Knihovna EnergyLib

TXV 003 65.01
šesté vydání
květen 2022
změny vyhrazeny

Historie změn

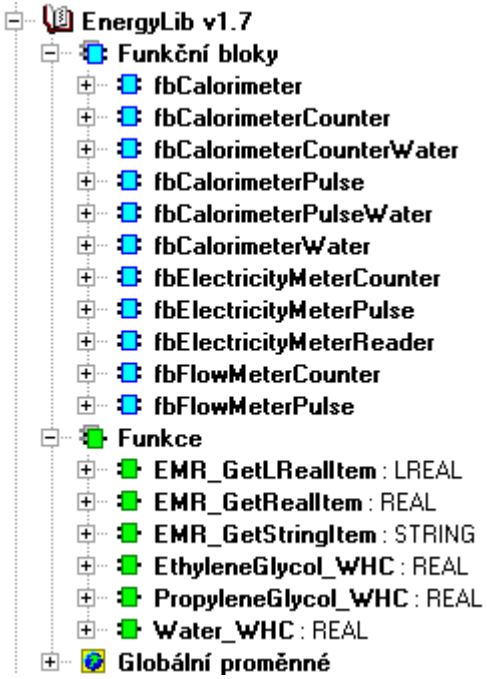
Datum	Vydání	Popis změn
Únor 2011	1	První vydání, popis odpovídá EnergyLib_v10
Červen 2011	2	Doplněn popis bloku fbElectricityMeterPulse(), popis odpovídá EnergyLib_v11
Únor 2012	3	Přidány bloky pro měření průtoku a tepla popis odpovídá EnergyLib_v14
Červen 2012	4	Popis k verzi EnergyLib_v16
Červen 2012	5	Přidány funkce EMR_GetRealItem, EMR_GetLRealItem() a EMR_GetStringItem() a příklad komunikace s elektroměrem ZPA ED 310 Popis odpovídá verzi EnergyLib_v17
Květen 2022	6	Přidán popis bloků fbElectricityMetersReader a fbFlowTotaliser Popis odpovídá verzi EnergyLib_v26

OBSAH

1 ÚVOD

Knihovna EnergyLib je standardně dodávaná jako součást programovacího prostředí Mosaic. Knihovna obsahuje funkční bloky umožňující načítat údaje z elektroměru připojeného přes opto hlavici nebo přes rozhraní RS-485 (komunikace podle normy ČSN EN 62056-21) a počítat spotřebu elektrické energie na základě pulzů S0 poskytovaných elektroměrem. Dále knihovna poskytuje funkce pro výpočet průtoku a dodaného tepla na základě údaje z průtokoměru a měřených teplot.

Následující obrázek ukazuje strukturu knihovny EnergyLib v prostředí Mosaic



Pokud chceme funkce z knihovny EnergyLib použít v aplikačním programu PLC, je třeba nejprve přidat tuto knihovnu do projektu. Současně je nezbytné, aby projekt obsahoval také knihovny StdLib a ComLib, protože některé funkce z těchto knihoven jsou použity v knihovně EnergyLib. Novější verze programu Mosaic (od v2.0.25) přidávají uvedené knihovny automaticky při přidání knihovny EnergyLib (pokud tyto knihovny ještě nejsou v projektu). Knihovna EnergyLib je dodávaná jako součást instalace prostředí Mosaic od verze v2.0.26.

Knihovna EnergyLib není podporovaná na systémech TC-650, u systému TC700 nelze knihovnu použít s procesorovými moduly CP-7002, CP-7003 a CP-7005.

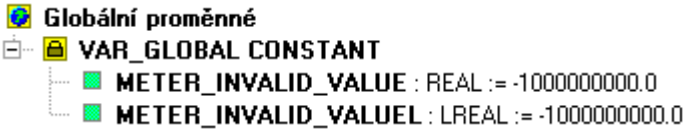
Funkce a funkční bloky knihovny EnergyLib jsou podporovány v centrálních jednotkách řady K (TC700 CP-7000 a CP-7004, všechny varianty systému Foxtrot) od verze v4.4.

2 DATOVÉ TYPY

V knihovně EnergyLib nejsou definovány žádné datové typy.

3 KONSTANTY

V knihovně EnergyLib jsou definovány následující konstanty:



Konstanta *METER_INVALID_VALUE* se používá ve funkčním bloku *fbElectricityMeterReader* jako návratová hodnota pro položky, které nejsou dostupné (např. v případě, že je elektroměr neposkytuje)

Identifikátor	***Typ***	***Hodnota***	***Význam***
METER_INVALID_VALUE	REAL	-1000000000.0	Neplatná hodnota
METER_INVALID_VALUEL	LREAL	-1000000000.0	Neplatná hodnota

4 GLOBÁLNÍ PROMĚNNÉ

V knihovně EnergyLib nejsou definovány žádné globální proměnné.

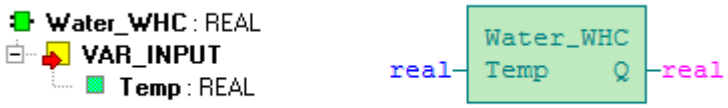
5 FUNKCE

V knihovně EnergyLib je definována následující funkce.

Funkce	Popis
Water_WHC	Výpočet objemové tepelné kapacity vody v závislosti na teplotě
EthyleneGlycol_WHC	Výpočet objemové tepelné kapacita ethlylglykolové teplonostné směsi
PropyleneGlycol_WHC	Výpočet objemové tepelné kapacita propylenglykolové teplo-nostného směsi
EMR_GetRealItem	Získat hodnotu z elektroměru podle OBIS kódu pole
EMR_GetLRealItem	Získat hodnotu z elektroměru podle OBIS kódu pole
EMR_GetStringItem	Získat hodnotu z elektroměru podle OBIS kódu pole

5.1 Funkce Water_WHC

Knihovna : *EnergyLib*



Funkce *Water_WHC* vrací objemovou tepelnou kapacitu vody při normálním atmosfe-rickém tlaku. Hodnoty jsou platné od 0 do 100 °C

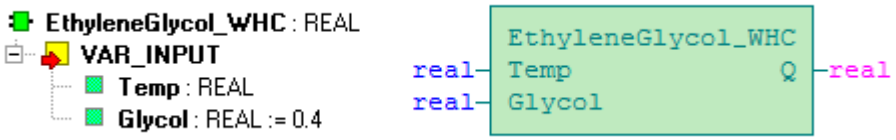
Popis proměnných :

	Proměnná	Typ	Význam
VAR_INPUT			
	temp	REAL	Teplota [°C]
Water_WHC			
	Návratová hodnota	REAL	Objemová tepelná kapacita [$kJm^{-3}h^{-1}$]

Příklad programu s voláním funkce *Water_WHC* je součástí popisu bloku *fbCalorimeter-Counter.*

5.2 Funkce EthyleneGlycol_WHC

Knihovna : *EnergyLib*



Funkce *EthyleneGlycol_WHC* vrací objemovou tepelnou kapacitu ethlylglykolové teplonosné směsi při normálním atmosferickém tlaku.

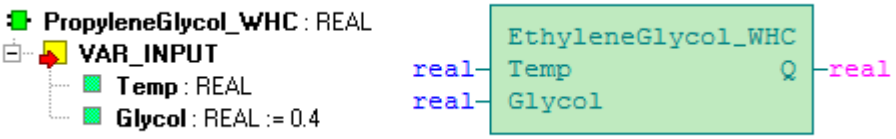
Popis proměnných :

	Proměnná	*Typ*	*Význam*
VAR_INPUT			
	temp	REAL	Teplota [°C]
	Glycol	REAL	Hmotnostní koncentrace glykolu v roztoku (0 - 1)
EthyleneGlycol_WHC			
	Návratová hodnota	REAL	Objemová tepelná kapacita [$kJm^{-3}h^{-1}$]

Příklad programu s voláním funkce *EthyleneGlycol_WHC* je součástí popisu bloku *fbCalorimeter.*

5.3 Funkce PropyleneGlycol_WHC

Knihovna : *EnergyLib*



Funkce *PropyleneGlycol_WHC* vrací objemovou tepelnou kapacitu ethlylglykolové teplonosné směsi při normálním atmosferickém tlaku.

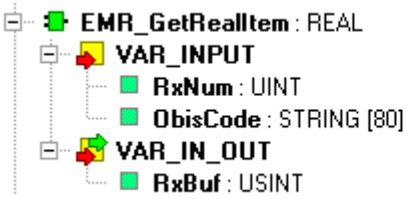
Popis proměnných :

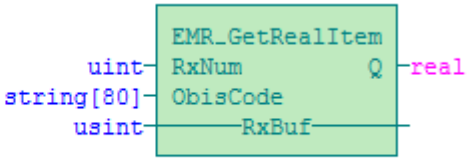
	Proměnná	*Typ*	*Význam*
VAR_INPUT			
	temp	REAL	Teplota [°C]
	Glycol	REAL	Hmotnostní koncentrace glykolu v roztoku (0 - 1)
PropyleneGlycol_WHC			
	Návratová hodnota	REAL	Objemová tepelná kapacita [$kJm^{-3}h^{-1}$]

Příklad programu s voláním funkce *PropyleneGlycol_WHC* je součástí popisu bloku *fbCalorimeterPulse*

5.4 Funkce EMR_GetRealItem

Knihovna : *EnergyLib*





Funkce *EMR_GetRealItem* vybere hodnotu z přijímacího bufferu funkčního bloku *fbElectricityMeterReader* podle zadaneho OBIS kodu a vrátí jí jako číslo typu REAL. Přijímací buffer musí být nejprve naplněn zprávou přijatou z elektroměru, což zajistí uvedený funkční blok. Komunikace s elektroměrem nesmí během volání funkce *EMR_GetRealItem* probíhat (musí být dokončená). Funkce *EMR_GetRealItem* slouží k získání těch hodnot, které nejsou k dispozici jako výstupy funkčního bloku *fbElectricityMeterReader*.

Funkce *EMR_GetRealItem* byla zařazena do knihovny EnergyLib od verze v1.7.

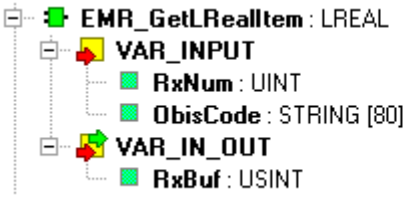
Popis proměnných :

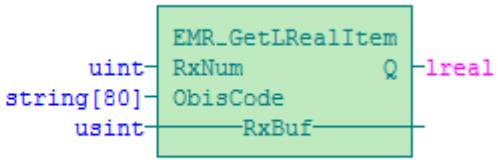
	Proměnná	***Typ***	***Význam***
VAR_INPUT			
	RxNum	UINT	Počet přijatých znaků v přijímacím bufferu funkčního bloku *fbElectricityMeterReader()*
	ObisCode	STRING	Název položky (OBIS kód), jejíž hodnotu chceme vrátit (např. '1.7.0' pro celkový výkon)
VAR_IN_OUT			
	RxBuf	USINT	První znak v přijímacím bufferu
EMR_GetRealItem			
	Návratová hodnota	REAL	Hodnota hledané položky Pokud položka není v přijímacím bufferu nalezena funkce vrátí *METER_INVALID_VALUE*

Příklad programu s voláním funkce *EMR_GetRealItem* je součástí popisu bloku *fbElectricityMeterReader.*

5.5 Funkce EMR_GetLRealItem

Knihovna : *EnergyLib*





Funkce *EMR_GetLRealItem* vybere hodnotu z přijímacího bufferu funkčního bloku *fbElectricityMeterReader* podle zadaneho OBIS kodu a vrátí jí jako číslo typu LREAL. Přijímací buffer musí být nejprve naplněn zprávou přijatou z elektroměru, což zajistí uvedený funkční blok. Komunikace s elektroměrem nesmí během volání funkce *EMR_GetLRealItem* probíhat (musí být dokončená). Funkce *EMR_GetLRealItem* slouží k získání těch hodnot, které nejsou k dispozici jako výstupy funkčního bloku *fbElectricityMeterReader*.

Funkce *EMR_GetLRealItem* byla zařazena do knihovny EnergyLib od verze v1.7.

