Knihovna XmlLibEx

TXV 003 78.01
druhé vydání
prosinec 2013
změny vyhrazeny

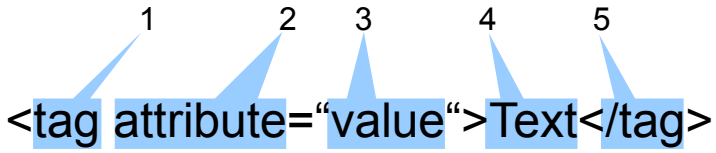
Historie změn

Datum	Vydání	Popis změn
Říjen 2012	1	První vydání, popis odpovídá XmlLibEx_v11
Prosinec 2013	2	Doplněn popis bloků fbXmlParser, fbXmlFileParser a fbXmlPageParser, popis odpovídá XmlLibEx_v13

OBSAH

1 ÚVOD

Knihovna XmlLibEx je standardně dodávaná jako součást programovacího prostředí Mosaic. Knihovna obsahuje funkční bloky umožňující práci s daty ve formátu Extensible Markup Language (XML).



Obecná struktura XML

1 – tag
2 – atribut
3 – hodnota atributu
4 – text
5 – ukončení tagu

XmlLibEx ve verzi 1.3 nepokrývá kompletně všechny možnosti zápisu XML dat. Omezení knihovny jsou následující:

- Komentáře jsou ignorovány
- Maximální délka názvu tagu je 80 znaků
- Maximální délka názvu atributu je 64 znaků
- Maximální délka hodnoty atributu je 192 znaků
- Maximální délka textu je 254 znaků
- Maximální počet atributů pro jeden tag je deset
- Není podporována sekce CDATA
- Nejsou podporovány zástupné xml entity

Následují obrázek ukazuje strukturu knihovny XmlLibEx v prostředí Mosaic

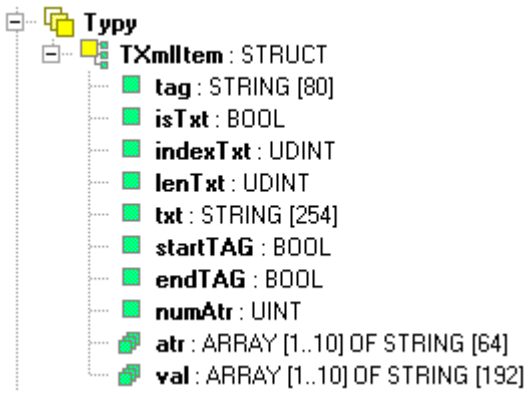
Pokud chceme funkce z knihovny XmlLibEx použít v aplikačním programu PLC, je třeba nejprve přidat tuto knihovnu do projektu. Knihovna je dodávaná jako součást instalace prostředí Mosaic od verze v2012.3.

Knihovna XmlLibEx využívá podporu ve firmware centrální jednotky PLC. Knihovnu lze použít na všech centrálních jednotkách řady Foxtrot od firmware v7.4. V systémech TC700 lze knihovnu použít na procesorech CP-7004 a CP-7007 od v7.4. Knihovnu XmlLibEx lze použít jako náhradu starší knihovny XmlLib. Kód XmlLibEx je kratší a běží rychleji než tomu bylo u původní knihovny XmlLib. Od XmlLibEx_v13 jsou navíc v knihovně k dispozici funkční bloky zjednodušující zpracování XML dokumentů.

Objednací číslo dokumentace ke knihovně XmlLibEx je TXV 003 78.01.

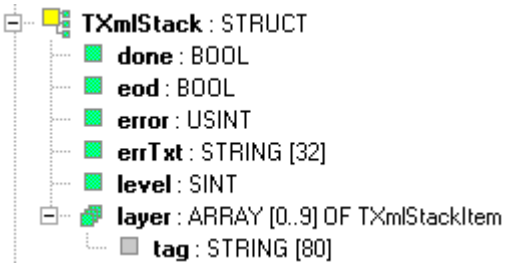
2 DATOVÉ TYPY

V knihovně XmlLibEx jsou definovány následující datové typy:



Datový typ *TXmlItem* je struktura obsahující informace o aktuálně načteném elementu XML dokumentu. Význam jednotlivých položek je následující:

Identifikátor	***Typ***	***Význam***
TXmlItem	STRUCT	Popis elementu XML
.tag	STRING[80]	Jméno tagu včetně klíčových znaků ?, /, atd.
.isTxt	BOOL	Příznak přítomnosti textu
.indexTxt	UDINT	Index začátku textu v xml bufferu
.txt	STRING[254]	Text před vlastním tagem (prvních 254 znaků)
.startTAG	BOOL	TRUE znamená začátek párového tagu (např. <ROOT>)
.endTAG	BOOL	TRUE znamená konec párového tagu (např. </ROOT>)
.numAtr	UINT	Počet atributů v tagu
.atr	ARRAY [1..10] OF STRING[64]	Názvy prvních deseti atributů
.val	ARRAY [1..10] OF STRING[192]	Hodnoty prvních deseti atributů

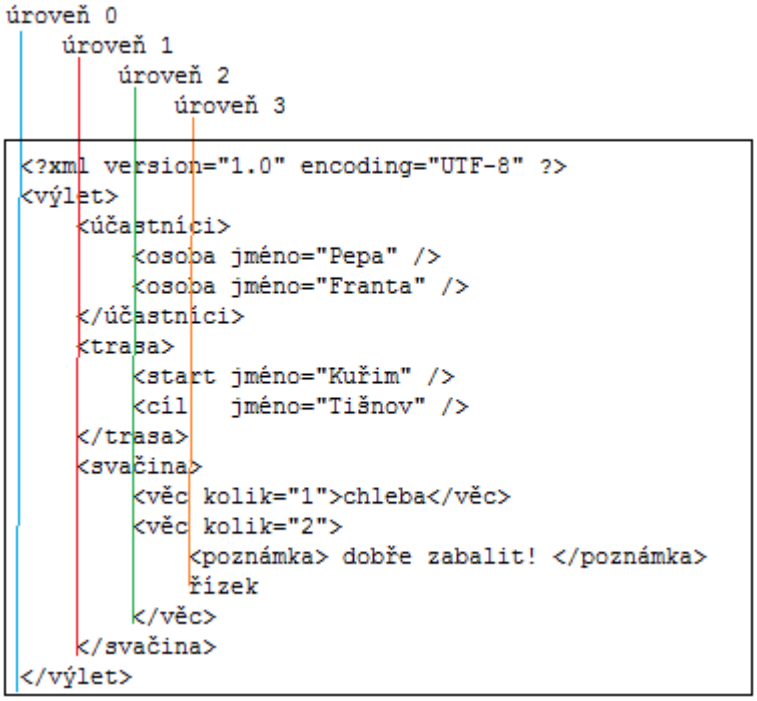


Datový typ *TXmlStack* je struktura obsahující informace o historii pársování XML dokumentu. Název každého vypársovaného tagu v XML dokumentu je uložen do XML stacku. Vnořené tagy zvyšují úroveň stacku. Význam jednotlivých položek je následující:

Identifikátor	***Typ***	***Význam***
TXmlStack	STRUCT	Historie parsování XML (vnoření tagů)
.done	BOOL	Příznak vypársování položky
.eod	BOOL	Konec XML dokumentu
.error	USINT	Kód chyby
.errTxt	STRING[32]	Popis chyby
.level	SINT	Úroveň stacku
.layer	ARRAY [0..9] OF TXmlStackItem	Názvy až deseti vnořených tagů

Datový typ *TXmlInfo* je struktura obsahující typy *TxmlItem* a *TxmlStack.*

Chování struktury *TxmlStack* během pársování XML dokumentu



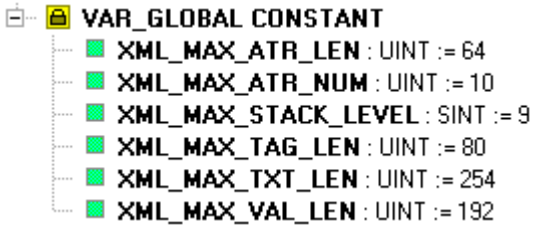
Pokud bude pársován výše uvedený XML soubor, pak bude jako první získán tag `<?xml>`. Do položky *TxmlStack.layer[0]* se tedy uloží název `?xml` a položka *TxmlStack.level* bude mít hodnotu 0. Atributy tohoto tagu budou uloženy do struktury *TXmlItem*. Při dalším volání funkčního bloku pro pársování (*fbXmlParser*, *fbXmlFileParser*, *fbXmlPageParser*) bude ze XML souboru získán tag `<výlet>`. Ten je také na úrovni 0 jako předchozí tag `<?xml>`, takže dojde k přepsání položky *TxmlStack.layer[0]* názvem tagu `výlet`. Úroveň stacku *TxmlStack.level* se nezmění (zůstane 0). A opět se odpovídajícím způsobem nastaví struktura *TxmlItem*. Dalším získaným tagem bude tag `<účastníci>`. To je tag vnořený v tagu `<výlet>`, úroveň vnoření je 1, takže se zvýší hodnota *TxmlStack.level* na 1 název tagu bude zapsán do položky *TxmlStack.layer[1]*. Dalším tagem v pořadí je tag `<osoba>`. Ten má úroveň vnoření 2 a jeho název bude tedy zapsán do položky *TxmlStack.layer[2]*, přičemž úroveň stacku *TxmlStack.level* se zvýší na hodnotu 2. Do struktury *TXmlItem* bude uložen atribut s názvem `jméno`, hodnota tohoto atributu bude `Pepa`. Následuje další tag `<osoba>`. Úroveň tohoto tagu je rovněž 2, takže úroveň stacku *TxmlStack.level* se nezmění (zůstane 2). Název tohoto tagu je shodný s předchozím tagem, takže hodnota položky *TxmlStack.layer[2]* zůstane `osoba`. Do struktury *TXmlItem* bude nyní uložen atribut s názvem `jméno`, hodnota tohoto atributu bude `Franta`. Dalším tagem je koncový tag `</účastníci>`. Tento tag je na úrovni 1, takže dojde ke snížení úrovně stacku *TxmlStack.level* na hodnotu 1. Položka *TxmlStack.layer[2]* se smaže (nahradí se prázdným řetězcem) a jméno tagu `/účastníci` je uloženo do položky *TxmlStack.layer[1]*. A tak dále stále dokola.

