Knihovna JsonLibEx

TXV 003 79.01
druhé vydání
listopad 2014
změny vyhrazeny

Historie změn

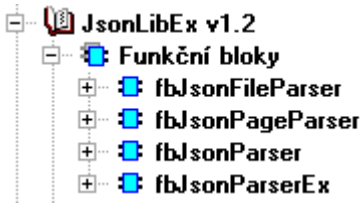
Datum	Vydání	Popis změn
Listopad 2012	1	První vydání, popis odpovídá JsonLibEx_v10
Listopad 2014	2	Doplněn popis bloků fbJsonParserEx, fbJsonFileParser a fbJson-PageParser , popis odpovídá JsonLibEx_v11

OBSAH

1 ÚVOD

Knihovna JsonLibEx je standardně dodávaná jako součást programovacího prostředí Mosaic. Knihovna obsahuje funkční bloky umožňující práci s daty ve formátu JavaScript Object Notation (JSON). JSON je textový, na jazyce zcela nezávislý formát, který se používá pro výměnu dat. Další podrobnosti o formátu JSON viz http://www.json.org/json-cz.html.

Následují obrázek ukazuje strukturu knihovny JsonLibEx v prostředí Mosaic



Pokud chceme funkce z knihovny JsonLibEx 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 JsonLibEx využívá podporu ve firmware centrální jednotky PLC. Knihovnu lze použít na všech centrálních jednotkách řady Foxtrot od firmware v7.5. V systémech TC700 lze knihovnu použít na procesorech CP-7004 a CP-7007 od v7.5.

Knihovna JsonLibEx nepokrývá kompletně všechny možnosti zápisu JSON dat. Omezení knihovny jsou následující:

- V objektech string jsou podporovány pouze ASCII znaky
- Není podporován zápis čísla ve vědecké notaci (např. 1.5E3)
- Maximální délka názvu položky je 255 znaků
- Maximální délka hodnoty položky je 255 znaků
- Vnoření struktur je podporováno do úrovně 10
  (Položka pole „spotřebuje“ 2 úrovně vnoření)

Princip činnosti

Funkční bloky v knihovně JsonLibEx analyzují JSON data a vrací název jedné položky v JSON dokumentu a její hodnotu. Pokud je blok volán opakovaně, vrací postupně všechny položky z JSON dokumentu. Na JSON data je pohlíženo jako na strukturu, takže názvy položek odpovídají způsobu, jakým se přistupuje k položce struktury v jazyce ST.

Například pokud budeme analyzovat následující JSON dokument

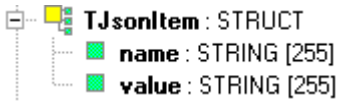
```
{
  "myObject":
  {
    "first": "John",
    "last": "Doe",
    "age": 39,
    "sex": "M",
    "salary": 70000,
    "registered": true,
    "interests": [ "Reading", "Mountain Biking", "Hacking" ],
    "favorites":
    {
      "color": "Blue",
      "sport": "Soccer",
      "food": "Spaghetti"
    }
  }
}
```

funkční blok fbJsonFileParser() bude postupně vracet následující informace

Volání bloku	Výstup bloku	
	jsonItem.name	jsonItem.value
1	myObject.first	John
2	myObject.last	Doe
3	myObject.age	39
4	myObject.sex	M
5	myObject.salary	70000
6	myObject.registered	true
7	myObject.interest[0]	Reading
8	myObject.interest[1]	Mountain Biking
9	myObject.interest[2]	Hacking
10	myObject.favorites.color	Blue
11	myObject.favorites.sport	Soccer
12	myObject.favorites.food	Spaghetti

2 DATOVÉ TYPY

V knihovně JsonLibEx jsou definovány následující datové typy:

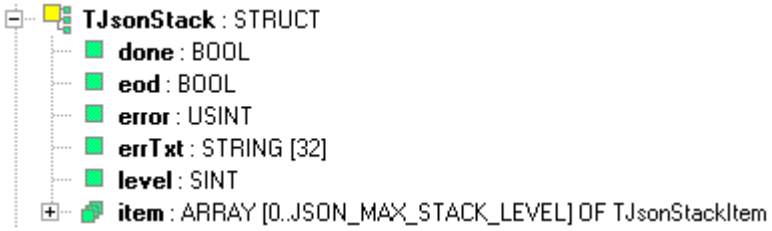


Datový typ *TJsonItem* je struktura obsahující informace o aktuálně načteném elementu JSON dokumentu . Význam jednotlivých položek je následující:

Popis proměnných :

	Proměnná	***Typ***	***Význam***
	TJsonItem	STRUCT	Struktura obsahující aktuálně načtené položky
	.name	STRING[255]	Celý název položky
	.value	STRING[255]	Hodnota položky

Dalším datovým typem je *TJsonStack* .Jedná se datovou strukturu obsahující informace o historii pársování JSON dokumentu.



Struktura vyparsovaných položek JSON dokumentu je obsažena v poli *item[ ]* .Pole *item[ ]* má strukturu *TJsonStackItem*, která je samostatně popsaná níže. Vnořené položky zvyšují úroveň stacku. Význam jednotlivých položek je následující:

Popis proměnných :

	Proměnná	***Typ***	***Význam***
	TJsonStack	STRUCT	Historie parsování JSON dokumentu
	.done	BOOL	byla vyparsovaná položka
	.eod	BOOL	konec dokumentu
	.error	USINT	kód chyby
	.errTxt	STRING[32]	popis chyby
	.maxLevel	SINT	maximální úroveň stacku
	.level	SINT	Aktuální úroveň stacku
	.item	ARRAY [0..JSON_MAX_STACK_LEVEL] OF TJsonStackItem	struktura položek v JSON dokumentu

Struktura *TJsonStackItem* obsahuje informace o historii parsování JSON dokument. Tato Struktura je součástí struktury *TJsonStack*.

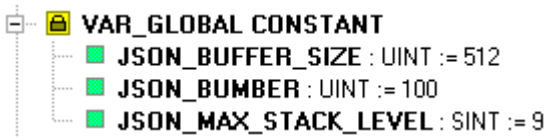
Popis proměnných :

	Proměnná	***Typ***	***Význam***
	TJsonStackItem	STRUCT	Historie parsování JSON (vnoření elementů)
	.stat	TJsonParserStat	stav parseru
	.isArray	BOOL	příznak pole
	.arrayIndex	UINT	index pole
	.name	STRING[31]	název položky

Datový typ *TJsonInfo* je struktura obsahující typy *TJsonItem* a *TjsonStack*.

3 KONSTANTY

V knihovně JsonLibEx jsou definovány následující konstanty:



Konstanty definují velikost zásobníku typu *TJsonStack.item*, také velikost bufferu a nárazníku. Význam konstant je následující:

Identifikátor	***Typ***	***Hodnota***	***Význam***
JSON_MAX_STACK_LEVEL	SINT	9	maximální úroveň JSON stacku
JSON_BUFFER_SIZE	UINT	768	Velikost pracovního bufferu pro JSON
JSON_BUMPER	UINT	256	Nárazník

4 GLOBÁLNÍ PROMĚNNÉ

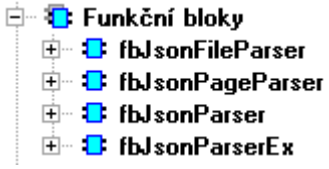
V knihovně JsonLibEx nejsou definovány žádné globální proměnné.

5 FUNKCE

V knihovně JsonLibEx nejsou definovány žádné funkce.