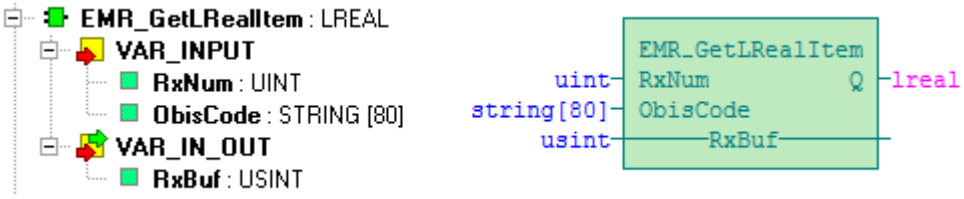
Popis proměnných :

	Proměnná	*Typ*	*Význam*
VAR_INPUT			
	RxNum	UINT	Počet přijatých znaků v přijímacím bufferu funkčního bloku *fbElectricityMeterReader()*
	ObisCode	STRING	Název položky (OBIS kód), jejíž hodnotu chceme vrátit (např. '1.7.0' pro celkový výkon)
VAR_IN_OUT			
	RxBuf	USINT	První znak v přijímacím bufferu
EMR_GetLRealItem			
	Návratová hodnota	LREAL	Hodnota hledané položky Pokud položka není v přijímacím bufferu nalezena funkce vrátí *METER_INVALID_VALUEL*

Příklad programu s voláním funkce *EMR_GetLRealItem* je součástí popisu bloku *fbElectricityMeterReader.*

5.6 Funkce EMR_GetStringItem

Knihovna : *EnergyLib*



Funkce *EMR_GetStringItem* vybere hodnotu z přijímacího bufferu funkčního bloku *fbElectricityMeterReader* podle zadaneho OBIS kodu a vrátí jí jako řetězec znaků typu STRING. Přijímací buffer musí být nejprve naplněn zprávou přijatou z elektroměru, což zajistí uvedený funkční blok. Komunikace s elektroměrem nesmí během volání funkce *EMR_GetStringItem* probíhat (musí být dokončená). Funkce *EMR_GetStringItem* slouží k získání těch hodnot, které nejsou k dispozici jako výstupy funkčního bloku *fbElectricityMeterReader* a nelze je přečíst předchozími funkcemi *EMR_GetRealItem* nebo *EMR_GetLRealItem*. Příkladem modou být hodnoty časových údajů (např. doba čítače dodávky v T1), které jsou elektroměrem vráceny v následujícím tvaru 'C.82.1(00000000:25#h:min)<CR><LF>'.

Funkce *EMR_GetStringItem* byla zařazena do knihovny EnergyLib od verze v1.7.

Popis proměnných :

	Proměnná	Typ	Význam
VAR_INPUT			
	RxNum	UINT	Počet přijatých znaků v přijímacím bufferu funkčního bloku *fbElectricityMeterReader()*
	ObisCode	STRING	Název položky (OBIS kód), jejíž hodnotu chceme vrátit (např. 'C.82.1' pro dobu čítače dodávky v T1)
VAR_IN_OUT			
	RxBuf	USINT	První znak v přijímacím bufferu
EMR_GetLRealItem			
	Návratová hodnota	LREAL	Hodnota hledané položky Pokud položka není v přijímacím bufferu nalezena funkce vrátí prázdný řetězec

Příklad programu s voláním funkce *EMR_ GetStringItem* je součástí popisu bloku *fbElectricityMeterReader.*

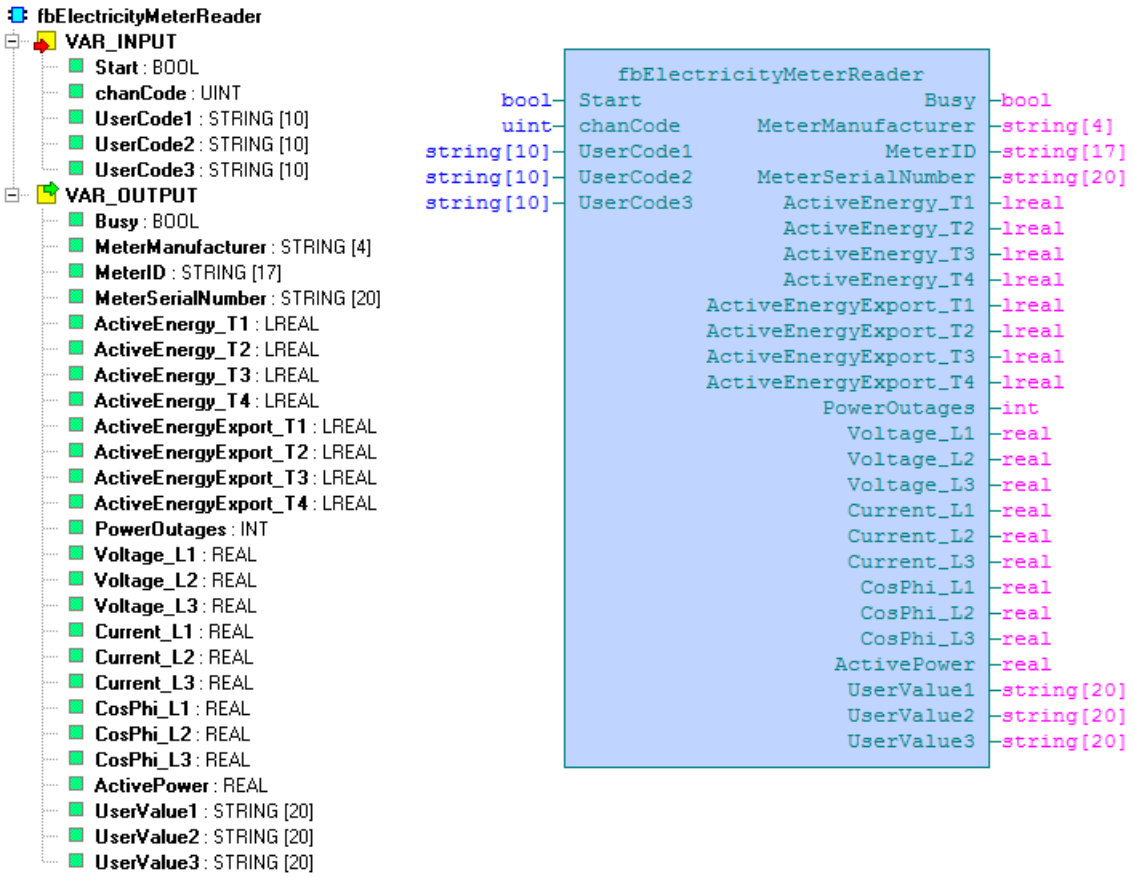
6 FUNKČNÍ BLOKY

V knihovně EnergyLib jsou definovány následující funkční bloky:

Funkční blok	***Popis***
fbElectricityMeterReader	Načte hodnoty z elektroměru přes optickou hlavici nebo přes rozhraní RS-485
fbElectricityMetersReader	Načte hodnoty z elektroměru přes rozhraní RS-485
fbElectricityMeterPulse	Na základě pulzů z elektroměru počítá okamžitou spotřebu, celkovou spotřebu a proud
fbElectricityMeterCounter	Na základě hodnoty čítače pulzů z elektroměru počítá okamžitou spotřebu, celkovou spotřebu a proud
fbFlowMeterPulse	Na základě pulzů z průtokoměru počítá okamžitý průtok a celkový proteklý objem
fbFlowMeterCounter	Na základě hodnoty čítače pulzů z průtokoměru počítá okamžitý průtok a celkový proteklý objem
fbFlowTotaliser	Měření celkového proteklého množství analogovým průtokoměrem
fbCalorimeter	Na základě průtoku a dvou teplot počítá okamžitý výkon a celkové teplo
fbCalorimeterPulse	Na základě pulzů z průtokoměru a dvou teplot počítá okamžitý výkon a celkové teplo
fbCalorimeterCounter	Na základě hodnoty čítače pulzů z průtokoměru a dvou teplot počítá okamžitý výkon a celkové teplo
fbCalorimeterWater	Varianta *fbCalorimeter* pro vodu jako teplonosné médium
fbCalorimeterPulseWater	Varianta *fbCalorimeterPulse* pro vodu jako teplonosné médium
fbCalorimeterCounterWater	Varianta *fbCalorimeterCounter* pro vodu jako teplonosné médium

6.1 Funkční blok fbElectricityMeterReader

Knihovna : *EnergyLib*



Funkční blok *fbElectricityMeterReader* slouží k načtení údajů z elektroměru, který je připojen přes optickou čtecí hlavu (např. hlava ZPA S10 IR) nebo přes rozhraní RS-485 (např. elektroměr ZPA ED 310). Elektroměr musí podporovat komunikaci podle normy ČSN EN 62056-21.

Funkční blok *fbElectricityMeterReader* vrací následující údaje :

- identifikační údaje elektroměru (kód výrobce, kód přístroje, výrobní číslo)
- stavy počítadel odebrané energie
- stavy počítadel dodané energie
- počet výpadků napájení
- napětí, proud a cos φ v jednotlivých fázích
- okamžitou spotřebu

Elektroměr nemusí nutně poskytovat všechny uvedené položky (např. jednofázový elektroměr neposkytuje údaje o fázích L2 a L3, navíc dodavatel elektroměru může nastavit, které údaje bude elektroměr poskytovat). Tyto položky pak mají na výstupu bloku hodnotu *METER_INVALID_VALUE.* Naopak elektroměr může poskytovat některé další údaje oproti uvedenému výčtu. Tyto položky lze získat zadáním OBIS kódu položky do vstupních proměnných *UserCode1* až *UserCode3* nebo zavoláním funkcí *EMR_GetRealItem, EMR_GetLRealItem* nebo *EMR_GetStringItem.*

Čtení údajů z elektroměru je zahájeno na náběžnou hranu vstupní proměnné *Start*, která musí mít hodnotu log. 1 po celou dobu vyčítání informací z elektroměru. Vstupní proměnná *chanCode* musí obsahovat číslo sériového kanálu, přes který probíhá komunikace (viz konstanty *CH1_uni až CH10_uni* z knihovny ComLib). Během načítání údajů je nastavena výstupní proměnná *Busy* na hodnotu *TRUE*. To může trvat několik desítek vteřin v závislosti na počtu údajů, které elektroměr poskytuje. Po načtení všech údajů se nastaví proměnná *Busy* na hodnotu

FALSE a do ostatních výstupních proměnných bloku se zapíší načtené údaje. V tomto okamžiku je možné volat funkce *EMR_GetRealItem, EMR_GetLRealItem* nebo *EMR_GetStringItem* pro získání těch hodnot, které nejsou dostupné jako výstupy bloku *fbElectricityMeterReader*. Vyčtení nových údajů je nutné spustit novou náběžnou hranou vstupu *Start*.

Funkční blok *fbElectricityMeterReader* je podporován na centrálních jednotkách řady K (TC700 CP-7000, CP-7004, CP-7007 a všechny verze Foxtrot) od verze firmware v4.4.

Popis proměnných :

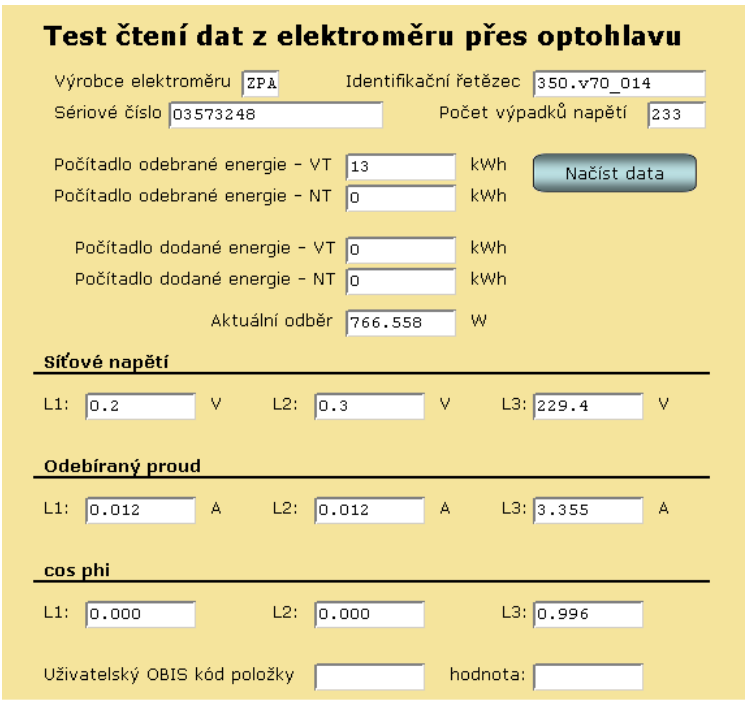
	Proměnná	***Typ***	***Význam***
VAR_INPUT			
	Start	BOOL	Řídicí proměnná. Náběžná hrana spouští vyčtení dat z elektroměru. Log 0 zastaví řešení funkčního bloku
	chanCode	UINT	Kód komunikačního kanálu, kam je připojena optická čtecí hlava *CH1_uni* sériový kanál CH1, režim uni .... *CH10_uni* sériový kanál CH10, režim uni
	UserCode1	STRING[10]	OBIS kód pole, které chce uživatel načíst
	UserCode2	STRING[10]	OBIS kód pole, které chce uživatel načíst
	UserCode3	STRING[10]	OBIS kód pole, které chce uživatel načíst
VAR_OUTPUT			
	Busy	BOOL	Příznak, že se právě načítají data z elektroměru
	MeterManufacturer	STRING[4]	Kód výrobce elektroměru
	MeterID	STRING[17]	Identifikační řetězec elektroměru
	MeterSerialNumber	STRING[20]	Výrobní číslo elektroměru 'C.1.0'
	ActiveEnergy_T1	LREAL	Stav počítadla odebrané energie pro tarif T1 '1.8.1' (zpravidla vysoký tarif)
	ActiveEnergy_T2	LREAL	Stav počítadla odebrané energie pro tarif T2 '1.8.2' (zpravidla nízký tarif)
	ActiveEnergy_T3	LREAL	Stav počítadla odebrané energie pro tarif T3 '1.8.3'
	ActiveEnergy_T4	LREAL	Stav počítadla odebrané energie pro tarif T4 '1.8.4'
	ActiveEnergyExport_T1	LREAL	Stav počítadla dodané energie pro tarif T1 '2.8.1' (zpravidla vysoký tarif)
	ActiveEnergyExport_T2	LREAL	Stav počítadla dodané energie pro tarif T2 '2.8.2' (zpravidla nízký tarif)
	ActiveEnergyExport_T3	LREAL	Stav počítadla dodané energie pro tarif T3 '2.8.3'