Takže struktura *TxmlStack* slouží k jednoznačnému určení aktuálně vypársovaného tagu. Například v okamžiku získání tagu `<start>` bude mít úroveň stacku *TxmlStack.level* hodnotu 2 a položky *TxmlStack.layer* budou naplněny následovně:

TxmlStack.layer[0] … `výlet`
TxmlStack.layer[1] … `trasa`
TxmlStack.layer[2] … `start`
TxmlStack.layer[3] … žádný text

3 KONSTANTY

V knihovně XmlLibEx jsou definovány následující konstanty:



Konstanty definují velikost položek v datovém typu *TxmlItem* (počet znaků). Význam konstant je následující:

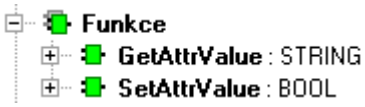
Identifikátor	*Typ*	*Hodnota*	*Význam*
XML_MAX_ATR_LEN	USINT	64	Max. délka názvu atributu
XML_MAX_ATR_NUM	USINT	10	Max. počet atributů v tagu
XML_MAX_STACK_LEVEL	SINT	9	Max. index položky TxmlStack.layer[]
XML_MAX_TAG_LEN	USINT	80	Max. délka názvu tagu
XML_MAX_TXT_LEN	USINT	254	Max. délka textu v tagu
XML_MAX_VAL_LEN	USINT	192	Max. délka hodnoty atributu

4 GLOBÁLNÍ PROMĚNNÉ

V knihovně XmlLibEx nejsou definovány žádné globální proměnné.

5 FUNKCE

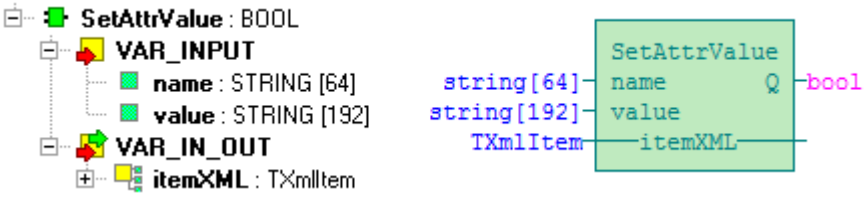
V knihovně XmlLibEx jsou definovány následující funkce.



Funkční blok	*Popis*
GetAttrValue	Přečte hodnotu atributu daného jména
SetAttrValue	Zapíše hodnotu do atributu daného jména

5.1 Funkce SetAttrValue

Knihovna : *XmlLibEx*



Funkce *SetAttrValue* pracuje nad strukturou *TXmlItem* popisující element XML dokumentu, která se předává na vstupu *itemXML*. V této struktuře nalezne podle zadaného jména na vstupu *name* příslušný atribut a nastaví mu hodnotu předávanou na vstupu *value*. Pokud atribut daného jména ve struktuře neexistuje, pak jej založí (zkopíruje jméno atributu na volné místo do struktury *itemXML.atr[i]*), nastaví jeho hodnotu (zkopíruje hodnotu atributun do *itemXML.val[i]*) a zvýší počet atributů ve struktuře (položka *itemXML.numAtr*).

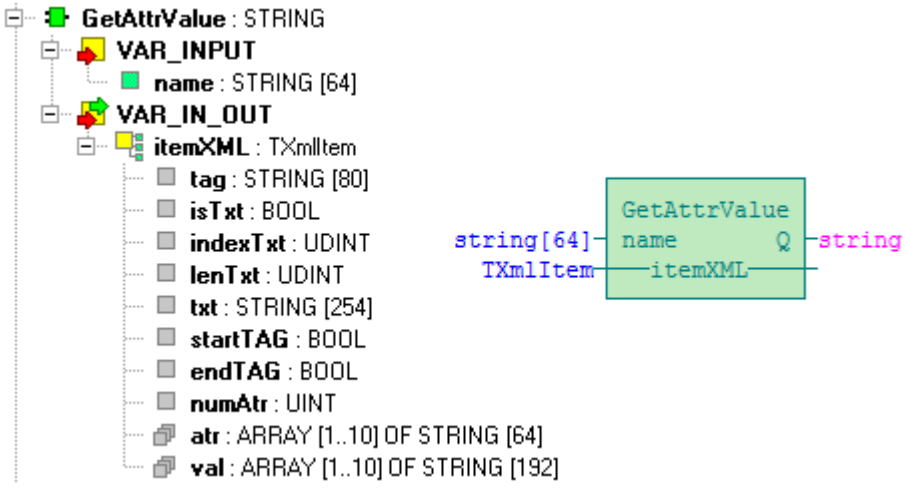
Pokud funkce zapíše požadovaný atribut a jeho hodnotu vrací TRUE. V případě, že se atribut nepodařilo zapsat, funce vrací FALSE.

Popis proměnných :

	Proměnná	***Typ***	***Význam***
VAR_INPUT			
	Name	STRING	Název atributu
	Value	STRING	Hodnota atributu
VAR_IN_OUT			
	itemXML	TXmlItem	Popis XML elementu
SetAttrValue			
	Návratová hodnota	BOOL	TRUE pokud se atribut podařilo zapsat, jinak FALSE

5.2 Funkce GetAttrValue

Knihovna : *XmlLibEx*



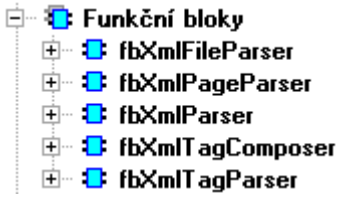
Funkce G*etAttrValue* pracuje nad strukturou popisující element XML dokumentu *TXmlItem*, která se předává na vstupu *itemXML*. V této struktuře nalezne podle zadaného jména na vstupu *name* příslušný atribut a vrátí jeho hodnotu. Pokud atribut není nalezen, funkce vrátí prázdný string.

Popis proměnných :

	Proměnná	***Typ***	***Význam***
VAR_INPUT			
	Name	STRING	Název atributu
VAR_IN_OUT			
	itemXML	TXmlItem	Popis zpracovaného XML elementu
GetAttrValue			
	Návratová hodnota	STRING	Hodnota atributu

6 FUNKČNÍ BLOKY

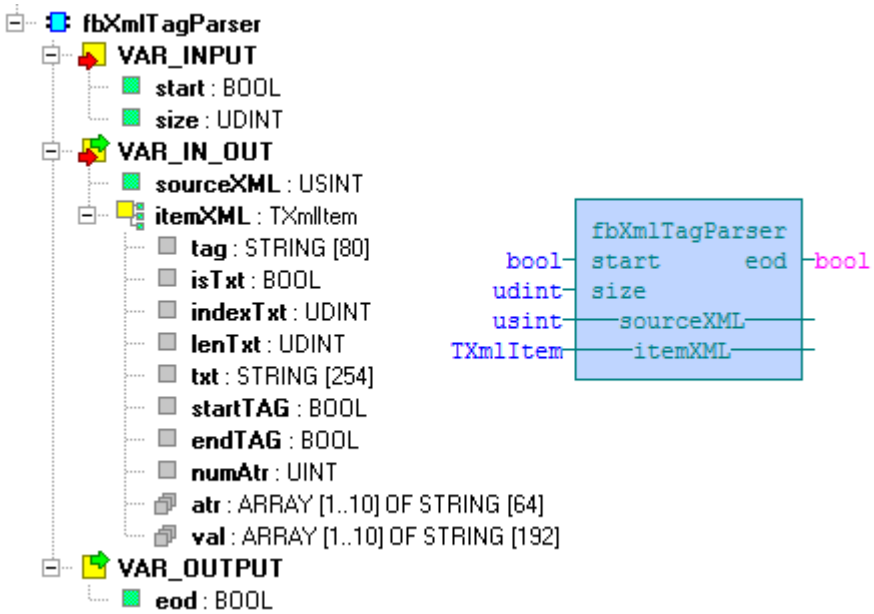
V knihovně XmlLibEx jsou definovány následující funkční bloky:



Funkční blok	*Popis*
fbXmlTagParser	Zpracovává elementy XML dokumentu
fbXmlTagComposer	Zapisuje elementy XML dokumentu
fbXmlParser	Zpracovává jednu položku XML dokumentu Tento blok je použit ve *fbXmlFileParser()* a *fbXmlPageParser()*
fbXmlFileParser	Pársování XML dokumentu ze souboru
fbXmlPageParser	Pársování XML dokumentu z web stránky

6.1 Funkční blok fbXmlTagParser

Knihovna : *XmlLibEx*



Funkční blok *fbXmlTagParser* slouží k rozebírání XML dokumentu po jednotlivých elementech. Na začátku rozebírání je nutné nastavit proměnnou *start* na hodnotu TRUE a proměnnou *size* na velikost proměnné, ve které je uložen XML dokument. Vlastní proměnná s XML dokumentem se předává na vstupu *sourceXML*. Výsledek zpracování se vrací ve struktuře *TXmlItem* na vstupu *itemXML*. Následující volání s proměnnou *start* nastavenou na hodnotu FALSE vrací další

elementy XML v pořadí, jak za sebou v dokumentu následují. Výstup *eod* je nastaven v případě, že byly přečteny všechny elementy XML dokumentu.

Ve struktuře *TXmlItem* položka *numAtr* udává počet atributů načteného tagu (0 až 10). Příznaky *startTAG* a *endTAG* v sobě nesou informaci o tom jestli se jedná o počáteční nebo koncový tag. Pokud mají oba příznaky hodnotu TRUE jedná se o nepárový tag. Příznak *isTxt* je nastaven v případě, že zpracovávaný tag obsahuje pole text. Délku textu pak obsahuje položka *lenTxt* a položka *indexTxt* obsahuje index od začátku *sourceXML*, na kterém text začíná. To má význam v případě, že je text delší než konstanta *XML_MAX_TXT_LEN* (254 znaků) a celý text se tím pádem nevejde do položky *txt*. Kompletní text je pak dostupný pouze v bufferu *sourceXML*, položka *indexTxt* říká ja daleko od začátku *sourceXML* text leží a položka *lenTxt* říká, jak je dlouhý.

