPY

V knihovně InternetLib jsou definovány následující datové typy:

Typ	*Popis*	*Základní typ*
TDnsQuery	Struktura dotazu na DNS server	STRUCT
TDnsQueryHeader	Struktura hlavičky dotazu na DNS server	STRUCT
TDnsReply	Struktura odpovědi DNS serveru	STRUCT
TDnsReplyHeader	Struktura hlavičky odpovědi DNS serveru	STRUCT
TFtpStoreState	Stavy komunikace FTP protokolem	ENUM
THttpBuffer	Pole pro přijatá data HTTP protokolem	ARRAY [0..511] OF USINT
THttpPostData	Pole pro data metody POST pro blok *fbHttpRequestL*	ARRAY [0..1535] OF USINT
THttpState	Stavy komunikace HTTP protokolem	ENUM
TNsLookUpItem	Dvojice IP adresa doménové jméno s příznaky	STRUCT
TnsLookUpTable	Pole dvojic IP adresa doménové jméno s příznaky	ARRAY [0..31] OF TNsLookUpItem
TSmtpState	Stavy komunikace SMTP protokolem	ENUM
TStringStreamOrigin	Stavy pro *fbStringStream*	ENUM
T_PING_INFO	Výsledek bloků *fbPingIP* a *fbPing*	STRUCT

Význam hodnot enumerací:

TSmtpState - Stavy komunikace SMTP protokolem		
0	ss_SmtpInit	Inicializace
1	ss_SmtpIdle	Není navázáno spojení, komunikace není aktivní
2	ss_SmtpSetIP	Nastavení IP adresy
3	ss_SmtpTxConnect	Navázání spojení se serverem
4	ss_SmtpRxConnect	Čekání na odezvu serveru číslo 220
5	ss_SmtpTxHelo	Odeslání příkazu HELO
6	ss_SmtpRxHelo	Čekání na odezvu serveru číslo 250
7	ss_SmtpTxAuthlogin	Odeslání příkazu AUTH (žádost o autorizované přihlášení)
8	ss_SmtpRxAuthlogin	Čekání na odezvu serveru číslo 334
9	ss_SmtpTxUserName	Odeslání uživatelského jména
10	ss_SmtpRxUserName	Čekání na odezvu serveru číslo 334
11	ss_SmtpTxPassword	Odeslání uživatelského hesla
12	ss_SmtpRxPassword	Čekání na odezvu serveru číslo 235
13	ss_SmtpTxMailFrom	Odeslání adresy odesilatele emailu (příkaz MAIL FROM)
14	ss_SmtpRxMailFrom	Čekání na odezvu serveru číslo 250
15	ss_SmtpTxRcptTo	Odeslání adres příjemců
16	ss_SmtpRxRcptTo	Čekání na odezvu serveru číslo 250 nebo 251
17	ss_SmtpTxData	Odeslání příkazu DATA
18	ss_SmtpRxData	Čekání na odezvu serveru číslo 354
19	ss_SmtpTxDataFrom	Odesílání těla zprávy - odesílatel
20	ss_SmtpTxDataTo	Odesílání těla zprávy - příjemce
21	ss_SmtpTxDataSubject	Odesílání těla zprávy - předmět
22	ss_SmtpTxMultipart	Odesílání těla zprávy - oddělovač částí
23	ss_SmtpTxDataText	Odesílání těla zprávy - text
24	ss_SmtpTxAttachement	Odesílání těla zprávy - oddělovač přílohy
25	ss_SmtpTxAttachementBody	Odesílání těla zprávy - příloha
26	ss_SmtpTxEndOfMail	Odesílání těla zprávy - konec emailu
27	ss_SmtpRxAck	Čekání na odezvu serveru číslo 250
28	ss_SmtpTxQuit	Odeslání příkazu QUIT pro ukončení spojení
29	ss_SmtpRxClose	Čekání na odezvu serveru číslo 221
30	ss_SmtpRxTimeout	Timeout komunikace vypršel
31	ss_SmtpRxError	Při komunikaci došlo k chybě
32	ss_SmtpTxDate	Odesílání času vytvoření zprávy
33	ss_SmtpTxContentType	Odesílání kódování těla zprávy
34	ss_SmtpTxMessageId	Odesílání unikátního Message-ID

TFtpStoreState - Stavy komunikace FTP protokolem		
0	fss_Init	Inicializace
1	fss_Idle	Není navázáno spojení, komunikace není aktivní
2	fss_OpenFile	Otevření souboru určeného k přenosu
3	fss_IpCom	Nastavení IP adresy pro spojení pro vysílání příkazů
4	fss_Connect	Navazování spojení
5	fss_Rx220	Čekání na odpověď s kódem 220
6	fss_TxUser	Vysílání uživatelského jména
7	fss_RxUser	Čekaní na potvrzení uživatelského jména
8	fss_TxPass	Vysílání hesla
9	fss_RxPass	Čekání na potvrzení přihlášení
10	fss_TxType	Požadavek na binární přenos
11	fss_RxType	Čekání na potvrzení módu přenosu
12	fss_TxPasv	Požadavek na pasivní režim
13	fss_RxPasv	Čekání na potvrzení pasivního režimu s IP adresou a portem
14	fss_TxStor	Příkaz pro uložení souboru na FTP
15	fss_IpDat	Nastavení datového spojení dle parametrů pasivního režimu
16	fss_WaitForOpen	Čekání na otevření spojení pro data
17	fss_TxData	Vysílání dat souboru
18	fss_RxComplete	Příjem potvrzení odvysílaných dat
19	fss_TxQuit	Požadavek na ukončení relace
20	fss_RxQuit	Čekání na potvrzení konce
21	fss_Close	Zavření spojení
22	fss_Timeout	Chybový stav při vypršení času pro odpověď (do verze 2.3)
22	fss_Error	Obecný chybový stav (od verze 2.3)
23	fss_UnexpectedReply	Chybový stav při neočekávané odpovědi serveru
24	fss_CannotOpen	Chybový stav při pokusu o otevření souboru (do verze 2.3)
24	fss_TxCreateDir	Příkaz pro vytvoření adresáře na FTP
25	fss_RxCreateDir	Čekání na potvrzení vytvoření adresáře na FTP
26	fss_ReadDir	Čtení struktury adresáře na paměťové kartě
27	fss_TxRetr	Odeslání požadavku na stažení souboru
28	fss_RxData	Příjem dat souboru
29	fss_TxSize	Vyslání dotazu na velikost souboru
30	fss_RxSize	Příjem odpovědi s velikostí souboru

THttpState - Stavy komunikace HTTP protokolem		
0	hs_HttpIdle	Není navázáno spojení, komunikace není aktivní
1	hs_HttpSetIP	Nastavení IP adresy
2	hs_HttpConnect	Čekání na navázání spojení
3	hs_HttpSend	Posílá se výzva serveru
4	hs_HttpReceivingData	Přijímají se data od serveru
5	hs_HttpSendPost	Odesílání dat metodou POST

3 KONSTANTY

V knihovně InternetLib jsou definovány následující konstanty:

Identifikátor	Typ	Hodnota	Význam
MAX_LEN_WS_BUFFER	UINT	1440	Max. délka bufferu pro WebSocket zprávu

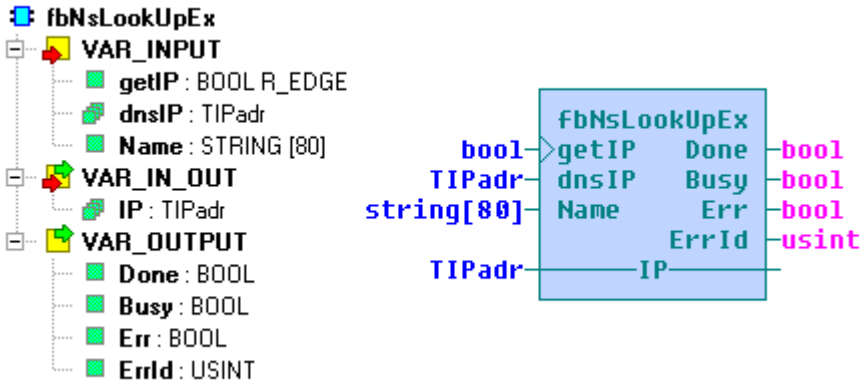
4 PŘEKLAD DOMÉNOVÝCH JMEN

Překlad doménových jmen využívá hierarchického systému doménových jmen DNS (Domain name system) pro získání IP adresy serverů s doménovým jménem.

IP adresy DNS serverů bývají v lokálních sítích stejné s adresou výchozí brány, routeru nebo proxy serveru. Kromě adres lokálních serverů lze použít i adresy přidělené poskytovatelem připojení nebo veřejných DNS serverů. V následujících příkladech se využívá veřejného DNS serveru poskytovaného společností OpenDNS zdarma. Dotazy jsou vysílány protokolem UDP na standardní port 53 (tento port nastavují bloky automaticky).

4.1 Funkční blok fbNsLookUpEx

knihovna: *InternetLib*



Funkční blok *fbNsLookUpEx* slouží pro získání IP adresy podle doménového jména. Blok využívá podporu firmware centrály. Podpora je obsažena v centrálách řady K a L od verze 7.1 a ve všech verzích řady I. Blok na rozdíl od *fbNsLookUp* a *fbNsLookUpByTable* nevyžaduje spojení na ethernet kanálu v režimu UNI.

Simultánně lze volat až sedm instancí funkčního bloku s různými doménovými jmény.

Žádost o IP adresu se vyvolá nastavení vstupu *getIP* na hodnotu TRUE. Žádosti všech instancí se řadí do fronty a jsou vyřizovány v pořadí v jakém byly vyvolány.

IP adresa DNS serveru se předává na vstupu d*nsIP.*

Doménové jméno, které chceme přeložit na IP adresu se zadává na vstupu *Name*.

Po dobu dotazu na DNS server je nastaven výstup *Busy*. V případě úspěšného dotazu je na jeden cyklus nastaven výstup *Done*. Pokud dotaz z nějakého důvodu selže, je nastaveny výstupy *Err* a *ErrId*. Hodnota *ErrId* určuje typ chyby, která nastala. Jednotlivé hodnoty jsou vysvětleny v popisu proměnných.

Popis proměnných :

	Proměnná	*Typ*	*Význam*
VAR_INPUT			
	getIP	BOOL R_EDGE	Řídící proměnná. Náběžná hrana (přechod z hodnotu FALSE na hodnotu TRUE) zahájí žádost o získání IP adresy
	dnsIP	TIPadr	IP adresa DNS serveru
	Name	STRING	Doménové jméno
VAR_IN_OUT			
	IP	TIPadr	IP adresa získaná z DNS serveru
VAR_OUTPUT			
	Done	BOOL	Má hodnotu TRUE v okamžiku kdy je získána IP adresa Jinak vrací FALSE
	Busy	BOOL	Příznak průběhu získávání adresy
	Err	BOOL	Příznak chyby Pokud operace dopadla úspěšně má hodnotu FALSE, jinak TRUE.
	ErrID	USINT	Chybový kód: ErrID = 0 operace dopadla úspěšně ErrID = 1 vypršel čas pro odpověď serveru ErrID = 2 chybný formát – DNS server nebyl schopen interpretovat dotaz ErrID = 3 selhání serveru – DNS server nebyl schopen zpracovat dotaz, kvůli problémům serveru ErrID = 4 chybné jméno – jméno odkazované v dotazu neexistuje ErrID = 5 není implementováno – DNS server nepodporuje tento typ dotazu ErrID = 6 odmítnuto – DNS server odmítl zpracovat dotaz na základě svých pravidel ErrID = 7–16 jiná chyba – rezervováno pro budoucí užití ErrID = 17 neplatné jméno – doménové jméno je příliš dlouhé nebo prázdné ErrID = 18 DNS tabulka je plná – příliš mnoho dotazů ErrID = 19 Odpověď s jiným číslem relace ErrID = 20 Chyba dekódování odpovědi ErrID = 254 nulová adresa DNS serveru

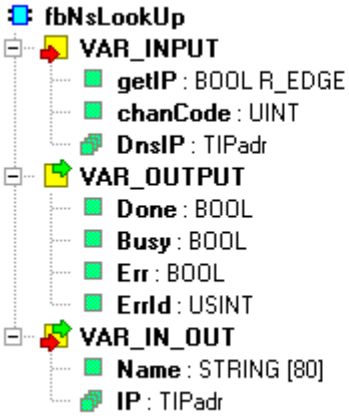
Příklad programu s voláním funkčního bloku *fbNsLookUp* :

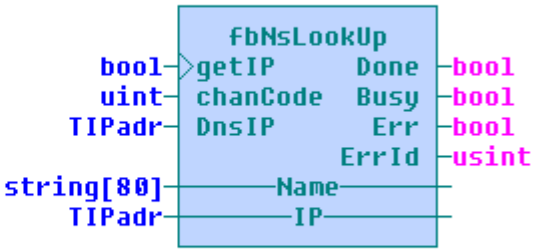
Proměnná *GetNtpIp* vyvolává žádost o IP adresu časového serveru, jehož doménové jméno je dáno proměnnou *DomName.* V případě úspěšného obdržení adresy je nastaven bit *NtpIpReady* na hodnotu TRUE.

```
VAR_GLOBAL
  GetNtpIP   : BOOL;
  NtpIpReady : BOOL;
END_VAR

PROGRAM prgExampleNsLookUpEx
  VAR
    NsLookUpEx : fbNsLookUpEx;
    ServerIP   : TIPadr;
    RSReady    : RS;
  END_VAR

  NsLookUpEx(getIP := GetNtpIP,
             DnsIP := STRING_TO_IPADR('208.67.222.222'),
             Name := 'cz.pool.ntp.org',
             IP := ServerIP);

  RSReady(S := NsLookUpEx.Done, R1 := NsLookUpEx.Err, Q1 => NtpIpReady);

END_PROGRAM
```

4.2 Funkční blok fbNsLookUp

knihovna: *InternetLib*





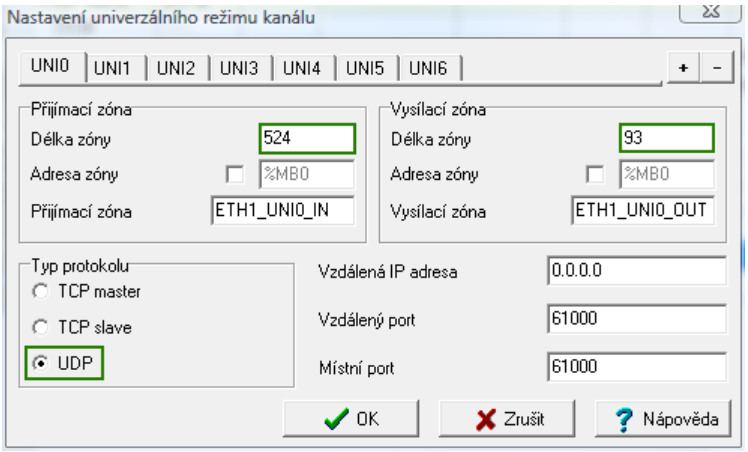
V nových projektech doporučujeme použít novější blok *fbNsLookUpEx.*

Funkční blok *fbNsLookUp* slouží pro získání IP adresy podle doménového jména. Žádost o IP adresu se vyvolá nastavení vstupu *getIP* na hodnotu TRUE. Žádost se provede přes spojení na ethernet kanálu v režimu UNI podle konstanty na vstupu *chanCode*. Spojení musí mít následujícími parametry: režim UDP, délka přijímací zóny 524 bytů, délka vysílací zóny 93 bytů. Pokud spojení není aktivní nebo nemá správné délky zón blok indikuje chybu na výstupech *Err* hodnotou TRUE a *ErrId* hodnotou 255.

IP adresa DNS serveru se předává na vstupu *DnsIP*, doménové jméno, které chceme přeložit na IP adresu je zadáno přes proměnou na vstupu *Name*.

Po dobu dotazu na DNS server je nastaven výstup *Busy*. V případě úspěšného dotazu je na jeden cyklus nastaven výstup *Done*. Pokud dotaz z nějakého důvodu selže, je nastaveny výstupy *Err* a *ErrId*. Hodnota *ErrId* určuje typ chyby, která nastala. Jednotlivé hodnoty jsou vysvětleny v popisu proměnných.

Pokud je v programu třeba získávat z DNS serveru více IP adres je vhodné využít blok *fbNsLookUpByTable*.



Nastavení spojení na ethernetovém kanálu v režimu UNI pro funkční blok fbNsLookUp

Popis proměnných :

	Proměnná	Typ	Význam
VAR_INPUT			
	getIP	BOOL R_EDGE	Řídící proměnná. Náběžná hrana (přechod z hodnotu FALSE na hodnotu TRUE) zahájí žádost o získání IP adresy
	chanCode	UINT	Kód spojení ETH1_uni0, ETH1_uni1,...
	DnsIP	TIPadr	IP adresa DNS serveru
VAR_IN_OUT			
	Name	STRING	Doménové jméno
	IP	TIPadr	IP adresa získaná z DNS serveru
VAR_OUTPUT			
	Done	BOOL	Má hodnotu TRUE v okamžiku kdy je získána IP adresa Jinak vrací FALSE
	Busy	BOOL	Příznak průběhu získávání adresy
	Err	BOOL	Příznak chyby Pokud operace dopadla úspěšně má hodnotu FALSE, jinak TRUE.
	ErrId	USINT	Chybový kód: *ErrId* = 0 operace dopadla úspěšně *ErrId* = 1 vypršel čas pro odpověď serveru *ErrId* = 2 chybný formát – DNS server nebyl schopen interpretovat dotaz *ErrId* = 3 selhání serveru – DNS server nebyl schopen zpracovat dotaz, kvůli problémům serveru *ErrId* = 4 chybné jméno – jméno odkazované v dotazu neexistuje *ErrId* = 5 není implementováno – DNS server nepodporuje tento typ dotazu *ErrId* = 6 odmítnuto – DNS server odmítl zpracovat dotaz na základě svých pravidel *ErrId* = 7–16 jiná chyba – rezervováno pro budoucí užití *ErrId* = 254 nulová adresa DNS serveru *ErrId* = 255 chybné nastavení spojení na ethernet kanálu

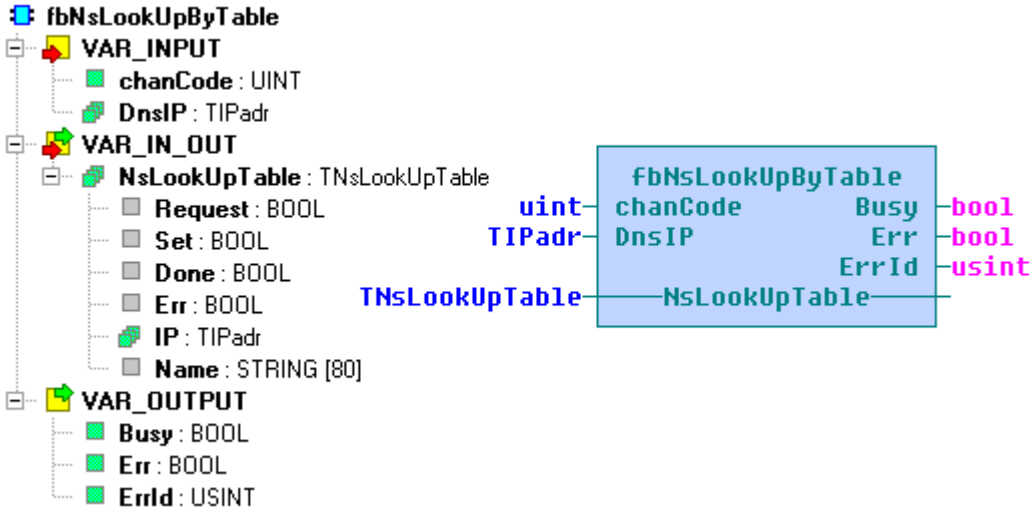
Příklad programu s voláním funkčního bloku *fbNsLookUp* :

Proměnná *GetNtpIp* vyvolává žádost o IP adresu časového serveru, jehož doménové jméno je dáno proměnnou *DomName.* V případě úspěšného obdržení adresy je nastaven bit *NtpIpReady* na hodnotu TRUE.

```
VAR_GLOBAL
  GetNtpIP   : BOOL;
  NtpIpReady : BOOL;
END_VAR

PROGRAM prgExampleNsLookUp
  VAR
    NsLookUp : fbNsLookUp;
    DomName  : STRING := 'cz.pool.ntp.org';
    ServerIP : TIPadr;
    RSReady  : RS;
  END_VAR

  NsLookUp(getIP := GetNtpIP,
           chanCode := ETH1_uni0,
           DnsIP := STRING_TO_IPADR('208.67.222.222'),
           Name := DomName,
           IP := ServerIP);

  RSReady(S := NsLookUp.Done, R1 := NsLookUp.Err, Q1 => NtpIpReady);

END_PROGRAM
```

4.3 Funkční blok fbNsLookUpByTable

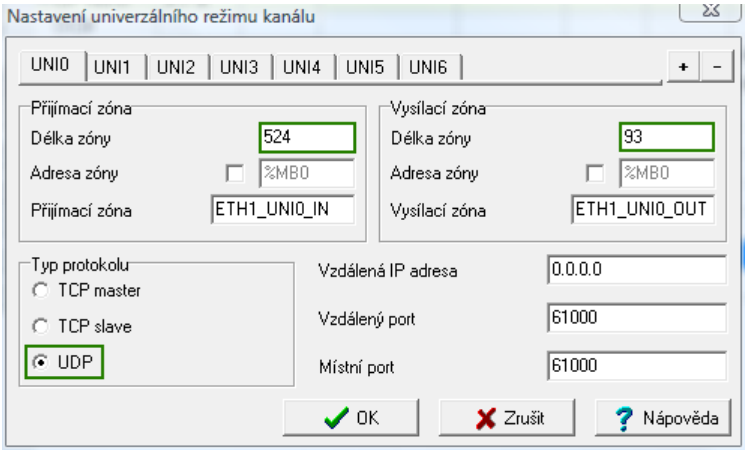
knihovna: *InternetLib*



Funkční blok *fbNsLookUpByTable* slouží k získání více IP adres podle doménových jmen přes jedno spojení. Blok vyžaduje spojení na ethernet kanálu v režimu UNI. Spojení musí mít následujícími parametry: režim UDP, délka přijímací zóny 524 bytů, délka vysílací zóny 93 bytů. Pokud spojení není aktivní nebo nemá správné délky zón blok indikuje chybu na výstupech *Err* hodnotou TRUE a *ErrId* hodnotou 255.