	Proměnná	Typ	Význam
	ActiveEnergyExport_T4	LREAL	Stav počítadla dodané energie pro tarif T4 '2.8.4'
	PowerOutages	INT	Počet výpadků napájení 'C.7.0'
	Voltage_L1	REAL	Napětí fáze L1 ve Voltech '12.4.L1'
	Voltage_L2	REAL	Napětí fáze L2 ve Voltech '12.4.L2'
	Voltage_L3	REAL	Napětí fáze L3 ve Voltech '12.4.L3'
	Current_L1	REAL	Proud ve fázi L1 v Ampérech '11.4.L1'
	Current_L2	REAL	Proud ve fázi L2 v Ampérech '11.4.L2'
	Current_L3	REAL	Proud ve fázi L3 v Ampérech '11.4.L3'
	CosPhi_L1	REAL	Kosinus φ fáze L1 '13.4.L1'
	CosPhi_L2	REAL	Kosinus φ fáze L2 '13.4.L2'
	CosPhi_L3	REAL	Kosinus φ fáze L3 '13.4.L3'
	ActivePower	REAL	Okamžitá spotřeba ve Watech '1.25' nebo vypočteno z napětí a proudu
	UserValue1	STRING[20]	Hodnota OBIS položky, která je požadovaná ve vstupní proměnné UserCode1
	UserValue2	STRING[20]	Hodnota OBIS položky, která je požadovaná ve vstupní proměnné UserCode2
	UserValue3	STRING[20]	Hodnota OBIS položky, která je požadovaná ve vstupní proměnné UserCode3
	RxNum	UINT	Počet znaků přijatých od elektroměru
	RxBuf	ARRAY[0..2048] OF USINT	Pole znaků přijatých od elektroměru

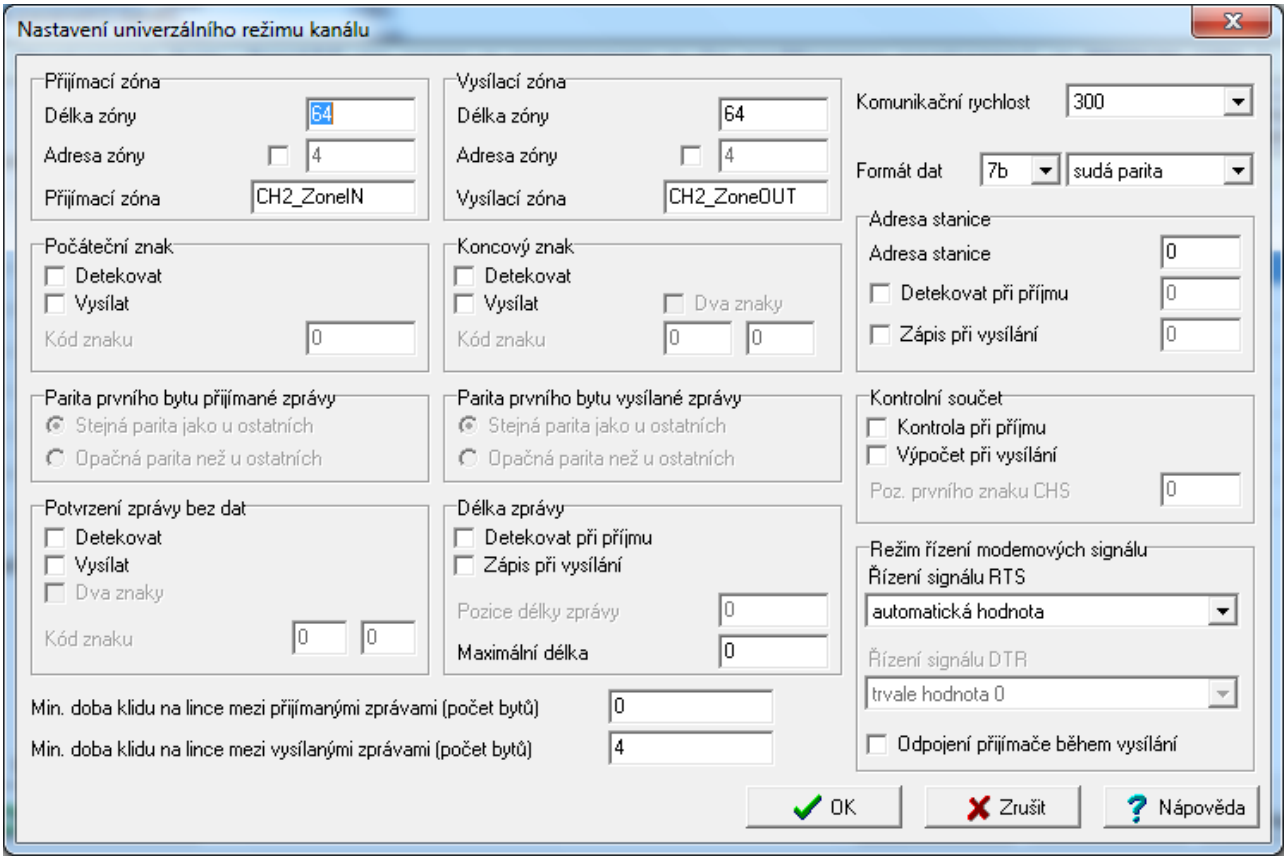
V následujícím příkladu je funkční blok *fbElectricityMeterReader* použit pro načtení údajů z elektroměru. Načtené údaje budou zveřejněny na web stránce. Proměnné *Valid_xxx* jsou použity pro řízení viditelnosti zobrazených údajů, takže viditelné budou pouze údaje, které elektroměr skutečně poskytuje.

```
PROGRAM prgMain
  VAR
     start        : bool;
     EMeter       : fbElectricityMeterReader;   // cteni elmeru pres optohlavici
     valid_L1     : bool;
     valid_L2     : bool;
     valid_L3     : bool;
     valid_Export : bool;
     valid_Power  : bool;
  END_VAR

  // cyklicke cteni udaju z elektromeru
  EMeter(Start := start, chanCode := CH1_uni);
  IF NOT EMeter.Busy THEN
    start := NOT start;
  END_IF;

  // pokud nektera polozka nebyla z elektromeru nactena,
  // je nastavena na hodnotu MeterInvalidValue
  valid_L1     := (EMETER.Voltage_L1             <> METER_INVALID_VALUE);
  valid_L2     := (EMETER.Voltage_L2             <> METER_INVALID_VALUE);
  valid_L3     := (EMETER.Voltage_L3             <> METER_INVALID_VALUE);
  valid_Export := (EMETER.ActiveEnergyExport_T1 <> METER_INVALID_VALUEL);
  valid_Power  := (EMETER.ActivePower            <> METER_INVALID_VALUE);

END_PROGRAM
```

Zobrazení vyčítaných dat ve web stránce může vypadat např. následovně.



Nastavení přenosových parametrů sériového kanálu CH1, na který je připojena optická čtecí hlava, je vidět z následujícího dialogu. Sériový kanál musí být nastaven v režimu uni, komunikační rychlost je 300 Baud, formát dat 7 bitů se sudou paritou, délka přijímací zóny 64 bytů, délka vysílací zóny 64 bytů, minimální délka klidu na lince mezi přijímanými zprávami je 0, minimální délka klidu na lince mezi vysílanými zprávami je 4, řízení RTS signálu je nastaveno na automatickou hodnotu.



Dalším příkladem budiž komunikace s elektroměrem ZPA ED 310 přes rozhraní RS-485. Nastavení přenosových parametrů sériového kanálu je shodné s předcházejícím příkladem. Elektroměr ZPA ED 310 může poskytovat řadu zajímavých údajů, které nejsou blokem *fbElectricityMeterReader* zpracovávány. Lze je však získat pomocí funkcí *EMR_GetRealItem* a *EMR_GetLRealItem*. Komunikace s elektroměrem probíhá cyklicky. Po zapnutí napájení PLC program nejprve čeká cca 1 minutu a potom zahájí komunikaci s elektroměrem. Po proběhnutí komunikace se načtené údaje zpracují a spustí se další komunikace. Údaje získané z elektroměru jsou ukládány do proměnné *eData*, odkud jsou pak zobrazovány ve web stránce a mohou být samozřejmě použity i dalším účelům.

```
TYPE
  TEmeterData : STRUCT
    comERROR              : BOOL;           // chyba komunikace s elektromerem
    comCntOk              : UDINT;          // pocet uspesnych komunikaci
    comCntErr             : UDINT;          // pocet neuspesnych komunikaci
    meterManufacturer     : STRING[4];      // kod výrobce elektroměru
    meterID               : STRING[17];     // identifikační řetězec elektroměru
    meterSerialNumber     : STRING[20];     // C.1.0 sériové číslo elektroměru
    dataValid             : BOOL;           // data jsou platná
    activeEnergy_T1       : LREAL;          // 1.8.1
    activeEnergy_T2       : LREAL;          // 1.8.2
    sumaEnergy            : LREAL;          // 1.8.0
    energyExport_T1       : LREAL;          // 2.8.1
    energyExport_T2       : LREAL;          // 2.8.2
    sumaEnergyExport      : LREAL;          // 2.8.0
    PowerOutage_L1        : REAL;           // C.7.1
    PowerOutage_L2        : REAL;           // C.7.2
    PowerOutage_L3        : REAL;           // C.7.3
    PowerOutages          : REAL;           // C.7.0
    effective_VL1         : REAL;           // 32.7
    effective_VL2         : REAL;           // 52.7
    effective_VL3         : REAL;           // 72.7
    effective_AL1         : REAL;           // 31.7
    effective_AL2         : REAL;           // 51.7
    effective_AL3         : REAL;           // 71.7
  END_STRUCT;
END_TYPE

VAR_GLOBAL
  eData                 : TEmeterData;
END_VAR

PROGRAM prgMain
  VAR
    startKom            : BOOL;
    Emeter              : fbElectricityMeterReader;
    valid_T1, valid_T2  : BOOL;
    timElektro1         : TON;            // timeout komunikace s elektromerem
    timElektro2         : TON;            // odmlka v komunikaci s elektromerem
  END_VAR

  // cteni udaju z elektromeru
  Emeter(Start := startKom, chanCode := CH2_uni);
  IF NOT Emeter.Busy THEN
    IF startKom THEN
      startKom := 0;
      // prevzit data standardne poskytovana blokem fbElectricityMeterReader()
      eData.meterManufacturer     := Emeter.MeterManufacturer;
      eData.meterID               := Emeter.MeterID;
      eData.meterSerialNumber     := Emeter.MeterSerialNumber;
      eData.activeEnergy_T1       := Emeter.ActiveEnergy_T1;
      eData.activeEnergy_T2       := Emeter.ActiveEnergy_T2;
      eData.energyExport_T1       := Emeter.ActiveEnergyExport_T1;
      eData.energyExport_T2       := Emeter.ActiveEnergyExport_T2;

      // nasledujici data fbElectricityMeterReader() nezjistuje - zjistime si je
      eData.sumaEnergy       := EMR_GetLRealItem(RxNum := Emeter.RxNum,
                                                 ObisCode := '1.8.0',
                                                 RxBuf := void( Emeter.RxBuf));
```