Popis proměnných :

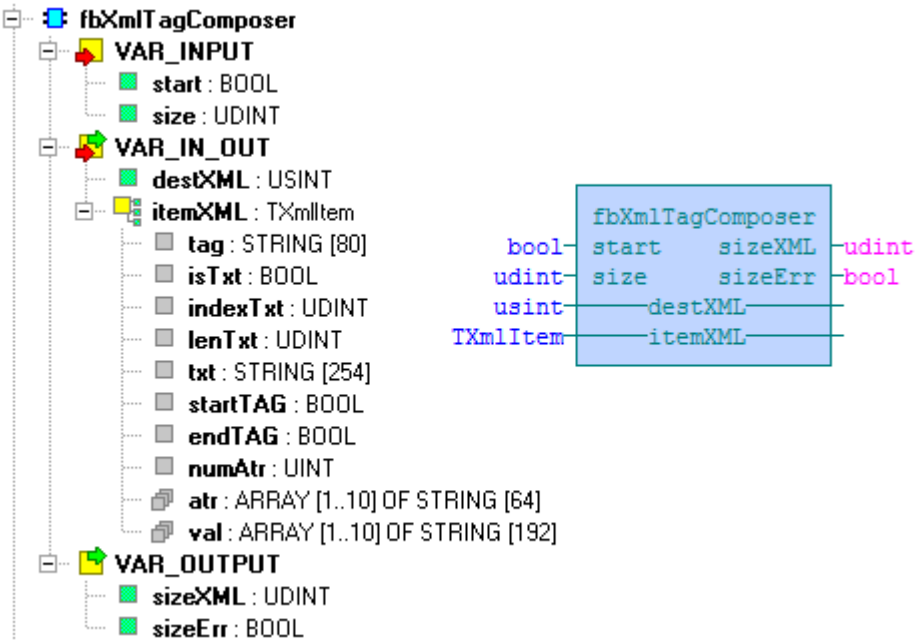
	Proměnná	***Typ***	***Význam***
VAR_INPUT			
	*start*	BOOL	TRUE začne procházet od začátku XML dat, FALSE pokračuje tam, kde se minule skončilo
	*size*	UDINT	Velikost XML dokumentu v bytech
VAR_IN_OUT			
	*sourceXML*	USINT	Proměnná obsahující XML dokument
	*itemXML*	TXmlItem	Popis zpracovaného XML elementu
VAR_OUTPUT			
	*eod*	BOOL	Konec dat, dokument byl přečten až do velikosti udané proměnnou *size*

Příklad použití (komentář „...“ je potřeba nahradit zpracováním XML tagů) :

```
VAR_GLOBAL
  xmlTemp : ARRAY [0..3] OF STRING := ['<data>', '<date></date>',
     '<value valid=""></value>', '</data>'];
END_VAR

PROGRAM prgXmlExample
  VAR
    ParserLineXML : fbXmlTagParser;
    itemXML       : TXmlItem;
  END_VAR

  ParserLineXML(start := 1, size := SIZEOF(xmlTemp),
    sourceXML := void(xmlTemp), itemXML := itemXML);
  IF itemXML.tag = 'data' THEN
    WHILE NOT ParserLineXML.eod AND itemXML.tag <> '/data' DO
      ParserLineXML(start := 0, sourceXML := void(xmlTemp), itemXML := itemXML);
      (* ... *)
    END_WHILE;
  END_IF;
END_PROGRAM
```

6.2 Funkční blok fbXmlTagComposer

Knihovna : *XmlLibEx*



Funkční blok *fbXmlTagComposer* slouží k zápisu XML dokumentu do proměnné. Na začátku zápisu je nutné nastavit proměnnou *start* na hodnotu TRUE a proměnnou *size* na velikost proměnné, ve které je bude výsledný XML dokument. Cílová proměnná se předává na vstupu dest*XML*. Následující volání s proměnnou *start* nastavenou na hodnotu FALSE zapisuje další elementy XML za poslední zapsaný. Výsledný zápis je ovlivněn příznaky *startTAG, endTAG* a *numAtr*, které musí být před voláním bloku *fbXmlTagComposer* nastaveny ve struktuře *itemXML*. Položka *numAtr* udává počet atributů zapisovaného tagu (0 až 10). Pokud jsou příznaky *startTAG* a *endTAG* oba TRUE, pak bude tag zapsán jako nepárový (např. <tag atrib=”value” />).

Popis proměnných :

	Proměnná	*Typ*	*Význam*
VAR_INPUT			
	start	BOOL	TRUE začne ukládat od začátku XML dat, FALSE pokračuje tam, kde se minule skončilo
	size	UDINT	Velikost proměnné, do které se ukládá XML dokument (počet bytů)
VAR_IN_OUT			
	destXML	USINT	Proměnná, obsahující vytvářený XML dokument
	itemXML	TXmlItem	Popis elementu, který bude zapsán do XML
VAR_OUTPUT			
	sizeXML	UDINT	Aktuální velikost XML dokumentu

	Proměnná	*Typ*	*Význam*
	*sizeErr*	BOOL	Nedostatek místa pro zápis elementu

Příklad programu s funkčním blokem *fbXmlTagComposer:*

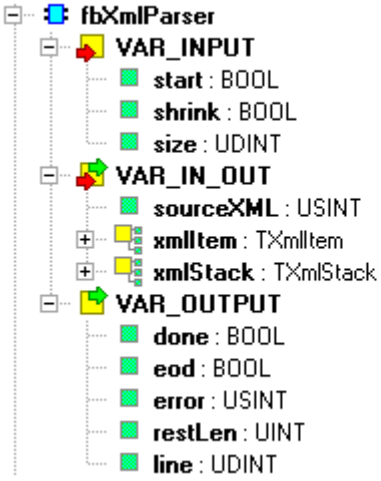
```
VAR_GLOBAL
  xmlResult : ARRAY [0..511] OF USINT;
END_VAR

PROGRAM prgExampleCompose
  VAR
    ComposerLineXML : fbXmlTagComposer;
    lineXML         : TXmlItem;
  END_VAR
  VAR_TEMP
    i : UINT;
  END_VAR

  FOR i := 0 TO 5 DO
    CASE i OF
      0 : lineXML.tag := 'data';
          lineXML.startTAG := true; lineXML.endTAG := false;
          lineXML.numAtr := 0;
      1 : lineXML.tag := 'date';
      2 : lineXML.tag := '/date';
          lineXML.txt := DT_TO_STRINGF(GetDateTime(), '%TYYYY/MM/DD-hh:mm:ss');
      3 : lineXML.tag := 'value'; lineXML.atr[1] := 'valid';
          lineXML.val[1] := BOOL_TO_STRING(
            NOT(r0_p3_AI0.STAT.UNR OR r0_p3_AI0.STAT.OVR));
          lineXML.numAtr := 1;
          lineXML.txt := '';
      4 : lineXML.startTAG := false; lineXML.endTAG := true;
          lineXML.tag := '/value';
          lineXML.txt := REAL_TO_STRINGF(r0_p3_AI0.ENG, '%5.2f');
          lineXML.numAtr := 0;
      5 : lineXML.tag := '/data';
          lineXML.txt := '';
    END_CASE;
    ComposerLineXML(start := i = 0,
        size := SIZEOF(xmlResult)-1, destXML := void(xmlResult),
        itemXML := lineXML);
  END_FOR;

  xmlResult[UDINT_TO_UINT(ComposerLineXML.sizeXML)] := 0;

END_PROGRAM
```

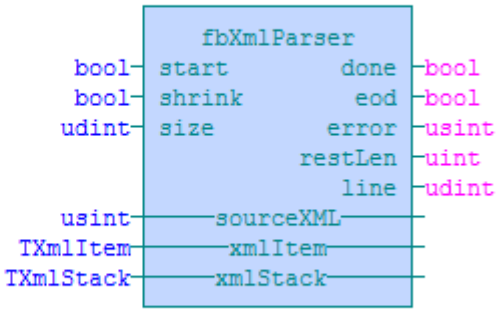
Výsledkem programu je následující XML dokument, který může vypadat například takto:

<data><date>2012/10/01-10:39:25</date><value valid="1">12.5</value></data>

6.3 Funkční blok fbXmlParser

Knihovna : *XmlLibEx*





Funkční blok *fbXmlParser* slouží k rozebírání XML dokumentu po jednotlivých elementech. Tento blok je rozšířením funkčního bloku *fbXmlTagParser*.

Na začátku rozebírání je nutné nastavit proměnnou *start* na hodnotu TRUE a proměnnou *size* na velikost bufferu, ve kterém je uložen XML dokument (nebo jeho část). Odkaz na buffer s XML dokumentem se předává na vstupu *sourceXML*. Výsledek zpracování se vrací ve strukturách *TXmlItem* a *TXmlStack*. Následující volání bloku s proměnnou *start* nastavenou na hodnotu FALSE vrací další elementy XML v pořadí, jak za sebou v dokumentu následují. Vstupní proměnná *shrink* umožňuje odmazat z bufferu tu část dokumentu, která již byla zpracovaná. V bufferu může být načtena pouze část dokumentu, po zpracování této části lze přes vstup *shrink* zpracovanou část z bufferu odstranit, čímž se v bufferu uvolní místo pro načtení další části XML dokumentu. Z uvedeného vyplývá, že funkční blok *fbXmlParser* umožňuje zpracování XML dokumentu po částech. To je výhodné zejména v případech, kdy není dopředu známá velikost XML dokumentu.

Výstup *done* udává, že byl vypársován jeden element XML dokumentu. Pokud má výstup *error* hodnotu různou od nuly, pak při párssování došlo k chybě. Výstup *restLen* udává počet znaků, které zbývají zpracovat do konce dokumentu (bufferu). Výstup *eod* je nastaven v případě, že byly přečteny všechny elementy XML dokumentu.

Struktura *TxmlItem* obsahuje informace o aktálně zpracovaném XML elementu. Položka *numAtr* v této struktuře udává počet atributů načteného tagu (0 až 10). Příznaky *startTAG* a *endTAG* v sobě nesou informaci o tom jestli se jedná o počáteční nebo koncový tag. Pokud mají oba příznaky hodnotu TRUE jedná se o nepárový tag. Příznak *isTxt* je nastaven v případě, že zpracovávaný tag obsahuje pole text. Délku textu pak obsahuje položka *lenTxt* a položka *indexTxt* obsahuje index od začátku *sourceXML*, na kterém text začíná. To má význam v případě, že je text delší než konstanta *XML_MAX_TXT_LEN* (254 znaků) a celý text se tím pádem nevejde do položky *txt*. Kompletní text je pak dostupný pouze v bufferu *sourceXML*, položka *indexTxt* říká ja daleko od začátku *sourceXML* text leží a položka *lenTxt* říká, jak je dlouhý.

Struktura *TXmlStack* je zásobník sloužící k analýze dokumentu. Položka *done* udává, že byla vypársovaná další položka. Položka *eod* signalizuje konec XML dokumentu. Nenulová hodnota položky *error* signalizuje, že při rozebírání dokumentu došlo k chybě. V takovém případě je popis chyby dostupný v položce *errTxt*. Položka *level* udává úroveň zaplnění zásobníku. Vlastní zá-

sobník je v poli *layer[]*. První položka pole *layer[0]* obsahuje název vrcholového tagu, další položky pole pak obsahují názvy vnořených tagů. Maximální podporovaná úroveň vnoření XML tagů je 10.