Blok má tři vstupy. Vstup *chanCode* určuje, přes které spojením bude blok pracovat, *DnsIP* adresu DNS serveru, od kterého budou informace získávány a *NsLookUpTable* odkazuje na strukturu s příznaky požadavků, doménovými jmény a IP adresami.

Struktura *NsLookUpTable* pojme až 32 páru doménové jméno, IP adresa. Každá tato dvojice je opatřena sadou bitových příznaků. Nastavením bitu *Request* dojde k zařazení požadavku na získání příslušné IP adresy. Tento bit je okamžitě po přijetí požadavku nulován. Ve chvíli, kdy je IP adresa získána, je nastaven bit *Done* a *Set.* Bit *Done* se nuluje v následujícím cyklu, bit *Set* až při dalšího požadavku. V případě chyby je nastavena proměnná *Err*. S proměnnou *Err* ve struktuře se nastavují společně i výstupy bloky *Err* a *ErrId* s upřesňujícím kódem chyby (Od verze knihovny 1.2 jsou v *ErrId* kódovány příznaky z „Response code“ uváděné RFC1035 pro lepší identifikaci problému. Kódování je provedeno dle klíče *ErrId* = „Response code“+1. Starší verze vrací *ErrId* = 2 při jakémkoli nenulovém „Response code“). Po dobu komunikace je nastaven výstup *Busy*.



Nastavení spojení na ethernetovém kanálu v režimu UNI pro funkční blok fbNsLookUpByTable

Popis proměnných :

	Proměnná	**Typ**	**Význam**
VAR_INPUT			
	chanCode	UINT	Kód spojení ETH1_uni0, ETH1_uni1,...
	DnsIP	TIPadr	IP adresa DNS serveru
VAR_IN_OUT			
	NsLookUpTable	T*NsLookUpTable*	Tabulka doménových jmen, příznaků a IP adres
	Request	BOOL	Bitový příznak žádosti o adresu
	Set	BOOL	Bitový příznak úspěšného získání IP adresy
	Done	BOOL	Náběžná hrana příznaku *Set*
	Err	BOOL	Bitový příznak chyby při získání IP adresy
	IP	TIPadr	IP adresa získaná z DNS serveru
	Name	STRING	Doménové jméno, ke kterému se hledá IP adresa
VAR_OUTPUT			
	Busy	BOOL	Má hodnotu TRUE po dobu komunikace s DNS serverem. Jinak vrací FALSE
	Err	BOOL	Příznak chyby Pokud poslední operace dopadla úspěšně má hodnotu FALSE, jinak TRUE.
	ErrID	USINT	Chybový kód: stejné jako *fbNsLookUp*

Příklad programu s voláním funkčního bloku fbNsLookUpByTable:

V následujícím příkladu jsou získány tři IP adresy níže uvedených adres po startu systému (to je zajištěno inicializací bitů *Request*). Příklad neukazuje použití příznakových bitů *Done* a *Set*. Tyto příznaky mohou být použity kdekoli dále v programu. Příznak *Done* lze použít pro odstartování akce okamžitě po získání IP adresy. Příkaz *Set* lze využít pro kontrolu, zda-li byla IP adresa úspěšně získána a je možné jí použít pro další komunikace.

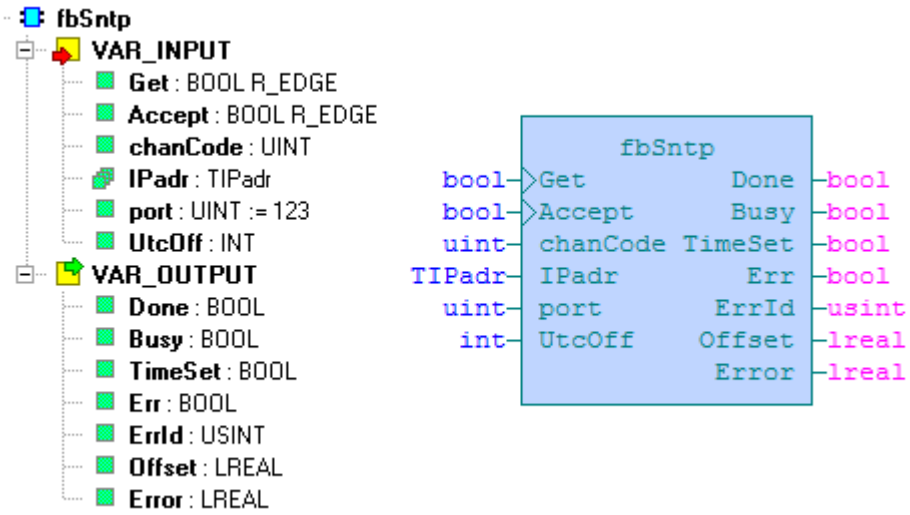
```
VAR_GLOBAL
  LookUpTable : TNsLookUpTable :=
    [(Request:= true, Name:= 'cz.pool.ntp.org'),
     (Request:= true, Name:= 'smtp.iol.cz'),
     (Request:= true, Name:= 'kamera.mukolin.cz')];
END_VAR

PROGRAM prgExampleNsLookUpByTable
  VAR
    NsLookUpByTable : fbNsLookUpByTable;
  END_VAR

  NsLookUpByTable(chanCode := ETH1_uni0,
                  DnsIP := STRING_TO_IPADR('208.67.222.222'),
                  NsLookUpTable := LookUpTable);

END_PROGRAM
```

5 SYNCHRONIZACE ČASU

Synchronizace času využívá SNTP (Simple Network Time Protocol) protokolu k získání časového rozdílu interních hodin proti času na časovém serveru. Tento rozdíl lze využít pro seřizování systémového času. Časový server může být provozován v lokální síti nebo lze využít veřejné servery. Seznam veřejných serverů lze najít na internetové adrese support.ntp.org. Dotazy jsou vysílány protokolem UDP na standardní port 123 (tento port nastavují bloky automaticky).

5.1 Funkční blok fbSntp

knihovna: *InternetLib*

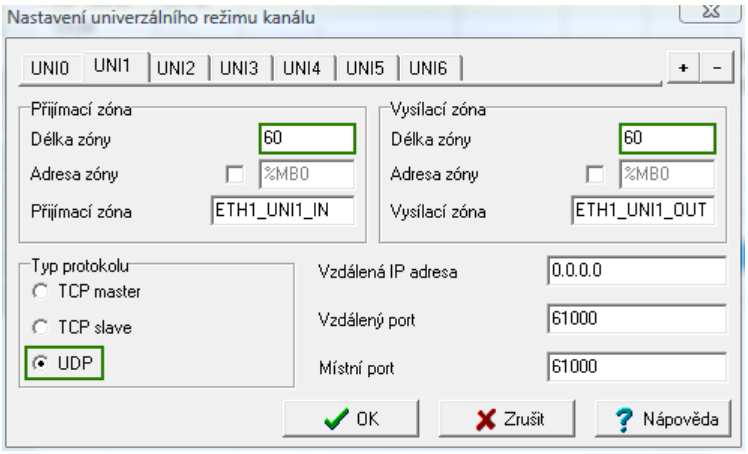


Funkční blok *fbSntp* slouží k získání časové rozdílu mezi serverem a systémovým časem PLC. Žádost o získání časové diference se vyvolá nastavení vstupu *Get* na hodnotu TRUE. Žádost se provede přes spojení na ethernet kanálu v režimu UNI podle konstanty na vstupu *chanCode*. Spojení musí mít následujícími parametry: režim UDP, délka přijímací a vysílací zóny 60 bytů. Pokud spojení není aktivní nebo nemá správné délky zón, blok indikuje chybu na výstupech *Err* hodnotou TRUE a *ErrId* hodnotou 255.

Adresa časového serveru se předává na vstupu *IPadr* a port, na kterém server přijímá požadavky se nastavuje vstupem *port* (výchozí hodnota pro protokol SNTP je 123). Na vstupu *UtcOff* se očekává posun časové zóny proti GMT v minutách. Pokud je v systému aktivován automatický přechod na letní čas, blok sám připočítá hodinový rozdíl k zadanému posunu.

Během žádosti o časový rozdíl je nastaven výstup *Busy*. Při úspěšném dokončení operace se objeví na výstupu *Offset* získaný časový rozdíl, na výstupu *Error* maximální chyba získaného rozdílu a je nastaven na jeden cyklus výstup *Done*. V případě neúspěchu je nastaven výstup *Err* a *ErrId*, kde je upřesňující kód chyby.

Po úspěšném získání časového rozdílu, lze s jeho pomocí synchronizovat systémový čas PLC nastavením vstupu *Accept* na hodnotu TRUE. Pokud je vstup *Accept* nastaven na hodnotu TRUE je systémový čas nastaven okamžitě po úspěšném získání časového rozdílu. Pokud byl časový rozdíl již úspěšně získán je provedena korekce systémového času s náběžnou hranou na vstupu *Accept*. Úspěšné nastavení systémového času PLC dle získaného časového rozdílu je indikováno nastavením výstupu *TimeSet*.



Nastavení spojení na ethernetovém kanálu v režimu UNI pro funkční blok fbSntp

Popis proměnných:

	Proměnná	Typ	Význam
VAR_INPUT			
	Get	BOOL R_EDGE	Řídící proměnná. Náběžná hrana zahájí žádost o časový rozdíl
	Accept	BOOL R_EDGE	Nastavení času dle získaného ofsetu.
	chanCode	UINT	Kód spojení ETH1_uni0, ETH1_uni1,...
	IPadr	TIPadr	IP adresa časového serveru
	port	UINT	Port časového serveru (výchozí hodnota pro protokol SNTP je 123)
	UtcOff	INT	Posun časové zóny proti GMT v minutách
VAR_OUTPUT			
	Done	BOOL	Má hodnotu TRUE v okamžiku kdy je úspěšně získán časový rozdíl. Jinak vrací FALSE
	Busy	BOOL	Má hodnotu TRUE po dobu získávání časového rozdílu
	TimeSet	BOOL	Má hodnotu TRUE pokud byl poslední získaný časový rozdíl použitý pro nastavení systémového času
	Err	BOOL	Příznak chyby Pokud poslední operace dopadla úspěšně má hodnotu FALSE, jinak TRUE.
	ErrId	USINT	Chybový kód: *ErrId* = 0 operace dopadla úspěšně *ErrId* = 1 vypršel čas pro odpověď serveru *ErrId* = 2 z odpovědi serveru se nepodařilo určit časový rozdíl *ErrId* = 254 nulová adresa časového serveru *ErrId* = 255 chybné nastavení spojení na ethernet kanálu
	Offset	LREAL	Získaný časový rozdíl
	Error	LREAL	Maximální chyba získaného časového rozdílu

Následující příklad ukazuje použití funkčního bloku *fbSntp* pro získání přesného času. Program každý den pět minut před půlnocí zažádá o IP adresu časového serveru, podle kterého nastaví systémový čas. Příklad využívá funkci *GetTime* z knihovny SysLib.

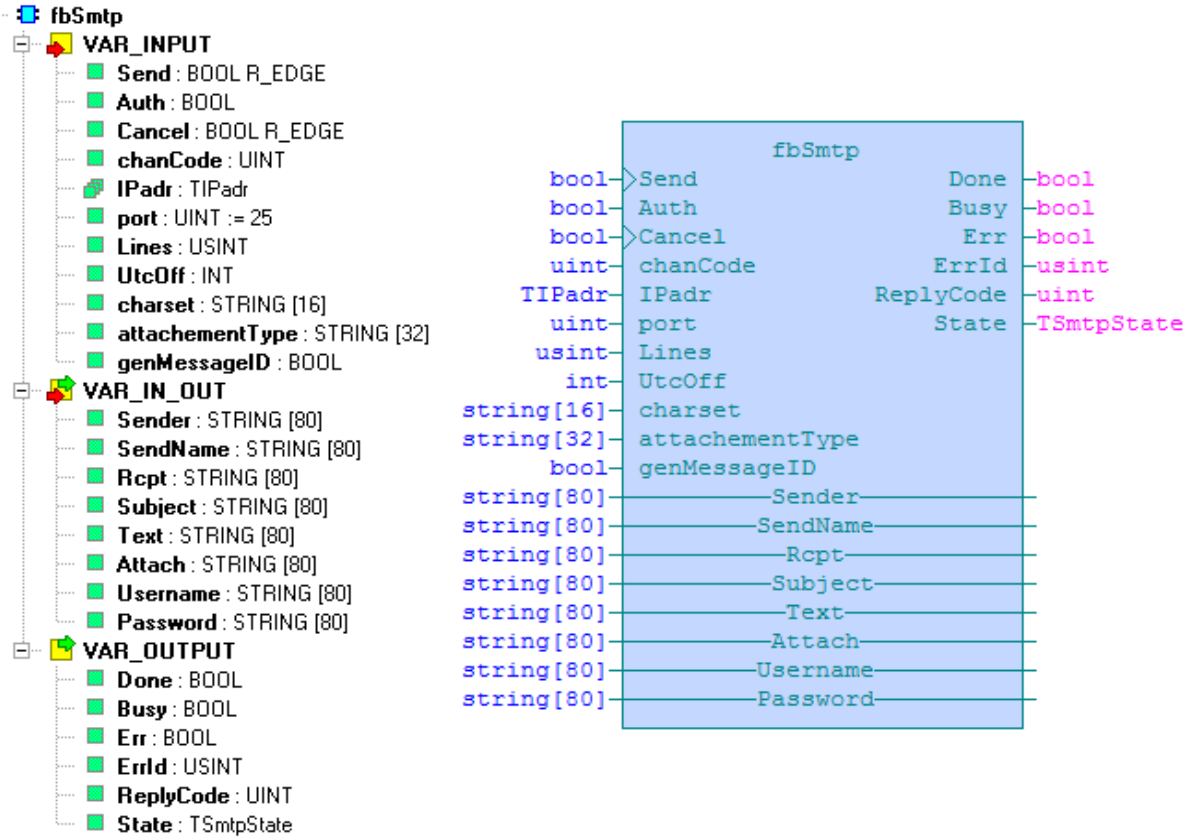
```
VAR_GLOBAL
  NtpName : STRING := 'cz.pool.ntp.org';
  NtpIP : TIPadr;
END_VAR

PROGRAM prgExampleSntp
  VAR_INPUT
  END_VAR
  VAR
    NsLookUp : fbNsLookUpEx;
    Sntp     : fbSntp;
    now      : TIME;
  END_VAR
  VAR_OUTPUT
  END_VAR
  VAR_TEMP
  END_VAR

  now := GetTime();

  NsLookUp(getIP := now > T#23:55:00.0,
           DnsIP :=  STRING_TO_IPADR('208.67.222.222'),
           Name := NtpName,
           IP := NtpIP);

  Sntp(Get := NsLookUp.Done, Accept := Sntp.Done, chanCode := ETH1_uni1,
      IPadr := NtpIP, UtcOff := 60);

END_PROGRAM
```

6 PRÁCE S ELEKTRONICKOU POŠTOU

Knihovna nabízí blok pro odesílání elektronické pošty pomocí protokolu SMTP. Jména SMTP serverů zveřejňují poskytovatelé emailových služeb.

6.1 Funkční blok fbSmtp

knihovna: *InternetLib*



Funkční blok *fbSmtp* slouží k odesílání emailových zpráv SMTP protokolem. Odesílání zprávy se zahájí nastavení vstupu *Send* na hodnotu TRUE. Odeslání se provede přes spojení na ethernet kanálu v režimu UNI podle konstanty na vstupu *chanCode*. Spojení musí mít následujícími parametry: režim TCP master, délka přijímací a vysílací zóny 255 bytů. Pokud spojení není aktivní nebo nemá správné délky zón, blok indikuje chybu na výstupech *Err* hodnotou TRUE a *ErrId* hodnotou 255.

Adresa SMTP serveru se předává na vstupu *IPadr* a port, na kterém server přijímá požadavky se nastavuje vstupem *port* (výchozí hodnota pro protokol SMTP je 25).

Na vstupu *Sender* je očekávána proměnná s emailovou adresou odesílatele, na vstupu *SendName* proměnná se jménem odesílatele, které se má zobrazit příjemci a na vstupu *Rcpt* proměnná s adresami příjemců oddělené středníky.

Vlastní zpráva se předává přes proměnné na vstupu *Subject*, kde se očekává předmět zprávy, a vstupu *Text*. Tělo zprávy musí mít formu pole textových řetězců standardní délky (ARRAY [1..*n*] OF STRING), kde *n* je počet řádků zprávy. Na vstupu *Text* se předává první řádek těla zprávy. Počet řádek, které budou skutečně odeslány, udává vstup *Lines*. Hodnota vstupu *Lines* může být menší nebo rovna *n*.

K odesílané zprávě lze připojit soubor z paměťové karty PLC jako přílohu. Jméno souboru se předává proměnnou na vstupu *Attach*. Pro email bez přílohy je na vstup *Attach* potřeba předat proměnnou s prázdným řetězcem.

Pokud server vyžaduje ověření pomocí uživatelského jména a hesla je nutné nastavit vstup *Auth* na hodnotu TRUE a na vstupech *UserName* a *Password* předat proměnné s uživatelským jménem a heslem. Pokud server ověření nevyžaduje, může být předána proměnná s prázdným řetězcem.

Na vstupu *UtcOff* se uvádí posun časové zóny proti GMT v minutách. Pokud je v systému aktivován automatický přechod na letní čas, blok sám připočítá hodinový rozdíl k zadanému posunu.

Od verze knihovny 3.7 lze na vstupu *charset* specifikovat použité kódování v těle a předmětu zprávy, aby byly správně zobrazeny národní znaky.

Od verze knihovny 4.1 lze definovat přesný typ přílohy na vstupu *attachementType*. Pokud typ není uveden použije se hodnota 'application/octet-stream', který pokrývá libovolná data. Upřesnění typu přílohy může zlepšit hodnocení antispamovými filtry. Výčet všech typů lze najít v dokumentu https://www.iana.org/assignments/media-types/media-types.xhtml.