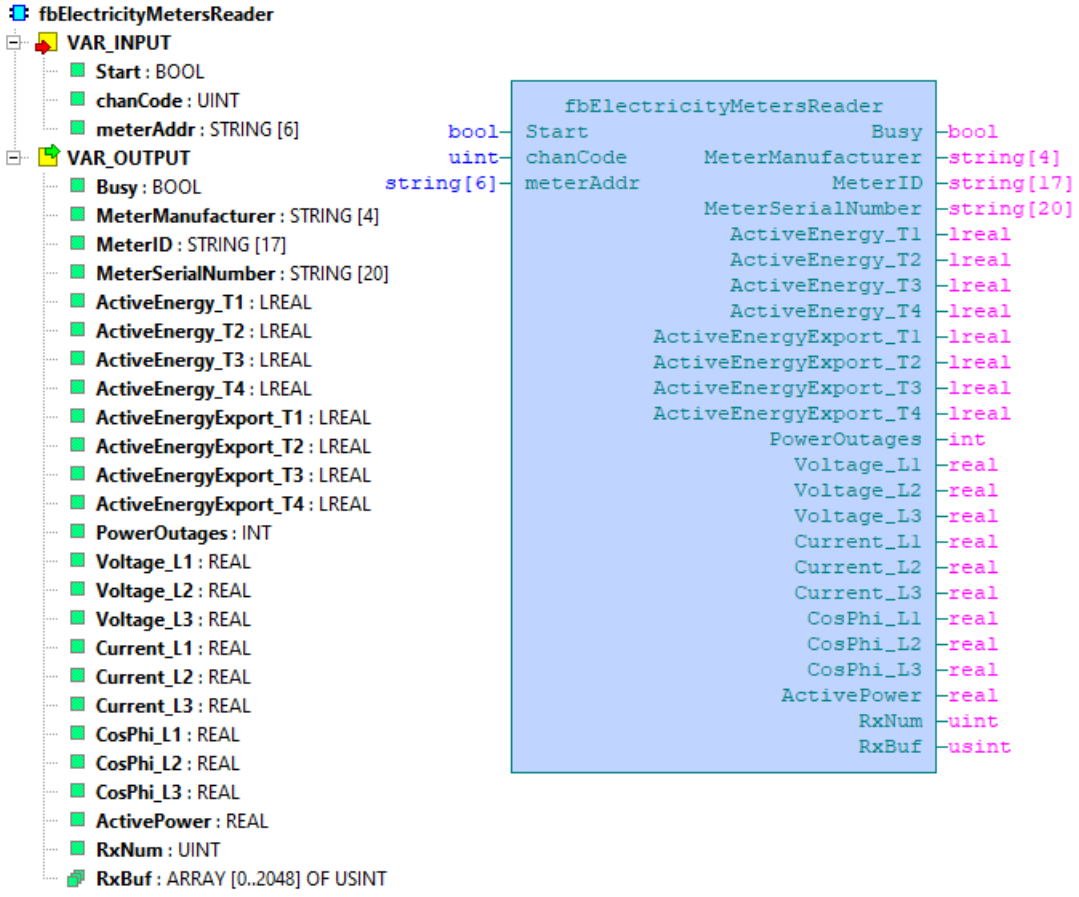
```
      eData.sumaEnergyExport := EMR_GetLRealItem(RxNum := Emeter.RxNum,
                                                 ObisCode := '2.8.0',
                                                 RxBuf := void( Emeter.RxBuf));
      eData.PowerOutage_L1   := EMR_GetRealItem (RxNum := Emeter.RxNum,
                                                 ObisCode := 'C.7.1',
                                                 RxBuf := void( Emeter.RxBuf));
      eData.PowerOutage_L2   := EMR_GetRealItem (RxNum := Emeter.RxNum,
                                                 ObisCode := 'C.7.2',
                                                 RxBuf := void( Emeter.RxBuf));
      eData.PowerOutage_L3   := EMR_GetRealItem (RxNum := Emeter.RxNum,
                                                 ObisCode := 'C.7.3',
                                                 RxBuf := void( Emeter.RxBuf));
      eData.PowerOutages     := EMR_GetRealItem (RxNum := Emeter.RxNum ,
                                                 ObisCode := 'C.7.0',
                                                 RxBuf := void( Emeter.RxBuf));
      eData.effective_VL1    := EMR_GetRealItem (RxNum := Emeter.RxNum ,
                                                 ObisCode := '32.7',
                                                 RxBuf := void( Emeter.RxBuf));
      eData.effective_VL2    := EMR_GetRealItem (RxNum := Emeter.RxNum ,
                                                 ObisCode := '52.7',
                                                 RxBuf := void( Emeter.RxBuf));
      eData.effective_VL3    := EMR_GetRealItem (RxNum := Emeter.RxNum ,
                                                 ObisCode := '72.7',
                                                 RxBuf := void( Emeter.RxBuf));
      eData.effective_AL1    := EMR_GetRealItem (RxNum := Emeter.RxNum ,
                                                 ObisCode := '31.7',
                                                 RxBuf := void( Emeter.RxBuf));
      eData.effective_AL2    := EMR_GetRealItem (RxNum := Emeter.RxNum ,
                                                 ObisCode := '51.7',
                                                 RxBuf := void( Emeter.RxBuf));
      eData.effective_AL3    := EMR_GetRealItem (RxNum := Emeter.RxNum ,
                                                 ObisCode := '71.7',
                                                 RxBuf := void( Emeter.RxBuf));

      valid_T1 := Emeter.ActiveEnergy_T1 <> METER_INVALID_VALUEL;
      valid_T2 := Emeter.ActiveEnergy_T2 <> METER_INVALID_VALUEL;
      IF valid_T1 AND valid_T2 THEN
        eData.comCntOk := eData.comCntOk + 1;
        eData.comERROR := 0; eData.dataValid := 1;
      ELSE
        eData.comCntErr := eData.comCntErr + 1;
        eData.comERROR := 1; eData.dataValid := 0;
      END_IF;
    ELSE
      timElektro2(IN := 1, PT := T#1m, Q => startKom);
    END_IF;
  END_IF;
  // timeout komunikace s elektromerem
  timElektro1(IN := startKom, PT := T#1m);
  IF timElektro1.Q THEN
    startKom := 0; timElektro1(IN := 0); timElektro2(IN := 0);
    eData.comCntErr := eData.comCntErr + 1; eData.comERROR := 1;
  END_IF;

END_PROGRAM
```

Výpis komunikace mezi PLC a elektroměrem je pro názornější představu uveden v následujícím odstavci. Data vysílaná z PLC jsou černě, odpovědi elektroměru jsou modře. Z výpisu je patrné, že PLC nemůže žádným způsobem ovlivnit jaká data budou v odpovědi obsažena. To je dáno pouze výrobcem elektroměru.

```
/?!<CR><LF>
/ZPA4ZE310.v30_012<CR><LF>

<ACK>000<CR><LF>
<STX>
F.F(000000)<CR><LF>
C.1.0(04591994)<CR><LF>
C.90(591994)<CR><LF>
1.8.1(0000016.5#kWh)<CR><LF>
1.8.2(0000002.4*kWh)<CR><LF>
1.8.0(0000018.9*kWh)<CR><LF>
2.8.1(0000004.2#kWh)<CR><LF>
2.8.2(0000001.1*kWh)<CR><LF>
2.8.0(0000005.4*kWh)<CR><LF>
21.8.0(0000011.9*kWh)<CR><LF>
41.8.0(0000003.5*kWh)<CR><LF>
61.8.0(0000003.5*kWh)<CR><LF>
C.9.3(28-05-12 09:44)<CR><LF>
C.9.1()<CR><LF>
C.9.2()<CR><LF>
C.7.1(0013)<CR><LF>
C.7.2(0031)<CR><LF>
C.7.3(0026)<CR><LF>
C.7.0(0031)<CR><LF>
0.3.3(00250.000*i\kWh)<CR><LF>
0.2.1(ED310_DR_30)<CR><LF>
C.8.1(00000359:17#h:min)<CR><LF>
C.8.2(00000000:06*h:min)<CR><LF>
C.8.0(00000359:23*h:min)<CR><LF>
C.82.1(00000000:25#h:min)<CR><LF>
C.82.2(00000000:03*h:min)<CR><LF>
C.82.0(00000000:28*h:min)<CR><LF>
C.50(00000361:35*h:min)<CR><LF>
32.7(227.8*V)<CR><LF>
52.7(000.4*V)<CR><LF>
72.7(000.2*V)<CR><LF>
31.7(000.191*A)<CR><LF>
51.7(000.008*A)<CR><LF>
71.7(000.010*A)<CR><LF>
91.7(000.187*A)<CR><LF>
1.6.1(00.0284*kW)<CR><LF>
1.6.2(00.0000*kW)<CR><LF>
1.6.3(00.0001*kW)<CR><LF>
1.7.0(0000.0284*kW)<CR><LF>
33.7(0.652*cos)<CR><LF>
53.7(0.000*cos)<CR><LF>
73.7(0.000*cos)<CR><LF>
31.6.0(000.195*A)<CR><LF>
51.6.0(000.007*A)<CR><LF>
71.6.0(000.007*A)<CR><LF>
91.6.0(000.203*A)<CR><LF>
21.6.0(000.027*kW)<CR><LF>
41.6.0(000.000*kW)<CR><LF>
61.6.0(000.000*kW)<CR><LF>
81.6.0(000.027*kW)<CR><LF>
0.9.3(00000071:15*h:min)<CR><LF>
0.6.0(1)<CR><LF>
!<CR><LF>
<ETX>
```

6.2 Funkční blok fbElectricityMetersReader

Knihovna : *EnergyLib*



Funkční blok *fbElectricityMetersReader* je funkčně podobný bloku *fbElectricityMeterReader* s následujícími rozdíly:

Blok umožňuje zadat na vstupu *meterAddr* adresu elektroměru, takže je možné komunikovat s více elektroměry na jedné lince.

Blok nemá vstupy UserCode1 … UserCode3. Pokud je potřeba získat pole s uživatelsky zadaným OBIS kódem, je možné použít funkce *EMR_GetLRealItem*, *EMR_GetRealItem* nebo *EMR_GetStringItem*.

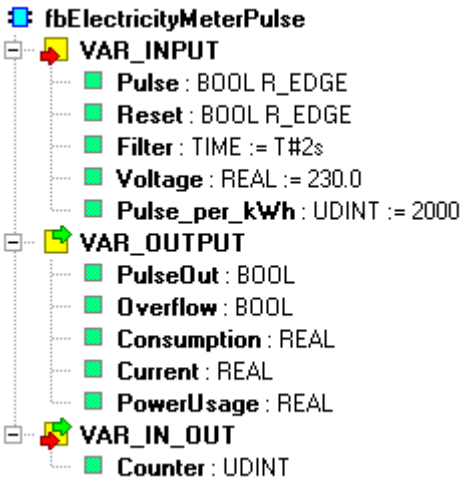
Popis proměnných :

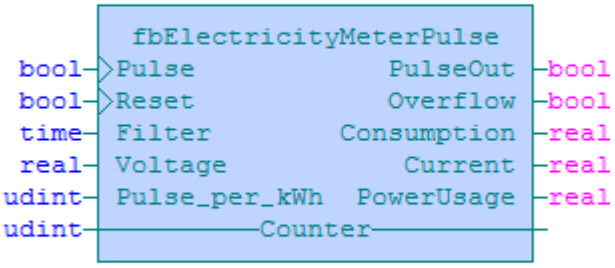
	Proměnná	Typ	Význam
VAR_INPUT			
	Start	BOOL	Řídicí proměnná. Náběžná hrana spouští vyčtení dat z elektroměru. Log 0 zastaví řešení funkčního bloku
	chanCode	UINT	Kód komunikačního kanálu, kam je připojena optická čtecí hlava *CH1_uni* sériový kanál CH1, režim uni …. *CH10_uni* sériový kanál CH10, režim uni
	meterAddr	STRING[6]	adresa elektroměru (posledních 6 znaků výrobního čísla)