6 FUNKČNÍ BLOKY

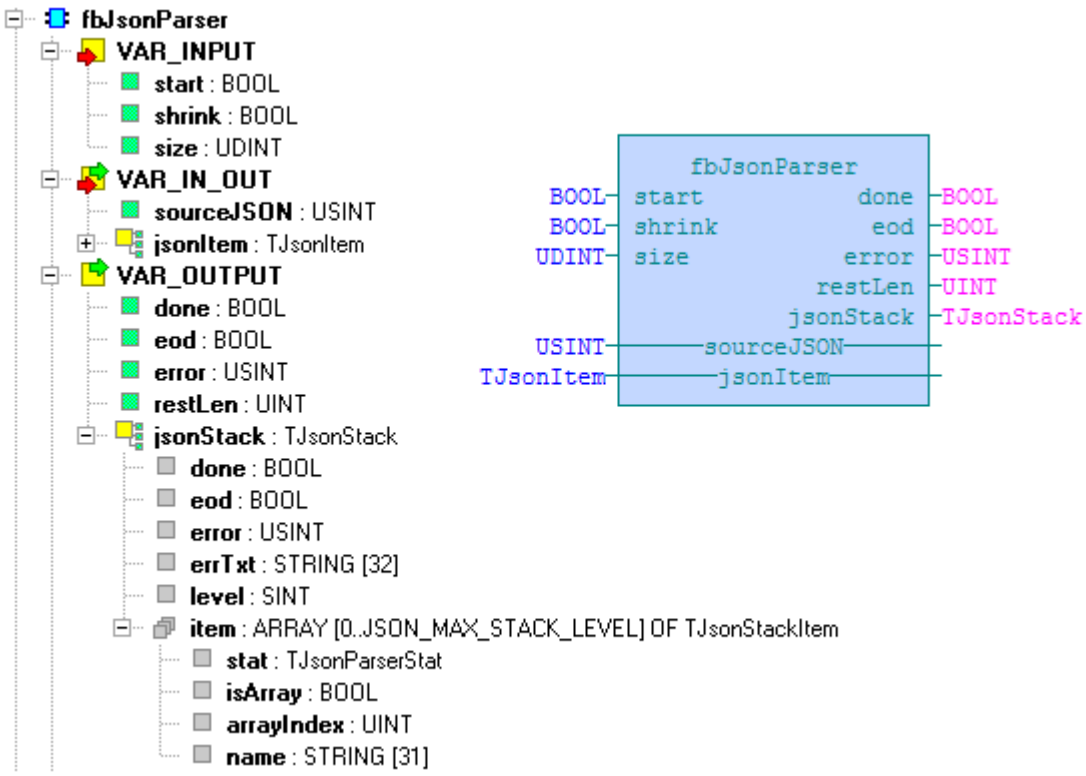
V knihovně JsonLibEx jsou definovány následující funkční bloky:



Funkční blok	***Popis***
fbJsonParser	Zpracovává jednu položku JSON dokumentu
fbJsonParserEx	Zpracovává jednu položku JSON dokumentu Tento blok je použit ve *fbJsonFileParser() a fbJsonPageParser()* získané informace jsou obsaženy ve struktuře *TJsonInfo*
fbJsonPageParser	Pársování JSON dokumentu ze souboru, získané informace jsou obsaženy ve struktuře *TJsonInfo*
fbJsonPageParser	Pársování JSON dokumentu z web stránky, získané informace jsou obsaženy ve struktuře *TJsonInfo*

6.1 *Funkční blok* fbJsonParser

Knihovna : *JsonLibEx*



Funkční blok *fbJsonParser* slouží k rozebírání JSON dokumentu na jednotlivé elementy. Na začátku rozebírání je nutné nastavit proměnnou *start* na hodnotu TRUE a proměnnou *size* na velikost proměnné, ve které je uložen JSON dokument. Vlastní proměnná s JSON dokumentem se předává na vstupu *sourceJSON*. Aktuální výsledky zpracování blok ukládá do proměnné *JsonItem*. Následující volání s proměnnou *done* nastavenou na hodnotu *TRUE* vrací další elementy JSON v pořadí, jak za sebou v dokumentu následují. S každým vyparsovaným elementem je také nastavena výstupní proměnná JsonStack. Výstup *eod* je nastaven na hodnotu *TRUE* v případě, že byly přečteny všechny elementy JSON dokumentu.

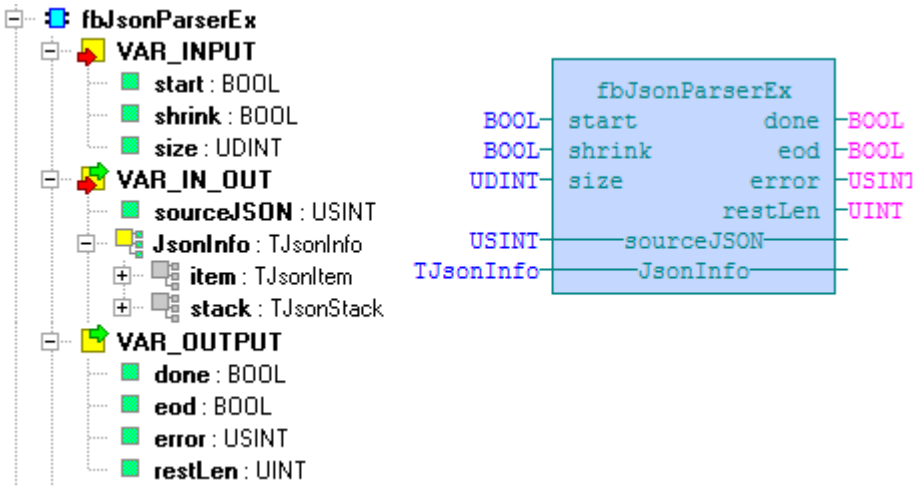
Popis proměnných :

	Proměnná	*Typ*	*Význam*
VAR_INPUT			
	start	BOOL	TRUE začne parsovat od začátku JSON dat, FALSE pokračuje tam, kde se minule skončilo
	shrink	BOOL	TRUE odstraní z bufferu již zpracovanou část a nezpracovaný zbytek přesune na začátek bufferu
	size	UDINT	velikost bufferu s JSON dokumentem
VAR_OUTPUT			
	done	BOOL	byla vyparsována další položka
	eod	BOOL	konec JSON dokumentu
	error	USINT	error <> 0 ... došlo k chybě
	restLen	UINT	počet bytů, který zbývá zpracovat
	jsonStack	TJsonStack	zásobník pro analýzu dokumentu
VAR_IN_OUT			
	sourceJSON	USINT	začátek JSON dat
	jsonItem	TJsonItem	výstup JSON parseru

Ve struktuře *jsonStack* je pole *item[ ],* které slouží jako zásobník s načtenými položkami JSON dokumentu s dalšími parametry. Index načtených položek je 0 až 9.

6.2 *Funkční blok* fbJsonParserEx

Knihovna : *JsonLibEx*



Funkční blok *fbJsonParserEx* slouží k rozebírání JSON dokumentu na jednotlivé elementy. Na začátku rozebírání je nutné nastavit proměnnou *start* na hodnotu TRUE a proměnnou *size* na velikost proměnné, ve které je uložen JSON dokument. Vlastní proměnná s JSON dokumentem se předává na vstupu *sourceJSON*. Aktuální výsledky zpracování se ukládají do proměnné *JsonInfo*. Následující volání s proměnnou *done* nastavenou na hodnotu *TRUE* vrací další elementy JSON v pořadí, jak za sebou v dokumentu následují. Výstup *eod* je nastaven na hodnotu *TRUE* v případě, že byly přečteny všechny elementy JSON dokumentu.

Ve struktuře *JsonInfo.stack* je pole *item[ ]*, která slouží jako zásobník s načtenými položkami JSON dokumentu s dalšími parametry. Index načtených položek je 0 až 9.