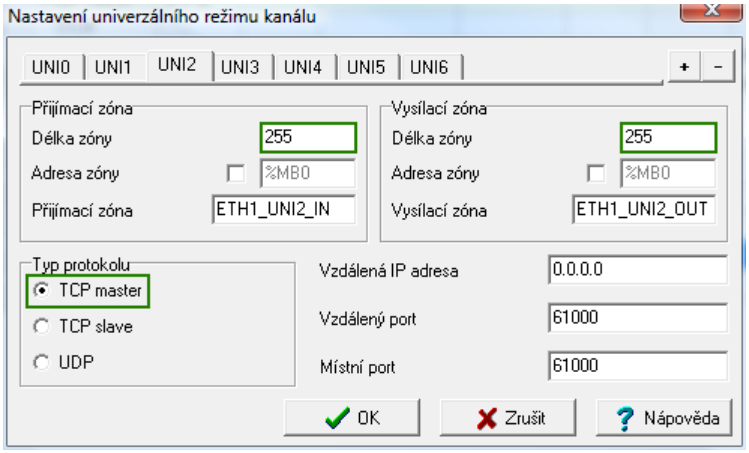
Blok je dále od verze 4.1 schopen generovat unikátní Message-ID. Tato vlastnost se aktivuje nastavením vstupu *genMessageID* na TRUE. Tato volba může zlepšit hodnocení antispamovými filtry v případě, že Message-ID negeneruje použitý SMTP server.

Během odesílání zprávy je nastaven výstup *Busy* na hodnotu TRUE. Na výstupu *State* se aktualizuje stav komunikace ze serverem (viz enumerace *TSmtpState*). Kódy odpovědí serveru jsou vraceny na výstupu *ReplyCode*. Význam jednotlivých kódu je podrobně popsán a vysvětlen v RFC 2821. Obecně platí, že první číslice odpovědi určuje její typ následujícím způsobem:

- **1**yz – **Předběžná pozitivní odpověď** – Příkaz byl přijat, ale vykonání je odloženo. Tuto odpověď používají jen rozšířené příkazy SMTP, které funkční blok nepoužívá
- **2**yz – **Pozitivní odpověď** – Příkaz byl přijat a vykonán. (Například spojení se serverem končí kódem 221)
- **3**yz – **Pozitivní okamžitá odpověď** – Příkaz byl přijat, očekávají se další informace. (Do této skupiny patří například odpovědi při ověřování uživatele 334, nebo při odesílání těla zprávy 354)
- **4**yz – **Dočasná negativní odpověď** – Příkaz nebyl přijat, důvod zamítnutí není trvalý, je možné příkaz zkusit znovu. (Odpovědi s tímto typem kódu jsou většinou příznakem zaneprázdnění nebo nedostatku prostředků na straně poštovního serveru)
- **5**yz – **Trvale negativní odpověď** – Příkaz nebyl přijat, důvod je trvalý, nedoporučuje se opakovat žádost ze stejnými parametry. (Nejčastěji se tento kód v odpovědi objevuje, nedostal-li server ověření, které vyžadoval, nebo pokud ověření uživatele nezdaří.)

V případě úspěšného odeslání zprávy je nastaven na dobu jednoho cyklu výstup *Done*.

V případě chyby je nastaven výstup *Err* a ErrId, kde je upřesňující chybový kód.



Nastavení spojení na ethernetovém kanálu v režimu UNI pro funkční blok fbSmtp

Popis proměnných :

	Proměnná	Typ	Význam
VAR_INPUT			
	Send	BOOL R_EDGE	Řídící proměnná. Náběžná hrana zahájí odesílání emailu.
	Auth	BOOL	Zapíná funkci ověřování uživatele jménem a heslem.
	Cancel	BOOL R_EDGE	Náběžná hrana předčasně ukončí probíhající odesílání.
	chanCode	UINT	Kód spojení ETH1_uni0, ETH1_uni1,...
	IPadr	TIPadr	IP adresa SMTP serveru
	port	UINT	Port časového serveru (výchozí hodnota pro protokol SMTP je 25)
	Lines	USINT	Počet řádků textu k odeslání
	UtcOff	INT	posun časového pásma v minutách
	charset	STRING[16]	znaková sada těla zprávy ('windows-1250', 'UTF-8',...)
	attachement-Type	STRING[32]	uživatelsky definovaný MINE typ přílohy. Není-li specifikován, je použit 'application/octet-stream' (další možné hodnoty jsou 'image/jpeg', 'image/png', 'text/plain; charset=windows-1250',...)
	genMessageID	BOOL	vygeneruje unikátní Message-ID
VAR_IN_OUT			
	Sender	STRING	Emailová adresa odesílatele
	SendName	STRING	Jméno odesilatele zobrazené příjemci (může obsahovat jen základní znaky bez diakritiky)
	Rcpt	STRING	Emailové adresy příjemců oddělené středníky
	Subject	STRING	Předmět zprávy
	Text	STRING	První řádek těla zprávy
	Attach	STRING	Jméno souboru pro připojení k emailové zprávě
	Username	STRING	Uživatelské jméno
	Password	STRING	Uživatelské heslo

	Proměnná	Typ	Význam
VAR_OUTPUT			
	Done	BOOL	Má hodnotu TRUE v okamžiku kdy je email úspěšně odeslán. Jinak vrací FALSE
	Busy	BOOL	Má hodnotu TRUE po dobu odesílání emailu.
	Err	BOOL	Příznak chyby Pokud poslední operace dopadla úspěšně má hodnotu FALSE, jinak TRUE.
	ErrId	USINT	Chybový kód: *ErrId* = 0 operace dopadla úspěšně *ErrId* = 1 vypršel čas pro odpověď serveru *ErrId* = 2 neočekávaná odpověď serveru (více viz ReplyCode) *ErrId* = 3 nelze otevřít soubor, email bude odeslán bez přílohy *ErrId* = 254 nulová adresa SMTP serveru *ErrId* = 255 chybné nastavení spojení na ethernet kanálu
	ReplyCode	UINT	Kód odpovědi SMTP serveru
	State	TSmtpState	Stav komunikace se serverem (viz enumerace TSmtpState)

Následující příklad ukazuje použití funkčního bloku *fbSmtp* pro odeslání emailové zprávy. Proměnná *HeatingIsOn* představuje stav topení (zapnuto/vypnuto), který se porovnává s posledním stavem ukládaným do lokální proměnné *LastHeatingState*. V případě změny stavu je proveden dotaz na DNS server na IP adresu SMTP serveru a sestavena zpráva. Základ zprávy je definován konstantou *BodyTemplate*, do které je doplněn aktuální datum a teploty ze vstupů PLC. K modifikaci těla zprávy jsou využité formátovací funkce z knihovny ToStringLib, aktuální datum a čas je získán funkcí *GetDateTime* z knihovny SysLib.

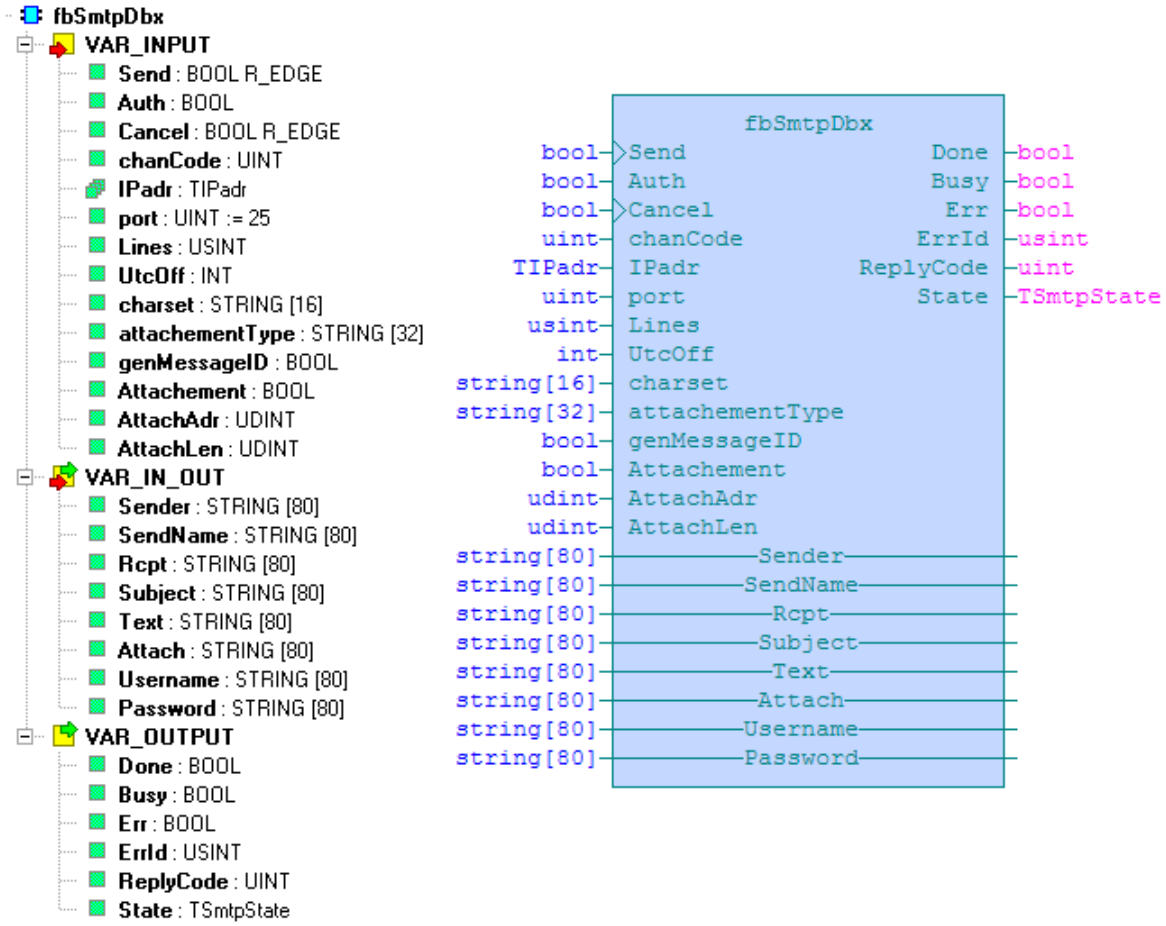
Po úspěšném dotazu na DNS server je zpráva odeslána.

```
VAR_GLOBAL
  SmtpName     : STRING := 'smtp.seznam.cz';
  SmtpIP       : TIPadr;
  TempOutdoor AT r0_p3_AI0.ENG : REAL;
  TempIndoor  AT r0_p3_AI1.ENG : REAL;
  TempHeating AT r0_p3_AI2.ENG : REAL;
  HeatingIsOn : BOOL;
END_VAR

VAR_GLOBAL CONSTANT
  NumberOfLines : USINT := 5;
END_VAR

TYPE
  TEmailBody : ARRAY [1..NumberOfLines] OF STRING;
END_TYPE
```

```
PROGRAM prgExampleSmtp
  VAR
    NsLookUp          : fbNsLookUpEx;
    Smtp              : fbSmtp;
    LastHeatingState  : BOOL;
    Sender            : STRING := 'TestPLC@seznam.cz';
    SenderName        : STRING := 'Do not reply';
    UserName          : STRING := 'TestPLC@seznam.cz';
    Password          : STRING := '********';

    Recipient         : STRING := 'notavailable@seznam.cz';
    Subject           : STRING := 'Heating status report';
    Attachement       : STRING;
    Body              : TEmailBody;

  END_VAR

  IF LastHeatingState <> HeatingIsOn THEN
    Body[1] := DT_TO_STRINGF(in := GetDateTime(),
                format := 'Status report %TDD.MM.YYYY$A0hh:mm');
    IF HeatingIsOn THEN
      Body[2] := 'Heating is switched on';
    ELSE
      Body[2] := 'Heating is switched off';
    END_IF;
    Body[3] := REAL_TO_STRINGF(in := TempOutdoor,
                format := 'Outdoor temperature is %5.1f°C');
    Body[4] := REAL_TO_STRINGF(in := TempIndoor,
                format := 'Indoor temperature is %5.1f°C');
    Body[5] := REAL_TO_STRINGF(in := TempHeating,
                format := 'Heating temperature is %5.1f°C');
  END_IF;

  NsLookUp(getIP := LastHeatingState <> HeatingIsOn,
           DnsIP :=  STRING_TO_IPADR('208.67.222.222'),
           Name := SmtpName,
           IP := SmtpIP);

  LastHeatingState := HeatingIsOn;

  Smtp(Send := NsLookUp.Done, Auth := true,
       chanCode := ETH1_uni2, IPadr := SmtpIP,
       Lines := NumberOfLines, Sender := Sender,
       SendName := SenderName, Rcpt := Recipient,
       Subject := Subject, Attach := Attachement,
       Username := UserName, Password := Password,
       Text := Body[1]);

END_PROGRAM
```

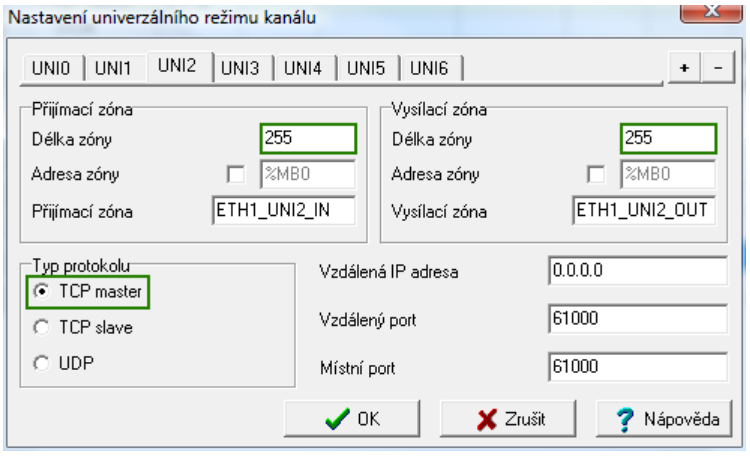
6.2 Funkční blok fbSmtpDbx

knihovna: *InternetLib*



Funkční blok *fbSmtpDbx* je variantou funkčního bloku *fbSmtp* s možností přidat obsah databoxu jako přílohu.

Vstup *Attach* v této variantě neoznačuje soubor v PLC, ale pouze jméno přílohy, ve které bude obsah databoxu uložen. Adresa začátku dat v databoxu je dána vstupem *AttachAdr* a délku určuje *AttachLen*. Zda bude zpráva poslána s přílohou rozhoduje vstup Attachement, který musí být nastaven do TRUE.



Nastavení spojení na ethernetovém kanálu v režimu UNI pro funkční blok fbSmtp

Popis proměnných :

	Proměnná	Typ	Význam
VAR_INPUT			
	Send	BOOL R_EDGE	řídící proměnná. Náběžná hrana zahájí odesílání emailu
	Auth	BOOL	zapíná funkci ověřování uživatele jménem a heslem
	Cancel	BOOL R_EDGE	náběžná hrana předčasně ukončí probíhající odesílání
	chanCode	UINT	kód spojení ETH1_uni0, ETH1_uni1,...
	IPadr	TIPadr	IP adresa SMTP serveru
	port	UINT	port časového serveru (výchozí hodnota pro protokol SMTP je 25)
	Lines	USINT	počet řádků textu k odeslání
	UtcOff	INT	posun časového pásma v minutách
	charset	STRING[16]	znaková sada těla zprávy ('windows-1250', 'UTF-8',...)
	attachementType	STRING[32]	uživatelsky definovaný MINE typ přílohy. Není-li specifikován, je použit 'application/octet-stream' (další možné hodnoty jsou 'image/jpeg', 'image/png', 'text/plain; charset=windows-1250',...)
	genMessageID	BOOL	vygeneruje unikátní Message-ID
	Attachement	BOOL	poslat s přílohou
	AttachAdr	UDINT	adresa přílohy v databoxu
	AttachLen	UDINT	délka přílohy v databoxu v bytech
VAR_IN_OUT			
	Sender	STRING[80]	emailová adresa odesílatele
	SendName	STRING[80]	jméno odesilatele zobrazené příjemci (může obsahovat jen základní znaky bez diakritiky)
	Rcpt	STRING[80]	emailové adresy příjemců oddělené středníky
	Subject	STRING[80]	předmět zprávy
	Text	STRING[80]	první řádek těla zprávy
	Attach	STRING[80]	jméno souboru s přilohou
	Username	STRING[80]	uživatelské jméno pro SMTP server
	Password	STRING[80]	heslo pro SMTP server
VAR_OUTPUT			
	Done	BOOL	má hodnotu TRUE v okamžiku kdy je email úspěšně odeslán. Jinak vrací FALSE
	Busy	BOOL	má hodnotu TRUE po dobu odesílání emailu
	Err	BOOL	příznak chyby pokud poslední operace dopadla úspěšně má hodnotu FALSE, jinak TRUE.

	Proměnná	***Typ***	***Význam***
	ErrId	USINT	chybový kód: *ErrId* = 0 operace dopadla úspěšně *ErrId* = 1 vypršel čas pro odpověď serveru *ErrId* = 2 neočekávaná odpověď serveru (více viz Reply-Code) *ErrId* = 3 nelze otevřít soubor, email bude odeslán bez přílohy *ErrId* = 254 nulová adresa SMTP serveru *ErrId* = 255 chybné nastavení spojení na ethernet kanálu
	ReplyCode	UINT	kód odpovědi SMTP serveru
	State	TSmtpState	stav komunikace se serverem (viz enumerace TSmtpState)

Následující příklad je určen pro PLC řady Foxtrot CP-2xxx a ukazuje možnost využít blok pro odesílání emailů přes zabezpečené spojení. Pro správnou funkci je nutné v nastavení použitého google účtu povolit přístup aplikací s nižším zabezpečením

```
PROGRAM prgExampleSmtpSsl
  VAR
    NsLookUp          : fbNsLookUpEx;
    Smtp : fbSmtpDbx := ( Auth := true, port := 465, Lines := 3,
                          charset := 'windows-1250',
                          attachementType := 'text/plain',
                          genMessageID := true,
                          Attachement := true,
                          AttachAdr := 0);
    SendEmail         : BOOL;
    Sender            : STRING := 'teco@gmail.com';
    SenderName        : STRING := 'Do not reply';
    UserName          : STRING := 'teco@gmail.com';
    Password          : STRING := '*********';
    Recipient         : STRING := 'notavailable@seznam.cz';
    Subject           : STRING;
    Body              : ARRAY [0..2] OF STRING :=
                        ['Dobrý den,',
                         'data naleznete v příloze.',
                         'PLC Foxtrot'];
    Attachement       : STRING := 'record.txt';

    DbxAdr1           : UDINT := 0;
    DbxLen1           : UDINT;
    DbxAdr2           : UDINT := 10240;
    DbxLen2           : UDINT;
    RecordEnabled     : BOOL;
    RecordLine        : STRING[255];
    RecordLineLen     : UDINT;
    UseSecond         : BOOL;
  END_VAR

  IF RecordEnabled AND System_S.R_EDGE_1SEC THEN
    RecordLine := DT_TO_STRINGF(GetDateTime(), '%TYY-MM-DD-hh:mm:ss;') +
                  USINT_TO_STRINGF(System_S.CPU_TEMPERATURE, '%i;') +
                  UINT_TO_STRINGF(System_S.LAST_CYCLE_TIME_100US, '%i;') +
                  TIME_TO_STRINGF(UDINT_TO_TIME(System_S.COUNTER_1MS),
                  '%Thh:mm:ss.zzz;$0D$0A');
    RecordLineLen := LEN(RecordLine);
    IF UseSecond THEN
      WriteToDBx(dataBoxAddress := DbxAdr2 + DbxLen2,
                 length := LEN(RecordLine),
                 varAddress := PTR_TO_UDINT(ADR(RecordLine)));
      DbxLen2 := DbxLen2 + RecordLineLen;
    ELSE
      WriteToDBx(dataBoxAddress := DbxAdr1 + DbxLen1,
                 length := LEN(RecordLine),
                 varAddress := PTR_TO_UDINT(ADR(RecordLine)));
      DbxLen1 := DbxLen1 + RecordLineLen;
    END_IF;
  END_IF;
```

```
  IF DbxLen1 > 10240 - 255 THEN
    UseSecond := true;
    DbxLen2   := 0;
    SendEmail := true;
    Subject   := DT_TO_STRINGF(GetDateTime(), 'Record %TYY-MM-DD-hh:mm');
    Smtp.AttachAdr := DbxAdr1;
    Smtp.AttachLen := DbxLen1;
    DbxLen1   := 0;
  END_IF;

  IF DbxLen2 > 10240 - 255 THEN
    UseSecond := false;
    DbxLen1   := 0;
    SendEmail := true;
    Subject   := DT_TO_STRINGF(GetDateTime(), 'Record %TYY-MM-DD-hh:mm');
    Smtp.AttachAdr := DbxAdr2;
    Smtp.AttachLen := DbxLen2;
    DbxLen2   := 0;
  END_IF;

  NsLookUp(getIP := SendEmail,
           Name := 'smtp.gmail.com',
           IP := Smtp.IPadr);

  SendEmail := false;

  IF NsLookUp.Done THEN
    Smtp.Send := true;
    Smtp.chanCode := OpenUniSocket(protocol := UNI_SSL_CLIENT);
    SetUniLog(chanHandle := Smtp.chanCode,
              logMode := ONE_TIME_LOG + TEXT_LOG_ONLY,
              logSizeKB := 32, logName := 'SEND_EMAIL.LOG');
  END_IF;

  Smtp(UtcOff := System_S.UTC_OFFSET,
       Sender := Sender,
       SendName := SenderName,
       Rcpt := Recipient,
       Subject := Subject,
       Text := Body[0],
       Attach := Attachement,
       Username := UserName,
       Password := Password);

  IF NOT Smtp.Busy AND Smtp.chanCode <> 0 THEN
    CloseUniChannel(chanHandle := Smtp.chanCode);
  END_IF;

END_PROGRAM
```