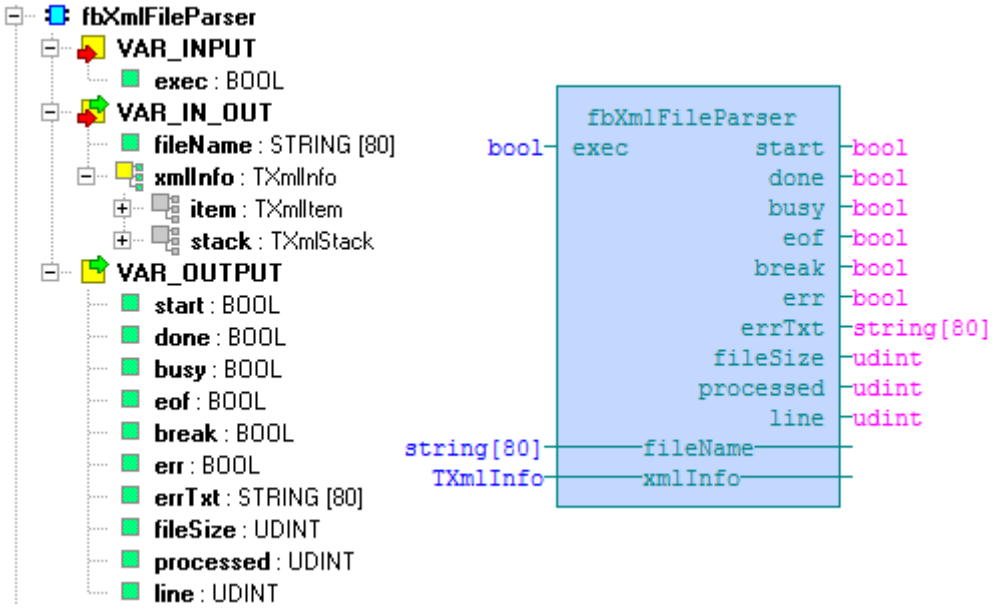
Popis proměnných :

	Proměnná	***Typ***	***Význam***
VAR_INPUT			
	start	BOOL	TRUE začne procházet od začátku XML dat, FALSE pokračuje tam, kde se minule skončilo
	shrink	BOOL	TRUE odstraní z bufferu již zpracovanou část XML dokumentu a nezpracovanou část přesune na začátek bufferu
	size	UDINT	Velikost XML dokumentu v bytech
VAR_IN_OUT			
	sourceXML	USINT	Proměnná obsahující XML dokument
	xmlItem	TXmlItem	Popis zpracovaného XML elementu
	xmlStack	TXmlStack	Zásobník pro analýzu dokumentu
VAR_OUTPUT			
	done	BOOL	Byla vypársována další položka
	eod	BOOL	Konec dat, dokument byl celý zpracován
	error	USINT	Obsahuje kód chyby pokud při rozebírání dokumentu dojde k chybě
	restLen	UINT	Počet znaků v bufferu, které zbývá zpracovat
	line	UDINT	Číslo řádku v XML dokumentu

Funkční blok *fbXmlParser* je používán funkčními bloky *fbXmlFileParser* a *fbXmlPageParser* (viz následující kapitoly).

6.4 Funkční blok fbXmlFileParser

Knihovna : *XmlLibEx*



Funkční blok *fbXmlFileParser* slouží k rozebírání XML dokumentu, který je uložen v souboru. Na začátku rozebírání je nutné nastavit proměnnou *exec* na hodnotu TRUE a do proměnné *fileName* uvést název souboru, ve kterém je uložen XML dokument. Na náběžnou hranu proměnné *exec* se otevře uvedený soubor, do výstupu *filesize* se nastaví velikost souboru a načte se první část XML dokumentu. Poté se v dokumentu najde první XML element. Od nastavení proměnné exec na hodnotu TRUE do nalezení prvního XML tagu uběhne typicky několik cyklů programu (tj. několik volání instance funkčního bloku *fbXmlFileParser*). Po celou dobu zpracování je nastaven výstup *busy na TRUE*. V okamžiku nalezení XML tagu je nastaven výstup *done* na TRUE a výsledky zpracování jsou uloženy ve struktuře *TXmlInfo*, která obsahuje jak informace o nalezeném XML elementu (název tagu, počet atributů, jejich názvy a hodnoty, atd. v položce *TXmlInfo.item*) tak informace o historii pársování dokumentu (úroveň vnoření XML tagů a jejich názvy, informace o případných chybách při pársování, atd. v položce *TXmlInfo.stack*). Aplikační program se pak rozhodne, jestli použije některou z informací ze struktury *TXmlInfo*. Poté je možné cyklicky volat blok *fbXmlFileParser* až do chvíle, kdy je výstup bloku *break* nastaven na TRUE. Po každém volání je ve struktuře *TXmlInfo* informace o dalším nalezeném XML elementu. Nastavením výstupu *break* na TRUE blok signalizuje, že je zpracovaná aktuálně načtená část souboru. Při dalším volání bloku se načte další část XML dokumentu ze souboru a pársování souboru pokračuje stejně jako po načtení první části dokumentu. Při dosažení konce souboru je nastaven výstup *eof* na TRUE. Výstup *processed* udává průběžně počet zpracovaných znaků, výstup *line* obsahuje číslo aktuálně zpracované řádky v XML souboru.

Struktura *TXmlItem* obsahuje informace o aktálně zpracovaném XML elementu. Položka *numAtr* v této struktuře udává počet atributů načteného tagu (0 až 10). Příznaky *startTAG* a *endTAG* v sobě nesou informaci o tom jestli se jedná o počáteční nebo koncový tag. Pokud mají oba příznaky hodnotu TRUE jedná se o nepárový tag. Příznak *isTxt* je nastaven v případě, že zpracovávaný tag obsahuje pole text. Délku textu pak obsahuje položka *lenTxt* a položka *indexTxt* obsahuje index od začátku *sourceXML*, na kterém text začíná. To má význam v případě, že je text delší než konstanta *XML_MAX_TXT_LEN* (254 znaků) a celý text se tím pádem nevejde do položky

txt. Kompletní text je pak dostupný pouze v bufferu *sourceXML*, položka *indexTxt* říká ja daleko od začátku *sourceXML* text leží a položka *lenTxt* říká, jak je dlouhý.

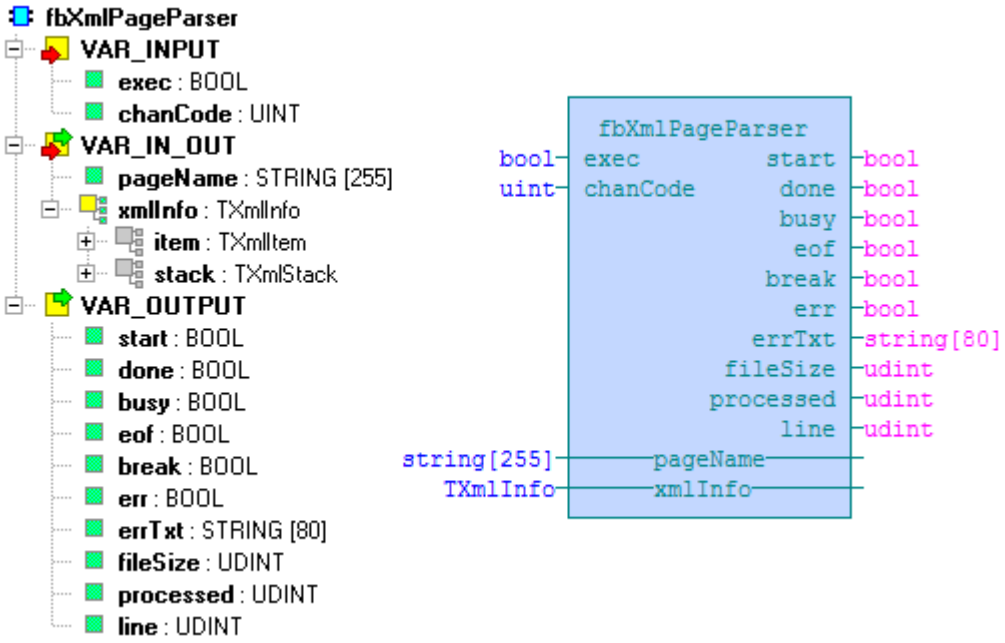
Struktura *TXmlStack* je zásobník sloužící k analýze dokumentu. Položka *done* udává, že byla vypársovaná další položka. Položka *eod* signalizuje konec XML dokumentu. Nenulová hodnota položky *error* signalizuje, že při rozebírání dokumentu došlo k chybě. V takovém případě je popis chyby dostupný v položce *errTxt*. Položka *level* udává úroveň zaplnění zásobníku. Vlastní zásobník je v poli *layer[]*. První položka pole *layer[0]* obsahuje název vrcholového tagu, další položky pole pak obsahují názvy vnořených tagů. Maximální podporovaná úroveň vnoření XML tagů je 10.

	Proměnná	***Typ***	***Význam***
VAR_INPUT			
	exec	BOOL	TRUE znamená pársovat XML soubor, na náběžnou hranu se otevře soubor a zahájí se jeho pársování. Tato proměnná musí být TRUE po celou dobu rozebírání (pársování) dokumentu.
VAR_IN_OUT			
	fileName	STRING	Jméno souboru s XML dokumentem
	xmlInfo	TXmlInfo	Popis zpracovaného XML elementu (obsahuje *TXmlItem* a *TXmlStack*)
VAR_OUTPUT			
	start	BOOL	Má hodnotu TRUE v okamžiku inicializace bloku *fbXmlFileParser* Dá se použít pro inicializaci proměnných, do kterých se budou ukládat výsledky pársování XML dokumentu
	done	BOOL	Byl vypársován jeden element XML dokumentu, výsledek pársování je k dispozici v *TXmlInfo*
	busy	BOOL	Blok je zaneprázdněn (pracuje na získání dalšího XML elementu ze souboru)
	eof	BOOL	Příznak dosažení konce souboru (byl vypársován celý XML soubor)
	break	BOOL	Aktuálně načtená část XML souboru byla zpracovaná, činnost bloku bude pokračovat v dalším cyklu načtením další části souboru
	err	BOOL	TRUE znamená, že při práci bloku došlo k chybě. Popis chyby viz *errTxt*
	errTxt	STRING	Popis chyby (pokud žádná chyba nenastala, pak string je prázdný)
	fileSize	UDINT	Velikost zpracovávaného souboru (počet znaků)
	processed	UDINT	Kolik již bylo ze souboru zpracováno (počet znaků)
	line	UDINT	Číslo řádku v XML dokumentu, který byl právě zpracován