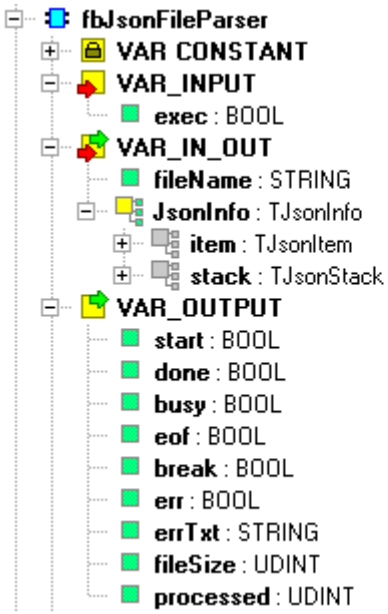
Blok *fbJsonParserEx* se chová prakticky stejně, jako blok 6.1 Funkční blok fbJsonParser. Rozdíl je v tom, že jsou výsledky parsování uloženy v jedné proměnné, která se předává při volání bloku v parametru *JsonInfo*.

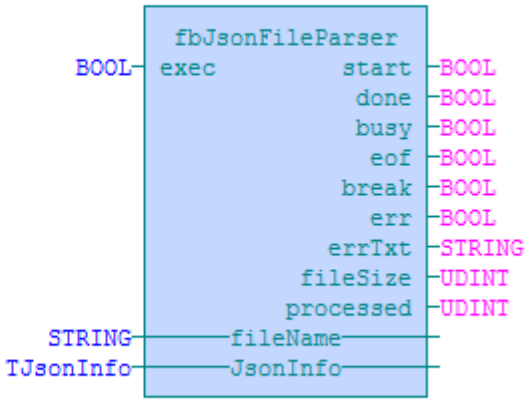
Popis proměnných :

	Proměnná	*Typ*	*Význam*
VAR_INPUT			
	start	BOOL	TRUE začne parsovat od začátku JSON dat, FALSE pokračuje tam, kde se minule skončilo
	shrink	BOOL	TRUE odstraní z bufferu již zpracovanou část a nezpracovaný zbytek přesune na začátek bufferu
	size	UDINT	velikost bufferu s JSON dokumentem
VAR_OUTPUT			
	done	BOOL	byla vyparsována další položka
	eod	BOOL	konec JSON dokumentu
	error	USINT	error <> 0 ... došlo k chybě
	restLen	UINT	počet bytů, který zbývá zpracovat
VAR_IN_OUT			
	sourceJSON	USINT	začátek JSON dat
	JsonInfo	TJsonInfo	výstup JSON parseru

6.3 *Funkční blok* fbJsonFileParser

Knihovna : *JsonLibEx*





Funkční blok *fbJsonFileParser* slouží k rozebírání JSON 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 JSON dokument. Na náběžnou hranu proměnné *exec* se otevře uvedený soubor, do výstupu *filesize* se nastaví velikost souboru a načte se první část JSON dokumentu. Poté se v dokumentu najde první JSON element. Od nastavení proměnné exec na hodnotu TRUE do nalezení prvního JSON elementu uběhne typicky několik cyklů programu (tj. několik volání instance funkčního bloku *fbJsonFileParser*). Po celou dobu zpracování je nastaven výstup *busy* na TRUE. V okamžiku nalezení dalšího element struktury JSON je nastaven výstup *done* na TRUE a výsledky zpracování jsou uloženy ve struktuře *JsonInfo*, která obsahuje jak informace o nalezeném JSON elementu (obsahují název a hodnotu elementu v položce *JsonInfo.item*) tak informace o historii pársování dokumentu (úroveň vnoření JSON elementů a jejich názvy, informace o případných chybách při pársování, atd. v položce *JsonInfo.stack*). Aplikační program se pak rozhodne, jestli použije některou z informací ze struktury *JsonInfo*. Poté je možné cyklicky volat blok *fbJsonFileParser* až do chvíle, kdy je výstup bloku *break* nastaven na TRUE. Po každém volání jsou ve struktuře *JsonInfo* informace o dalším nalezeném JSON elementu. Nastavením výstupu *break* na *TRUE* blok signalizuje, že je zpracovaná aktuálně načtená část souboru. Při dalším volání bloku se načte další část JSON 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 v JSON souboru. Struktura *JsonInfo.item* obsahuje informace o aktuálně zpracovaném JSON elementu. Položka *name* obsahuje název aktuální položky .Položka *value* obsahuje aktuální načtenou hodnotu. Všechny položky *JsonInfo.item* jsou typu *string*. Struktura *JsonInfo.stack* je zásobník sloužící k analýze dokumentu. Položka *done* udává, že byla vypársovaná další položka. Položka *eod* signalizuje konec JSON 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 *item[]*.

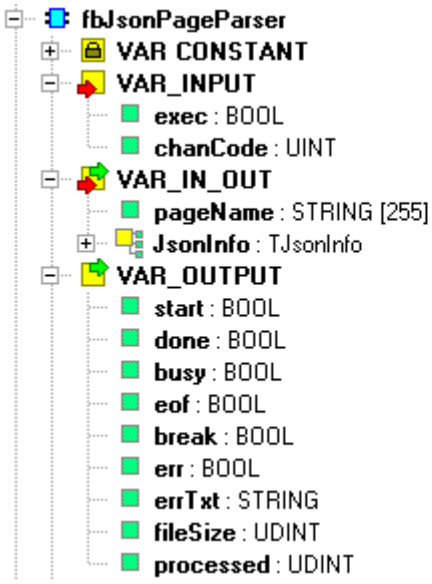
Příklad aplikačního programu je uveden v kap. 7.2 Použití fbJsonFileParser.

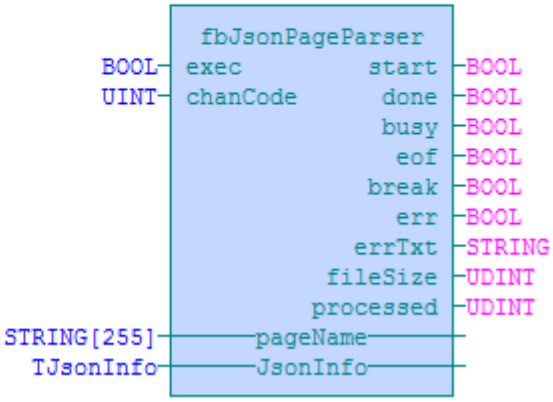
Popis proměnných :

	Proměnná	*Typ*	*Význam*
VAR_INPUT			
	exec	BOOL	parsovat JSON soubor
VAR_OUTPUT			
	start	BOOL	příznak inicializace bloku
	done	BOOL	výsledek parsování je připraven v JsonInfo
	busy	BOOL	blok je zaneprázdněn
	eof	BOOL	konec souboru
	break	BOOL	buffer zpracován, ukončit volání bloku, pársování bude pokračovat příští cyklus
	err	BOOL	příznak chyby
	errTxt	STRING	popis chyby
	fileSize	UDINT	velikost souboru
	processed	UDINT	zpracováno (počet znaků)
VAR_IN_OUT			
	fileName	STRING	jméno souboru s JSON dokumentem
	JsonInfo	TJsonInfo	výsledek parsování

6.4 *Funkční blok* fbJsonPageParser

Knihovna : *JsonLibEx*





Funkční blok *fbJsonPageParser* slouží k rozebírání JSON 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 JSON 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 JSON dokumentu a do výstupu *filesize* se nastaví velikost aktuálně načtené části souboru. Poté se v dokumentu najde první JSON element. Od nastavení proměnné exec na hodnotu TRUE do nalezení prvního JSON element uběhne typicky několik cyklů programu (tj. několik volání instance funkčního bloku *fbJsonPageParser*). Po celou dobu zpracování je nastaven výstup *busy na TRUE*. V okamžiku nalezení JSON elementu je nastaven výstup *done* na TRUE a výsledky zpracování jsou uloženy ve struktuře *JsonInfo*, která obsahuje jak informace o nalezeném JSON elementu (název a hodnota elementu v položce *JsonInfo.item*) tak informace o historii pársování dokumentu (úroveň vnoření JSON elementů ,informace o případných chybách při pársování, atd. v položce *JsonInfo.stack*). Aplikační program se pak rozhodne, jestli použije některou z informací ze struktury *JsonInfo*. Poté je možné cyklicky volat blok *fbJsonPageParser* až do chvíle, kdy je výstup bloku *break* nastaven na TRUE. Po každém volání je ve struktuře *JsonInfo* informace o dalším nalezeném JSON elementu. Nastavením výstupu *break* na TRUE blok signalizuje, že je zpracovaná aktuálně načtená část souboru. Při dalším volání bloku se načte další část JSON 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 v JSON 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.