7 KOMUNIKACE HTTP PROTOKOLEM

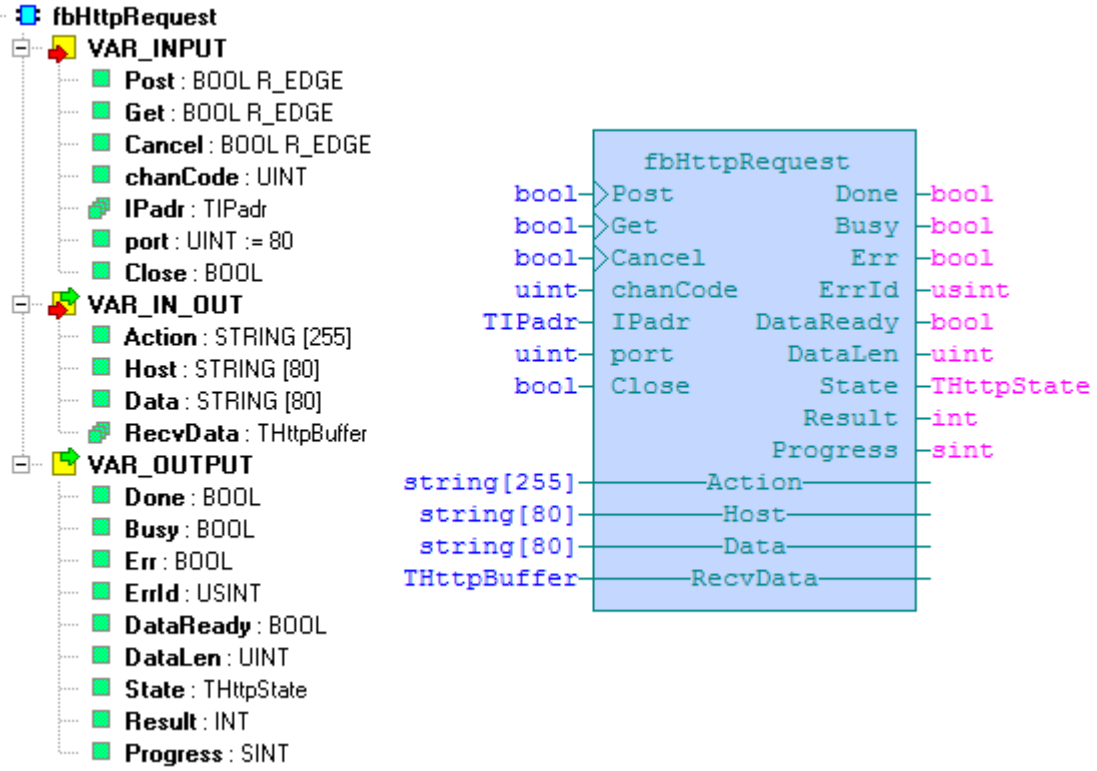
Knihovna nabízí funkční blok pro komunikaci s webovým serverem přes protokol HTTP. Blok implementuje ze souboru metod HTTP protokolu metody GET a POST.

Metoda GET slouží pro získání dat z webového serveru. Typicky lze využít k získání obrazu z IP kamery, stahování receptur ze serveru dispečinku nebo získávání dat z veřejných serverů (údaje o počasí apod.).

Metoda POST se využívá k odesílání dat na webový server. Typické použití je automatizovaný sběr dat vysíláním na centrální server.

7.1 Funkční blok fbHttpRequest

knihovna: *InternetLib*

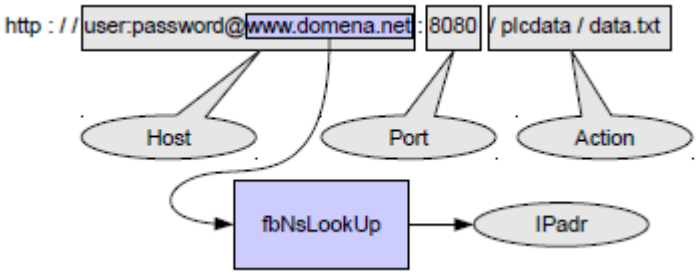


Funkční blok *fbHttpRequest* slouží ke komunikaci s webovým serverem přes protokol HTTP 1.0. Blok implementuje ze souboru metod HTTP protokolu metody GET a POST. Komunikace probíhá přes spojení na ethernet kanálu v režimu UNI podle konstanty na vstupu *chanCode*. Spojení musí mít následujícími parametry: režim TCP master, délka přijímací a vysílací zóny 512 bytů. Pokud spojení není aktivní nebo nemá správné délky zón, blok indikuje chybu na výstupech *Err* hodnotou TRUE a *ErrId* hodnotou 255.

Adresa stahovaných dat se předává na čtyrech vstupech. Na vstupu *IPadr* se očekává adresa serveru (typicky získaná z doménového jména serveru blokem *fbNsLookUp* nebo *fbNsLookUpByTable* nebo *fbNsLookUpEx*), na vstupu *Port* se předává číslo portu, na kterém server naslouchá (výchozí hodnota pro HTTP protokol je 80). Na vstupu *Host* se očekává proměnná s doménovým jménem serveru. Od verze knihovny 2.4 je na vstupu *Host* možné také předat parametry základní autentizace jako součást doménového jména ve tvaru „jméno:heslo@“. Na vstupu *Action* se očekává proměnná s cestou k datům na serveru (cesta vždy začíná znakem lomítko!).

Od verze knihovny 3.4 je možné z adresy naplnit proměnné automaticky pomocí bloku *fbSplitUrlAddress*.

Na obrázku níže je naznačeno, jak souvisí data na adresním řádku webového prohlížeče s hodnotami předávanými na jednotlivých vstupech.



Převod dat z adresního řádku webového prohlížeče na vstupy funkčního bloku
Port nemusí být uveden, v takovém případě má port výchozí hodnotu 80
Parametry základní autentizace user:password@ jsou taktéž nepovinné

Komunikace se podle zvolené metody zahájí nastavením stavu *Get* nebo *Post*. Metoda *Post* očekává proti metodě *Get* navíc data v proměnné předané na vstupu *Data.* Pro snadné zpracování na straně serveru by měla mít proměnná na vstupu *Data* následující formát:

NázevHodnoty1	=	Hodnota1	&	NázevHodnoty2	=	Hodnota_2	&	...	&		NázevHodnotyN	=	HodnotaN

Například: `temp1=20.4&state=1&error=0`

Řetězce v proměnných na vstupu *Action* a *Data* musí být ve formátu URI (Uniform Resource Identifier) dle RFC 2396. Obecně platí, že tyto řetězce mohou obsahovat jen číslice a písmena bez diakritiky, ostatní symboly včetně mezer by měly být kódovány ve tvaru % následované dvěma hexadecimálními číslicemi, které vyjadřují hodnotu znaku v ASCII tabulce (například „%20" je zástupný kód pro mezeru).

Během komunikace je nastaven výstup *Busy* na hodnotu TRUE. V případě úspěšného ukončení je nastaven na jeden cyklus výstup *Done*. V případě neúspěchu je nastaven výstup *Err* a *ErrId*, který obsahuje upřesňující číslo chyby.

V případě chyby *ErrId* = 3 přesáhla velikost hlavičky protokolu velikost vysílacího bufferu. Tento stav nastane pokud jsou proměnné *Host*, *Action* a *Data* v součtu příliš dlouhé. V takovém případě je možné využít zkracovače jmen (např. službu bit.ly) nebo pro metodu post využít blok *fbHttpRequestL*, kde proměnná *Data* není součástí hlavičky.

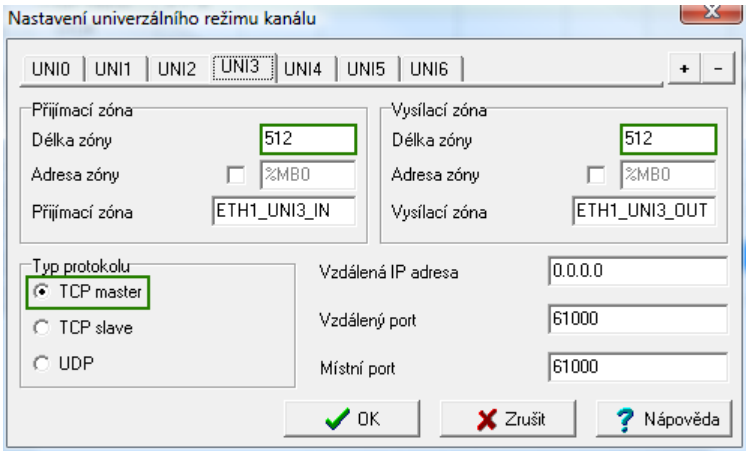
Výstup *State* indikuje aktuální stav komunikace. Po načtení hlavičky zprávy ze serveru se nastaví výstup *Result* se stavovým kódem (viz tabulka níže) a pokud je dostupná délka následujících dat vrací se na výstupu *Progress* průběh stahovaní v procentech (0 až 100). Ve všech jiných stavech nebo pokud není délka známa vrací *Progress* hodnotu -1.

Pokud je nastaven vstup *Close* na TRUE, blok nečeká na server až uzavře spojení, ale zavírání zahájí aktivně sám poté, co přijme všechna data.

Význam nejčastějších stavových kódu na výstupu *Result*

Kód	Význam
200	OK – data nalezena
302	Found – data byla přesunuta
403	Forbidden – přístup odepřen
500	Internal Server Error – vnitřní chyba serveru
další kódy viz RFC 2616	

Data ze serveru přichází v po sobě následujících blocích. Každý cyklus může být vrácen jeden blok. Přítomnost nových dat je signalizována hodnotou TRUE na výstupu *DataReady*. Blok dat se vrací do proměnné na vstupu *RecvData* a jeho délku indikuje výstup *DataLen*.



Nastavení spojení na ethernetovém kanálu v režimu UNI pro funkční blok fbHttpRequest

Popis proměnných :

	Proměnná	*Typ*	*Význam*
VAR_INPUT			
	Post	BOOL R_EDGE	Náběžná hrana zahájí komunikaci metodou POST
	Get	BOOL R_EDGE	Náběžná hrana zahájí komunikaci metodou GET
	Cancel	BOOL R_EDGE	Náběžná hrana přeruší právě probíhající komunikaci
	chanCode	UINT	Kód spojení ETH1_uni0, ETH1_uni1,...
	IPadr	TIPadr	IP adresa webového serveru
	port	UINT	Port webového serveru (výchozí hodnota pro protokol HTTP je 80)
	Close	BOOL	Pokud je nastaveno, blok nečeká na server až zavře spojení a zavře jej aktivně, jakmile získá všechny data
VAR_IN_OUT			
	Action	STRING	Cesta k datům na serveru (vždy začíná znakem /)
	Host	STRING	Doménové jméno serveru
	Data	STRING	Data pro metodu POST
	RecvData	THttpBuffer	Blok přijatých dat
VAR_OUTPUT			
	Done	BOOL	Má hodnotu TRUE v okamžiku kdy je úspěšně ukončena komunikace se serverem. Jinak vrací FALSE
	Busy	BOOL	Má hodnotu TRUE po komunikace se serverem
	Err	BOOL	Příznak chyby Pokud poslední operace dopadla úspěšně má hodnotu FALSE, jinak TRUE.
	ErrId	USINT	Chybový kód: *ErrId* = 0 operace dopadla úspěšně *ErrId* = 1 vypršel čas pro odpověď serveru *ErrId* = 2 nepodařilo se získat všechna data ze serveru *ErrId* = 3 hlavička protokolu HTTP přesáhla 512 bytů *ErrId* = 254 nulová adresa webového serveru *ErrId* = 255 chybné nastavení spojení na ethernet kanálu
	DataReady	BOOL	Hodnota TRUE indikuje nový blok na vstupu *RecvData*
	DataLen	UINT	Délka přijatého bloku dat
	State	THttpState	Stav komunikace se serverem (viz enumerace THttpState)
	Result	INT	Stavový kód vrácený serverem
	Progress	SINT	Během stahování dat ze serveru indikuje průběh 0 až 100%. Jinak vrací -1

Následující příklad ukazuje použití funkčního bloku *fbHttpRequest* pro stažení obrázku z webové kamery. Parametry pro blok *fbHttpRequest* jsou připraveny z URL adresy blokem *fbSplitUrlAddress*. Příklad využívá funkční blok *WriteToFileSeq* z knihovny *FileLib* pro ukládání přijatých dat na paměťovou kartu. Stažení obrázku se iniciuje nastavením proměnné *GetPicture* a je podmíněné úspěšným vytvořením cesty pro ukládání souborů, což indikuje proměnná *PathOk*. Pro vytváření jmen souborů s inkrementujícím se indexem je použita formátovací funkce z knihovny *ToStringLib*.

```
VAR_GLOBAL CONSTANT
  PathTemplate     : STRING := 'WWW/PICT/';
  FileNameTemplate : STRING := PathTemplate + 'PICT%04d.JPG';
END_VAR

VAR_GLOBAL
  Url          : STRING :=
  'http://posta.mukolin.cz/axis-cgi/jpg/image.cgi?resolution=CIF';
  Path         : STRING := PathTemplate;
  FileName     : STRING;

  GetPicture   : BOOL;
  PathOk       : BOOL;
  PictIndx     : INT;
END_VAR

PROGRAM prgExampleHttpGet
  VAR
    Split       : fbSplitUrlAddress;
    HttpIP      : TIPadr;
    HttpName    : STRING;
    Action      : STRING;
    HttpRequest : fbHttpRequest;
    WriteToFile : WriteToFileSeq;
    CPath       : CreatePath;
    Empty       : STRING[1];
    Data        : THttpBuffer;
  END_VAR

  CPath(exec := NOT PathOk, fileName := Path, done => PathOk);

  Split(split := PathOk AND GetPicture,
        urlAddress := Url,
        host := HttpName,
        action := Action,
        ipAdr := HttpIP,
        port := HttpRequest.Port);

  HttpRequest(Get := Split.Done, chanCode := ETH1_uni0,
              IPadr := HttpIP,
              Action := Action,
              Host := HttpName,
              Data := Empty, RecvData := Data);

  FileName := INT_TO_STRINGF(in := PictIndx,
                             format := FileNameTemplate);
```

```
  WriteToFile(fileName := FileName,
              srcVar := void(Data),
              write := HttpRequest.Result = 200 & HttpRequest.DataReady,
              close := HttpRequest.Done OR HttpRequest.Err,
              size := UINT_TO_UDINT(HttpRequest.DataLen));

  IF HttpRequest.Done THEN
     PictIndx := PictIndx + 1;
  END_IF;
END_PROGRAM
```

Druhý příklad ukazuje použití funkčního bloku *fbHttpRequest* pro odesílání dat do databáze na webovém serveru metodou POST. Proměnná *HeatIsOn* představuje stav topení (zapnuto/vypnuto), který se porovnává s posledním stavem ukládaným do lokální proměnné *LastHeatState*. V případě změny stavu funkční blok *fbSplitUrlAddress* získá IP adresu serveru a připraví parametry dle zadané URL adresy.

Ve chvíli, kdy jsou parametry připraveny jsou data transformovaná do řetězce odeslána. Po vyslání dat se v přijaté odpovědi hledá řetězec „OK“, který skript na serveru (uvedený níže) vrací v případě úspěšného uložení dat. Pro kopírování přijatých dat z buferu do řetězce je použita funkce *memcpy* z knihovny *SysLib*.

```
VAR_GLOBAL
  TempOut     AT r0_p3_AI0.ENG : REAL;
  TempIn      AT r0_p3_AI1.ENG : REAL;
  TempHeat    AT r0_p3_AI2.ENG : REAL;
  HeatIsOn                     : BOOL;

  PostUrl : STRING := 'foxtrot.howto.cz/index.php';
END_VAR

PROGRAM prgExampleHttpPost
  VAR
    Split         : fbSplitUrlAddress;
    HttpPostIP     : TIPadr;
    HttpPostName   : STRING;
    HttpPostAction : STRING;
    HttpRequest    : fbHttpRequest;
    DataIn         : THttpBuffer;
    DataInString   : STRING;
    DataOut        : STRING;
    LastHeatState : BOOL;
    PostSuccesful : BOOL;
  END_VAR

  Split.split := LastHeatState <> HeatIsOn;

  IF Split.split THEN
    PostSuccesful := false;
  END_IF;

  Split(urlAddress := PostUrl,
        host := HttpPostName,
        action := HttpPostAction,
        ipAdr := HttpPostIP,
        port := HttpRequest.Port);

  LastHeatState := HeatIsOn;
```

```
  DataOut := 'Heat=' + BOOL_TO_STRING(HeatIsOn) +
             '&TempOut=' + REAL_TO_STRING(TempOut) +
             '&TempIn=' + REAL_TO_STRING(TempIn) +
             '&TempHeat=' + REAL_TO_STRING(TempHeat);

  HttpRequest(Post   := Split.Done, chanCode := ETH1_uni3,
              IPadr  := HttpPostIP,
              Action := HttpPostAction,
              Host   := HttpPostName,
              Data   := DataOut, RecvData := DataIn);

  IF HttpRequest.DataReady THEN
    Memcpy(length := min(80, HttpRequest.DataLen),
           source := VOID(DataIn),
           dest := VOID(DataInString));
    IF FIND(IN1 := DataInString, IN2 := 'OK') > 0 THEN
      PostSuccesful := true;
    END_IF;
  END_IF;

END_PROGRAM
```

Následující PHP skript na straně serveru ukládána data vyslaná metodou POST do SQL databáze. Kromě této funkce navíc skript jako reakci na metodu GET generuje přehledovou tabulku se všemi zaznamenanými daty během aktuálního dne. Soubor je na serveru uložen v souboru *index.php*, na který se odkazuje příklad výše. Proměnné *$db_server*, *$db_name*, *$db_user*, *$db_pass* obsahují údaje pro připojení k serveru s SQL databází. K datům vyslaným z PLC je přistupováno pomocí globální proměnné *$POST*, kde je jako index použitý název hodnoty.