Příklad použití viz kap.7.2 Použití fbXmlFileParser

6.5 Funkční blok fbXmlPageParser

Knihovna : *XmlLibEx*



Funkční blok *fbXmlPageParser* slouží k rozebírání XML dokumentu, který je načten z web serveru metodou GET. Na začátku rozebírání je nutné nastavit proměnnou *exec* na hodnotu TRUE a do proměnné *pageName* uvést název odkazu na soubor obsahující XML dokument. V proměnné *chanCode* musí být uveden kód kanálu, který bude použit pro komunikaci s web serverem. Na náběžnou hranu proměnné *exec* se naváže spojení s web severem, ze kterého bude načten uvedený soubor, načte se první část XML dokumentu a do výstupu *filesize* se nastaví velikost aktuálně načtené části souboru. Poté se v dokumentu najde první XML element. Od nastavení proměnné exec na hodnotu TRUE do nalezení prvního XML tagu uběhne typicky několik cyklů programu (tj. několik volání instance funkčního bloku *fbXmlPageParser*). Po celou dobu zpracování je nastaven výstup *busy na TRUE*. V okamžiku nalezení XML tagu je nastaven výstup *done* na TRUE a výsledky zpracování jsou uloženy ve struktuře *TXmlInfo*, která obsahuje jak informace o nalezeném XML elementu (název tagu, počet atributů, jejich názvy a hodnoty, atd. v položce *TXmlInfo.item*) tak informace o historii pársování dokumentu (úroveň vnoření XML tagů a jejich názvy, informace o případných chybách při pársování, atd. v položce *TXmlInfo.stack*). Aplikační program se pak rozhodne, jestli použije některou z informací ze struktury *TXmlInfo*. Poté je možné cyklicky volat blok *fbXmlPageParser* až do chvíle, kdy je výstup bloku *break* nastaven na TRUE. Po každém volání je ve struktuře *TXmlInfo* informace o dalším nalezeném XML elementu. Nastavením výstupu *break* na TRUE blok signalizuje, že je zpracovaná aktuálně načtená část souboru. Při dalším volání bloku se načte další část XML dokumentu z web serveru a pársování souboru pokračuje stejně jako po načtení první části dokumentu. Při dosažení konce souboru je nastaven výstup *eof* na TRUE. Výstup *processed* udává průběžně počet zpracovaných znaků, výstup *line* obsahuje číslo aktuálně zpracované řádky v XML souboru.

Pro komunikaci je nutné použít rozhraní ETH1 v režimu uni – TCP master, velikost přijímací zóny 512 bytů, velikost vysílací zóny 512 bytů, vzdálená IP adresa 0.0.0.0, vzdálený port 80, místní port 0. Komunikace může probíhat jak v lokální síti tak přes internet.

Ve struktuře *TXmlItem* položka *numAtr* udává počet atributů načteného tagu (0 až 10). Příznaky *startTAG* a *endTAG* v sobě nesou informaci o tom jestli se jedná o počáteční nebo koncový tag. Pokud mají oba příznaky hodnotu TRUE jedná se o nepárový tag. Příznak *isTxt* je nastaven v případě, že zpracovávaný tag obsahuje pole text. Délku textu pak obsahuje položka *lenTxt* a položka *indexTxt* obsahuje index od začátku *sourceXML*, na kterém text začíná. To má význam v případě, že je text delší než konstanta *XML_MAX_TXT_LEN* (254 znaků) a celý text se tím pádem nevejde do položky *txt*. Kompletní text je pak dostupný pouze v bufferu *sourceXML*, položka *indexTxt* říká ja daleko od začátku *sourceXML* text leží a položka *lenTxt* říká, jak je dlouhý.

Struktura *TXmlStack* je zásobník sloužící k analýze dokumentu. Položka *done* udává, že byla vypársovaná další položka. Položka *eod* signalizuje konec XML dokumentu. Nenulová hodnota položky *error* signalizuje, že při rozebírání dokumentu došlo k chybě. V takovém případě je popis chyby dostupný v položce *errTxt*. Položka *level* udává úroveň zaplnění zásobníku. Vlastní zásobník je v poli *layer[]*. První položka pole *layer[0]* obsahuje název vrcholového tagu, další položky pole pak obsahují názvy vnořených tagů. Maximální podporovaná úroveň vnoření XML tagů je 10.

Popis proměnných :

	Proměnná	***Typ***	***Význam***
VAR_INPUT			
	exec	BOOL	TRUE znamená pársovat XML soubor, na náběžnou hranu se naváže spojení s web serverem, načte se první část souboru a zahájí se jeho pársování. Tato proměnná musí být TRUE po celou dobu rozebírání (pársování) dokumentu.
	chanCode	UINT	Kód komunikačního kanálu *ETH1_uni0 ... ETH1_uni7*
VAR_IN_OUT			
	pageName	STRING	Odkaz na soubor s XML dokumentem (jméno web stránky)
	xmlInfo	TXmlInfo	Popis zpracovaného XML elementu (obsahuje *TXmlItem* a *TXmlStack*)
VAR_OUTPUT			
	start	BOOL	Má hodnotu TRUE v okamžiku inicializace bloku *fbXmlPageParser* Dá se použít pro inicializaci proměnných, do kterých se budou ukládat výsledky pársování XML dokumentu
	done	BOOL	Byl vypársován jeden element XML dokumentu, výsledek pársování je k dispozici v *TXmlInfo*
	busy	BOOL	Blok je zaneprázdněn (pracuje na získání dalšího XML elementu ze souboru)
	eof	BOOL	Příznak dosažení konce souboru (byl vypársován celý XML soubor)
	break	BOOL	Aktuálně načtená část XML souboru byla zpracovaná, činnost bloku bude pokračovat v dalším cyklu načtením další části souboru
	err	BOOL	TRUE znamená, že při práci bloku došlo k chybě. Popis chyby viz *errTxt*

	Proměnná	***Typ***	***Význam***
	errTxt	STRING	Popis chyby (pokud žádná chyba nenastala, pak string je prázdný)
	fileSize	UDINT	Aktuální velikost souboru načteného z web serveru (počet znaků)
	processed	UDINT	Kolik již bylo ze souboru zpracováno (počet znaků)
	line	UDINT	Číslo řádku v XML dokumentu, který byl právě zpracován

Příklad použití viz kap. 7.3 Použití fbXmlPageParser

7 PŘÍKLADY POUŽITÍ

7.1 Použití fbXmlTagParser a fbXmlTagComposer

Následující příklad ukazuje možnosti použití funkčních bloků *fbXmlTagParser* a *fbXmlTagComposer*. Pomocí bloku *fbXmlTagParser* je vyčtena struktura XML dokumentu z proměnné *templateXml*. Načtená struktura je doplňována programem a opět převáděna na XML dokument, blokem *fbXmlTagComposer*.

```
VAR_GLOBAL
  templateXml : ARRAY [0..3] OF STRING :=
    ['<data>',
     '<date></date>',
     '<value valid=""></value>',
     '</data>'];
  resultXml : ARRAY [0..511] OF USINT;
  resultTxt AT resultXml : STRING[255];
END_VAR

PROGRAM prgXmlExample
  VAR
    ParserLineXML   : fbXmlTagParser;
    ComposerLineXML : fbXmlTagComposer;
    lineXML         : TXmlItem;
  END_VAR
  VAR_TEMP
    i : UINT;
  END_VAR

  ParserLineXML( start := 1, size := SIZEOF(templateXml),
                 sourceXML := void(templateXml), itemXML := lineXML);
  IF lineXML.tag = 'data' THEN
    ComposerLineXML( start := 1, size := SIZEOF(resultXml)-1,
                     destXML := void(resultXml),
                     itemXML := lineXML);
    WHILE NOT ParserLineXML.EOD DO
      ParserLineXML( start := 0, sourceXML := void(templateXml),
                     itemXML := lineXML);
      IF lineXML.tag = '/date' THEN
        lineXML.txt := DT_TO_STRINGF(GetDateTime(), '%TYYYY/MM/DD-hh:mm:ss');
      END_IF;
      IF  lineXML.tag = 'value' THEN
        FOR i := 1 TO lineXML.numAtr DO
          IF lineXML.atr[i] = 'valid' THEN
            lineXML.val[i] := BOOL_TO_STRING(
                              NOT(r0_p3_AI0.STAT.UNR OR r0_p3_AI0.STAT.OVR));
            EXIT;
          END_IF;
        END_FOR;
      END_IF;
      IF  lineXML.tag = '/value' THEN
        lineXML.txt := REAL_TO_STRINGF(r0_p3_AI0.ENG, '%5.2f');
      END_IF;
      ComposerLineXML( start := 0,
                       destXML := void(resultXml),
                       itemXML := lineXML);
    END_WHILE;
```

```
  END_IF;

  resultXml[UDINT_TO_UINT(ComposerLineXML.sizeXML)] := 0;
END_PROGRAM
```

Program nejprve zkontroluje, zda-li je počáteční tag <data>. Pokud ano, hledá koncový tag </date> před, který přidá datum zformátovaný funkcí z knihovny ToStringLib. Když narazí na tag <value> zkusí vyhledat atribut valid, do kterého doplní stav z analogového vstupu. Při nalezení koncového tagu </value> je před něj doplněn text z hodnotou analogového vstupu.

Po sestavení celého dokumentu je přidán na konec znak 0. Výsledek může vypadat například takto:

<data><date>2012/10/01-09:14:28</date><value valid =“1“>12.5</value></data>

7.2 Použití fbXmlFileParser

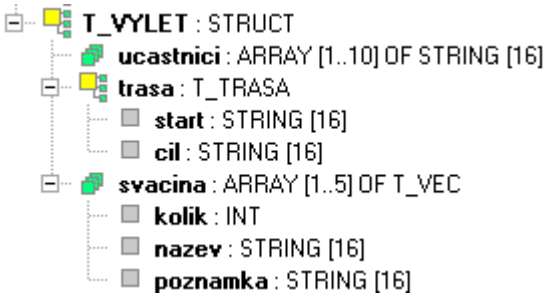
Funkční blok *fbXmlFileParser* umožňuje získat informace ze XML dokumentu, který je uložen na SD kartě v PLC. Předpokládejme, že se XML soubor bude jmenovat vylet.xml a jeho obsah bude následující:

```
<?xml version="1.0" encoding="UTF-8" ?>
<výlet>
    <účastníci>
        <osoba jméno="Pepa" />
        <osoba jméno="Franta" />
    </účastníci>
    <trasa>
        <start jméno="Kuřim" />
        <cíl   jméno="Tišnov" />
    </trasa>
    <svačina>
        <věc kolik="1">chleba</věc>
        <věc kolik="2">
            <poznámka> dobře zabalit! </poznámka>
            řízek
        </věc>
    </svačina>
</výlet>
```