Příklad aplikačního programu je uveden v podkapitole 7.3 Použití fbJsonPageParser .

Popis proměnných :

	Proměnná	*Typ*	*Význam*
VAR_INPUT			
	exec	BOOL	parsovat JSON soubor
	chanCode	UINT	komunikační kanál (ETH1_uni0, ETH1_uni1,...)
VAR_OUTPUT			
	start	BOOL	příznak inicializace bloku
	done	BOOL	výsledek parsování je připraven v jsonInfo
	busy	BOOL	blok je zaneprázdněn
	eof	BOOL	konec souboru
	break	BOOL	buffer zpracován, ukončit volání bloku, parsování bude pokračovat příští cyklus
	err	BOOL	příznak chyby
	errTxt	STRING	popis chyby
	fileSize	UDINT	velikost souboru
	processed	UDINT	zpracováno (počet znaků)
VAR_IN_OUT			
	pageName	STRING[255]	název web stránky s s JSON dokumentem
	JsonInfo	TJsonInfo	výsledek parsování

7 PŘÍKLADY POUŽITÍ

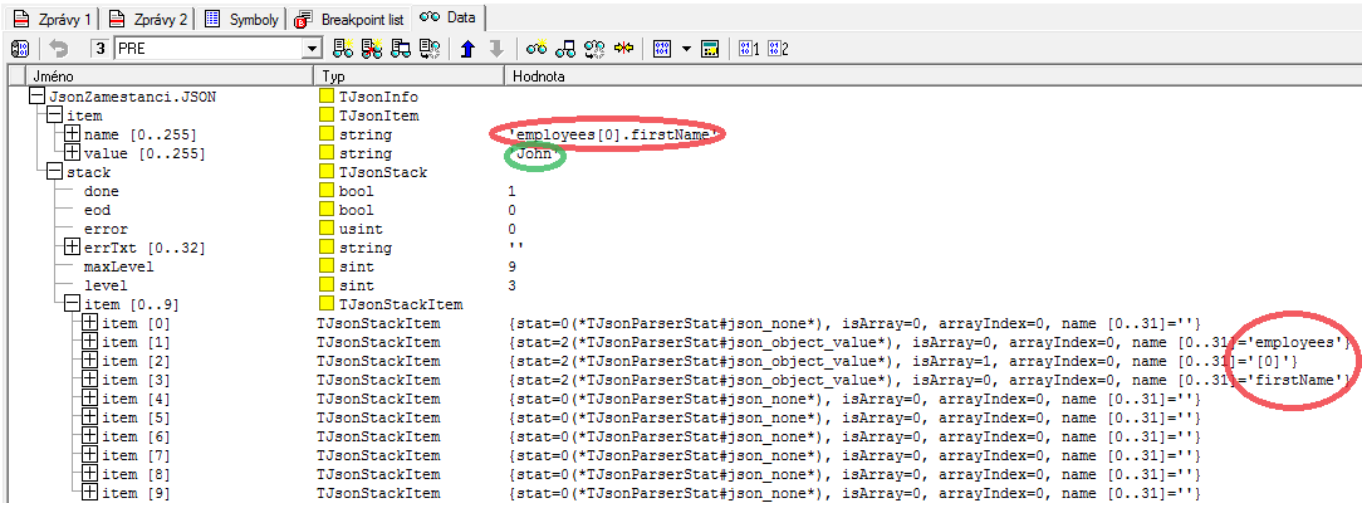
V této kapitole se seznámíme s praktickými ukázkami použití knihovny JsonLibEx.

7.1 Použití fbJsonParserEx

Následující příklad ukazuje použití funkčního bloku *fbJsonParserEx*. Pomocí tohoto bloku vyčteme strukturu z JSON souboru, který je uložen na SD kartě v PLC systému, název souboru je `employees.jsn`. Naše struktura má podobu jmen a příjmení zaměstnanců, kteří pracují ve firmě, jak je patrné níže. Tato jména načteme do struktury *Name_Employees* pro další zpracování. Načtený JSON dokument vypadá takto :

```
{
    "employees": [
        {
            "firstName": "John",
            "lastName": "Doe"
        },
        {
            "firstName": "Anna",
            "lastName": "Smith"
        },
        {
            "firstName": "Peter",
            "lastName": "Jones"
        }
    ]
}
```

Po načtení souboru do bufferu začne samotné parsování pomocí funkčního bloku *fbJsonParserEx*. Parsováním se buffer rozdělí na jednotlivé elementy z JSON struktury. Jak je patrné z okna „Data“ aktuálně načtená struktura JSON dokumentu se objeví ve struktuře *item*. Tam je rozdělena na název položky (červený kroužek) a hodnotu (zelený kroužek). V položce *item.name* je uloženo celé jméno právě načteného elementu, jenž je posléze rozděleno na jednotlivé položky struktury v zásobníku *stack.item*. V zásobníku *stack.item* je dobře vidět, že poslední položka *firstName* se nachází v třetí vrstvě zásobníku.



Podle třetí položky(*firstName* nebo *lastName*) v zásobníku jsou posléze v programu ukládána jména a příjmení do struktury *Name_Employees*.

```
VAR_GLOBAL CONSTANT
  JSON_BUFFER_SIZE_EXAMPLE : UINT := 512;     // velikost bufferu
END_VAR
TYPE
  T_TABULKA_Employees : STRUCT
    firstName   : STRING;
    lastName  : STRING;
  END_STRUCT;
END_TYPE
VAR_GLOBAL
Name_Employees  :  array[0..2] of   T_TABULKA_Employees;
END_VAR
PROGRAM employees
  VAR_INPUT
  END_VAR
  VAR_OUTPUT
  END_VAR
  VAR
    init            : BOOL;
    rqFile          : BOOL;
    rqParse         : BOOL;
    rqStart         : BOOL;
    JSON            : TJsonInfo;
    fileName        : STRING := 'employees.jsn';
    ReadFile        : ReadFromFile;
    jsonBuffer      : ARRAY[0..JSON_BUFFER_SIZE_EXAMPLE ] OF USINT;
    JsonParser      : fbJsonParserEx;
    sumaLen         : UDINT;
    actLen          : UDINT;
    restLen         : UINT;
    fInfo           : TFileInfo;
    index           : UINT;// index strukture
  END_VAR
  VAR_TEMP
  END_VAR
  // odstartovat cteni JSON dokumentu
  IF NOT JsonParser.eod THEN
    IF NOT init THEN
      init := FileInfo( fileName := fileName, fileDesc := fInfo);
            // kolik budu cist
      IF init THEN
        rqFile := 1; rqParse := 0;
        rqStart := 1; restLen := 0; sumaLen := 0;
      END_IF;
    END_IF;
    // nacitat v kazdem cyklu max. 512 bytu
    ReadFile(fileName := fileName, dstVar := void(jsonBuffer[restLen]),
              exec := rqFile, seek := sumaLen,
              size := UINT_TO_UDINT(JSON_BUFFER_SIZE_EXAMPLE  - restLen));
    IF ReadFile.done THEN
      actLen  := UINT_TO_UDINT(restLen) + ReadFile.actSize;
      // aktualni velikost json bufferu
      sumaLen := sumaLen + ReadFile.actSize;
      // celkove jiz nacteno ze souboru
      rqFile := 0; rqParse := 1;
      // shodit nabeznou hranu ReadFile.exec
      ReadFile( fileName := fileName, dstVar := void(jsonBuffer),
                exec := 0, seek := 0, size := 0);
    END_IF;
    IF rqParse  THEN
```

```
            // parsovani dat v JSON souboru
            JsonParser( start := rqStart, shrink := 0, size := actLen,
                        sourceJSON := void( jsonBuffer),JsonInfo := JSON);
            restLen := JsonParser.restLen;
                  // nastavit, kolik jeste zbyva zpracovat
            rqStart := 0;
            IF JSON.stack.item[3].name =  'firstName'  THEN
                Name_Employees[index].firstName := JSON.item.value;
            ELSIF JSON.stack.item[3].name = 'lastName'  THEN
               Name_Employees[index].lastName := JSON.item.value;
               index := index+1;
            END_IF;
          END_IF;
        ELSE
        rqParse := 0;
        index:=0;
        END_IF;
```