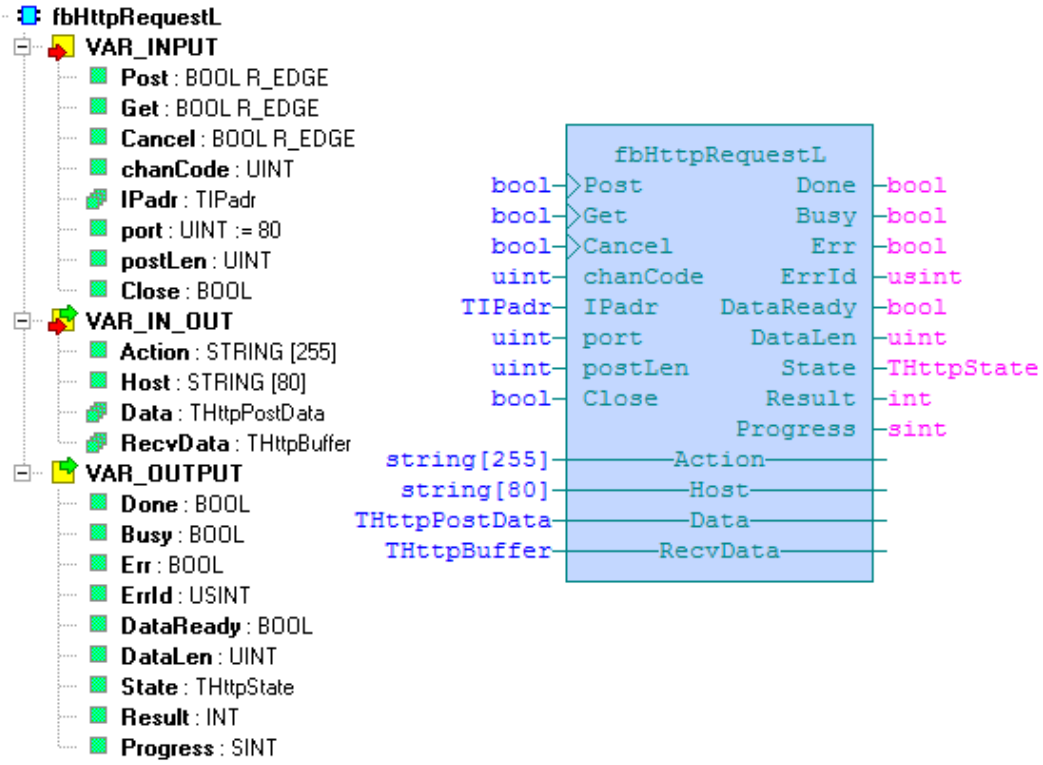
```
<?php
  $db_server = "mysql.ic.cz";
  $db_name   = "ht_foxtrot";
  $db_user   = "ht_foxtrot";
  $db_pass   = "**********";

  $link = mysql_connect($db_server, $db_user, $db_pass)
          or die("ERR - " . mysql_error());
  mysql_select_db($db_name) or die("ERR - unable to select database");

  if(!empty($_POST)) {
    header("Content-type: text/plain");
    $query = "INSERT INTO plc_data VALUES ('".date("Y-m-d-H:i:s").
             "', '".$_POST['Heat']."', '".$_POST['TempOut']."', '".
             $_POST['TempIn']."', '".$_POST['TempHeat']."');";
    $result = mysql_query($query) or die("ERR - " . mysql_error());
    echo "OK";
  } else {
    echo "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.01 Transitional//EN\">";
    echo "<html><head><title>PLC data</title></head><body><center>";

    $query = "SELECT * FROM plc_data WHERE datetime LIKE '".date("Y-m-d")."%';";
    $result = mysql_query($query) or die("ERR - " . mysql_error());

    print "<br><br><table border='1' cellpadding='2' cellspacing='1'>";
    print "<tr><th>Time stamp</th><th>State</th><th>Outdoor temperature</th>".
          "<th>Indoor temperature</th><th>Heating temperature</th></tr>";
    while ($line = mysql_fetch_array($result, MYSQL_ASSOC)) {
      print "<tr><td>".$line['datetime']."</td><td>".$line['Heat'].
            "</td><td>".number_format($line['TempOut'],1)."</td><td>".
            number_format($line['TempIn'],1)."</td><td>".
            number_format($line['TempHeat'],1)."</td></tr>";
    }
    print "</table></center>";
    echo "</body></html>";
  }
  mysql_close($link);
?>
```

Tabulka v databázi využívaná PHP skriptem byla založena následujícím SQL příkazem:

```
CREATE TABLE `plc_data` (
  `datetime` timestamp NOT NULL default CURRENT_TIMESTAMP,
  `Heat` tinyint(1) NOT NULL, `TempOut` double NOT NULL,
  `TempIn` double NOT NULL, `TempHeat` double NOT NULL,
  UNIQUE KEY `datetime` (`datetime`)
);
```

7.2 Funkční blok fbHttpRequestL

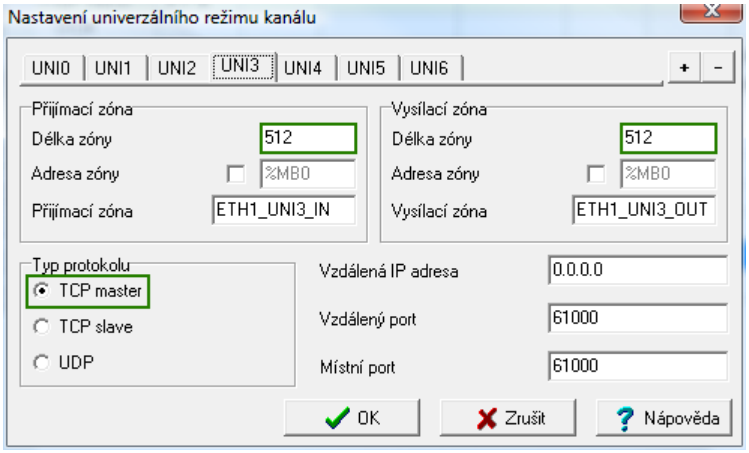
knihovna: *InternetLib*



Funkční blok *fbHttpRequestL* slouží ke komunikaci s webovým serverem přes protokol HTTP 1.0 stejně jako *fbHttpRequest. fbHttpRequestL* se liší od bloku *fbHttpRequest* možností odeslat metodou POST větší množství dat (až 1535 bytů).

Data pro metodu POST se nepředávají jako u *fbHttpRequest* přes typ STRING, ale přes obecnou VAR_IN_OUT proměnnou *Data*. Délka přenášených dat se předává na vstupu *postLen.* Tato délka musí být menší nebo rovna velikosti proměnná předané na vstupu *Data*. Všechny ostatní vstupy a výstupy mají stejný význam a funkci jako u bloku *fbHttpRequest.*

Nastavení spojení a chování je shodné s *fbHttpRequest.*



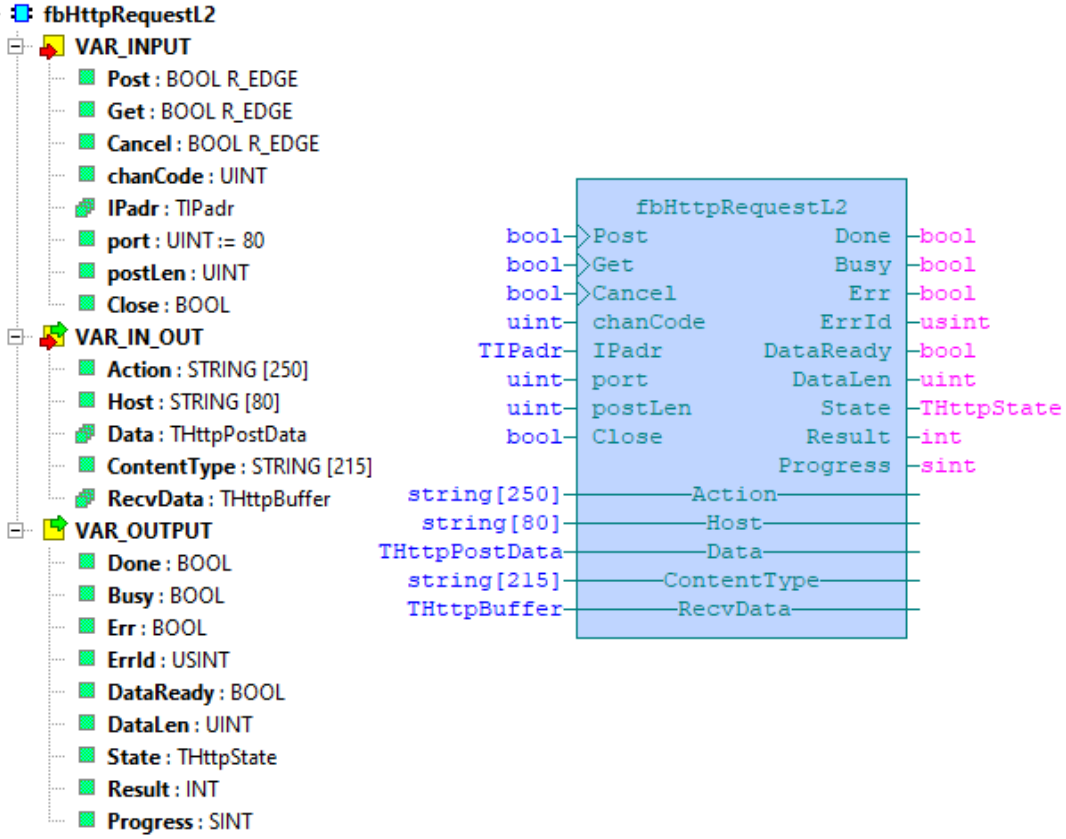
Nastavení spojení na ethernetovém kanálu v režimu UNI pro funkční blok fbHttpRequestL

Popis proměnných :

	Proměnná	*Typ*	*Význam*
VAR_INPUT			
	Post	BOOL R_EDGE	Náběžná hrana zahájí komunikaci metodou POST
	Get	BOOL R_EDGE	Náběžná hrana zahájí komunikaci metodou GET
	Cancel	BOOL R_EDGE	Náběžná hrana přeruší právě probíhající komunikaci
	chanCode	UINT	Kód spojení ETH1_uni0, ETH1_uni1,...
	IPadr	TIPadr	IP adresa webového serveru
	port	UINT	Port webového serveru (výchozí hodnota pro protokol HTTP je 80)
	postLen	UINT	Délka dat pro metodu POST
	Close	BOOL	Pokud je nastaveno, blok nečeká na server až zavře spojení a zavře jej aktivně, jakmile získá všechny data
VAR_IN_OUT			
	Action	STRING	Cesta k datům na serveru (vždy začíná znakem /)
	Host	STRING	Doménové jméno serveru
	Data	THttpPost-Data	Data pro metodu POST
	RecvData	THttpBuffer	Blok přijatých dat
VAR_OUTPUT			
	Done	BOOL	Má hodnotu TRUE v okamžiku kdy je úspěšně ukončena komunikace se serverem. Jinak vrací FALSE
	Busy	BOOL	Má hodnotu TRUE po komunikace se serverem
	Err	BOOL	Příznak chyby Pokud poslední operace dopadla úspěšně má hodnotu FALSE, jinak TRUE.
	ErrId	USINT	Chybový kód: *ErrId* = 0 operace dopadla úspěšně *ErrId* = 1 vypršel čas pro odpověď serveru *ErrId* = 2 nepodařilo se získat všechna data ze serveru *ErrId* = 3 hlavička protokolu HTTP přesáhla 512 bytů *ErrId* = 254 nulová adresa webového serveru *ErrId* = 255 chybné nastavení spojení na ethernet kanálu
	DataReady	BOOL	Hodnota TRUE indikuje nový blok na vstupu *RecvData*
	DataLen	UINT	Délka přijatého bloku dat
	State	THttpState	Stav komunikace se serverem (viz enumerace THttpState)
	Result	INT	Stavový kód vrácený serverem
	Progress	SINT	Během stahování dat ze serveru indikuje průběh 0 až 100%. Jinak vrací -1

7.3 Funkční blok fbHttpRequestL2

knihovna: *InternetLib*



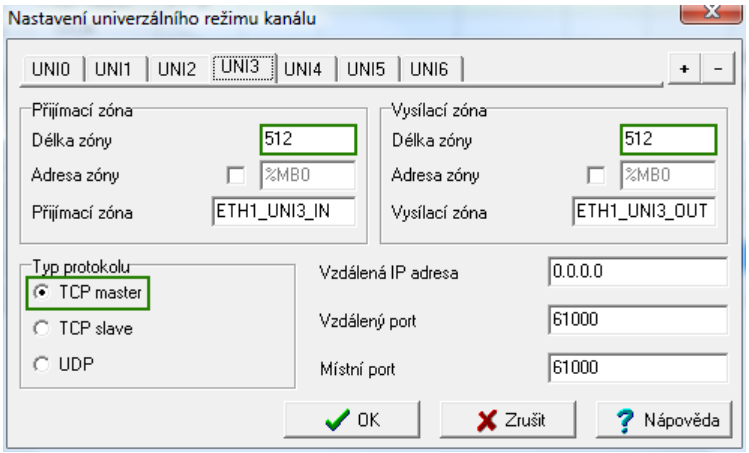
Funkční blok fbHttpRequestL2 je variantou bloku fbHttpRequestL rozšířenou o možnost nastavit typ přenášených dat (Content-Type).

Seznam hodnot, které může proměnná na vstupu *ContentType* nabívat lze nalézt na stránkách http://www.iana.org/assignments/media-types/media-types.xhtml

Samotný blok nijak odesílaná data nezpracovává a je na uživately, aby data byla formátována tak aby odpovídala zvolenému *Content-Type*.

Pokud je na vstupu *ContentType* proměnná obsahující prázdný řetězec, je použita výchozí hodnota *application/x-www-form-urlencoded* a chování se poté plně shoduje s funkčním blokem fbHttpRequestL.

Nastavení spojení je shodné s *fbHttpRequestL*.



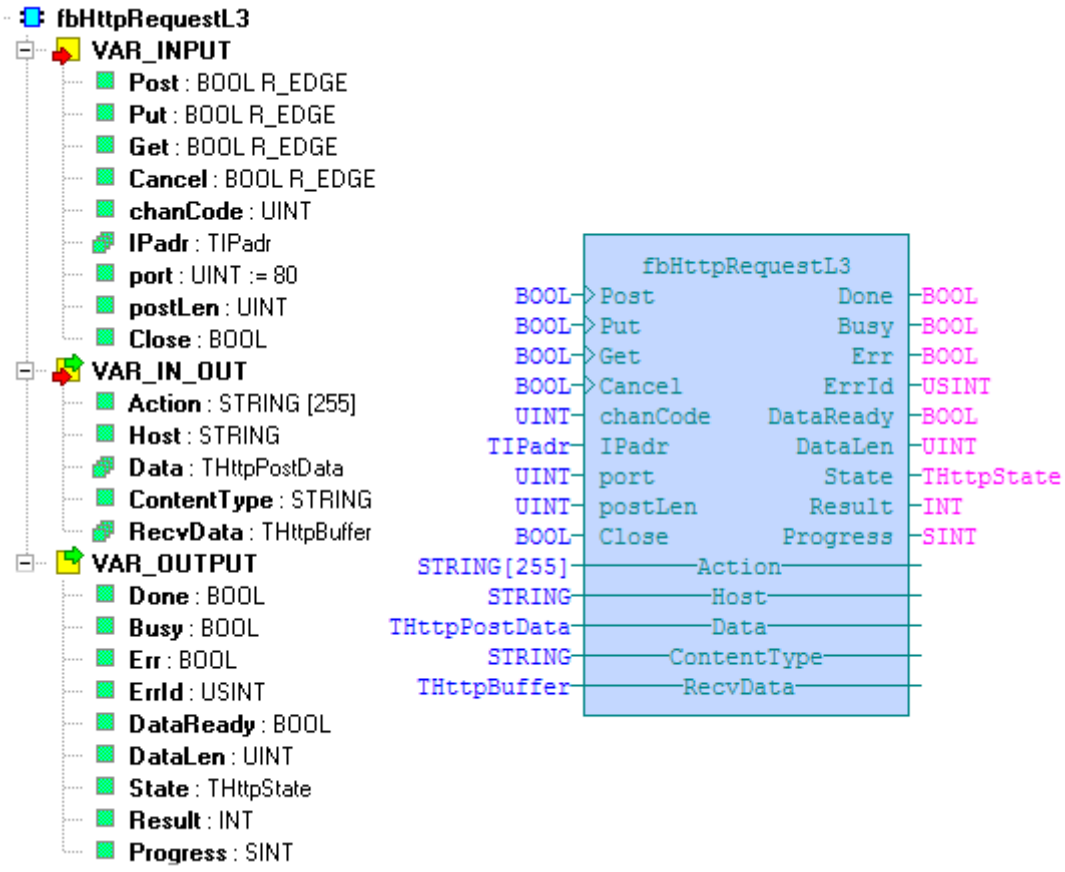
Nastavení spojení na ethernetovém kanálu v režimu UNI pro funkční blok fbHttpRequestL2

Popis proměnných :

	Proměnná	*Typ*	*Význam*
VAR_INPUT			
	Post	BOOL R_EDGE	Náběžná hrana zahájí komunikaci metodou POST
	Get	BOOL R_EDGE	Náběžná hrana zahájí komunikaci metodou GET
	Cancel	BOOL R_EDGE	Náběžná hrana přeruší právě probíhající komunikaci
	chanCode	UINT	Kód spojení ETH1_uni0, ETH1_uni1,...
	IPadr	TIPadr	IP adresa webového serveru
	port	UINT	Port webového serveru (výchozí hodnota pro protokol HTTP je 80)
	postLen	UINT	Délka dat pro metodu POST
	Close	BOOL	Pokud je nastaveno, blok nečeká na server až zavře spojení a zavře jej aktivně, jakmile získá všechny data
VAR_OUTPUT			
	Done	BOOL	Má hodnotu TRUE v okamžiku kdy je úspěšně ukončena komunikace se serverem. Jinak vrací FALSE
	Busy	BOOL	Má hodnotu TRUE po komunikace se serverem
	Err	BOOL	Příznak chyby Pokud poslední operace dopadla úspěšně má hodnotu FALSE, jinak TRUE.
	ErrId	USINT	Chybový kód: *ErrId* = 0 operace dopadla úspěšně *ErrId* = 1 vypršel čas pro odpověď serveru *ErrId* = 2 nepodařilo se získat všechna data ze serveru *ErrId* = 3 hlavička protokolu HTTP přesáhla 512 bytů *ErrId* = 254 nulová adresa webového serveru *ErrId* = 255 chybné nastavení spojení na ethernet kanálu
	DataReady	BOOL	Hodnota TRUE indikuje nový blok na vstupu *RecvData*
	DataLen	UINT	Délka přijatého bloku dat
	State	THttpState	Stav komunikace se serverem (viz enumerace THttpState)
	Result	INT	Stavový kód vrácený serverem
	Progress	SINT	Během stahování dat ze serveru indikuje průběh 0 až 100%. Jinak vrací -1
VAR_IN_OUT			
	Action	STRING[255]	Cesta k datům na serveru (vždy začíná znakem /)
	Host	STRING[80]	Doménové jméno serveru
	Data	THttpPostData	Data pro metodu POST
	ContentType	STRING[215]	Typ obsahu (Content-Type)
	RecvData	THttpBuffer	Blok přijatých dat

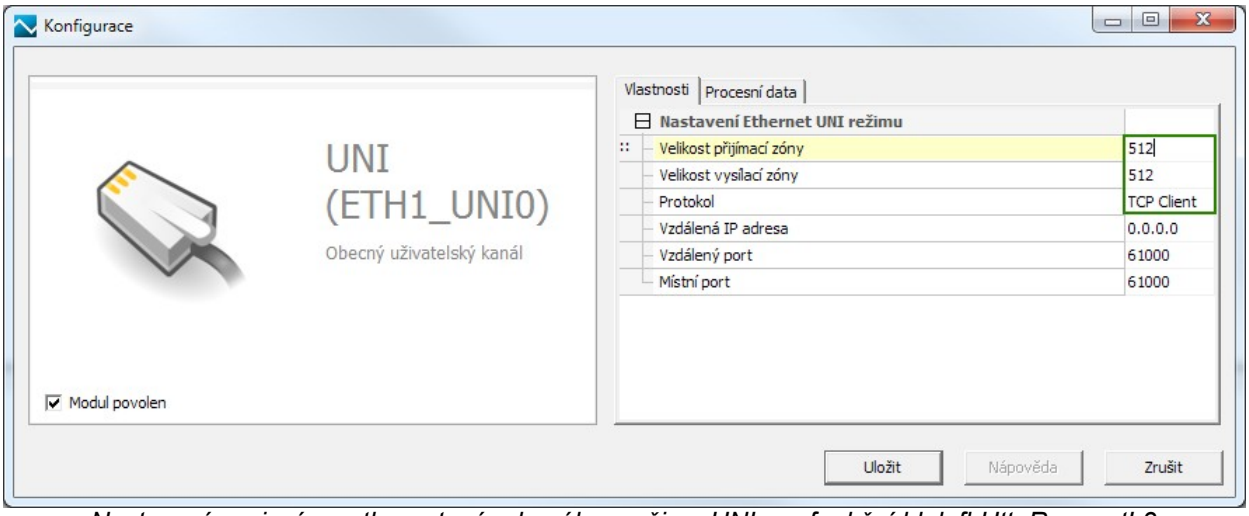
7.4 Funkční blok fbHttpRequestL3

knihovna: *InternetLib*



Funkční blok fbHttpRequestL3 je variantou bloku fbHttpRequestL2 rozšířenou o možnost použít metodu PUT.

Nastavení spojení je shodné s *fbHttpRequestL* a *fbHttpRequestL2*.