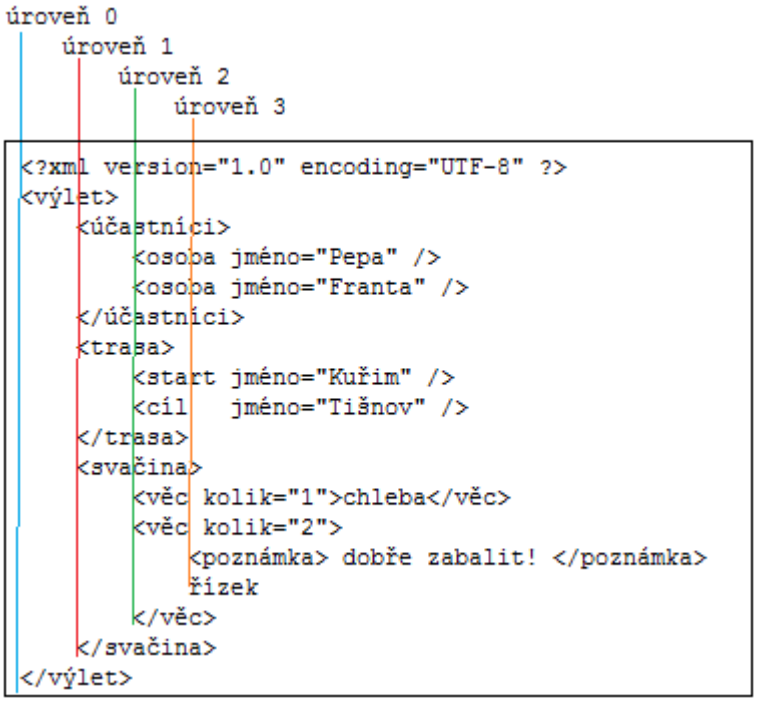
Prvním krokem bude návrh struktury dat, do které budeme ukládat informace získané rozebíráním (pársováním) xml souboru. V tomto případě bude vhodné, aby navržená datová struktura kopírovala strukturu tagů v xml souboru. Strukturu nazveme `T_VYLET` a její položky nazveme `ucastnici`, `trasa` a `svacina`. Tyto položky pak obsahují informace uložené uvnitř odpovídajících tagů. Například tag `<účastníci>` obsahuje vnořené tagy `<osoba jméno="Pepa" />` kde jméno účastníka je atributem tagu `<osoba>`. V naší struktuře tedy připravíme pole jmen pro 10 účastníků výletu `ucastnici : ARRAY[1..10] OF STRING[16];`. Podobný postup je použit i pro tagy `<trasa>` a `<svačina>`. Jen je třeba poznamenat, že názvy položek v datové struktuře nemohou obsahovat háčky a čárky (je možno použít pouze písmena anglické abecedy). Definice struktury bude vypadat následovně:

```
TYPE
  T_TRASA : STRUCT
    start      : STRING[16];
    cil        : STRING[16];
  END_STRUCT;

  T_VEC : STRUCT
    kolik      : INT;
    nazev      : STRING[16];
    poznamka   : STRING[16];
  END_STRUCT;

  T_VYLET : STRUCT
    ucastnici : ARRAY[1..10] OF STRING[16];
    trasa     : T_TRASA;
    svacina   : ARRAY[1..5] OF T_VEC;
  END_STRUCT;
END_TYPE
```

Graficky vyjádřeno bude struktura dat vypadat následovně:



Pro další postup je dobré si uvědomit, že každý tag v xml dokumentu je jednoznačně identifikován názvem tagu a úrovní, na které je v xml struktuře uložen.



V našem případě jsou na úrovni 0 tagy `<?xml>` a `<výlet>`, na úrovni 1 jsou tagy `<účastníci>`, `<trasa>` a `<svačina>`, na úrovni 2 jsou tagy `<osoba>`, `<start>`, `<cíl>` a `<věc>` a na úrovni 3 je tag `<poznámka>`.

Takže například pro nalezení tagu `<start>` platí následující podmínka: pokud je na úrovni 0 nalezen tag `<výlet>` a na úrovni 1 je tag `<trasa>` a úrovni 2 najdeme tag `<start>` pak je tag nalezen a můžeme použít jeho atribut `jméno`. K jednoduchému vyhodnocení uvedené podmínky v programu slouží struktura *TXmlStack*, kterou nastavuje funkční blok *fbXmlFileParser*. Jméno každého nalezeného tagu je uloženo do položky *layer[i]*, kde index položky *i* odpovídá úrovni, na které byl tag nalezen.

Ukázkový příklad používá pro pársování xml souboru instanci funkčního blok *fbXmlFileParser*, která je nazvaná `XmlDocParser`. Zpracování je zahájeno na náběžnou hranu proměnné `rqDoc`. Proměnná `docName` obsahuje jméno xml souboru. Instance `XmlDocParser` je volaná v cyklu `REPEAT`. Cyklus je vykonáván až do okamžiku, kdy se nastaví výstup `XmlDocParser.break` na hodnotu TRUE. To znamená, že blok potřebuje načíst další část souboru a zpracování bude pokračovat v následujícím cyklu programu. Pokud je nastaven výstup `XmlDocParser.start` tak program provede inicializaci pomocných proměnných potřebných k rozebírání dokumentu. Když je nastaven výstup `XmlDocParser.done` tak to znamená, že byl vypársován jeden element XML dokumentu a informace o tomto elementu jsou uloženy v proměnné `xml` (ta obsahuje položky typu *TXmlItem* a *TXmlStack*) Následuje zpracování těchto informací a výsledky se uloží do proměnné `vylet`.

```
REPEAT
  // cist XML ze souboru
  XmlDocParser( exec := rqDoc, fileName := docName, xmlInfo := xml);

  IF XmlDocParser.start THEN
    //
    // blok XmlDocParser zahajil parsovani -> zahajime ho taky
    //
  END_IF;

  IF XmlDocParser.done THEN
    //
    // vyparsovan jeden radek dokumentu -> zpracujeme ho
    //
  END_IF;
UNTIL XmlDocParser.break END_REPEAT;
```

Celý program pro zpracování xml souboru vylet.xml pak vypadá následovně:

```
TYPE
  T_TRASA : STRUCT
    start      : STRING[16];
    cil        : STRING[16];
  END_STRUCT;

  T_VEC : STRUCT
    kolik      : INT;
    nazev      : STRING[16];
    poznamka   : STRING[16];
  END_STRUCT;

  T_VYLET : STRUCT
    ucastnici : ARRAY[1..10] OF STRING[16];
    trasa      : T_TRASA;
    svacina    : ARRAY[1..5] OF T_VEC;
  END_STRUCT;

  T_PARSE_LEVEL_0 : (PARSE_0_NONE, PARSE_0_VYLET);
  T_PARSE_LEVEL_1 : (PARSE_1_NONE, PARSE_1_UCAST,
                     PARSE_1_TRASA, PARSE_1_SVACINA);
END_TYPE

VAR_GLOBAL
  vylet : T_VYLET;          // data získaná z xml dokumentu
END_VAR

PROGRAM prgExampleFileParse
  VAR
    rqDoc          : BOOL := 1;              // žádost o načtení a vypársovaní xml
    docName        : STRING := 'vylet.xml'; // jméno souboru s XML dokumentem
    XmlDocParser   : fbXmlFileParser;        // FB pro pársování souboru
    xml            : TXmlInfo;               // struktura pro výsledky pársování
    lastErr        : STRING;                 // popis případné chyby

    indUcastnici   : INT;
    indSvacina     : INT;
    parseLevel0    : T_PARSE_LEVEL_0;
    parseLevel1    : T_PARSE_LEVEL_1;
  END_VAR
  VAR_TEMP
    tmpStr         : STRING;
  END_VAR

  REPEAT
    // cist XML ze souboru
    XmlDocParser( exec := rqDoc, fileName := docName, xmlInfo := xml);
    IF XmlDocParser.start THEN
      // blok XmlDocParser zahajil parsovani -> zahajime ho taky
      indUcastnici := 1;                    // nastavit indexy poli na zacatek
      indSvacina   := 1;
      parseLevel0 := PARSE_0_NONE;          // nastavit uvodni stavy pro parsovani
      parseLevel1 := PARSE_1_NONE;
    END_IF;

    IF XmlDocParser.done THEN               // vyparsovan jeden radek dokumentu

      IF xml.stack.level = 0 THEN
        IF    xml.stack.layer[0].tag = 'výlet' THEN
              parseLevel0 := PARSE_0_VYLET;
        ELSE  parseLevel0 := PARSE_0_NONE;
        END_IF;
      END_IF;
```

```
      IF xml.stack.level = 1 AND parseLevel0 = PARSE_0_VYLET THEN
        IF    xml.stack.layer[1].tag = 'účastníci' THEN
              parseLevel1 := PARSE_1_UCAST;
        ELSIF xml.stack.layer[1].tag = 'trasa'     THEN
              parseLevel1 := PARSE_1_TRASA;
        ELSIF xml.stack.layer[1].tag = 'svačina'   THEN
              parseLevel1 := PARSE_1_SVACINA;
        ELSE  parseLevel1 := PARSE_1_NONE;
        END_IF;
      END_IF;

      IF xml.stack.level = 2 AND parseLevel0 = PARSE_0_VYLET THEN
        CASE parseLevel1 OF
          PARSE_1_UCAST :
            IF xml.stack.layer[2].tag = 'osoba' THEN
              IF indUcastnici <= 10 THEN
                vylet.ucastnici[indUcastnici] := GetAttrValue(name := 'jméno',
                                                              itemXML := xml.item);
                indUcastnici := indUcastnici + 1;
              END_IF;
            END_IF;

          PARSE_1_TRASA :
            IF xml.stack.layer[2].tag = 'start' THEN
              vylet.trasa.start := GetAttrValue(name := 'jméno',
                                                itemXML := xml.item);
            ELSIF xml.stack.layer[2].tag = 'cíl' THEN
              vylet.trasa.cil := GetAttrValue(name := 'jméno',
                                              itemXML := xml.item);
            END_IF;

          PARSE_1_SVACINA :
            IF indSvacina <= 5 THEN
              IF xml.stack.layer[2].tag = 'věc' THEN
                tmpStr := GetAttrValue(name := 'kolik', itemXML := xml.item);
                vylet.svacina[indSvacina].kolik := STRING_TO_INT( tmpStr);
              ELSIF xml.stack.layer[2].tag = '/věc' THEN
                vylet.svacina[indSvacina].nazev := xml.item.txt;
                indSvacina := indSvacina + 1;
              END_IF;
            END_IF;
        END_CASE;
      END_IF;

      IF xml.stack.level = 3 AND parseLevel0 = PARSE_0_VYLET THEN
        IF parseLevel1 = PARSE_1_SVACINA THEN
          IF xml.stack.layer[3].tag = '/poznámka' THEN
            IF indSvacina <= 5 THEN
              vylet.svacina[indSvacina].poznamka := xml.item.txt;
            END_IF;
          END_IF;
        END_IF;
      END_IF;
    END_IF;
  UNTIL XmlDocParser.break END_REPEAT;

  IF XmlDocParser.exec THEN
    IF XmlDocParser.eof OR XmlDocParser.err THEN  // parsovani ukonceno
      rqDoc := 0;
      IF XmlDocParser.err THEN
        lastErr := XmlDocParser.errTxt;           // zachytit chybu
      END_IF;
    END_IF;
  END_IF;
END_PROGRAM
```