7.2 Použití fbJsonFileParser

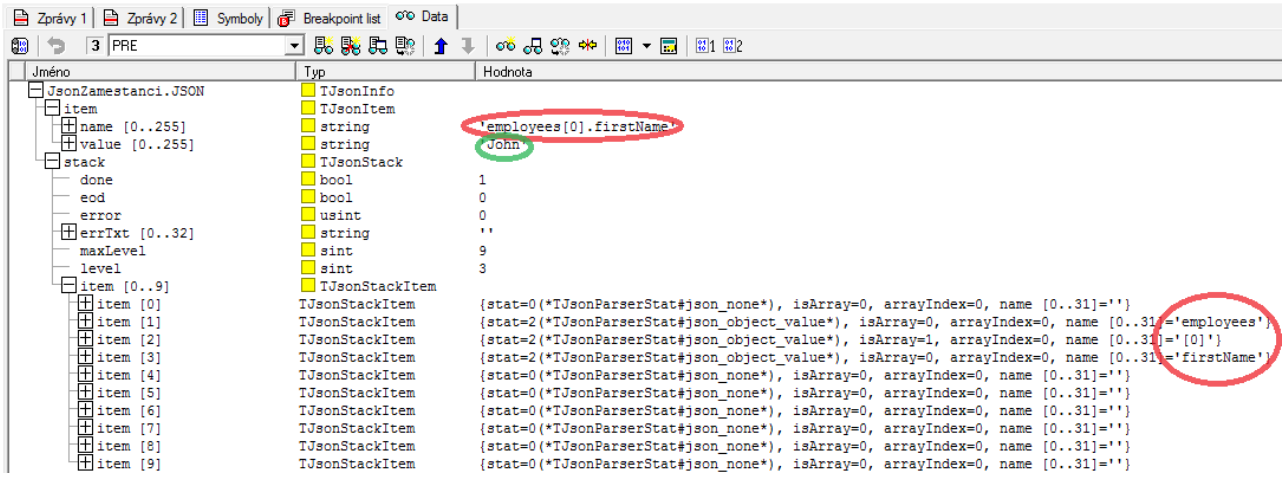
Funkční blok *fbJsonFileParser* umožňuje získat informace z JSON dokumentu, který je uložen na SD kartě v PLC. Předpokládejme, že se JSON soubor bude jmenovat `employees.jsn` a bude obsahovat následující:

```
{
    "employees": [
        {
            "firstName": "John",
            "lastName": "Doe"
        },
        {
            "firstName": "Anna",
            "lastName": "Smith"
        },
        {
            "firstName": "Peter",
            "lastName": "Jones"
        }
    ]
}
```

Prvním krokem bude návrh struktury dat, do které budeme ukládat informace získané rozebíráním (pársováním) JSON souboru. V tomto případě bude vhodné, aby navržená datová struktura kopírovala strukturu elementů v JSON souboru. Dle struktury v souboru je patrné, že obsahem struktury jsou jména a příjmení zaměstnanců ve firmě. Tyto informace si budeme ukládat do vlastní struktury pro další zpracování. Proto je zvolena následující struktura:

```
      TYPE
        T_Table_Employees : STRUCT
          firstName    : STRING;
          lastName     : STRING;
        END_STRUCT;
      END_TYPE
      VAR_GLOBAL
      Fisrt_last_name_Employees  :  array[0..2] of   T_Table_Employees;
END_VAR
```

Dále je nutné zjistit v jaké vrstvě struktury se nacházejí informace, které chceme z JSON dokumentu vyparsovat. Naše požadované informace *firstName* a *lastName* se nacházejí ve třetí vrstvě a najdeme je v *item.value* (zelený kroužek). Jednotlivé položky struktury načtené z JSON dokumentu se nachází v zásobníku *stack.item* (spodní červený kroužek).



Ukázkový příklad používá pro pársování JSON souboru instanci funkčního blok *fbJsonFileParser*, která je nazvaná *JsonDocParser*. Zpracování je zahájeno na náběžnou hranu proměnné *rqDoc*. Proměnná *docName* obsahuje jméno JSON souboru. Instance *JsonDocParser* je volaná v cyklu *REPEAT*. Cyklus je vykonáván až do okamžiku, kdy se nastaví výstup *JsonDocParser.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 *JsonDocParser.start* tak program provede inicializaci pomocných proměnných potřebných k rozebírání dokumentu. Když je nastaven výstup *JsonDocParser.done* tak to znamená, že byl vypársován jeden element JSON dokumentu a informace o tomto elementu jsou uloženy v proměnné *JSON* (ta obsahuje položky typu *item* a *stack*) Následuje zpracování těchto informací a výsledky se uloží do proměnné *Fisrt_last_name_Employees.*

Celý program pro zpracování JSON souboru *employees.jsn* pak vypadá následovně:

```
PROGRAM employeesFile
  VAR_INPUT
  END_VAR
  VAR_OUTPUT
  END_VAR
  VAR
     rqDoc          : BOOL := 1;// žádost o načtení a vypársovaní json
    docName         : STRING := 'employees.jsn';
      // jméno souboru s JSON dokumentem
    JsonDocParser : fbJsonFileParser;// FB pro pársování souboru
    JSON            : TJsonInfo;// struktura pro výsledky pársování
    lastErr         : STRING;// popis případné chyby
    index           : UINT;
  END_VAR
  VAR_TEMP
  END_VAR
 REPEAT
   JsonDocParser( exec := rqDoc, fileName := docName, JsonInfo := JSON);
   IF JsonDocParser.start THEN
```

```
            // blok JsonDocParser zahajil parsovani -> zahajime ho taky
      END_IF;
      IF JsonDocParser.done THEN     // vyparsovan jeden radek
        IF JSON.stack.item[3].name =  'firstName'  THEN
            Fisrt_last_name_Employees[index].firstName := JSON.item.value;
        ELSIF JSON.stack.item[3].name = 'lastName'  THEN
            Fisrt_last_name_Employees[index].lastName := JSON.item.value;
            index := index+1;
        END_IF;
      END_IF;
    UNTIL JsonDocParser.break END_REPEAT;
     IF JsonDocParser.exec THEN
       IF JsonDocParser.eof OR JsonDocParser.err THEN  // parsovani ukonceno
         rqDoc := 0;
         IF JsonDocParser.err THEN
           lastErr := JsonDocParser.errTxt;           // zachytit chybu
         END_IF;
       END_IF;
     END_IF;
   END_PROGRAM
```