Nastavení spojení na ethernetovém kanálu v režimu UNI pro funkční blok fbHttpRequestL3

Popis proměnných :

	Proměnná	*Typ*	*Význam*
VAR_INPUT			
	Post	BOOL R_EDGE	Náběžná hrana zahájí komunikaci metodou POST
	Put	BOOL R_EDGE	Náběžná hrana zahájí komunikaci metodou PUT
	Get	BOOL R_EDGE	Náběžná hrana zahájí komunikaci metodou GET
	Cancel	BOOL R_EDGE	Náběžná hrana přeruší právě probíhající komunikaci
	chanCode	UINT	Kód spojení ETH1_uni0, ETH1_uni1,...
	IPadr	TIPadr	IP adresa webového serveru
	port	UINT	Port webového serveru (výchozí hodnota pro protokol HTTP je 80)
	postLen	UINT	Délka dat pro metodu POST
	Close	BOOL	Pokud je nastaveno, blok nečeká na server až zavře spojení a zavře jej aktivně, jakmile získá všechny data
VAR_OUTPUT			
	Done	BOOL	Má hodnotu TRUE v okamžiku kdy je úspěšně ukončena komunikace se serverem. Jinak vrací FALSE
	Busy	BOOL	Má hodnotu TRUE po komunikace se serverem
	Err	BOOL	Příznak chyby Pokud poslední operace dopadla úspěšně má hodnotu FALSE, jinak TRUE.
	ErrId	USINT	Chybový kód: *ErrId* = 0 operace dopadla úspěšně *ErrId* = 1 vypršel čas pro odpověď serveru *ErrId* = 2 nepodařilo se získat všechna data ze serveru *ErrId* = 3 hlavička protokolu HTTP přesáhla 512 bytů *ErrId* = 254 nulová adresa webového serveru *ErrId* = 255 chybné nastavení spojení na ethernet kanálu
	DataReady	BOOL	Hodnota TRUE indikuje nový blok na vstupu *RecvData*
	DataLen	UINT	Délka přijatého bloku dat
	State	THttpState	Stav komunikace se serverem (viz enumerace THttpState)
	Result	INT	Stavový kód vrácený serverem
	Progress	SINT	Během stahování dat ze serveru indikuje průběh 0 až 100%. Jinak vrací -1
VAR_IN_OUT			
	Action	STRING[255]	Cesta k datům na serveru (vždy začíná znakem /)
	Host	STRING[80]	Doménové jméno serveru
	Data	THttpPostData	Data pro metodu POST
	Content-Type	STRING[80]	Typ obsahu (Content-Type)
	RecvData	THttpBuffer	Blok přijatých dat

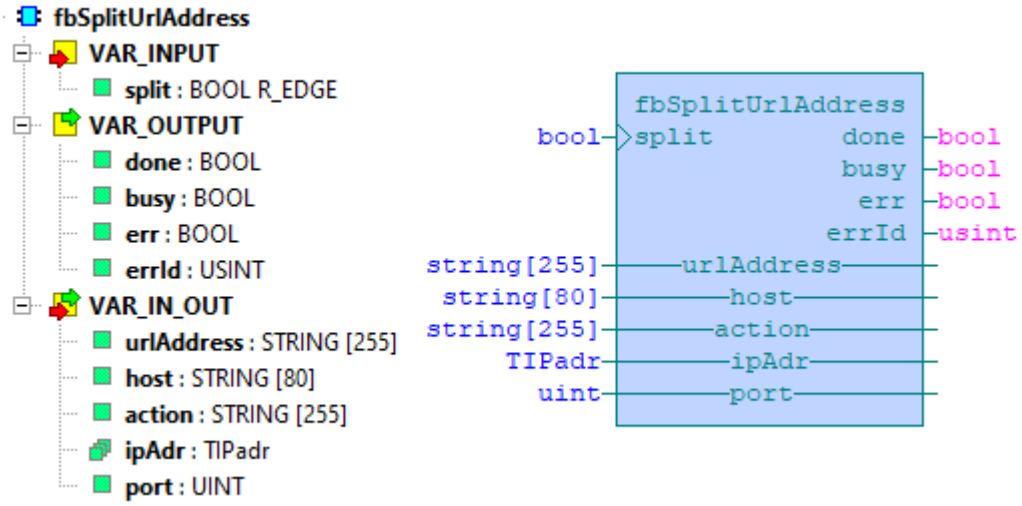
Následující příklad ukazuje vyslání dat ve formátu JSON metodou PUT na testovací server a uložení odpovědi do souboru.

Příklad je určen pro řady PLC Foxtrot 2, ze kterých využívá pokročilé funkce na otevření UNI spojení bez nutnosti jej zakládat v konfiguraci a komunikaci šifrovaným protokolem HTTPS. Pro starší systémy je nutné vybrat server, který komunikuje nešifrovaným protokolem HTTP, proměnnou *HttpRequest.chanCode* naplnit konstantou *ETH1_uniX* označující spojení definované v konfiguraci a vynechat volání *OpenUniSocket*, *SetUniLog* a *CloseUniSocket*).

```
PROGRAM prgPutExample
  VAR
    url : STRING           := 'https://postman-echo.com/put';
    data : STRING[255]     := '{"test" : true}';
    content : STRING[16]   := 'application/json';
    file    : STRING       := 'PUTREPLY.TXT';

    host : STRING;
    action : STRING[255];
    ip : TIPadr;
    port : UINT;
    buffer : THttpBuffer;

    SplitUrlAddress : fbSplitUrlAddress := (split := true);
    HttpRequest     : fbHttpRequestL3;
    WTFSeq          : WriteToFileSeq;
  END_VAR

  SplitUrlAddress(urlAddress := url,
                  host   := host,
                  action := action,
                  ipAdr  := ip,
                  port   := port);

  IF SplitUrlAddress.done THEN
    HttpRequest.chanCode := OpenUniSocket(protocol := UNI_SSL_CLIENT);
    SetUniLog(chanHandle := HttpRequest.chanCode,
              logMode := ONE_TIME_LOG,
              logSizeKB := 32,
              logName := 'TESTPUT.LOG');
  END_IF;

  HttpRequest(Put := SplitUrlAddress.done, ContentType := content,
              IPadr := ip, port := port, Action := action, Host := host,
              postLen := LEN(data),
              Data := void(data),
              RecvData := buffer);

  WTFSeq(fileName := file,
         srcVar := void(buffer),
         write := HttpRequest.DataReady,
         close := NOT HttpRequest.Busy,
         size := TO_UDINT(HttpRequest.DataLen));

  IF NOT HttpRequest.Busy AND HttpRequest.chanCode <> 0 THEN
    CloseUniSocket(chanHandle := HttpRequest.chanCode);
  END_IF;

END_PROGRAM
```

7.5 Funkční blok fbSplitUrlAddress

knihovna: *InternetLib*



Funkční blok *fbSplitUrlAddress* rozebírá zadanou URL adresu na jednotlivé položky vyžadované bloky *fbHttpRequest*, *fbHttpRequestL* a *fbHttpRequestL2*. Vstupem bloku je proměnná předaná přes *urlAddress*, která nese URL adresu tak jak je uvedena ve webovém prohlížeči.

Pro získání adresy serveru blok vnitřně využívá *fbNsLookUpEx*, což vyžaduje řadu centrály K nebo L s firmware verze 7.1 nebo vyšším nebo jakoukoli verzi centrály řady I.

Popis proměnných :

	Proměnná	Typ	Význam
VAR_INPUT			
	split	BOOL R_EDGE	Rozdělení URL adresy
VAR_OUTPUT			
	done	BOOL	URL adresa rozdělena a IP adresa úspěšně získána
	busy	BOOL	Operace probíhá
	err	BOOL	Nastala chyba
	errId	USINT	Číslo chyby
VAR_IN_OUT			
	urlAddress	STRING[255]	URL adresa tak jak je uvedena ve webovém prohlížeči
	host	STRING	Jméno hostitele
	action	STRING[255]	Cesta k datům na serveru (vždy začíná znakem '/')
	ipAdr	TIPadr	IP adresa vzdáleného serveru
	port	UINT	Číslo portu vzdáleného serveru

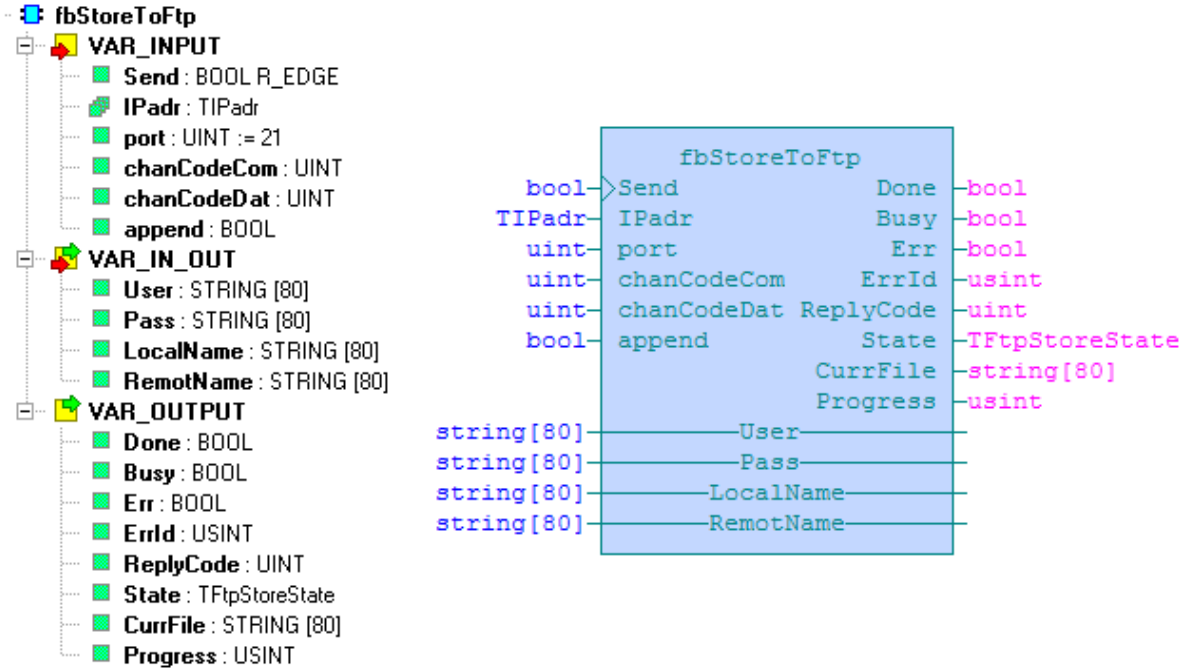
Příklad viz *fbHttpRequest*.

8 KOMUNIKACE FTP PROTOKOLEM

Knihovna od verze 1.3 obsahuje funkční blok pro ukládání souborů na FTP serveru. Pro správnou funkci musí server umožňovat práci v pasivním režimu.

8.1 Funkční blok fbStoreToFtp

knihovna: *InternetLib*



Funkční blok *fbStoreToFtp* slouží k uložení souboru z paměťové karty PLC na FTP server. Od verze knihovny 2.3 blok umožňuje vytvořit neexistující adresáře na FTP serveru a přenášet celé adresáře. Blok pracuje se serverem v pasivním režimu, což znamená, že se PLC aktivně připojuje pro přenos dat na port určený vzdáleným serverem.

Komunikace probíhá přes dvě spojení na ethernetovém kanálu v režimu UNI podle konstant na vstupech *chanCodeCom* a *chanCodeDat*. Spojení pro musí mít následujícími parametry: pro *chanCodeCom* – režim TCP master, délka přijímací a vysílací zóny 255 bytů. Pro *chanCodeDat* – režim TCP master, délka vysílací zóny 255 bytů. Pokud spojení není aktivní nebo nemá správné délky zón, blok indikuje chybu na výstupech *Err* hodnotou TRUE a *ErrId* hodnotou 255 pro *chanCodeCom* a 253 pro *chanCodeDat*.

Adresa FTP serveru se předává na vstupu *IPadr* na vstupu *port* se očekává číslo portu, na kterém server naslouchá (výchozí hodnota pro FTP protokol je 21). Na vstupech *User* a *Pass* se očekávají přihlašovací údaje – uživatelské jméno a heslo. Proměnná se jménem a cestou k souboru nebo adresáři na paměťové kartě PLC se očekává na vstupu *LocalName*. Pokud soubor nebo adresář neexistuje funkční blok vyhlásí na výstupu *ErrId* chybu číslo 3. Proměnná na vstupu *RemotName* udává jméno a cestu, do které bude soubor uložen na serveru. Cesta na FTP serveru od verze knihovny 2.3 již nemusí existovat (ve starších verzích akce skončila v případě neexistující cesty neúspěchem – chyba číslo 2). Pokud na FTP serveru již soubor existuje, je, pokud tomu nebrání přístupová práva, přepsán.

Pro přenos celých adresářů je nutné, aby jak *LocalName* tak *RemotName* obsahoval cestu k adresáři ukončenou oddělovačem /. Pokud tato podmínka není splněna je vyhlášena chyba číslo 7.

Přenos souboru nebo adresáře se zahájí na náběžnou hranu na vstupu *Send*. Během komunikace je nastaven výstup *Busy* na hodnotu TRUE. V případě úspěšného ukončení je nastaven na jeden cyklus výstup *Done*. V případě neúspěchu je nastaven výstup *Err* a *ErrId*, který obsahuje upřesňující číslo chyby.

Od verze knihovny 2.3 je během přenosu souboru na výstupu *CurrFile* jméno souboru na paměťové kartě a na výstupu *Progress* je indikován průběh odesílání souboru v procentech.

Přenos adresářů probíhá včetně podadresářů. Maximální počet vnoření podadresářů jsou 4. Pokud obsahuje adresář větší hloubku vnoření je vyhlášena chyba číslo 5.

Výstup *State* indikuje aktuální stav komunikace. Na výstupu *ReplyCode* se objevují kódy odpovědí serveru, které lze využít k další diagnostice, pokud je vyhlášena na výstupu *ErrId* chyba číslo 2 – neočekávaná odpověď serveru.

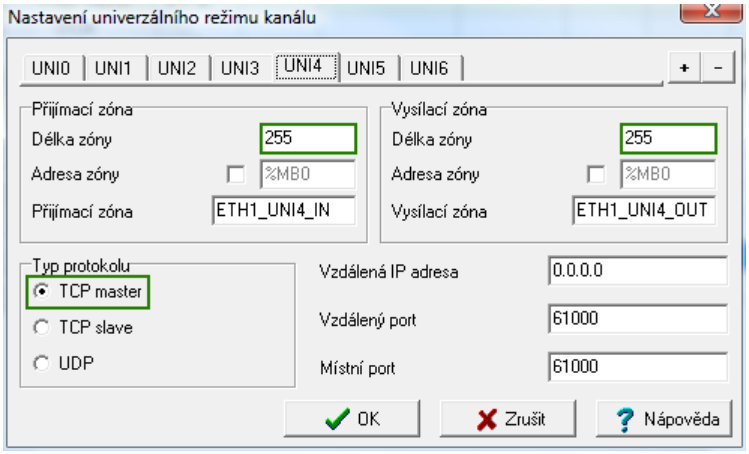
Od verze 4.4. má blok vstup *append*. Pokud je vstup nastaven na hodnotu FALSE, blok se chová stejně jako ve starších verzích, kdy se soubory na FTP přepíší nově příchozími. Pokud se vstup *append* nastaví na TRUE, jsou data příchozích souborů připojeny k existujícím souborům nakonec. Pokud soubory na straně FTP neexistují, blok pracuje shodně nezávisle na nastavení vstupu *append*.

Seznam očekávaných kódu odpovědí serveru pro jednotlivé stavy

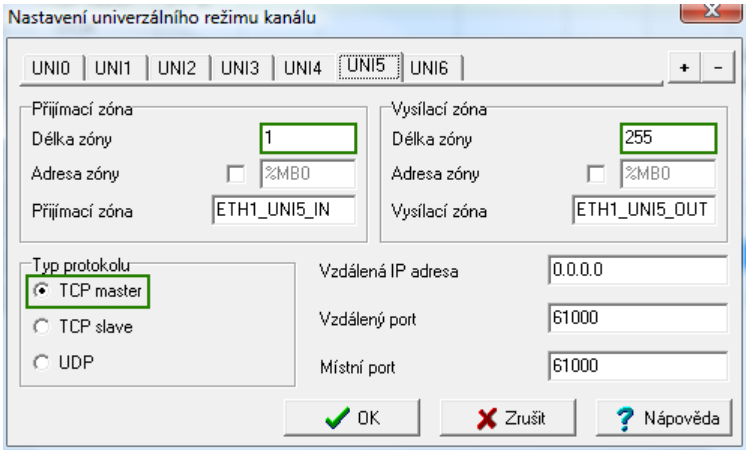
Stav	**Očekávaný kód odpovědi serveru**
fss_Rx220	220
fss_RxUser	331
fss_RxPass	230
fss_RxType	200
fss_RxPasv	227
fss_WaitForOpen	150 nebo 125 (InternetLib 1.7) nebo 550 (InternetLib 2.3) – vyvolá pokus o vytvoření cesty na FTP
fss_RxCreateDir (InternetLib 2.3)	257 nebo 550 pokud adresář již existuje nebo nelze vytvořit
fss_RxComplete	226
fss_RxQuit	221

Bližší informace o významech kódů odpovědí viz RFC 959.

Nastavení příkazového a datového spojení na ethernetovém kanálu v režimu UNI:



Nastavení spojení na ethernetovém kanálu v režimu UNI pro funkční blok fbStoreToFTP vstup chanCodeCom (příkazové spojení)



Nastavení spojení na ethernetovém kanálu v režimu UNI pro funkční blok fbStoreToFTP vstup chanCodeDat (datové spojení)

Popis proměnných :

	Proměnná	***Typ***	***Význam***
VAR_INPUT			
	Send	BOOL R_EDGE	Odešle soubor/adresář na FTP
	IPadr	TIPadr	IP adresa FTP serveru
	port	UINT	Port FTP serveru
	chanCodeCom	UINT	Přenosový kanál TCP Master, in: 255, out: 255
	chanCodeDat	UINT	Přenosový kanál TCP Master, in: 1, out: 255
	append	BOOL	Pokud soubor na straně serveru existuje budou data přidána na jeho konec
VAR_OUTPUT			

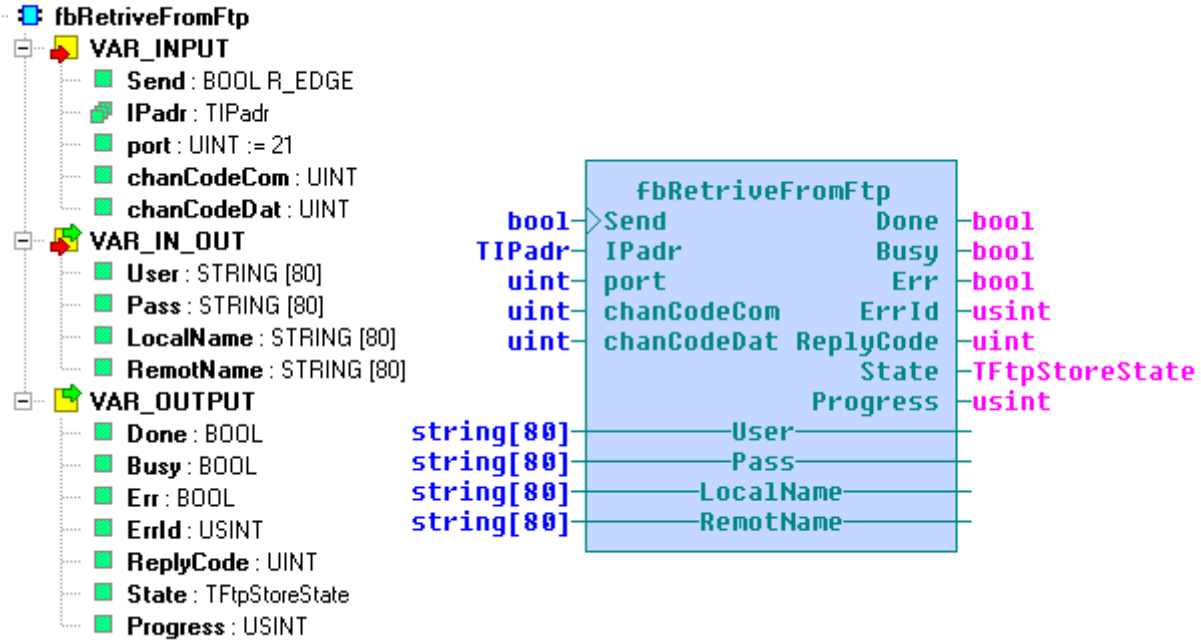
	Proměnná	Typ	Význam
	Done	BOOL	Soubor byl uložen na FTP server
	Busy	BOOL	Soubor se ukládá na FTP server
	Err	BOOL	Nastala chyba
	ErrId	USINT	Číslo chyby
	ReplyCode	UINT	Kód odpovědi serveru
	State	TFtpStoreState	Stav komunikace
	CurrFile	STRING[80]	Jméno aktuálně nahrávaného souboru
	Progress	USINT	Průběh nahrávání souboru v procentech
VAR_IN_OUT			
	User	STRING[80]	Uživatelské jméno
	Pass	STRING[80]	Heslo
	LocalName	STRING[80]	Jméno souboru nebo složky na paměťové kartě
	RemotName	STRING[80]	Jméno souboru nebo složky na FTP serveru

Následující příklad ukazuje použití funkčního bloku *fbStoreToFtp*. Při nastavení proměnné *SendFileToFtp* do logické jedničky dojde k odeslání souboru s obrázkem na FTP server.

```
VAR_GLOBAL
  SendFileToFtp  : BOOL;
END_VAR

PROGRAM prgExampleStoreToFtp
  VAR
    NsLookUp     : fbNsLookUpEx;
    StoreToFtp   : fbStoreToFtp;
    FtpIP        : TIPadr;
    FtpName      : STRING := 'foxtrot.howto.cz';
    User         : STRING := 'foxtrot.howto.cz';
    Pass         : STRING := '******';
    FileName     : STRING := '/WWW/IMAGES/TOP.PNG';
    FtpFileName  : STRING := 'TOP.PNG';
  END_VAR

  NsLookUp(getIP := SendFileToFtp,
           DnsIP :=  STRING_TO_IPADR('208.67.222.222'),
           Name := FtpName,
           IP := FtpIP);

  StoreToFtp(Send := NsLookUp.Done, IPadr := FtpIP,
             chanCodeCom := ETH1_uni4, chanCodeDat := ETH1_uni5,
             User := User, Pass := Pass,
             LocalName := FileName,
             RemotName := FtpFileName);
END_PROGRAM
```

8.2 Funkční blok fbRetriveFromFtp

knihovna: *InternetLib*



Funkční blok *fbRetriveFromFtp* stažení souboru z FTP serveru na paměťovou kartu PLC. Blok pracuje se serverem v pasivním režimu, což znamená, že se PLC aktivně připojuje pro přenos dat na port určený vzdáleným serverem.