	Proměnná	Typ	Význam
VAR_OUTPUT			
	Busy	BOOL	Příznak, že se právě načítají data z elektroměru
	MeterManufacturer	STRING[4]	Kód výrobce elektroměru
	MeterID	STRING[17]	Identifikační řetězec elektroměru
	MeterSerialNumber	STRING[20]	Výrobní číslo elektroměru 'C.1.0'
	ActiveEnergy_T1	LREAL	Stav počítadla odebrané energie pro tarif T1 '1.8.1' (zpravidla vysoký tarif)
	ActiveEnergy_T2	LREAL	Stav počítadla odebrané energie pro tarif T2 '1.8.2' (zpravidla nízký tarif)
	ActiveEnergy_T3	LREAL	Stav počítadla odebrané energie pro tarif T3 '1.8.3'
	ActiveEnergy_T4	LREAL	Stav počítadla odebrané energie pro tarif T4 '1.8.4'
	ActiveEnergyExport_T1	LREAL	Stav počítadla dodané energie pro tarif T1 '2.8.1' (zpravidla vysoký tarif)
	ActiveEnergyExport_T2	LREAL	Stav počítadla dodané energie pro tarif T2 '2.8.2' (zpravidla nízký tarif)
	ActiveEnergyExport_T3	LREAL	Stav počítadla dodané energie pro tarif T3 '2.8.3'
	ActiveEnergyExport_T4	LREAL	Stav počítadla dodané energie pro tarif T4 '2.8.4'
	PowerOutages	INT	Počet výpadků napájení 'C.7.0'
	Voltage_L1	REAL	Napětí fáze L1 ve Voltech '12.4.L1'
	Voltage_L2	REAL	Napětí fáze L2 ve Voltech '12.4.L2'
	Voltage_L3	REAL	Napětí fáze L3 ve Voltech '12.4.L3'
	Current_L1	REAL	Proud ve fázi L1 v Ampérech '11.4.L1'
	Current_L2	REAL	Proud ve fázi L2 v Ampérech '11.4.L2'
	Current_L3	REAL	Proud ve fázi L3 v Ampérech '11.4.L3'
	CosPhi_L1	REAL	Kosinus φ fáze L1 '13.4.L1'
	CosPhi_L2	REAL	Kosinus φ fáze L2 '13.4.L2'
	CosPhi_L3	REAL	Kosinus φ fáze L3 '13.4.L3'
	ActivePower	REAL	Okamžitá spotřeba ve Watech '1.25' nebo vypočteno z napětí a proudu
	RxNum	UINT	Počet znaků přijatých od elektroměru
	RxBuf	ARRAY [0..2048] OF USINT	Pole znaků přijatých od elektroměru

6.3 Funkční blok fbElectricityMeterPulse

Knihovna : *EnergyLib*





Funkční blok *fbElectricityMeterPulse* slouží k výpočtu okamžité spotřeby, celkové spotřeby a proudu na základě pulzů poskytovaných elektroměrem. Pozor, hodnoty okamžitého spotřeby a proudu jsou pouze orientační. Pokud hodnoty aktuální spotřeby a proudu vykazují příliš prudké změny je možné upravit jejich průběh filtrem. Aktuální proud je počítán dle zadaného napětí na vstupu *Voltage* dle vzorce:

$$Current = \frac{Consumption}{Voltage}$$

Celková spotřeba je uchovávána v proměnné na vstupu *Couter* jako počet pulzů. Maximální celková spotřeba je 4294967295 pulzů. Po dosažení této hodnoty je nastaven příznak *Overflow a* čítání se zastaví. Pro většinu aplikací by neměl limit čítače být překážkou.

Čítat celkové spotřeby je možné vynulovat náběžnou hranou na vstupu *Reset*.

Pro zachování hodnoty celkové spotřeby během výpadků napájení je nutné proměnnou na vstupu *Counter* definovat jako VAR_GLOBAL RETAIN.

Popis proměnných :

	Proměnná	***Typ***	***Význam***
VAR_INPUT			
	Pulse	BOOL R_EDGE	Pulzy z elektroměru
	Reset	BOOL R_EDGE	Nulování čítače celkové spotřeby
	Filter	TIME	Časová konstanta filtru okamžité spotřeby (implicitní hodnota 2 sec)
	Voltage	REAL	Napětí (implicitní hodnota 230V)
	Pulse_per_kWh	UDINT	Počet pulzů na jednu kWh (implicitní hodnota 2000 pulzů/kWh)

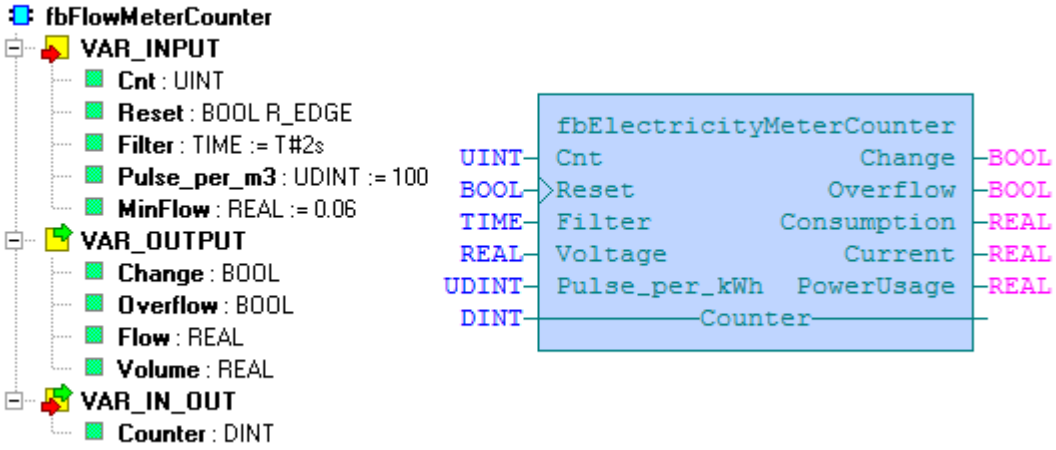
	Proměnná	Typ	Význam
VAR_OUTPUT			
	PulseOut	BOOL	Kopie vstupních pulzů
	Overflow	BOOL	Příznak naplnění čítače – čítání je zastaveno, je nutné vynulovat čítač celkové spotřeby (*Reset*)
	Consumption	REAL	Okamžitá spotřeba [kW]
	Current	REAL	Proud [A]
	PowerUsage	REAL	Celková spotřeba [kWh]
VAR_IN_OUT			
	Counter	UDINT	Počítadlo pulzů celkové spotřeby (musí být RETAIN!)

V následujícím příkladu je funkční blok *fbElectricityMeterPulse* použit pro výpočet aktuální a celkové spotřeby.

```
VAR_GLOBAL RETAIN
  ECounter : UDINT;            // čítač pulzů pro výpočet celkové spotřeby
END_VAR

PROGRAM prgExampleMeterPulse
  VAR
     PMeter       : fbElectricityMeterPulse; // FB pro zpracování pulzů z elměru
     aktSpotreba  : REAL;
     celkSpotreba : REAL;
     proud        : REAL;
  END_VAR

  PMeter( Pulse          := r0_p3_DI.DI0,
          Filter         := T#2s,
          Voltage        := 230.0,
          Pulse_per_kWh  := 2000,
          Counter        := ECounter,
          Consumption    => aktSpotreba,
          PowerUsage     => celkSpotreba,
          Current        => proud);

END_PROGRAM
```

6.4 Funkční blok fbElectricityMeterCounter

Knihovna : *EnergyLib*



Funkční blok *fbElectricityMeterCounter* je variantou bloku *fbElectricityMeterPulse,* pro případ, že vstupem nejsou pulsy, ale hodnota čítače. Výpočet okamžité spotřeby a proudu je u obou bloků shodný.

Blok je primárně navržen pro spolupráci s modulem C-AM-0600I. Z tohoto důvodu je vstup pro hodnotu čítače *Cnt* typu UINT. Blok lze použít i s 32bitovými čítači s použitím konverzní funkce UDINT_TO_UINT.

Celková spotřeba je uchovávána v proměnné na vstupu *Couter* jako počet pulzů. Maximalní celková spotřeba je 2147483647 pulzů. Po dosažení této hodnoty je nastaven příznak *Overflow a* čítání se zastaví. Pro většinu aplikací by neměl limit čítače být překážkou.

Čítat celkové spotřeby je možné vynulovat náběžnou hranou na vstupu *Reset*.

Pro zachování hodnoty celkové spotřeby během výpadků napájení je nutné proměnnou na vstupu *Counter* definovat jako VAR_GLOBAL RETAIN.

Popis proměnných :

	Proměnná	***Typ***	***Význam***
VAR_INPUT			
	Cnt	UINT	Čítač pulzů z elektroměru
	Reset	BOOL R_EDGE	Nulování čítače celkové spotřeby
	Filter	TIME	Časová konstanta filtru okamžité spotřeby (implicitní hodnota 2 sec)
	Voltage	REAL	Napětí (implicitní hodnota 230V)
	Pulse_per_kWh	UDINT	Počet pulzů na jednu kWh (implicitní hodnota 2000 pulzů/kWh)

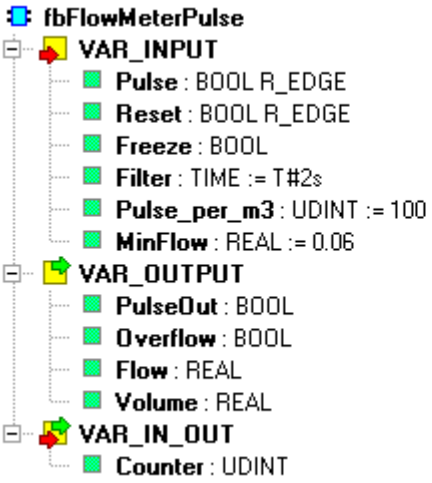
	Proměnná	Typ	Význam
VAR_OUTPUT			
	Change	BOOL	Čítač pulzů změnil hodnotu
	Overflow	BOOL	Příznak naplnění čítače – čítání je zastaveno, je nutné vynulovat čítač celkové spotřeby (*Reset*)
	Consumption	REAL	Okamžitá spotřeba [kW]
	Current	REAL	Proud [A]
	PowerUsage	REAL	Celková spotřeba [kWh]
VAR_IN_OUT			
	Counter	DINT	Počítadlo pulzů celkové spotřeby (musí být RETAIN!)

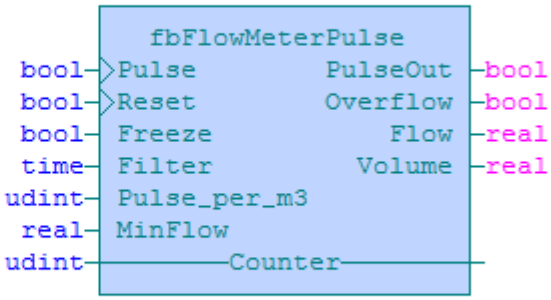
V následujícím příkladu je funkční blok *fbElectricityMeterPulse* použit pro výpočet aktuální a celkové spotřeby.

```
VAR_GLOBAL RETAIN
  ECounter2 : DINT;           // čítač pulzů pro výpočet celkové spotřeby
END_VAR

PROGRAM prgExampleMeterCounter
  VAR
     PMeter  : fbElectricityMeterCounter; //FB pro zpracování pulzů z elměru
     aktSpotreba  : REAL;
     celkSpotreba : REAL;
     proud        : REAL;
  END_VAR

  PMeter( Cnt           := C_AM_0600I_IN.AI1,
          Filter        := T#2s,
          Voltage       := 230.0,
          Pulse_per_kWh := 2000,
          Counter       := ECounter2,
          Consumption   => aktSpotreba,
          PowerUsage    => celkSpotreba,
          Current       => proud);

END_PROGRAM
```

6.5 Funkční blok fbFlowMeterPulse

Knihovna : *EnergyLib*





Funkční blok *fbFlowMeterPulse* slouží k výpočtu okamžitého průtoku a celkového proteklého množství (objemu) na základě pulzů poskytovaných průtokoměrem. Pozor, hodnota okamžitého průtoku je pouze orientační. Pokud hodnota okamžitého průtoku vykazuje příliš prudké změny je možné upravit její průběh filtrem.

Pokud je k dispozici informace o běhu čerpadla v uzavřeném okruhu, lze její negaci využít k vnucení nulového průtoku v čase, kdy čerpadlo neběží a tím zpřesnit měření. K tomuto účelu slouží vstup *Freeze.*

Celkové proteklé množství je uchováváno v proměnné na vstupu *Couter* jako počet pulzů. Maximální proteklé množství je 4294967295 pulzů. Po dosažení této hodnoty je nastaven příznak *Overflow* a čítání se zastaví. Pro většinu aplikací by neměl limit čítače být překážkou.

Čítat celkové spotřeby je možné vynulovat náběžnou hranou na vstupu *Reset*.

Pro zachování hodnoty celkového proteklého množství během výpadků napájení je nutné proměnnou na vstupu *Counter* definovat jako VAR_GLOBAL RETAIN.