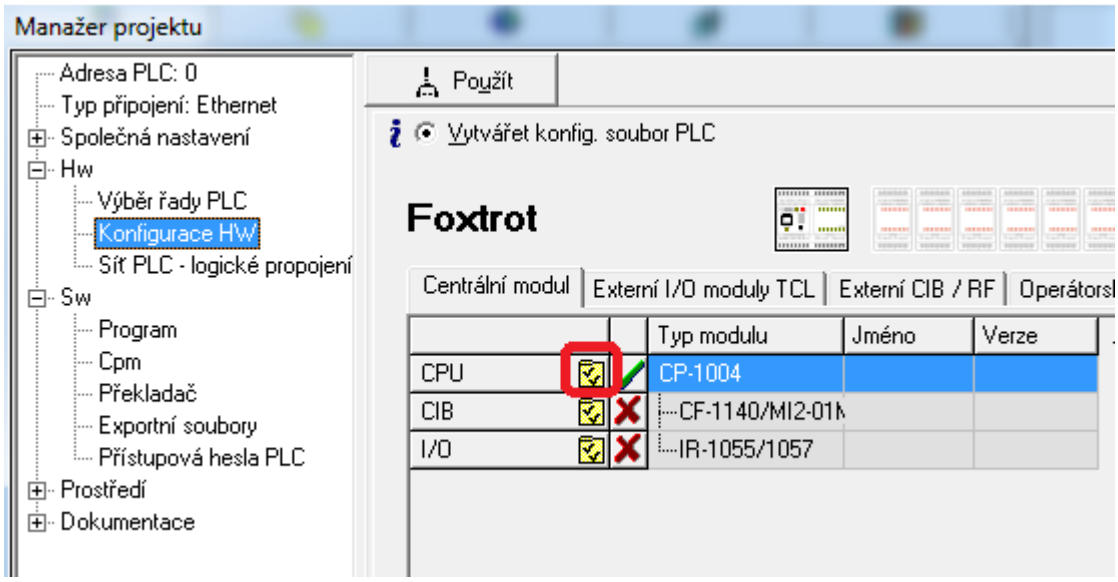
POZNÁMKA

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 *employees.jsn* a nakopírujeme do něho výše uvedený JSON text. Adresář ROOT v simulátoru nahrazuje SD kartu PLC.

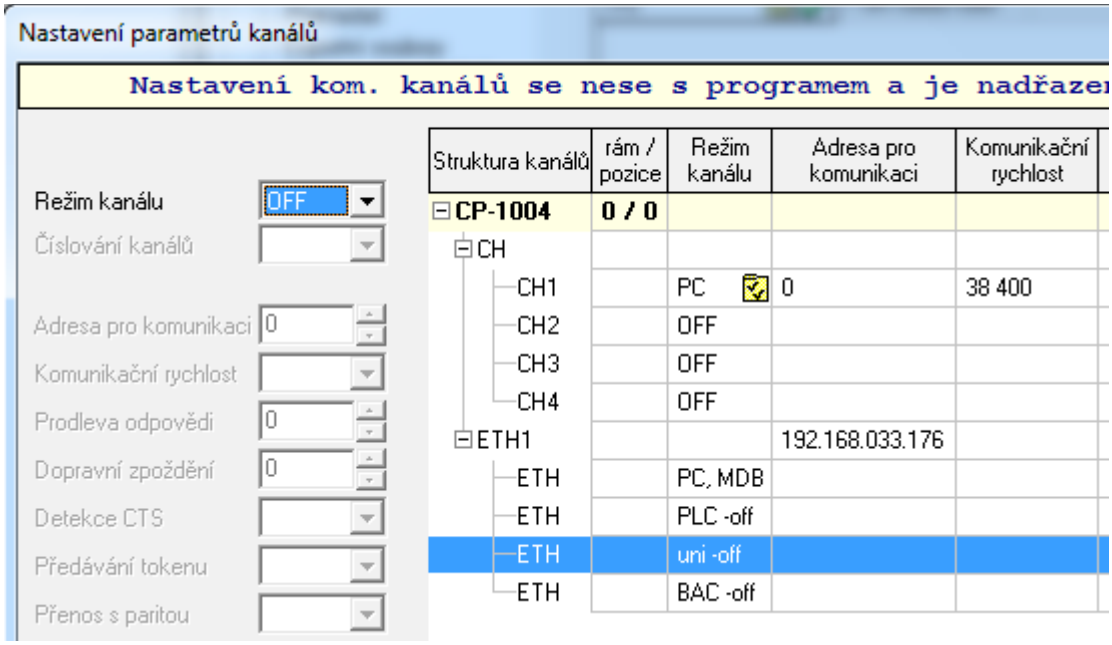
7.3 Použití fbJsonPageParser

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

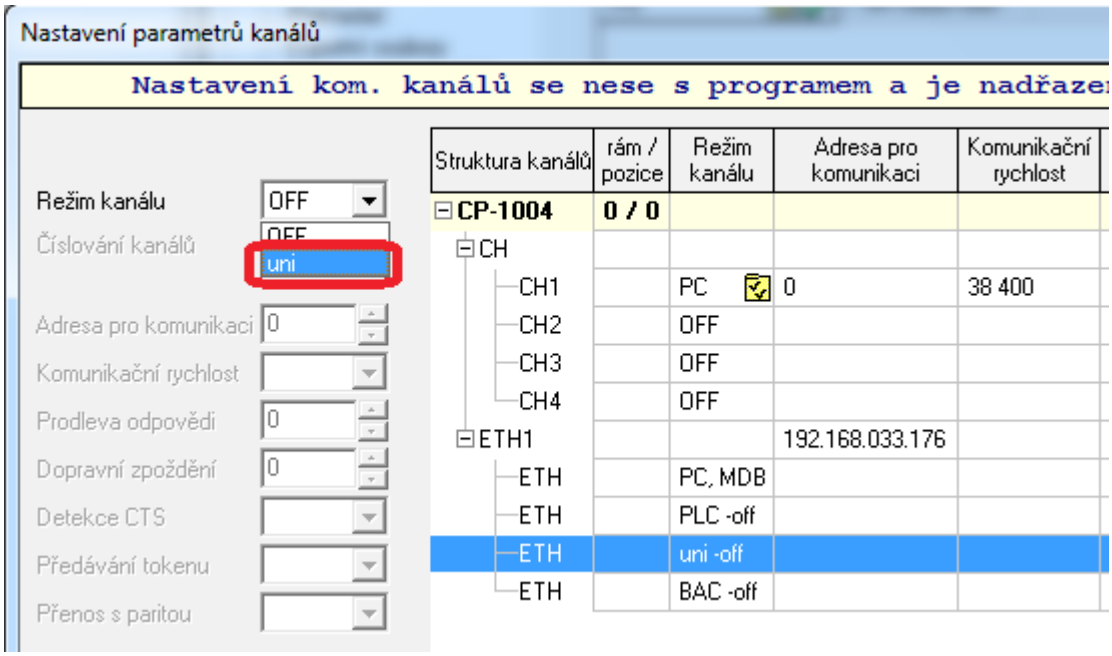
Nejprve je tedy třeba nastavit komunikační kanál pro spojení s web serverem. Prvním potřebným krokem je zapnout podporu režimu uni na rozhraní ethernet. Toto se v prostředí Mosaic provede pomocí Manažeru projektu. Po spuštění Manažera projektu (např. CTRL+ALT+F11) vybereme myší uzel HW konfigurace. Dále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