Komunikace probíhá přes dvě spojení na ethernetovém kanálu v režimu UNI podle konstant na vstupech *chanCodeCom* a *chanCodeDat*. Spojení pro musí mít následujícími parametry: pro *chanCodeCom* – režim TCP master, délka přijímací a vysílací zóny 255 bytů. Pro *chanCodeDat* – režim TCP master, délka vysílací zóny 255 bytů, přijímací 1 byte. Pokud spojení není aktivní nebo nemá správné délky zón, blok indikuje chybu na výstupech *Err* hodnotou TRUE a *ErrId* hodnotou 255 pro *chanCodeCom* a 253 pro *chanCodeDat*.

Adresa FTP serveru se předává na vstupu *IPadr* na vstupu *port* se očekává číslo portu, na kterém server naslouchá (výchozí hodnota pro FTP protokol je 21). Na vstupech *User* a *Pass* se očekávají přihlašovací údaje – uživatelské jméno a heslo. Proměnná se jménem a cestou, pod kterým bude soubor uložen na paměťové kartě PLC, se očekává na vstupu *LocalName*. Cesta k souboru musí existovat, blok ji sám nezakládá. Pokud se zápis souboru nepovede vrací blok na výstupu *ErrId* chybu číslo 3.

Proměnná na vstupu *RemotName* udává jméno a cestu k souboru serveru. Pokud soubor na serveru neexistuje blok skončí s chybu číslo 2.

Přenos souboru se zahájí na náběžnou hranu na vstupu *Send*. Během komunikace je nastaven výstup *Busy* na hodnotu TRUE. V případě úspěšného ukončení je nastaven na jeden cyklus výstup *Done*. V případě neúspěchu je nastaven výstup *Err* a *ErrId*, který obsahuje upřesňující číslo chyby.

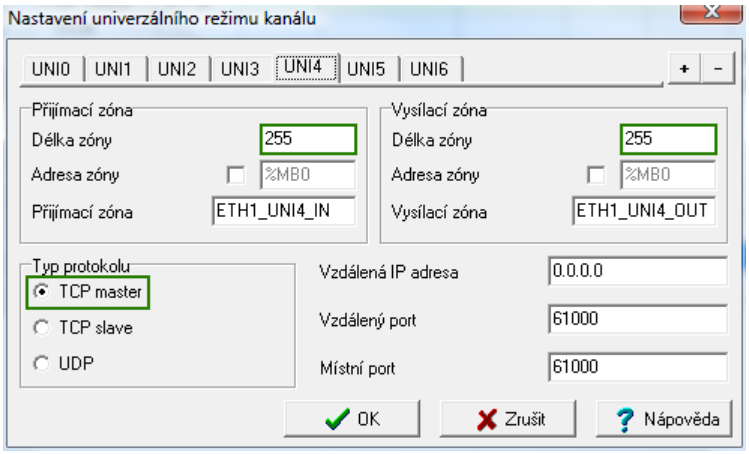
Na výstupu *Progress* je indikován průběh stahování souboru v procentech.

Výstup *State* indikuje aktuální stav komunikace. Na výstupu *ReplyCode* se objevují kódy odpovědí serveru, které lze využít k další diagnostice, pokud je vyhlášena na výstupu *ErrId* chyba číslo 2 – neočekávaná odpověď serveru.

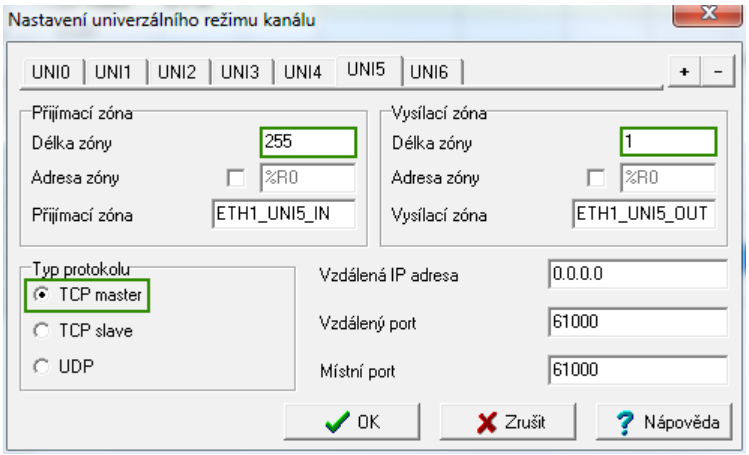
Stav	Očekávaný kód odpovědi serveru
fss_Rx220	220
fss_RxUser	331
fss_RxPass	230
fss_RxType	200
fss_RxPasv	227
fss_RxSize	213 (v případě jiné odpovědi nevyvolá chybu, pouze nedokáže indikovat průběh stahování)
fss_RxData	150, 125 a 226
fss_RxQuit	221

Bližší informace o významech kódů odpovědí viz RFC 959.

Nastavení příkazového a datového spojení na ethernetovém kanálu v režimu UNI:



Nastavení spojení na ethernetovém kanálu v režimu UNI pro funkční blok fbRetriveFromFtp vstup chanCodeCom (příkazové spojení)



Nastavení spojení na ethernetovém kanálu v režimu UNI pro funkční blok fbRetriveFromFtp vstup chanCodeDat (datové spojení)

Popis proměnných :

	Proměnná	Typ	Význam
VAR_INPUT			
	Send	BOOL R_EDGE	Náběžná hrana zahájí přenos souboru
	IPadr	TIPadr	IP adresa FTP serveru
	port	UINT	Port FTP serveru
	chanCodeCom	UINT	Přenosový kanál TCP Master, in: 255, out: 255
	chanCodeDat	UINT	Přenosový kanál TCP Master, in: 255, out: 1
VAR_OUTPUT			
	Done	BOOL	Soubor byl uložen na paměťovou kartu
	Busy	BOOL	Soubor se ukládá na paměťovou kartu
	Err	BOOL	Nastala chyba
	ErrId	USINT	Chybový kód: *ErrId* = 0 operace dopadla úspěšně *ErrId* = 1 vypršel čas pro odpověď serveru *ErrId* = 2 nepodařilo se získat všechna data ze serveru *ErrId* = 3 chyba při zápisu přijatých dat *ErrId* = 253 chybné nastavení spojení pro data *ErrId* = 254 nulová adresa webového serveru *ErrId* = 255 chybné nastavení spojení pro příkazy
	ReplyCode	UINT	Kód odpovědi serveru
	State	TFtpStoreState	Stav komunikace
	Progress	USINT	Průběh nahrávání souboru v procentech
VAR_IN_OUT			
	User	STRING[80]	Uživatelské jméno
	Pass	STRING[80]	Heslo
	LocalName	STRING[80]	Jméno souboru na paměťové kartě
	RemotName	STRING[80]	Jméno souboru na FTP serveru

Následující příklad ukazuje použití funkčního bloku *fbRetriveFromFtp.* Při nastavení proměnné *RetrieveFileFromFtp* do logické jedničky dojde ke stažení souboru s obrázkem z FTP server.

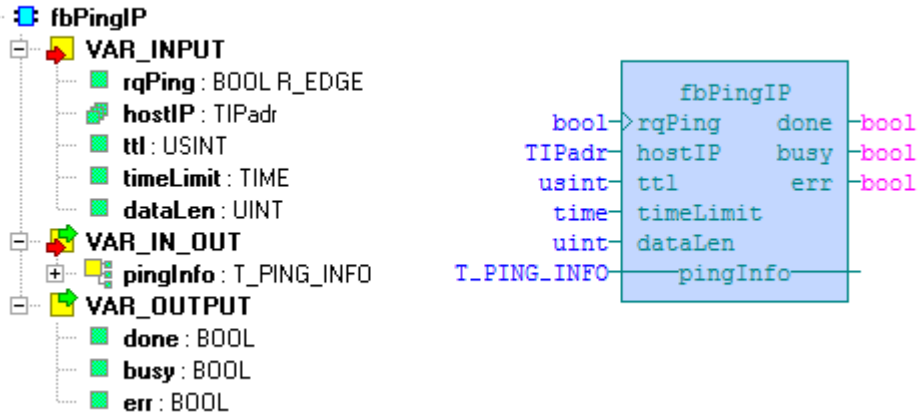
```
VAR_GLOBAL
  RetrieveFileFromFtp  : BOOL;
END_VAR

PROGRAM prgExampleRetriveFromFtp
  VAR
    NsLookUp     : fbNsLookUpEx;
    RetriveFromFtp  : fbRetriveFromFtp;
    FtpIP        : TIPadr;
    FtpName      : STRING := 'srv80.endora.cz';
    User         : STRING := 'user';
    Pass         : STRING := '********';
    FileName     : STRING := '/WWW/TOP.DATA.TXT';
    FtpFileName : STRING := 'DATA.TXT';
  END_VAR

  NsLookUp(getIP := RetrieveFileFromFtp,
           Name := FtpName,
           IP := FtpIP);

  RetriveFromFtp(Retrieve := NsLookUp.Done,
      IPadr := FtpIP,
      chanCodeCom := ETH1_uni4, chanCodeDat := ETH1_uni5,
      User := User, Pass := Pass,
      LocalName := FileName,
      RemotName := FtpFileName);

END_PROGRAM
```

9 OVĚŘOVÁNÍ DOSTUPNOSTI - PING

Knihovna od verze 3.8 obsahuje funkční bloky *fbPing* a *fbPingIP* umožňující ověřit dostupnost síťových prostředků odesíláním příkazu ICMP protokolu Echo. Podpora je obsažena v centrálách řady K a L od verze 9.9.

9.1 Funkční blok fbPingIP

knihovna: *InternetLib*



Funkční blok *fbPingIP* ověří spojení mezi PLC a síťovým rozhraním s danou IP adresou ICMP protokolem pomocí zprávy Echo. Funkční blok umožňuje nastavit parametry zprávy jako je životnost paketu TTL, časový limit pro platnou odpověď a délku dat.

Popis proměnných :

	Proměnná	Typ	Význam
VAR_INPUT			
	rqPing	BOOL R_EDGE	Žádost o odeslání Ping
	hostIP	TIPadr	IP adresa hostitele
	ttl	USINT	Životnost paketu TTL
	timeLimit	TIME	Časový limit pro odpověď
	dataLen	UINT	Délka dat
VAR_OUTPUT			
	done	BOOL	Operace byla úspěšně dokončena
	busy	BOOL	Operace probíhá
	err	BOOL	Nastala chyba
VAR_IN_OUT			
	pingInfo	T_PING_INFO	Informace o Pingu

Výsledky operace jsou uloženy ve struktuře T_PING_INFO předávané na vstupu *pingInfo*. Struktura obsahuje následující informace:

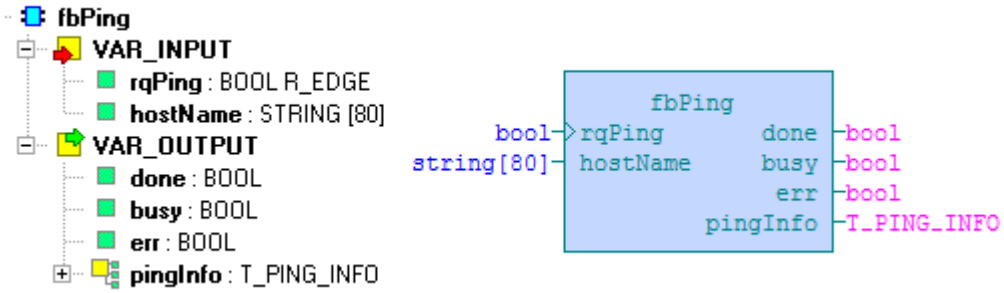
Proměnná	***Typ***	***Význam***
T_PING_INFO	STRUCT	
.result	DINT	0 = probíhá, 1 = dokončeno bez chyb, -1 = chyba
.hostIP	TIPadr	IP adresa hostitele
.responseTime	TIME	čas odezvy
.pingNumber	UDINT	číslo pingu
.dataLen	UDINT	délka dat
.ttl	USINT	TTL odpovědi
.errMsg	STRING[80]	chybová zpráva

Příklad níže ukazuje možnost použití *fbPingIP* k ověření dostupnost výchozí brány PLC. K ověření dojde nastavení vstupu *rqPing* na TRUE. Výsledek operace je uložen do proměnné *pingInfo*.

```
PROGRAM prgMain
  VAR
    myEthAdr    : TLocalEthAdr;
    myDnsIp     : TIPAdr;
    SendPing    : fbPingIP := (ttl := 60, timeLimit := T#3s, dataLen := 32);
    pingInfo    : T_PING_INFO;
  END_VAR

  // zjistit vlastni nastaveni
  GetIPaddress( ethChan := ETH1, ethAdr := myEthAdr);

  // stiskem tlacitka z web stranky odeslat ping
  SendPing(hostIP := myEthAdr.GW, pingInfo := pingInfo);
END_PROGRAM
```

9.2 Funkční blok fbPing

knihovna: *InternetLib*



Funkční blok *fbPing* ověří spojení mezi PLC a hostitelem daného jména. Blok vrací stejné informace jako *fbPingIP*. Navíc nabízí automatický překlad jmen a přednastavení parametrů, tak aby implementace byla co nejjednodušší.

Blok využívá vnitřně fbPingIP s následujícím nastavením:
ttl := 60, *timeLimit* := T#3s, *dataLen* := 32

Výsledky operace blok vrací na výstupu *pingInfo*. Popis struktury viz *fbPingIP*.

Popis proměnných :

	Proměnná	***Typ***	***Význam***
VAR_INPUT			
	rqPing	BOOL R_EDGE	Žádost o odeslání Ping
	hostName	STRING[80]	Jméno hostitele
VAR_OUTPUT			
	done	BOOL	Operace byla úspěšně dokončena
	busy	BOOL	Operace probíhá
	err	BOOL	Nastala chyba
	pingInfo	T_PING_INFO	Informace o Pingu

Následující příklad ukazuje možnost použití *fbPing* k ověření dostupnosti služby TecoRoute. K ověření dojde nastavení vstupu *rqPing* na TRUE. Výsledek je uložen na výstupu *pingInfo*. Funkce *GetIPaddress* a *GetDNS_IP* slouží pouze pro kontrolu nastavení síťového rozhraní PLC.

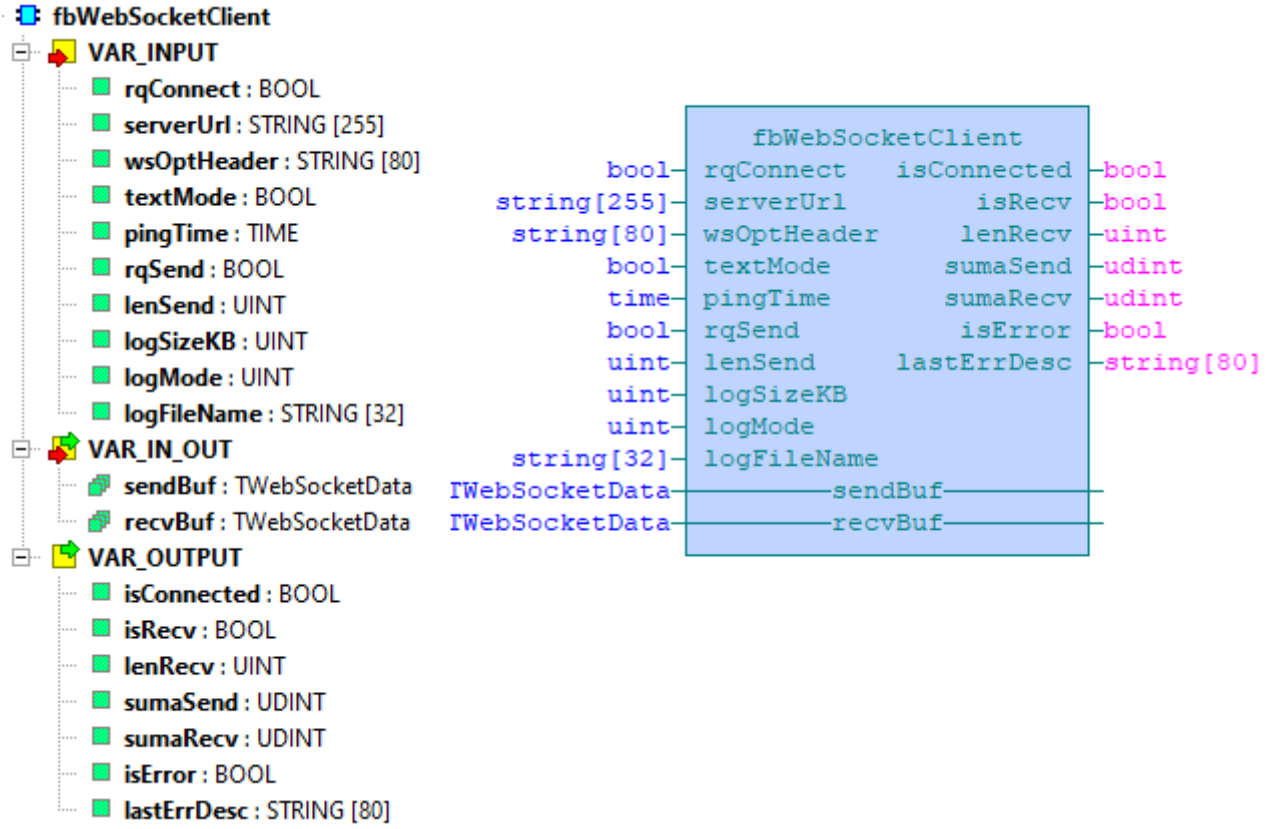
```
PROGRAM prgMain
  VAR
    myEthAdr    : TLocalEthAdr;
    myDnsIp     : TIPAdr;
    SendPing    : fbPing := ( hostName := 'route.tecomat.com');
  END_VAR

  // zjistit vlastni nastaveni
  GetIPaddress( ethChan := ETH1, ethAdr := myEthAdr);
  myDnsIp := GetDNS_IP ( ethChan := ETH1);

  // stiskem tlacitka z web stranky odeslat ping
  SendPing();
END_PROGRAM
```

10 WEBSOCKET KOMUNIKACE

Knihovna od verze 5.2 obsahuje funkční blok *fbWebSocketClient* umožňující komunikaci WebSocket protokolem. Podpora je obsažena v centrálách řady I (Foxtrot II a TC800), na starších systémech Foxtrot a TC-700 není tato komunikace podporována.

10.1 Funkční blok fbWebSocketClient

knihovna: *InternetLib*



Funkční blok *fbWebSocketClient* slouží pro WebSocket komunikaci PLC se serverem, PLC je v roli klienta. Komunikace probíhá přes TCP port číslo 80 (nebo 443 v případě TLS-šifrovaných spojení) a umožňuje plně duplexní výměnu dat. Protokol WebSocket byl standardizován jako RFC 6455 v roce 2011.