Popis proměnných :

	Proměnná	***Typ***	***Význam***
VAR_INPUT			
	Pulse	BOOL R_EDGE	Pulzy z průtokoměru
	Reset	BOOL R_EDGE	Nulování čítače celkového objemu
	Freeze	BOOL	Blokování čítání od vypnutého čerpadla
	Filter	TIME	Časová konstanta filtru okamžité spotřeby (implicitní hodnota 2 sec)
	Pulse_per_m3	UDINT	Počet pulzů na jeden kubický metr (1000 l) (implicitní hodnota 100 pulzů/m^3)
	MinFlow	REAL	Minimální měřitelný průtok [m^3/h]

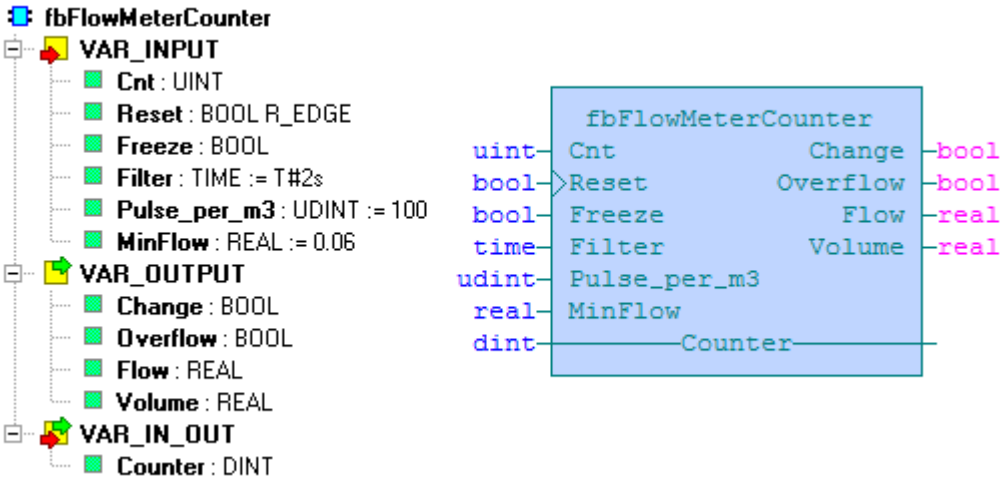
	Proměnná	Typ	Význam
VAR_OUTPUT			
	PulseOut	BOOL	Kopie vstupních pulzů
	Overflow	BOOL	Příznak naplnění čítače – čítání je zastaveno, je nutné vynulovat čítač celkové objemu (*Reset*)
	Flow	REAL	Okamžitý průtok (přibližná hodnota) [m^3/h]
	PowerUsage	REAL	Celkový objem [m^3]
VAR_IN_OUT			
	Counter	UDINT	Počítadlo pulzů celkového průtoku (musí být RETAIN!)

V následujícím příkladu je funkční blok *fbFlowMeterPulse* použit pro výpočet aktuálního průtoku a celkové spotřeby.

```
VAR_GLOBAL RETAIN
  FCounter : UDINT;           // čítač pulzů pro výpočet celkové spotřeby
END_VAR

PROGRAM prgExampleFlowMeterPulse
  VAR
     FMeter       : fbFlowMeterPulse;        // FB pro zpracování pulzů
     aktPrutok    : REAL;
     celkObjem    : REAL;
  END_VAR

  FMeter( Pulse          := r0_p3_DI.DI0,
          Filter         := T#2s,
          Pulse_per_m3   := 100,
          MinFlow        := 0.06,
          Counter        := FCounter,
          Flow           => aktPrutok,
          Volume         => celkObjem);

END_PROGRAM
```

6.6 Funkční blok fbFlowMeterCounter

Knihovna : *EnergyLib*



Funkční blok *fbFlowMeterCounter* je variantou bloku *fbFlowMeterPulse* pro případ, že vstupem nejsou pulsy, ale hodnota čítače. Výpočet okamžitého průtoku je u obou bloků shodný.

Blok je primárně navržen pro spolupráci s modulem C-AM-0600I. Z tohoto důvodu je vstup pro hodnotu čítače *Cnt* typu UINT. Blok lze použít i s 32bitovými čítači s použitím konverzní funkce UDINT_TO_UINT.

Pokud je k dispozici informace o běhu čerpadla v uzavřeném okruhu, lze její negaci využít k vnucení nulového průtoku v čase, kdy čerpadlo neběží a tím zpřesnit měření. K tomuto účelu slouží vstup *Freeze.*

Celkové proteklé množství je uchováváno v proměnné na vstupu *Couter* jako počet pulzů. Maximalní proteklé množství je 2147483647 pulzů. Po dosažení této hodnoty je nastaven příznak *Overflow a* čítání se zastaví. Pro většinu aplikací by neměl limit čítače být překážkou.

Čítat celkové spotřeby je možné vynulovat náběžnou hranou na vstupu *Reset*.

Pro zachování hodnoty celkového proteklého množství během výpadků napájení je nutné proměnnou na vstupu *Counter* definovat jako VAR_GLOBAL RETAIN.

Popis proměnných :

	Proměnná	***Typ***	***Význam***
VAR_INPUT			
	Cnt	UINT	Čítač pulzů z průtokoměru
	Reset	BOOL R_EDGE	Nulování čítače celkového objemu
	Freeze	BOOL	Blokování čítání od vypnutého čerpadla
	Filter	TIME	Časová konstanta filtru okamžité spotřeby (implicitní hodnota 2 sec)
	Pulse_per_m3	UDINT	Počet pulzů na jeden kubický metr (1000 l) (implicitní hodnota 100 pulzů/m^3)
	MinFlow	REAL	Minimální měřitelný průtok [m^3/h]

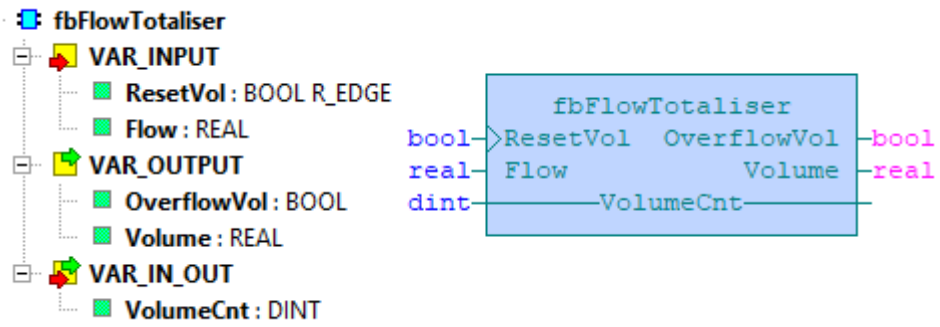
	Proměnná	Typ	Význam
VAR_OUTPUT			
	Change	BOOL	Čítač pulzů změnil hodnotu
	Overflow	BOOL	Příznak naplnění čítače – čítání je zastaveno, je nutné vynulovat čítač celkové objemu (*Reset*)
	Flow	REAL	Okamžitý průtok (přibližná hodnota) [m^3/h]
	PowerUsage	REAL	Celkový objem [m^3]
VAR_IN_OUT			
	Counter	DINT	Počítadlo pulzů celkového průtoku (musí být RETAIN!)

V následujícím příkladu je funkční blok *fbFlowMeterCounter* použit pro výpočet aktuálního průtoku a celkové spotřeby.

```
VAR_GLOBAL RETAIN
  FCounter2 : DINT;            // čítač pulzů pro výpočet celkové spotřeby
END_VAR

PROGRAM prgExampleFlowMeterCounter
  VAR
     FMeter        : fbFlowMeterCounter;        // FB pro zpracování pulzů
     aktPrutok     : REAL;
     celkObjem     : REAL;
  END_VAR

  FMeter( Cnt            := UDINT_TO_UINT(r0_p3_CNT_IN1.VALA),
          Filter         := T#2s,
          Pulse_per_m3   := 100,
          MinFlow        := 3.6,
          Counter        := FCounter2,
          Flow           => aktPrutok,
          Volume         => celkObjem);

END_PROGRAM
```

6.7 Funkční blok fbFlowTotaliser

Knihovna : *EnergyLib*



Funkční blok *fbFlowTotaliser* slouží k měření celkového proteklého množství z hodnoty průtoku měřené analogovým průtokoměrem. Průtok z měřidla se přivede na vstup *Flow*. Předpokládané jednotky jsou m^3/h. Hodnota celkového proteklého objemu je vracena na výstupu *Volume* v m^3.

Celkový proteklý objem je uchováván celočíselně v proměnné na vstupu *VolumeCnt* jako počet proteklých litrů. Tato proměnná by měla být založena jako RETAIN, aby se proteklý objem zachoval i po restartu PLC. Maximální proteklé množství je 2147483647 litrů. Po dosažení této hodnoty dojde k zastavení čítání na této hodnotě a nastavení příznaku *OverflowVol*.

Blok je možné použít i pro jiné jednotky. Například pro kg/h bude výstup vracet počet proteklých kilogramů a hodnota bude uchována jako celé číslo v gramech.

Popis proměnných :

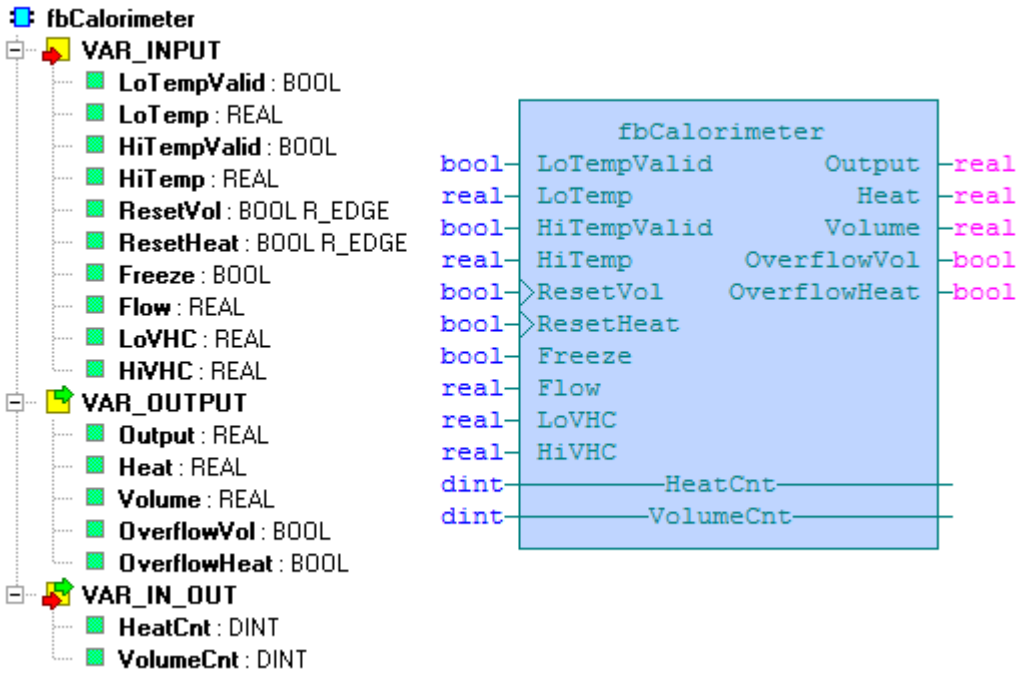
	Proměnná	***Typ***	***Význam***
VAR_INPUT			
	ResetVol	BOOL R_EDGE	Nulování počítadla proteklého množství
	Flow	REAL	Objemový průtok [m^3/h]
VAR_OUTPUT			
	OverflowVol	BOOL	Přetečení čítače pulzů, nastavte reset do logické 1
	Volume	REAL	Celkový objem [m^3]
VAR_IN_OUT			
	VolumeCnt	DINT	Počítadlo pulzů celkového průtoku [l] (musí být RETAIN!)

Příklad použití funkčního bloku pro vstup z jednotky C-AM-0600I. Dělení měřené hodnoty konstantou 0,06, převádí litry za minutu na kubické metry za hodinu.

```
PROGRAM prgExampleFlowTotaliser
  VAR
    FlowTotaliser : fbFlowTotaliser;
  END_VAR
  VAR RETAIN
    VolumeCnt : DINT;
  END_VAR
  VAR_TEMP
  END_VAR

  FlowTotaliser(Flow := C_AM_0600I_IN.AV23.FLOW / 0.06,
                VolumeCnt := VolumeCnt);

END_PROGRAM
```

6.8 Funkční blok fbCalorimeter

Knihovna : *EnergyLib*



Funkční blok *fbCalorimeter* slouží k výpočtu spotřebovaného nebo dodaného tepla, aktuálního výkonu a celkového proteklého objemu teplonosného média.

Na vstupech *HiTemp* a *LoTemp* se očekává teplota vstupujícího a výstupujícího média. Aby byl přírůstek tepla kladný musí platit *HiTemp* > *LoTemp.* Vstupy *LoTempValid* a *HiTempValid* podmiňují platnost měřených teplot pro měření tepla a výkonu. Pokud nejsou oba vstupy *LoTempValid* a *HiTempValid* v logické 1, je počítáno pouze proteklé množství.

Aktuální průtok se předává na vstupu *Flow* v m^3 za hodinu (0,06 m^3/h = 1 l/m).