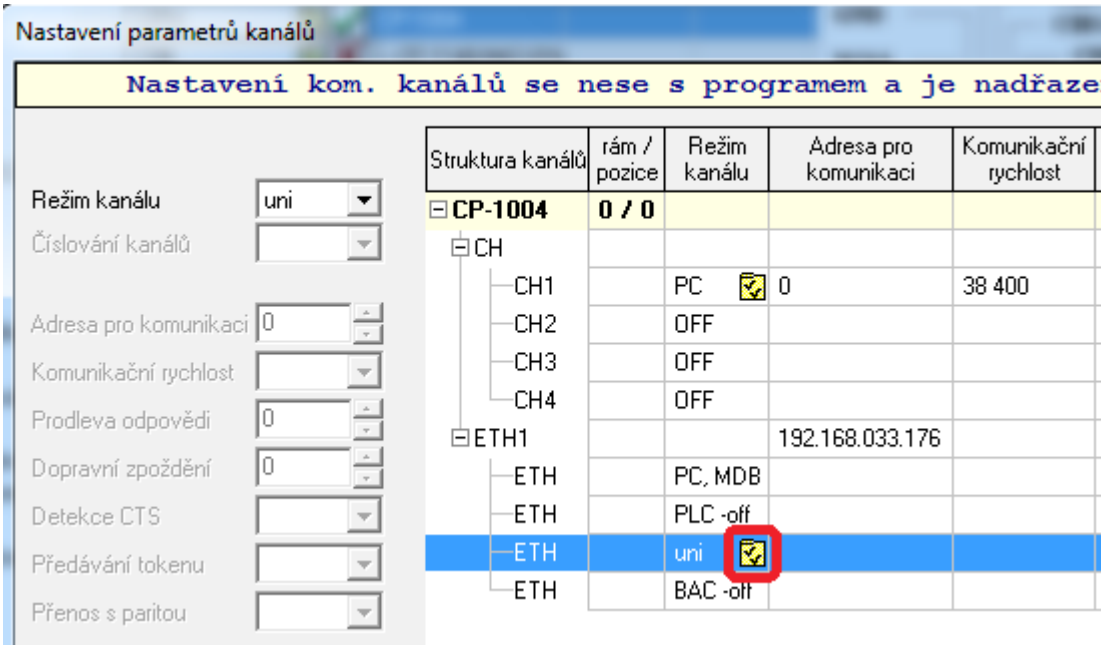
Poté klikneme na řádek s nastavením režimu uni pro rozhraní Ethernet (viz řádek ETH – uni-off) a ten se zbarví modře. V novém projektu je uni režim pro rozhraní ethernet vypnutý (viz pole Režim kanálu = OFF).



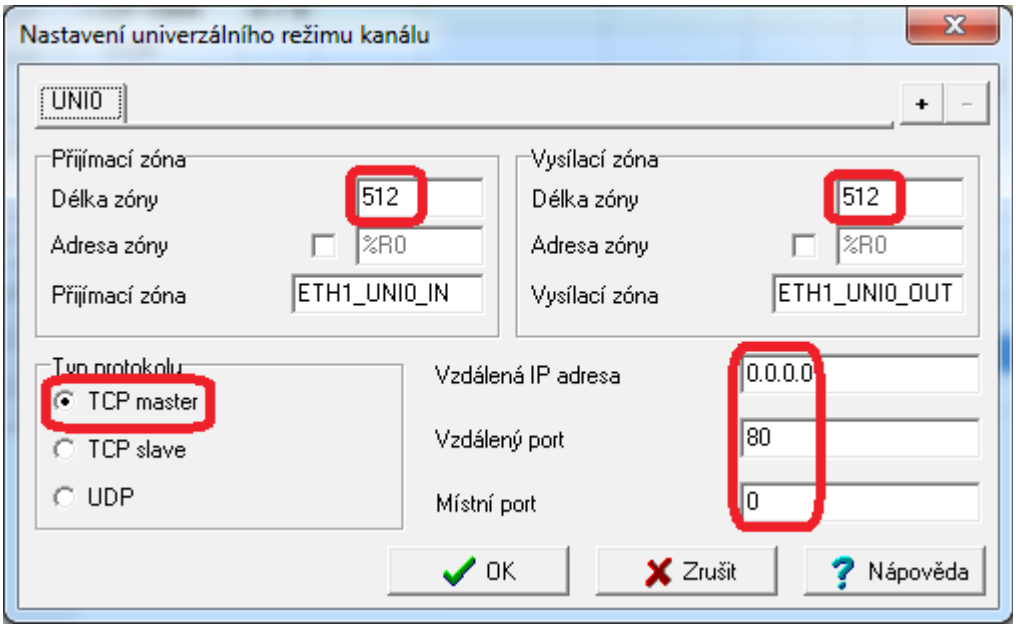
Poté je třeba zvolit režim kanálu *uni*, což se provede pomocí rozbalovaného menu jak ukazuje následující obrázek.



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*.

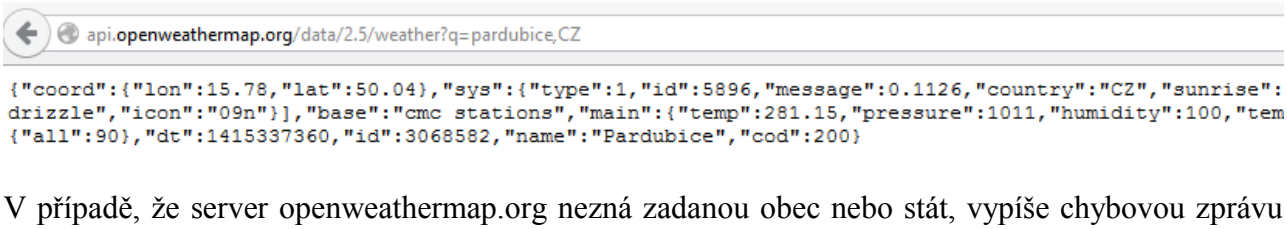


Objeví se dialog s názvem „Nastavení univerzálního režimu kanálu“. V něm nastavíme následující parametry pro první ethernet spojení (*ETH1_uni0*): zvolíme délku přijímací zóny 512 bytů, délku vysílací zóny 512 bytů, typ protokolu TCP master, vzdálená IP adresa 0.0.0.0, vzdálený port 80, místní port 0.

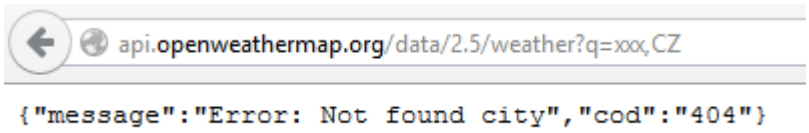


Po stisku tlačítka OK je ethernet rozhraní PLC nastaveno pro komunikaci s web serverem. Tím je nastavení komunikačního kanálu hotové.

Pro příklad použijeme službu serveru openweathermap.org, který nám vrátí aktuální stav počasí pro dané město a stát. Tyto informace je tento server schopen zasílat v XML a JSON formátu. My se zaměříme na JSON formát. Službu lze snadno vyzkoušet a to tak, že do internetové prohlížeče zadáme http://api.openweathermap.org/data/2.5/weather?q=**obec**,**stát,** kde místo *obce* zadáme příslušnou obec. Místo státu zadáme zkratku státu ve kterém se obec vyskytuje. Například výpis počasí pro Pardubice v České republice bude vypadat následovně:



V případě, že server openweathermap.org nezná zadanou obec nebo stát, vypíše chybovou zprávu ve tvaru:



Odpověď serveru může vypadat např. následovně:

```
{
    "coord": {
        "lon": 15.78,
        "lat": 50.04
    },
    "sys": {
        "type": 1,
        "id": 5896,
        "message": 0.1126,
        "country": "CZ",
        "sunrise": 1415339841,
        "sunset": 1415373831
    },
    "weather": [
        {
            "id": 300,
            "main": "Drizzle",
            "description": "light intensity drizzle",
            "icon": "09n"
        }
    ],
    "base": "cmc stations",
    "main": {
        "temp": 281.15,
        "pressure": 1011,
        "humidity": 100,
        "temp_min": 281.15,
        "temp_max": 281.15
    },
    "wind": {
        "speed": 1.5,
        "deg": 269.002
    },
    "clouds": {
        "all": 90
    },
    "dt": 1415337360,
    "id": 3068582,
    "name": "Pardubice",
    "cod": 200
}
```