Celý postup zůstane beze změny i v případě, že bude xml dokumet doplněn o další informace. Podmínkou je, aby struktura původních tagů zůstala zachována.

```
<?xml version="1.0" encoding="UTF-8" ?>
<výlet>
    <účastníci>
        <osoba jméno="Pepa" />
        <osoba jméno="Franta" />
    </účastníci>
    <trasa>
        <start jméno="Kuřim" />
        <přes  jméno="Čebín" />
        <cíl   jméno="Tišnov" />
    </trasa>
    <sraz>
        <místo>Kuřim nádraží</místo>
        <kdy datum="2013-05-11" čas="08:25:00" />
        <GPS N="49°18'3,507" E="16°32'4,673" />
    </sraz>
    <svačina>
        <věc kolik="1">chleba</věc>
        <věc kolik="2">
            <poznámka> dobře zabalit! </poznámka>
            řízek
        </věc>
    </svačina>
</výlet>
```

Pársování výše uvedeného souboru bude mít stejný výsledek jako v předchozím případě.

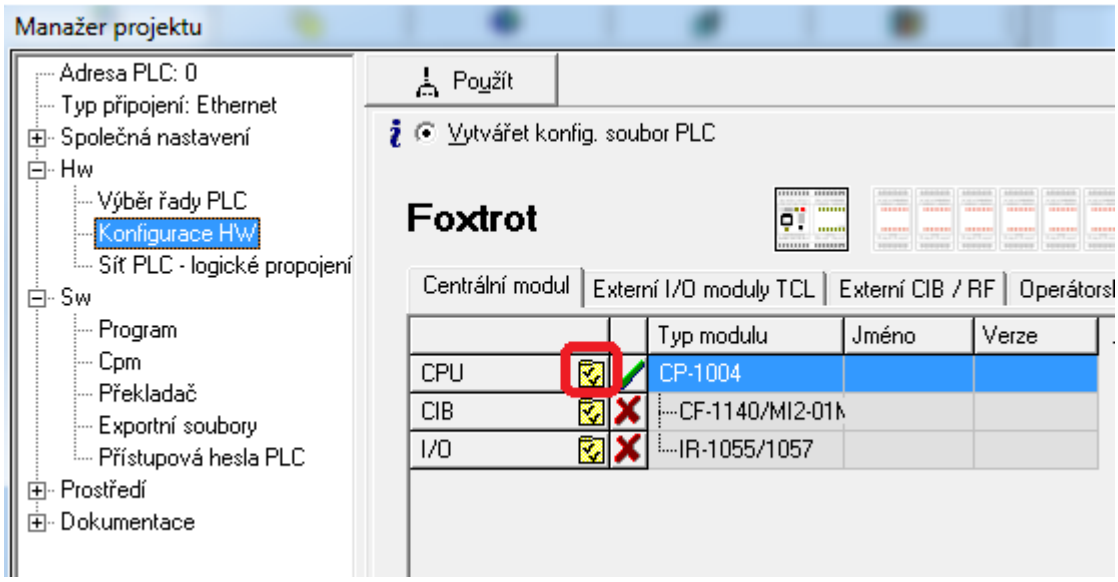
POZNÁMKA

Pokud chceme ladit uvedený příklad se simulátorem PLC v prostředí Mosaic (kde je možnost využít krokování programu), pak je třeba v adresáři projektu založit adresář ROOT (pokud již neexistuje). V tomto adresáři vytvoříme textový soubor vylet.xml a nakopírujeme do něho výše uvedený xml text. Adresář ROOT v simulátoru nahrazuje SD kartu PLC.

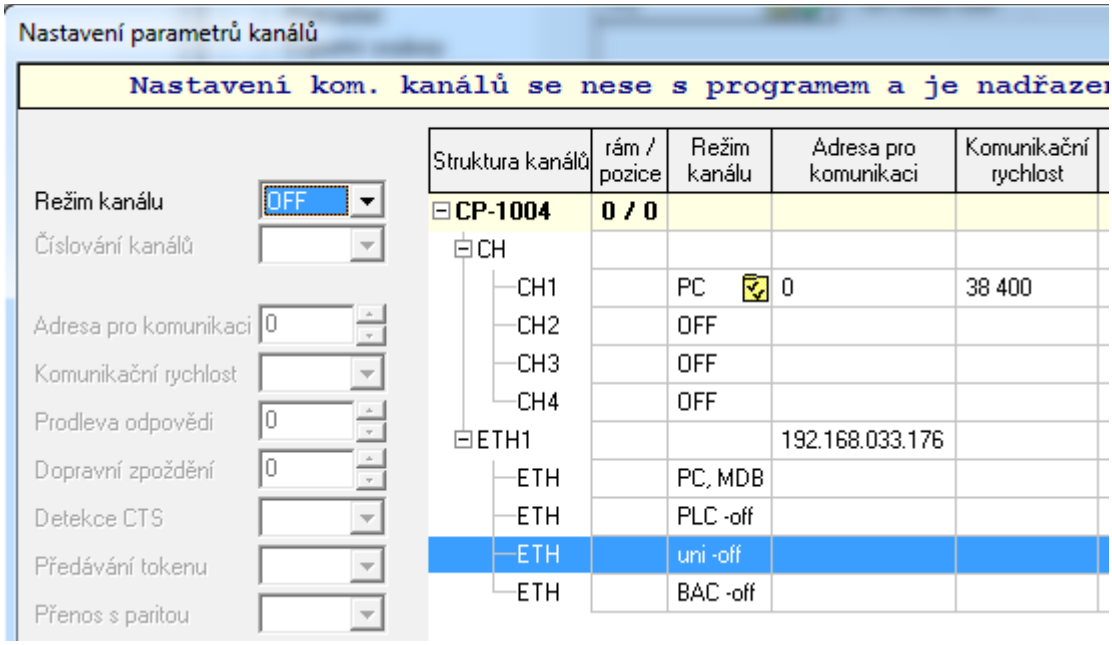
7.3 Použití fbXmlPageParser

Funkční blok *fbXmlPageParser* umožňuje získat informace ze XML dokumentu, který je poskytován web serverem. Princip práce bloku je shodný s blokem *fbXmlFileParser*, rozdíl je pouze v tom, jakým způsobem se získává obsah xml souboru. V případě bloku *fbXmlPageParser* se data získávají komunikací s web serverem (HTTP protokol, port 80, metoda GET).

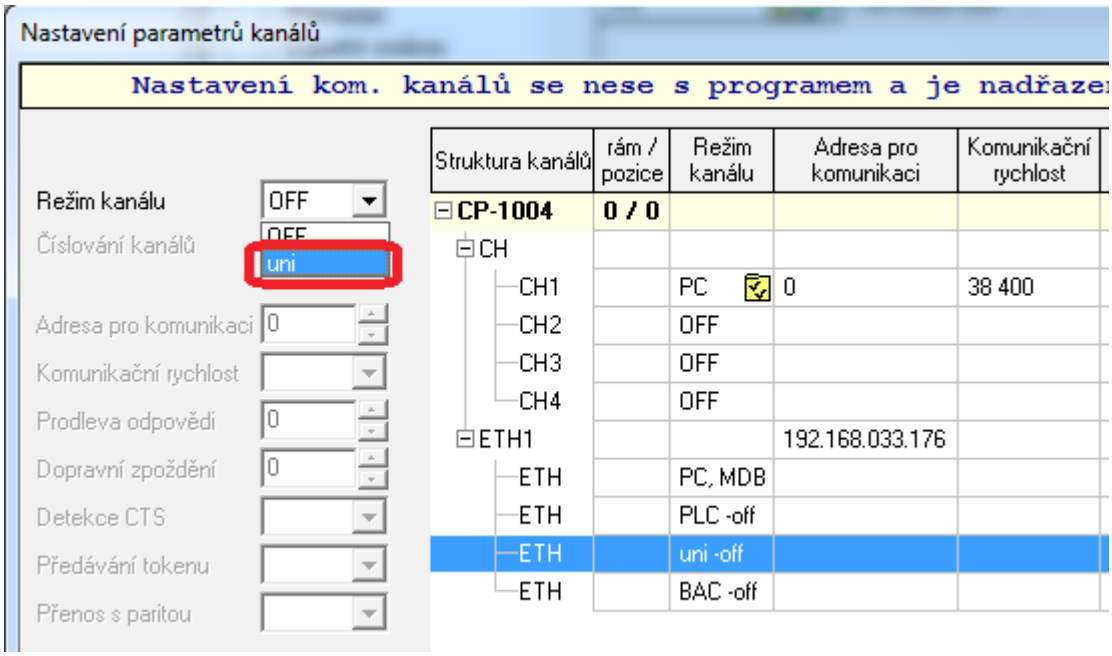
Nejprve je tedy třeba nastavit komunikační kanál. Pro spojení s web serverem je třeba nejprve zapnout podporu režimu uni na rozhraní ethernet. Toto se v prostředí Mosaic provede pomocí Manažeru projektu. Po spuštění Manažera projektu (např. CTRL+ALT+F11) vybereme myší uzel HW konfigurace. Dále je třeba vyvolat dialog pro nastavení komunikačních kanálů centrální jednotky PLC, což se provede kliknutím na ikonu v řádku CPU.