K navázání WebSocket komunikace mezi PLC a serverem dojde při náběžné hraně na vstupu *rqConnect*. Úspěšné navázání komunikace (včetně handshake) je signalizováno výstupem *isConnected*. Pokud má tento výstup hodnotu TRUE, pak je možno přenášet data WebSocket protokolem mezi PLC a serverem. URI adresu serveru udává vstup *serverUrl*, ze kterého funkční blok zjistí IP adresu serveru a zvolí typ komunikace. Pro adresy začínající ws:// (například 'ws://echo.websocket.org') použije funkční blok nešifrovanou komunikaci a výchozí port 80. Komunikace se servery, jejichž URI začíná wss:// (například 'wss://echo.websocket.org'), bude probíhat šifrovaně a výchozím portem bude 443. A konečně vstup *textMode* udává, jestli bude datová výměna probíhat v binárním režimu (*textMode* = FALSE) nebo v textovém režimu (*textMode* = TRUE). Nastavení vstupu *rqConnect* na FALSE uzavře spojení se serverem a ukončí komunikaci.

Vysílání dat na server

Data, která mají být odeslána na server, musí být připravena v proměnné *sendBuf*, což je typicky pole, jehož maximální velikost je daná konstantou MAX_LEN_WS_BUFFER. Aktuální délku odesílaných dat udává vstup *lenSend*. K odeslání dat blokem *fbWebSocketClient* dojde v okamžiku kdy se nastaví vstup *rqSend* na hodnotu TRUE.

Příjem dat ze serveru

Data přijatá ze serveru jsou uložena v proměnné *recvBuf*. Maximální velikost této proměnné je omezena konstantou MAX_LEN_WS_BUFFER. Příjem dat blokem *fbWebSocketClient* je signalizován výstupem *isRecv*, který se při příjmu dat nastaví na TRUE. Aktuální velikost přijatých dat udává výstup *lenRecv*.

Signalizace chyb

Pokud vznikne chyba v bloku *fbWebSocketClient* tak je nastaven výstup *isError* na TRUE a spojení se serverem se přeruší. Textový popis chyby je uveden v *lastErrDesc*. Pro obnovení spojení je třeba nastavit vstup *rqConnect* nejprve na FALSE a vzápětí zpět na TRUE (náběžná hrana na vstupu *rqConnect* naváže nové spojení).

Záznam komunikace

WebSocket komunikaci mezi PLC a serverem lze zaznamenat do souboru. Záznam komunikace se zapne, pokud je v proměnné *logSizeKB* nastavena nenulová velikost log souboru (v kilobytech). Tato proměnná se vyhodnocuje při navázání komunikace se serverem. Log soubor je uložen v adresáři WWW/LOGS/ a jeho název udává proměnná *logFileName*. Při výpadku napájení PLC je log soubor ztracen. Záznam dat do log souboru je řízen proměnnou *logMode*. Pokud je *logMode* = ONE_TIME_LOG, pak je komunikace zaznamenávaná od navázání spojení se serverem až do doby, kdy velikost log souboru překročí hodnotu danou proměnnou *logSizeKB*. Poté se záznam komunikace ukončí. Pokud je *logMode* = CYCLIC_LOG, pak se komunikace zaznamenává plynule a log soubor funguje jako kruhový buffer – v okamžiku, kdy se překročí velikost souboru daná proměnnou *logSizeKB*, tak ukládání pokračuje od začátku souboru. Soubor tak obsahuje záznam posledních několika kilobytů komunikace.

Pokud má *logSizeKB* hodnotu 0, tak se log soubor nezaloží (komunikace se nebude zaznamenávat).

Implementace WebSocket komunikace

Blok *fbWebSocketClient* podporuje nešifrovanou i šifrovanou (TLS) komunikaci. Jeden rámec WebSocket protokolu může obsahovat maximálně 1440 bytů (viz konstanta MAX_LEN_WS_BUFFER). Podporovány jsou všechny standardní typy rámců (text frame, binary frame, connection close, ping, pong). Pokud je v navázaném spojení klid po dobu *pingTime* (ani jedna strana nevysílá data) a *pingTime* je různý od hodnoty 0 sekund, tak blok *fbWebSocketClient* automaticky odešle ping rámec. Příchozí ping rámec je automaticky potvrzován rámcem pong.

Fragmentovaná WebSocket komunikace není podporována. Dále nejsou podporovány Sec-WebSocket-Extensions.

Do hlavičky HTTP protokolu, která se vysílá při navazování komunikace je možné vložit volitelné položky. Jejich obsah je možné zadat do vstupu *wsOptHeader*. Každá položka musí být zakončena dvojicí znaků '$0D$0A'.

Popis proměnných :

	Proměnná	Typ	Význam
VAR_INPUT			
	rqConnect	BOOL	žádost o spojení s WebSocket serverem
	serverUrl	STRING[255]	URI serveru (např. 'wss://echo.websocket.org')
	wsOptHeader	STRING[80]	volitelná položka hlavičky web socketu (např. 'Sec-WebSocket-Protocol: ocpp1.6$0D$0A')
	textMode	BOOL	0 = binární výměna dat, 1 = textová výměna dat
	pingTime	TIME	předvolba pro odesílání ping paketu (při T#0s je spontánní vysílání ping vypnuto)
	rqSend	BOOL	žádost o odeslání zprávy na server
	lenSend	UINT	délka odesílané zprávy
	logSizeKB	UINT	max. velikost log souboru [KB] (0 = bez logování)
	logMode	UINT	režim logu (jednorázový = ONE_TIME_LOG, cyklický = CYCLIC_LOG)
	logFileName	STRING[32]	jméno log souboru (bude uložen ve WWW/LOGS/)
VAR_IN_OUT			
	sendBuf	TWebSocketData	buffer pro vysílaná data
	recvBuf	TWebSocketData	buffer pro přijímaná data
VAR_OUTPUT			
	isConnected	BOOL	spojení s WebSocket serverem navázáno
	isRecv	BOOL	přišla zpráva od serveru
	lenRecv	UINT	délka přijaté zprávy
	sumaSend	UDINT	celkem odesláno
	sumaRecv	UDINT	celkem přijato
	isError	BOOL	došlo k chybě
	lastErrDesc	STRING[80]	popis chyby

Příklad

Následující příklad ukazuje WebSocket komunikaci PLC Foxtrot se serverem 'wss://echo.websocket.org'.

Služba použitá v tomto příkladu již bohužel není dostupná. Pro otestování příkladu je možné použít jakýkoli websocketový server, který opakuje přijatá data.

K navázání spojení se serverem dojde po nastavení proměnné *rqWebSocket* na hodnotu TRUE. Na server jsou zapisovány stavy globálních proměnných *timeMark* a *myStruct*. Výměna dat probíhá v textovém režimu, proměnné jsou na server odesílána ve formátu JSON, který je vytvořen pomocí funkce *VarApiToJsonBuf*. K odeslání dat dojde v okamžiku, kdy se proměnná *Main.rqSend* nastaví na hodnotu TRUE.

Server echuje zasílaná data, která se tak vrací zpět do PLC. Po příjmu každého echa je nastavena proměnná *isRecv* na TRUE a proměnná *lenRecv* obsahuje velikost přijatých dat. Přijatá data jsou rovněž ve formátu JSON a tak je můžeme zpracovat funkcí *JsonBufToVarApi*, která podle obsahu JSON nastaví proměnné v paměti PLC. V tomto konkrétním případě, kdy server pouze opakuje zaslaná data, to nemá žádný praktický význam. Je to ale ukázka toho, jak se dá zpracovávat JSON formát na straně příjmu.

Komunikace je logovaná do souboru WWW/LOGS/UNI_WEB_SOCKET.LOG.

```
TYPE
  TMyStruct : STRUCT            // testovaci struktura
    cnt     : UDINT;
    cpuTemp : USINT;
    desc    : STRING;
    note    : STRING;
  END_STRUCT;
END_TYPE

VAR_GLOBAL
  rqWebSocket : BOOL;           // pozadavek na otevreni web socketu
  Url : STRING := 'wss://echo.websocket.org';    // URI serveru

  // promenne PLC zverejnovane na server
  timeMark  {PUBLIC_API} : DT;
  myStruct  {PUBLIC_API} : TMyStruct;
  varList   : STRING := 'timeMark&myStruct';  // seznam zverejnenych promennych
END_VAR

PROGRAM prgMain
  VAR
    rqSend       : BOOL;     // zadost o odeslani zpravy na server
    lenSend      : UINT;     // delka odesilane zpravy
    sendMes      : ARRAY[1..MAX_LEN_WS_BUFFER] OF USINT; // send buffer
    sendMesCnt   : UDINT;    // citac odeslanych zprav

    isRecv       : BOOL;     // prisla zprava od serveru
    lenRecv      : UINT;     // delka prijate zpravy
    recvMes      : ARRAY[1..MAX_LEN_WS_BUFFER] OF USINT; // recv buffer
    recvMesCnt   : UDINT;    // citac prijatych zprav

    WebSocket    : fbWebSocketClient;  // blok pro web socket komunikaci
    jsonError    : STRING;
  END_VAR

  // priprava vysilanych zprav ========================================
  IF rqWebSocket AND WebSocket.isConnected THEN
    IF rqSend THEN
```

```
      // pripravit data odesilana na server -> nastavit neco do myStruct
      timeMark := GetDateTime();
      myStruct.cnt := myStruct.cnt + 1;
      myStruct.cpuTemp := System_S.CPU_TEMPERATURE;
      myStruct.desc := REAL_TO_STRINGF(
                         UINT_TO_REAL(System_S.LAST_CYCLE_TIME_100US)/10.0,
                         'Last cycle: %3.1f [ms]');
      IF myStruct.cnt.0 THEN
        myStruct.note := '';
      ELSE
        myStruct.note := 'this packet is longer then 126 bytes';
      END_IF;

      // sestavit JSON, ktery odesleme web socketem
      lenSend := UDINT_TO_UINT( VarApiToJsonBuf( varNames := varList,
                                                 buffer := void(sendMes),
                                                 maxBufLen := sizeof(sendMes)));
      IF lenSend = 0 THEN
        jsonError := GetLastJsonError();  // chyba pri tvorbe JSON
        rqSend := 0;
      ELSE
        sendMesCnt := sendMesCnt + 1;  // dalsi odeslana zprava
      END_IF;
    END_IF;
  END_IF;


  // obsluha web socketu ================================================
  WebSocket( rqConnect := rqWebSocket, serverUrl := Url, textMode := 1,
             rqSend := rqSend, lenSend := lenSend, sendBuf := void(sendMes),
             recvBuf := void(recvMes), isRecv => isRecv, lenRecv => lenRecv,
             logSizeKB := 32, logFileName := 'UNI_WEB_SOCKET.LOG',
             logMode := ONE_TIME_LOG);

  rqSend := 0;


  // zpracovani prijatych zprav =========================================
  IF rqWebSocket AND WebSocket.isConnected THEN
    IF isRecv THEN
      // zpracovat JSON, ktery jsme dostali od serveru
      IF JsonBufToVarApi( buffer := void( recvMes), strict := 0) > 0 THEN
        recvMesCnt := recvMesCnt + 1;  // pocet prijatych zprav
      ELSE
        jsonError := GetLastJsonError();
      END_IF;
      isRecv := 0;
    END_IF;
  END_IF;

END_PROGRAM
```

Záznam komunikace v souboru WWW/LOGS/UNI_WEB_SOCKET.LOG vypadá následovně (navázání spojení, WebSocket handshake, odeslání paketu s délkou 124 a příjem echa, údržba spojení – odeslání ping a příjem pong, odeslání paketu s délkou 162 a příjem echa, ukončení spojení):

```
ETH UNI CHANNEL (/UNI_WEB_SOCKET.LOG), F2x CP2005I v1.9.029b (Dec 13 2019 15:53:08),
start 2019-12-17 09:13:24.023772
================================================================================
2019-12-17 09:13:24.023959   ---- UNI start ---- F2x CP2005I v1.9.029b (Dec 13 2019
15:53:08)
2019-12-17 09:13:24.024062  SSL client - myIP: 0.0.0.0:0, hisIP: 0.0.0.0:0
2019-12-17 09:13:24.027032  TCP - myIP: 0.0.0.0:0, hisIP: 174.129.224.73:443
2019-12-17 09:13:24.027408  TCP socket opened
2019-12-17 09:13:24.227653  TCP connection established
2019-12-17 09:13:24.649306  SSL handshake successful, Version: TLSv1.2, Cipher: AES128-
SHA
2019-12-17 09:13:24.653770  SEND 157
 47 45 54 20 2F 20 48 54 54 50 2F 31 2E 31 0D 0A  GET./.HTTP/1.1..
 48 6F 73 74 3A 20 65 63 68 6F 2E 77 65 62 73 6F  Host:.echo.webso
 63 6B 65 74 2E 6F 72 67 0D 0A 55 70 67 72 61 64  cket.org..Upgrad
 65 3A 20 77 65 62 73 6F 63 6B 65 74 0D 0A 43 6F  e:.websocket..Co
 6E 6E 65 63 74 69 6F 6E 3A 20 55 70 67 72 61 64  nnection:.Upgrad
 65 0D 0A 53 65 63 2D 57 65 62 53 6F 63 6B 65 74  e..Sec-WebSocket
 2D 4B 65 79 3A 20 39 6D 38 45 43 79 43 46 55 6C  -Key:.9m8ECyCFUl
 41 53 49 61 4A 5A 74 31 49 70 49 77 3D 3D 0D 0A  ASIaJZt1IpIw==..
 53 65 63 2D 57 65 62 53 6F 63 6B 65 74 2D 56 65  Sec-WebSocket-Ve
 72 73 69 6F 6E 3A 20 31 33 0D 0A 0D 0A           rsion:.13....
2019-12-17 09:13:24.849637  RECV 201
 48 54 54 50 2F 31 2E 31 20 31 30 31 20 57 65 62  HTTP/1.1.101.Web
 20 53 6F 63 6B 65 74 20 50 72 6F 74 6F 63 6F 6C  .Socket.Protocol
 20 48 61 6E 64 73 68 61 6B 65 0D 0A 43 6F 6E 6E  .Handshake..Conn
 65 63 74 69 6F 6E 3A 20 55 70 67 72 61 64 65 0D  ection:.Upgrade.
 0A 44 61 74 65 3A 20 54 75 65 2C 20 31 37 20 44  .Date:.Tue,.17.D
 65 63 20 32 30 31 39 20 30 38 3A 31 30 3A 32 33  ec.2019.08:10:23
 20 47 4D 54 0D 0A 53 65 63 2D 57 65 62 53 6F 63  .GMT..Sec-WebSoc
 6B 65 74 2D 41 63 63 65 70 74 3A 20 4C 51 51 4E  ket-Accept:.LQQN
 36 56 70 30 78 72 68 6D 64 47 4C 4F 30 37 55 41  6Vp0xrhmdGLO07UA
 42 54 33 6E 6B 64 59 3D 0D 0A 53 65 72 76 65 72  BT3nkdY=..Server
 3A 20 4B 61 61 7A 69 6E 67 20 47 61 74 65 77 61  :.Kaazing.Gatewa
 79 0D 0A 55 70 67 72 61 64 65 3A 20 77 65 62 73  y..Upgrade:.webs
 6F 63 6B 65 74 0D 0A 0D 0A                       ocket....
2019-12-17 09:13:26.104725  SEND 124
 81 F6 69 7E F5 6D 12 5C 81 04 04 1B B8 0C 1B 15  ..i..m.\........
 D7 57 49 5C C7 5D 58 47 D8 5C 5B 53 C4 5A 3D 4E  .WI\.]XG.\[S.Z=N
 CC 57 58 4D CF 5F 5F 50 C4 5D 59 24 D7 41 4B 13  .WXM.__P.]Y$.AK.
 8C 3E 1D 0C 80 0E 1D 5C CF 16 4B 1D 9B 19 4B 44  .>.....\..K...KD
 D5 5E 45 5C 96 1D 1C 2A 90 00 19 5C CF 4D 5F 4F  .^E\...*...\.M_O
 D9 4F 0D 1B 86 0E 4B 44 D5 4F 25 1F 86 19 49 1D  .O....KD.O%...I.
 8C 0E 05 1B CF 4D 5D 50 C6 4D 32 13 86 30 4B 52  .....M]P.M2..0KR
 D7 03 06 0A 90 4F 53 5E D7 4F 14 03              .....OS^.O..
2019-12-17 09:13:26.379093  RECV 120
 81 76 7B 22 74 69 6D 65 4D 61 72 6B 22 3A 20 22  .v{"timeMark":."
 32 30 31 39 2D 31 32 2D 31 37 54 30 39 3A 31 33  2019-12-17T09:13
 3A 32 36 2E 31 30 30 5A 22 2C 22 6D 79 53 74 72  :26.100Z","myStr
 75 63 74 22 3A 7B 22 63 6E 74 22 3A 20 33 2C 22  uct":{"cnt":.3,"
 63 70 75 54 65 6D 70 22 3A 20 36 31 2C 22 64 65  cpuTemp":.61,"de
 73 63 22 3A 20 22 4C 61 73 74 20 63 79 63 6C 65  sc":."Last.cycle
 3A 20 34 2E 33 20 5B 6D 73 5D 22 2C 22 6E 6F 74  :.4.3.[ms]","not
 65 22 3A 20 22 22 7D 7D                          e":.""}}
2019-12-17 09:13:36.393433  SEND 2
 89 00                                            ..
2019-12-17 09:13:36.614629  RECV 2
 8A 00                                            ..
2019-12-17 09:13:38.248347  SEND 162
 81 FE 00 9A E5 57 3E 10 9E 75 4A 79 88 32 73 71  .....W>..uJy.2sq
 97 3C 1C 2A C5 75 0C 20 D4 6E 13 21 D7 7A 0F 27  .<.*.u...n.!.z.'
 B1 67 07 2A D4 64 04 23 DD 79 0C 24 D6 0D 1C 3C  .g.*.d.#.y.$...<
 C7 3A 47 43 91 25 4B 73 91 75 04 6B C7 34 50 64  .:GC.%Ks.u.k.4Pd
 C7 6D 1E 24 C9 75 5D 60 90 03 5B 7D 95 75 04 30  .m.$.u]`..[}.u.0
 D3 66 12 32 81 32 4D 73 C7 6D 1E 32 A9 36 4D 64  .f.2.2Ms.m.2.6Md
 C5 34 47 73 89 32 04 30 D1 79 0F 30 BE 3A 4D 4D  .4Gs.2.0.y.0.:MM
 C7 7B 1C 7E 8A 23 5B 32 DF 77 1C 64 8D 3E 4D 30  .{...#[2.w.d.>M0
```

```
 95 36 5D 7B 80 23 1E 79 96 77 52 7F 8B 30 5B 62  .6]{.#.y.wR..0[b
 C5 23 56 75 8B 77 0F 22 D3 77 5C 69 91 32 4D 32  .#Vu.w."\i.2M2
 98 2A                                            .*
2019-12-17 09:13:38.459592  RECV 158
 81 7E 00 9A 7B 22 74 69 6D 65 4D 61 72 6B 22 3A  ....{"timeMark":
 20 22 32 30 31 39 2D 31 32 2D 31 37 54 30 39 3A  ."2019-12-17T09:
 31 33 3A 33 38 2E 32 34 33 5A 22 2C 22 6D 79 53  13:38.243Z","myS
 74 72 75 63 74 22 3A 7B 22 63 6E 74 22 3A 20 34  truct":{"cnt":.4
 2C 22 63 70 75 54 65 6D 70 22 3A 20 36 31 2C 22  ,"cpuTemp":.61,"
 64 65 73 63 22 3A 20 22 4C 61 73 74 20 63 79 63  desc":."Last.cyc
 6C 65 3A 20 34 2E 31 20 5B 6D 73 5D 22 2C 22 6E  le:.4.1.[ms]","n
 6F 74 65 22 3A 20 22 74 68 69 73 20 70 61 63 6B  ote":."this.pack
 65 74 20 69 73 20 6C 6F 6E 67 65 72 20 74 68 65  et.is.longer.the
 6E 20 31 32 36 20 62 79 74 65 73 22 7D 7D        n.126.bytes"}}
2019-12-17 09:13:41.872263  ---- UNI stop -----
```

TXV 003 54.01

Teco a.s. Průmyslová zóna Šťáralka 984, 280 02 Kolín, tel. +420 321 401 111,
e-mail: teco@tecomat.cz

Výrobce si vyhrazuje právo na změny dokumentace.
Poslední aktuální vydání je k dispozici na internetu www.tecomat.cz