Tuto odpověď zpracujeme programem v PLC. Získáme tím informaci o aktuálním počasí v příslušném městě.

```
TYPE
    T_Table_weather : STRUCT
      Zemepisna_sirka   : REAL;        //[°]
      zemepisna_delka   : REAL;        //[°]
      vychod_slunce     : DATE_AND_TIME;
      zapad_slunce      : DATE_AND_TIME;
      stav_oblohy       : STRING[255];
      Popis_pocasi      : STRING[255];
      teplota           : REAL;        //[°C]
      Tlak              : UINT;        //[hPa]
      Vlhkost           : UINT;        //[ % ]
```

```
            Maximalni_teplo    : REAL;        //[°C]
            Minimalni_teplo    : REAL;        //[°C]
            Rychlost_vetru     : REAL;        //[m/s]
            Orinetace_vetru    : INT;         //[°]
            Oblacnost          : UINT;        //[ % ]
            cas_datum          : DATE_AND_TIME;
            Jmeno_mesta        : STRING;
            ID_Mesta           : UINT;
          END_STRUCT;
        END_TYPE
        VAR_GLOBAL CONSTANT
        OPENWEATHERMAP_MASK_PARAMETR : ARRAY[0 ..17] OF STRING[10]
                  :=['dt','id','name','lon','lat','sunrise','sunset',
                     'temp','pressure','humidity','temp_min','temp_max',
                     'speed','deg','all','main','description']  ;
                    //maska parametru jednotlivých hodnot
        firstTime :  DATE_AND_TIME;   //pomocná proměnná na převod času
        END_VAR
        VAR_GLOBAL
          weather  :   T_Table_weather;
        END_VAR

      PROGRAM WeatherJSONPage
        VAR
         rqDoc              : BOOL := 1;// žádost o načtení a vypársovaní JSON
         PageName           : STRING := 'api.openweathermap.org/
      data/2.5/weather?q=Pardubice,CZ';   // jméno souboru s JSON dokumentem
         JsonDocParserW     : fbJsonPageParser;     // FB pro pársování souboru
         JSON               : TJsonInfo;// struktura pro výsledky pársování
         lastErr            : STRING;   // popis případné chyby
         parametr           : UINT;     //  urcuje cislo parametru
         i                  : UINT;     // pro pohyb poli masky
         helpTime           : UDINT;    // pomocna promena na převod Unix casu
        END_VAR

        REPEAT
            JsonDocParserW(exec := rqDoc, chanCode := ETH1_uni0 , pageName :=
                           PageName , JsonInfo := JSON );
          IF JsonDocParserW.start THEN
               // blok JsonDocParser zahajil parsovani -> zahajime ho taky
               parametr := 99;           //nastaví nexistující parametr
          END_IF;
          IF JsonDocParserW.done THEN     // vyparsovan jeden radek
              parametr  := 99;              //nenastaven, zadny parametr
              IF JSON.stack.level = 1  THEN
                FOR  i:=0  TO 2  DO
      //hledá stejnou parametr v poli masek, v případě nálezu vraci index na
      které položka leží, jinak parametr je 99
              IF  JSON.stack.item[1].name = OPENWEATHERMAP_MASK_PARAMETR[i] THEN
                    parametr:=i;
                    EXIT;
              END_IF;
                END_FOR;
                CASE parametr OF
          // závisloti na parametru předá z JSON.item.value hodnotu do struktury s
      příslušným převodem
                    0  : helpTime  := DATE_AND_TIME_TO_UDINT(firstTime)+
      STRING_TO_UDINT(JSON.item.value);
                         weather.cas_datum := UDINT_TO_DATE_AND_TIME(helpTime);
                    1  : weather.ID_Mesta    :=  STRING_TO_UINT(JSON.item.value);
                    2  : weather.Jmeno_mesta :=  JSON.item.value;
                END_CASE;
              END_IF;
              IF JSON.stack.level = 2 THEN
                  FOR  i := 3  TO 14  DO
```

```
                    //hledá stejnou parametr v poli masek, v případě nálezu
vraci index na které položka leží, jinak parametr je 99
             IF  JSON.stack.item[2].name = OPENWEATHERMAP_MASK_PARAMETR[i] THEN
                    parametr := i;
                    EXIT;
             END_IF;
                END_FOR;
                CASE parametr OF
                        //hledá stejnou parametr v poli masek, v případě nálezu
vraci index na které položka leží, jinak parametr je 99
                    3   : weather.Zemepisna_sirka  :=
                                STRING_TO_REAL(JSON.item.value);
                    4   : weather.zemepisna_delka  :=
                                STRING_TO_REAL(JSON.item.value);
                    5   : helpTime  := DATE_AND_TIME_TO_UDINT(firstTime)
                                     + STRING_TO_UDINT(JSON.item.value);
                  weather.vychod_slunce := UDINT_TO_DATE_AND_TIME(helpTime);
                    6   : helpTime  := DATE_AND_TIME_TO_UDINT(firstTime)
                                      +STRING_TO_UDINT(JSON.item.value);
                   weather.zapad_slunce := UDINT_TO_DATE_AND_TIME(helpTime);
                    7   : weather.teplota:=
                               STRING_TO_REAL( JSON.item.value)-270.15;
                    8   : weather.Tlak := STRING_TO_UINT ( JSON.item.value);
                    9   : weather.Vlhkost := STRING_TO_UINT( JSON.item.value);
                    10  : weather.Minimalni_teplo  :=
                               STRING_TO_REAL( JSON.item.value)-270.15;
                    11  : weather.Maximalni_teplo  :=
                               STRING_TO_REAL( JSON.item.value)-270.15;
                    12  : weather.Rychlost_vetru   :=
                                     STRING_TO_REAL( JSON.item.value);
                    13  : weather.Orinetace_vetru  :=
                                     STRING_TO_INT ( JSON.item.value);
                    14  : weather.Oblacnost:= STRING_TO_UINT( JSON.item.value);
                  END_CASE;
              END_IF;
              IF JSON.stack.level = 3 THEN
                 FOR  i := 15  TO 16  DO
         //hledá stejnou parametr v poli masek, v případě nálezu vraci index na
které položka leží, jinak parametr je 99
          IF  JSON.stack.item[3].name = OPENWEATHERMAP_MASK_PARAMETR[i] THEN
                    parametr:=i;
                    EXIT;
        END_IF;
                END_FOR;
                CASE parametr OF
           //hledá stejnou parametr v poli masek, v případě nálezu vraci index na
které položka leží, jinak parametr je 99
                    15  : weather.stav_oblohy        :=  JSON.item.value;
                    16  : weather.Popis_pocasi       :=  JSON.item.value;
                END_CASE;
              END_IF;
          END_IF;
        UNTIL JsonDocParserW.break END_REPEAT;
        IF JsonDocParserW.exec THEN
          IF JsonDocParserW.eof OR JsonDocParserW.err THEN  // parsovani ukon-
ceno
            rqDoc := 0;
            IF JsonDocParserW.err THEN
              lastErr := JsonDocParserW.errTxt;           // zachytit chybu
            END_IF;
          END_IF;
        END_IF;

      END_PROGRAM
```

Ukázkový příklad používá pro pársování JSON souboru instanci funkčního blok *fbJson-PageParser,* která je nazvaná *JsonDocParserW*. Zpracování je zahájeno na náběžnou hranu proměnné *rqDoc*. Proměnná *PageName* obsahuje název web stránky z JSON souborem. Instance *JsonDocParserW* je volaná v cyklu *REPEAT*. Cyklus je vykonáván až do okamžiku, kdy se nastaví výstup *JsonDocParserW.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 *JsonDocParserW.start* tak program provede inicializaci pomocných proměnných potřebných k rozebírání dokumentu. Když je nastaven výstup *JsonDocParserW.done* tak to znamená, že byl vypársován jeden element JSON dokumentu a informace o tomto elementu jsou uloženy v proměnné JSON (ta obsahuje položky typu *TJsonItem* a *TJsonStack*). Následuje zpracování těchto informací a výsledky se uloží do proměnné *weather*.

POZNÁMKA

Je pravděpodobné, že většina serverů, ze kterých bude PLC systém stahovat JSON 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.