Pokud je k dispozici informace o běhu čerpadla v uzavřeném okruhu, lze její negaci využít k vnucení nulového průtoku v čase, kdy čerpadlo neběží a tím zpřesnit měření. K tomuto účelu slouží vstup *Freeze.*

Vstupy *HiVHC* a *LoVHC* slouží pro zadání objemové tepelné kapacity teplonosného média v kJ/m^3 K. Tyto hodnoty jsou závislé na teplotě, a proto je nutné předávat hodnotu odpovídající teplotě *HiTemp* (*HiVHC*) a *LoTemp* (*LoVHC*) zvlášť. Konstantní tabulkovou hodnotu lze použít pouze u orientačních měření. Pokud je teplonosným médiem voda lze použít blok *fbCalorimeterWater*, který již výpočet objemové tepelné kapacity ve vztahu k měřeným teplotám obsahuje.

Celkový proteklý objem je uchováván v proměnné na vstupu *VolumeCnt* jako počet proteklých litrů. Maximální proteklé množství je 2147483647 litrů. Po dosažení této hodnoty dojde k zastavení čítání na této hodnotě a nastavení příznaku *OverflowVol*.

Čítač proteklého objemu je možné vynulovat náběžnou hranou na vstupu *ResetVol*.

Teplo je čítáno do proměnné na vstupu *HeatCnt* jako počet dodaných/spotřebovaných watt hodin. Maximální celkové teplo je 2,147483647 GWh. Po dosažení této hodnoty dojde k zastavení čítání na této hodnotě a nastavení příznaku *OverflowVol*.

Čítač tepla je možné vynulovat náběžnou hranou na vstupu *ResetHeat*.

Pro zachování hodnoty celkového proteklého množství a celkového tepla během výpadků napájení je nutné proměnné na vstupu *HeatCnt* a *VolumeCnt* definovat jako VAR_GLOBAL RETAIN.

Popis proměnných :

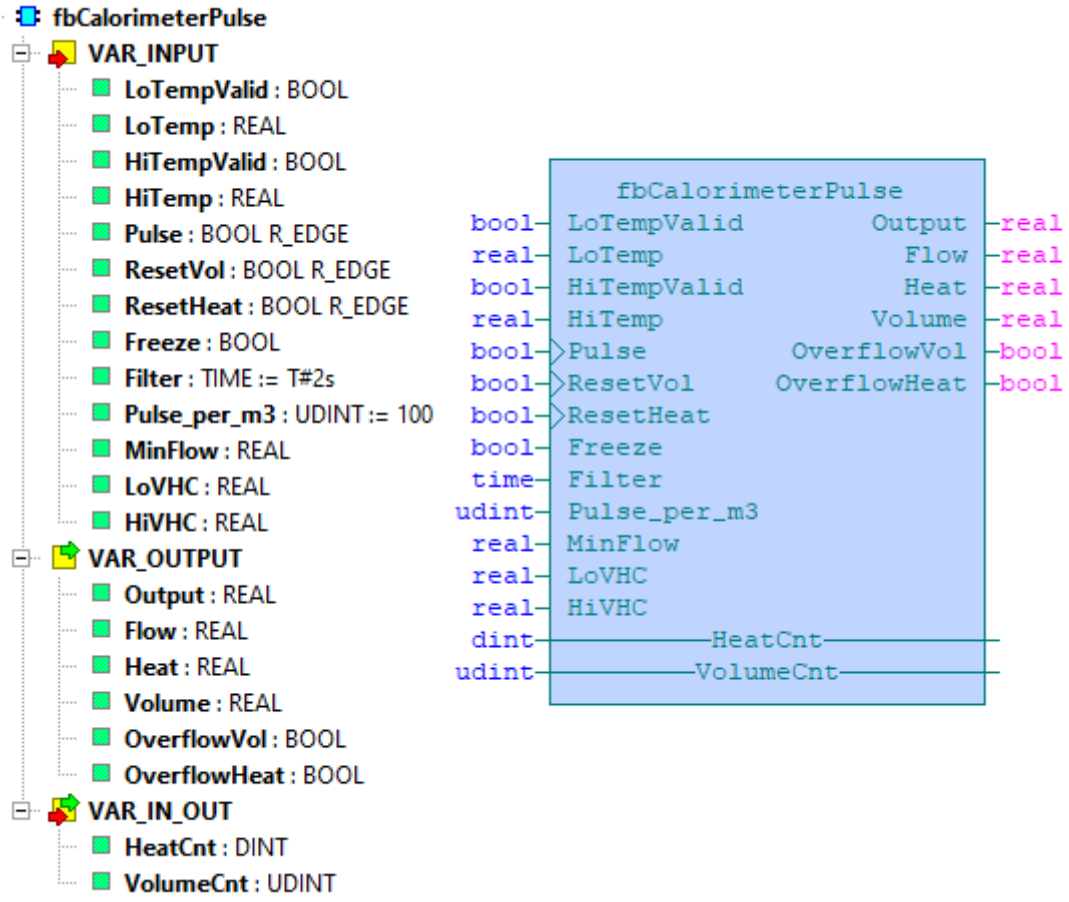
	Proměnná	Typ	Význam
VAR_INPUT			
	LoTempValid	BOOL	Nižší teplota je platná
	LoTemp	REAL	Nižší teplota
	HiTempValid	BOOL	Vyšší teplota je platná
	HiTemp	REAL	Vyšší teplota
	ResetVol	BOOL R_EDGE	Nulování počítadla proteklého objemu
	ResetHeat	BOOL R_EDGE	Nulování počítadla tepla
	Freeze	BOOL	Blokování čítání od vypnutého čerpadla
	Flow	REAL	Objemový průtok [m^3/h]
	LowVHC	REAL	Objemová tepelná kapacita pro nižší teplotu [$kJm^{-3}h^{-1}$]
	HiVHC	REAL	Objemová tepelná kapacita pro vyšší teplotu [$kJm^{-3}h^{-1}$]
VAR_OUTPUT			
	Output	REAL	Tepelný výkon [kW]
	Heat	REAL	Celkové teplo [kWh]
	Volume	REAL	Celkový objem [m^3]
	OverflowVol	BOOL	Přetečení proteklého objemu
	OverflowHeat	BOOL	Přetečení čítače tepla
VAR_IN_OUT			
	HeatCnt	DINT	Počítadlo teplo [Wh] (musí být RETAIN!)
	VolumeCnt	DINT	Počítadlo proteklého množství [l] (musí být RETAIN!)

V následujícím příkladu je funkční blok *fbCalorimeter* použit pro výpočet dodaného tepla a aktuálního výkonu pro ethlylglykolovou směs jako teplonosné médium. Pro určení objemové tepelné kapacity je použita funkce *EthyleneGlycol_WHC*.

```
VAR_GLOBAL RETAIN
  HeatCounter : DINT;
  VolCounter  : DINT;
END_VAR

VAR_GLOBAL CONSTANT
  AnalogValid : TAIStat := ( UNF := false, UNR := false,
                             OVR := false, OVF := false,
                             FLS := false);
END_VAR


PROGRAM prgExampleCalorimeter
  VAR
    Calorimeter : fbCalorimeter;
    aktVykon    : REAL;
    celkTeplo   : REAL;
  END_VAR

  Calorimeter(LoTempValid := r0_p3_AI0.STAT = AnalogValid,
              LoTemp := r0_p3_AI0.ENG,
              HiTempValid := MI_CIB1_IN.ID1_IN.STAT.VLD1 AND
                             NOT MI_CIB1_IN.ID1_IN.STAT.OUF1,
              HiTemp := MI_CIB1_IN.ID1_IN.AI1,
              Flow := MI_CIB1_IN.ID1_IN.AV23.FLOW / 0.06,
              LoVHC := EthyleneGlycol_WHC(Temp := r0_p3_AI0.ENG,
                                          Glycol := 0.5),
              HiVHC := EthyleneGlycol_WHC(Temp := MI_CIB1_IN.ID1_IN.AI1,
                                          Glycol := 0.5),
              HeatCnt := HeatCounter,
              VolumeCnt := VolCounter,
              Output => aktVykon,
              Heat => celkTeplo);

END_PROGRAM
```

6.9 Funkční blok fbCalorimeterPulse

Knihovna : *EnergyLib*



Funkční blok *fbCalorimeterPulse* slouží k výpočtu spotřebovaného nebo dodaného tepla, aktuálního výkonu a celkového proteklého objemu teplonosného média. Jedná se o variantu bloku *fbCalorimeter* s tím rozdílem, že průtok je aproximován z pulzů průtokoměru.

Pokud je k dispozici informace o běhu čerpadla v uzavřeném okruhu, lze její negaci využít k vnucení nulového průtoku v čase, kdy čerpadlo neběží a tím zpřesnit měření. K tomuto účelu slouží vstup *Freeze.*

Na vstupech *HiTemp* a *LoTemp* se očekává teplota vstupujícího a výstupujícího média. Aby byl přírůstek tepla kladný musí platit *HiTemp* > *LoTemp.* Vstupy *LoTempValid* a *HiTempValid* podmiňují platnost měřených teplot pro měření tepla a výkonu. Pokud nejsou oba vstupy *LoTemp-Valid* a *HiTempValid* v logické 1, je počítáno pouze proteklé množství.

Pulzy průtokeměru se předavají přes vstup *Pulse*.

Vstupy *HiVHC* a *LoVHC* slouží pro zadání objemové tepelné kapacity teplonosného média v kJ/m^3 K. Tyto hodnoty jsou závislé na teplotě, a proto je nutné předávat hodnotu odpovídající teplotě *HiTemp* (*HiVHC*) a *LoTemp* (*LoVHC*) zvlášť. Konstantní tabulkovou hodnotu lze použít pouze u orientačních měření. Pokud je teplonosným médiem voda lze použít blok *fbCalorime-terPulseWater*, který již výpočet objemové tepelné kapacity ve vztahu k měřeným teplotám obsahuje.

Hodnoty aktuálního výkonu *Output* a průtoku *Flow* jsou pouze orientační. Stejně je zatížen chybou aproximace průtoku z pulzů i výpočet celkového tepla *Heat*, jedná se tedy také o orientační hodnotu.

Celkový proteklý objem je uchováváno v proměnné na vstupu *VolumeCnt* jako počet pulzů. Maximální proteklé množství je 4294967295 pulzů. Po dosažení této hodnoty dojde k zastavení čítání na této hodnotě a nastavení příznaku *OverflowVol*.

Čítač proteklého objemu je možné vynulovat náběžnou hranou na vstupu *ResetVol*.

Teplo je čítáno do proměnné na vstupu *HeatCnt* jako počet dodaných/spotřebovaných watt hodin. Maximální celkové teplo je 2,147483647 GWh. Po dosažení této hodnoty dojde k zastavení čítání na této hodnotě a nastavení příznaku *OverflowVol*.

Čítač tepla je možné vynulovat náběžnou hranou na vstupu *ResetHeat*.

Pro zachování hodnoty celkového proteklého množství a celkového tepla během výpadků napájení je nutné proměnné na vstupu *HeatCnt* a *VolumeCnt* definovat jako VAR_GLOBAL RETAIN.