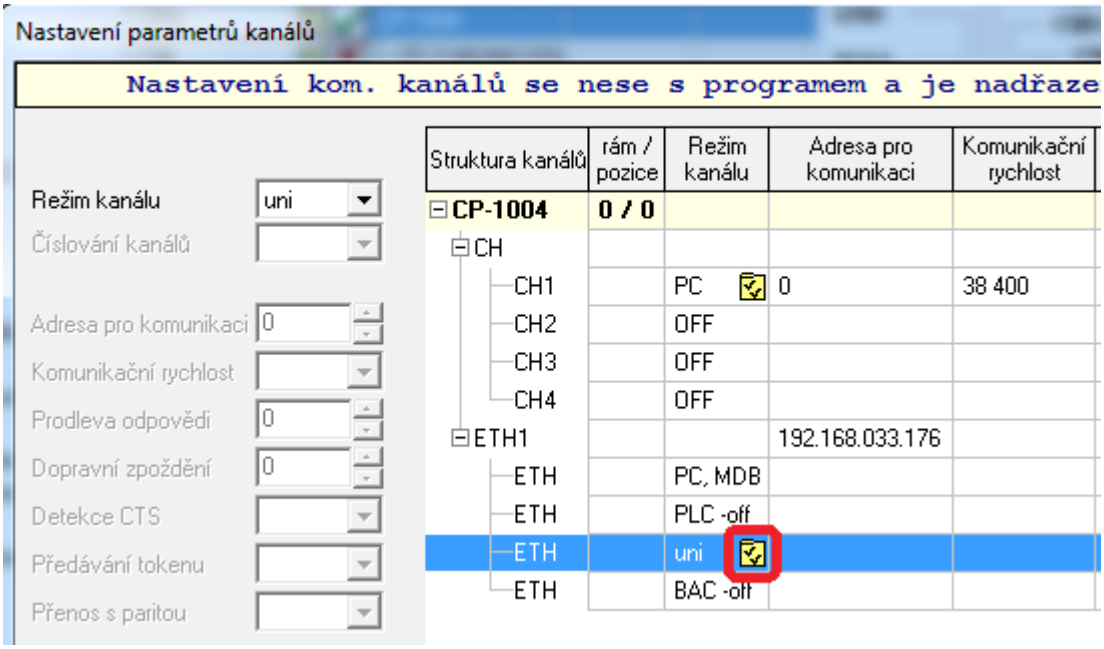
Poté klikneme na řádek s nastavením režimu uni pro rozhraní Ethernet (viz řádek ETH – uni-off) a ten se zbarví modře. V novém projektu je uni režim pro rozhraní ethernet vypnutý (viz pole Režim kanálu = OFF).



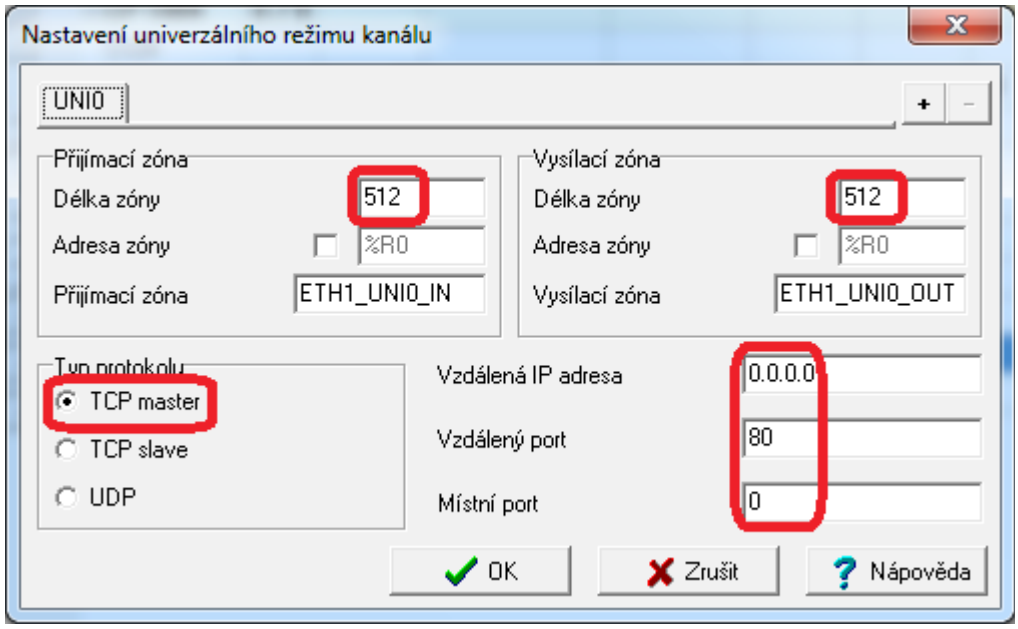
Poté je třeba zvolit režim kanálu *uni*, což se provede pomocí rozbalovacího menu jak ukazuje následující obrázek.



Následující obrázek ukazuje jak bude vypadat dialog po nastavení režimu *uni* pro kanál ethernet. Kliknutím na ikonu v řádku ETH-uni a vyvoláme dialog pro nastavení parametrů komunikace v režimu *uni*.



Objeví se dialog s názvem „Nastavení univerzálního režimu kanálu“. V něm nastavíme následující parametry pro první ethernet spojení (*ETH1_uni0*): zvolíme délku přijímací zóny 512 bytů, délku vysílací zóny 512 bytů, typ protokolu TCP master, vzdálená IP adresa 0.0.0.0, vzdálený port 80, místní port 0.



Po stisku tlačítka OK je ethernet rozhraní PLC nastaveno pro komunikaci s web serverem. Tím je nastavení komunikačního kanálu hotové.

Pro příklad použijeme službu serveru freegeoip.net, který vrací geografické informace podle IP adresy, ze které přišel dotaz. Tyto informace je server schopen zasílat v XML formátu. Službu lze snadno vyzkoušet, stačí zadat do adresního řádku webového prohlížeče odkaz http://freegeoip.net/xml/. Server vrátí xml soubor, který může vypadat např. následovně:

```
<?xml version="1.0" encoding="UTF-8"?>
<Response>
 <Ip>188.75.132.14</Ip>
 <CountryCode>CZ</CountryCode>
 <CountryName>Czech Republic</CountryName>
 <RegionCode>88</RegionCode>
 <RegionName>Stredocesky kraj</RegionName>
 <City>Kolin</City>
 <ZipCode></ZipCode>
 <Latitude>50.0269</Latitude>
 <Longitude>15.2023</Longitude>
 <MetroCode></MetroCode>
 <AreaCode></AreaCode>
</Response>
```

Odpověď serveru zpracujeme programem v PLC. Získáme tím informaci o aktuálním umístění PLC systému. Program může vypadat například následovně:

```
TYPE
  T_GEO_INFO   : STRUCT
    IP         : TIPadr;
    country    : STRING[32];
    region     : STRING[32];
    city       : STRING[32];
    latitude   : LREAL;
```

```
    longitude : LREAL;
  END_STRUCT;
END_TYPE

VAR_GLOBAL
  geoInfo     : T_GEO_INFO;            // data získaná z xml dokumentu
END_VAR

PROGRAM prgExamplePageParse
  VAR
    rqDoc         : BOOL := 1;         // žádost o načtení a vypársovaní xml
    xmlPage       : STRING := 'freegeoip.net/xml/';  // jméno web stránky
    XmlDocParser  : fbXmlPageParser;   // FB pro pársování souboru
    xml           : TXmlInfo;          // struktura pro výsledky pársování
    lastErr       : STRING;            // popis případné chyby
    parseEnable   : BOOL;              // pomocná pro pársování
  END_VAR

  REPEAT
    // cist XML ze souboru
    XmlDocParser( exec := rqDoc, chanCode := ETH1_uni0,
                  pageName := xmlPage, xmlInfo := xml);

    IF XmlDocParser.start THEN
      // blok XmlDocParser zahajil parsovani -> zahajime ho taky
      parseEnable := 0;                    // nastavit uvodni stav pro parsovani
    END_IF;

    IF XmlDocParser.done THEN              // vyparsovan jeden radek dokumentu

      IF xml.stack.level = 0 THEN
        IF xml.stack.layer[0].tag = 'Response' THEN parseEnable := 1;
        ELSE parseEnable := 0; END_IF;
      END_IF;

      IF xml.stack.level = 1 AND parseEnable THEN
        IF    xml.stack.layer[1].tag = '/Ip'          THEN
          geoInfo.IP        := STRING_TO_IPADR( xml.item.txt );
        ELSIF xml.stack.layer[1].tag = '/CountryName' THEN
          geoInfo.country   := xml.item.txt;
        ELSIF xml.stack.layer[1].tag = '/RegionName'  THEN
          geoInfo.region    := xml.item.txt;
        ELSIF xml.stack.layer[1].tag = '/City'        THEN
          geoInfo.city      := xml.item.txt;
        ELSIF xml.stack.layer[1].tag = '/Latitude'    THEN
          geoInfo.latitude  := STRING_TO_LREAL( xml.item.txt);
        ELSIF xml.stack.layer[1].tag = '/Longitude'   THEN
          geoInfo.longitude := STRING_TO_LREAL( xml.item.txt);
        END_IF;
      END_IF;

    END_IF;
  UNTIL XmlDocParser.break END_REPEAT;

  IF XmlDocParser.exec THEN
    IF XmlDocParser.eof OR XmlDocParser.err THEN  // parsovani ukonceno
      rqDoc := 0;
      IF XmlDocParser.err THEN
        lastErr := XmlDocParser.errTxt;           // zachytit chybu
      END_IF;
    END_IF;
  END_IF;
END_PROGRAM
```

Ukázkový příklad používá pro pársování xml souboru instanci funkčního blok *fbXmlPageParser*, která je nazvaná `XmlDocParser`. Zpracování je zahájeno na náběžnou hranu proměnné `rqDoc`. Proměnná `xmlPage` obsahuje název web stránky se xml souborem. Instance `XmlDocParser` je volaná v cyklu `REPEAT`. Cyklus je vykonáván až do okamžiku, kdy se nastaví výstup `XmlDocParser.break` na hodnotu TRUE. To znamená, že blok potřebuje načíst další část souboru a zpracování bude pokračovat v následujícím cyklu programu. Pokud je nastaven výstup `XmlDocParser.start` tak program provede inicializaci pomocných proměnných potřebných k rozebírání dokumentu. Když je nastaven výstup `XmlDocParser.done` tak to znamená, že byl vypársován jeden element XML dokumentu a informace o tomto elementu jsou uloženy v proměnné `xml` (ta obsahuje položky typu *TXmlItem* a *TxmlStack*). Následuje zpracování těchto informací a výsledky se uloží do proměnné `geoInfo`.

POZNÁMKA

Je pravděpodobné, že většina serverů, ze kterých bude PLC systém stahovat XML soubory, je umístěna na internetu. V takovém případě je potřeba, aby měl PLC správně nastavenou nejen IP adresu a masku sítě, ale také adresu brány (gateway). A pokud router, přes který je PLC k internetu připojen, obsahuje také DNS server, pak je užitečné nastavit i adresu DNS serveru. Není-li IP adresa DNS serveru nastavena, PLC systém použije DNS server s IP adresou 8.8.8.8. Nastavení uvedených adres lze provést například programem SetPlcIp, který je součástí instalace prostředí Mosaic nebo ho lze stáhnout z ftp://fw.tecomat.com/APP/SetPlcIP.zip.