Popis proměnných :

	Proměnná	***Typ***	***Význam***
VAR_INPUT			
	LoTempValid	BOOL	Nižší teplota je platná
	LoTemp	REAL	Nižší teplota
	HiTempValid	BOOL	Vyšší teplota je platná
	HiTemp	REAL	Vyšší teplota
	Pulse	BOOL R_EDGE	Pulz z průtokoměru
	ResetVol	BOOL R_EDGE	Nulování počítadla proteklého objemu
	ResetHeat	BOOL R_EDGE	Nulování počítadla tepla
	Freeze	BOOL	Blokování čítání od vypnutého čerpadla
	Filter	TIME	Časová konstanta filtru okamžité spotřeby
	Pulse_per_m3	UDINT	Počet pulzů na jeden kubický metr (1000 l)
	MinFlow	REAL	Minimální měřitelný průtok [m^3/h]
	LowVHC	REAL	Objemová tepelná kapacita pro nižší teplotu [$kJm^{-3}h^{-1}$]
	HiVHC	REAL	Objemová tepelná kapacita pro vyšší teplotu [$kJm^{-3}h^{-1}$]
VAR_OUTPUT			
	Output	REAL	Tepelný výkon [kW]
	Heat	REAL	Celkové teplo [kWh]
	Volume	REAL	Celkový objem [m^3]
	OverflowVol	BOOL	Přetečení proteklého objemu
	OverflowHeat	BOOL	Přetečení čítače tepla

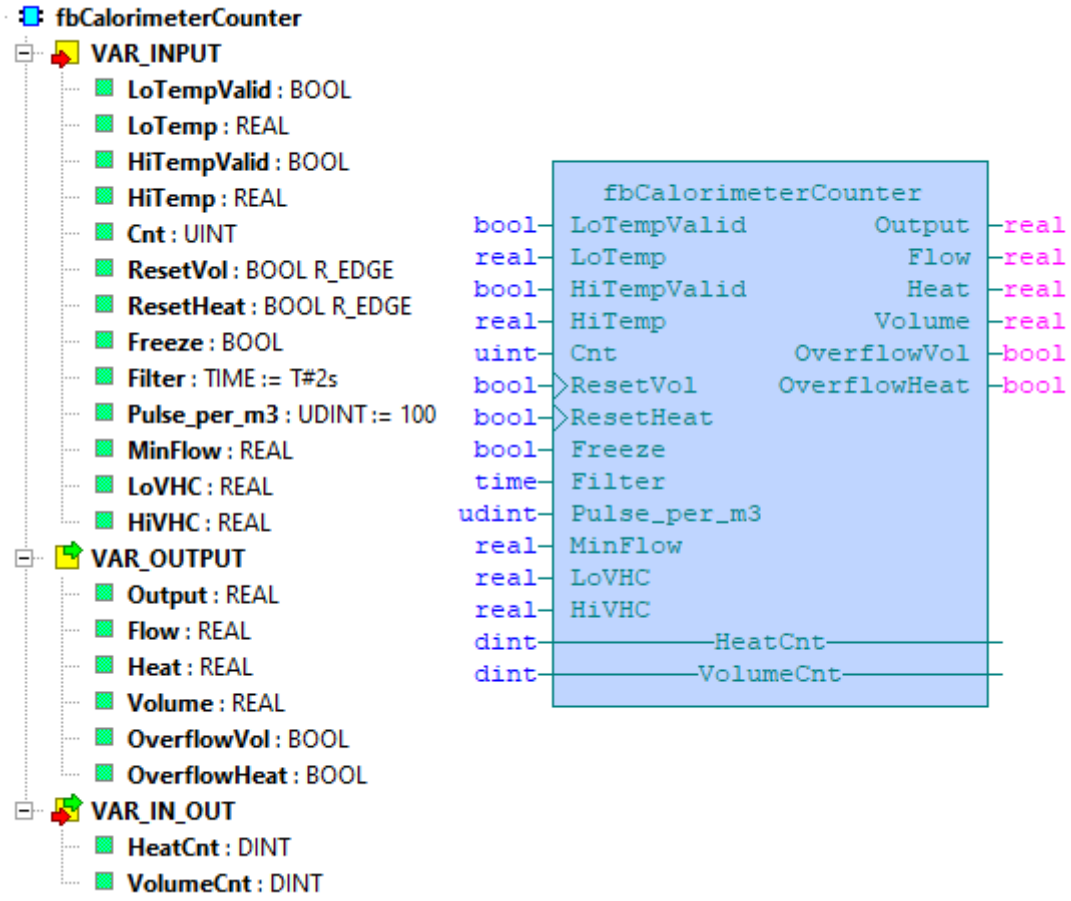
	Proměnná	Typ	Význam
VAR_IN_OUT			
	HeatCnt	DINT	Počítadlo teplo [Wh] (musí být RETAIN!)
	VolumeCnt	DINT	Počítadlo proteklého množství [l] (musí být RETAIN!)

V následujícím příkladu je funkční blok *fbCalorimeterPulse* použit pro výpočet dodaného tepla a aktuálního výkonu pro propylenglykolovou směs jako teplonosné médium. Pro určení objemové tepelné kapacity je použita funkce *PropyleneGlycol_WHC*.

```
VAR_GLOBAL RETAIN
  HeatCounter : DINT;
  VolCounter2 : UDINT;
END_VAR

VAR_GLOBAL CONSTANT
  AnalogValid : TAIStat := ( UNF := false, UNR := false,
                             OVR := false, OVF := false,
                             FLS := false);
END_VAR

PROGRAM prgExampleCalorimeterPulse
  VAR
    Calorimeter : fbCalorimeterPulse;
    aktVykon    : REAL;
    celkTeplo   : REAL;
  END_VAR

  Calorimeter(LoTempValid := r0_p3_AI0.STAT = AnalogValid,
              LoTemp := r0_p3_AI0.ENG,
              HiTempValid := MI_CIB1_IN.ID1_IN.STAT.VLD1 AND
                             NOT MI_CIB1_IN.ID1_IN.STAT.OUF1,
              HiTemp := MI_CIB1_IN.ID1_IN.AI1,
              Pulse  := r0_p3_DI.DI0,
              LoVHC := PropyleneGlycol_WHC(Temp := r0_p3_AI0.ENG,
                                           Glycol := 0.5),
              HiVHC := PropyleneGlycol_WHC(Temp := MI_CIB1_IN.ID1_IN.AI1,
                                           Glycol := 0.5),
              Pulse_per_m3 := 1000,
              HeatCnt := HeatCounter,
              VolumeCnt := VolCounter2,
              Output => aktVykon,
              Heat => celkTeplo);

END_PROGRAM
```

6.10 Funkční blok fbCalorimeterCounter

Knihovna : *EnergyLib*



Funkční blok *fbCalorimeterCounter* je variantou bloku *fbCalorimeterPulse* pro případ, že vstupem nejsou pulsy, ale hodnota čítače. Funkce obou bloků je po všech ostatních stránkách shodná.

Blok je primárně navržen pro spolupráci s modulem C-AM-0600I. Z tohoto důvodu je vstup pro hodnotu čítače *Cnt* typu UINT. Blok lze použít i s 32bitovými čítači s použitím konverzní funkce UDINT_TO_UINT.

Pokud je k dispozici informace o běhu čerpadla v uzavřeném okruhu, lze její negaci využít k vnucení nulového průtoku v čase, kdy čerpadlo neběží a tím zpřesnit měření. K tomuto účelu slouží vstup *Freeze.*

Celkový proteklý objem je uchováváno v proměnné na vstupu *VolumeCnt* jako počet pulzů. Maximalní proteklé množství je 2147483647 pulzů. Po dosažení této hodnoty dojde k zastavení čítání na této hodnotě a nastavení příznaku *OverflowVol*.

Čítač proteklého objemu je možné vynulovat náběžnou hranou na vstupu *ResetVol*.

Teplo je čítano do proměnné na vstupu *HeatCnt* jako počet dodaných/spotřebovaných watt hodin. Maximální celkové teplo je 2,147483647 GWh. Po dosažení této hodnoty dojde k zastavení čítání na této hodnotě a nastavení příznaku *OverflowVol*.

Čítač tepla je možné vynulovat náběžnou hranou na vstupu *ResetHeat*.

Pro zachování hodnoty celkového proteklého množství a celkového tepla během výpadků napájení je nutné proměnné na vstupu *HeatCnt* a *VolumeCnt* definovat jako VAR_GLOBAL RETAIN.

Popis proměnných :

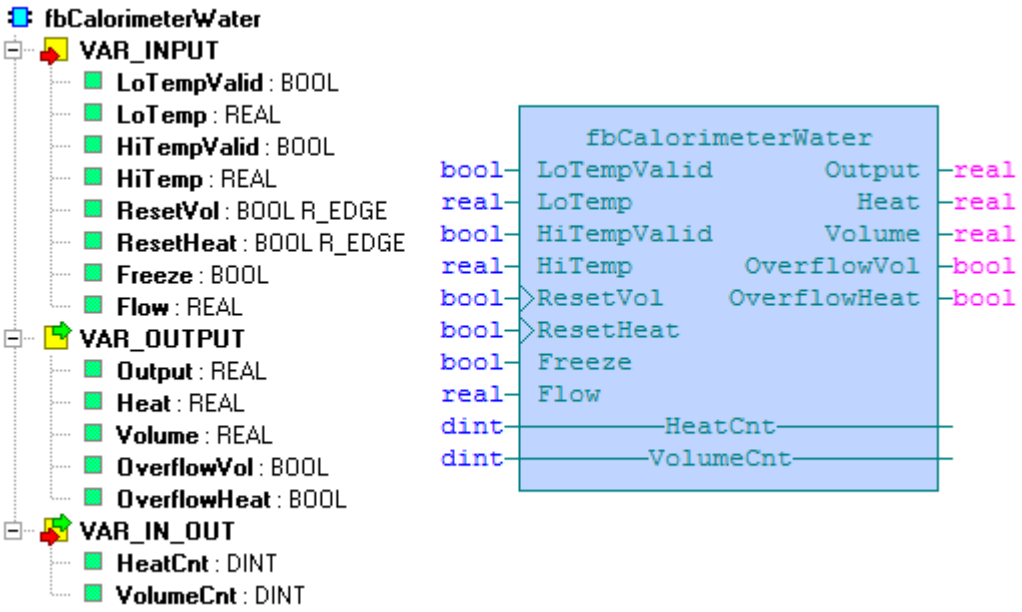
	Proměnná	Typ	Význam
VAR_INPUT			
	LoTempValid	BOOL	Nižší teplota je platná
	LoTemp	REAL	Nižší teplota
	HiTempValid	BOOL	Vyšší teplota je platná
	HiTemp	REAL	Vyšší teplota
	Cnt	UINT	Čítač pulzů z průtokoměru
	ResetVol	BOOL R_EDGE	Nulování počítadla proteklého objemu
	ResetHeat	BOOL R_EDGE	Nulování počítadla tepla
	Freeze	BOOL	Blokování čítání od vypnutého čerpadla
	Filter	TIME	Časová konstanta filtru okamžité spotřeby
	Pulse_per_m3	UDINT	Počet pulzů na jeden kubický metr (1000 l)
	MinFlow	REAL	Minimální měřitelný průtok [m^3/h]
	LowVHC	REAL	Objemová tepelná kapacita pro nižší teplotu [$kJm^{-3}h^{-1}$]
	HiVHC	REAL	Objemová tepelná kapacita pro vyšší teplotu [$kJm^{-3}h^{-1}$]
VAR_OUTPUT			
	Output	REAL	Tepelný výkon [kW]
	Heat	REAL	Celkové teplo [kWh]
	Volume	REAL	Celkový objem [m^3]
	OverflowVol	BOOL	Přetečení proteklého objemu
	OverflowHeat	BOOL	Přetečení čítače tepla
VAR_IN_OUT			
	HeatCnt	DINT	Počítadlo teplo [Wh] (musí být RETAIN!)
	VolumeCnt	DINT	Počítadlo proteklého množství [l] (musí být RETAIN!)

V následujícím příkladu je funkční blok *fbCalorimeterCounter* použit pro výpočet dodaného tepla a aktuálního výkonu pro vodu jako teplonosné médium. Pro určení objemové tepelné kapacity je použita funkce *Water_WHC*.

```
VAR_GLOBAL RETAIN
  HeatCounter : DINT;
  VolCounter  : DINT;
END_VAR

VAR_GLOBAL CONSTANT
  AnalogValid : TAIStat := ( UNF := false, UNR := false,
                             OVR := false, OVF := false,
                             FLS := false);
END_VAR

PROGRAM prgExampleCalorimeterCounter
  VAR
    Calorimeter : fbCalorimeterCounter;
    aktVykon    : REAL;
    celkTeplo   : REAL;
  END_VAR

  Calorimeter(LoTempValid := r0_p3_AI0.STAT = AnalogValid,
              LoTemp := r0_p3_AI0.ENG,
              HiTempValid := MI_CIB1_IN.ID1_IN.STAT.VLD1 AND
                             NOT MI_CIB1_IN.ID1_IN.STAT.OUF1,
              HiTemp := MI_CIB1_IN.ID1_IN.AI1,
              Cnt    := MI_CIB1_IN.ID1_IN.AI5,
              LoVHC := Water_WHC(Temp := r0_p3_AI0.ENG),
              HiVHC := Water_WHC(Temp := MI_CIB1_IN.ID1_IN.AI1),
              Pulse_per_m3 := 1000,
              HeatCnt := HeatCounter,
              VolumeCnt := VolCounter,
              Output => aktVykon,
              Heat => celkTeplo);

END_PROGRAM
```

6.11 Funkční blok fbCalorimeterWater

Knihovna : *EnergyLib*



Funkční blok *fbCalorimeterWater* je verzí bloku *fbCalorimeter* pro případ, kdy je teplonosným médiem voda. Blok má integrovanou funkci *Water_WHC*. Tím odpadá nutnost vstupů pro objemovou tepelnou kapacitu. Funkčnost bloku je jinak identická s *fbCalorimeter.*

Popis proměnných :

	Proměnná	*Typ*	*Význam*
VAR_INPUT			
	LoTempValid	BOOL	Nižší teplota je platná
	LoTemp	REAL	Nižší teplota
	HiTempValid	BOOL	Vyšší teplota je platná
	HiTemp	REAL	Vyšší teplota
	ResetVol	BOOL R_EDGE	Nulování počítadla proteklého objemu
	ResetHeat	BOOL R_EDGE	Nulování počítadla tepla
	Freeze	BOOL	Blokování čítání od vypnutého čerpadla
	Flow	REAL	Objemový průtok [m^3/h]

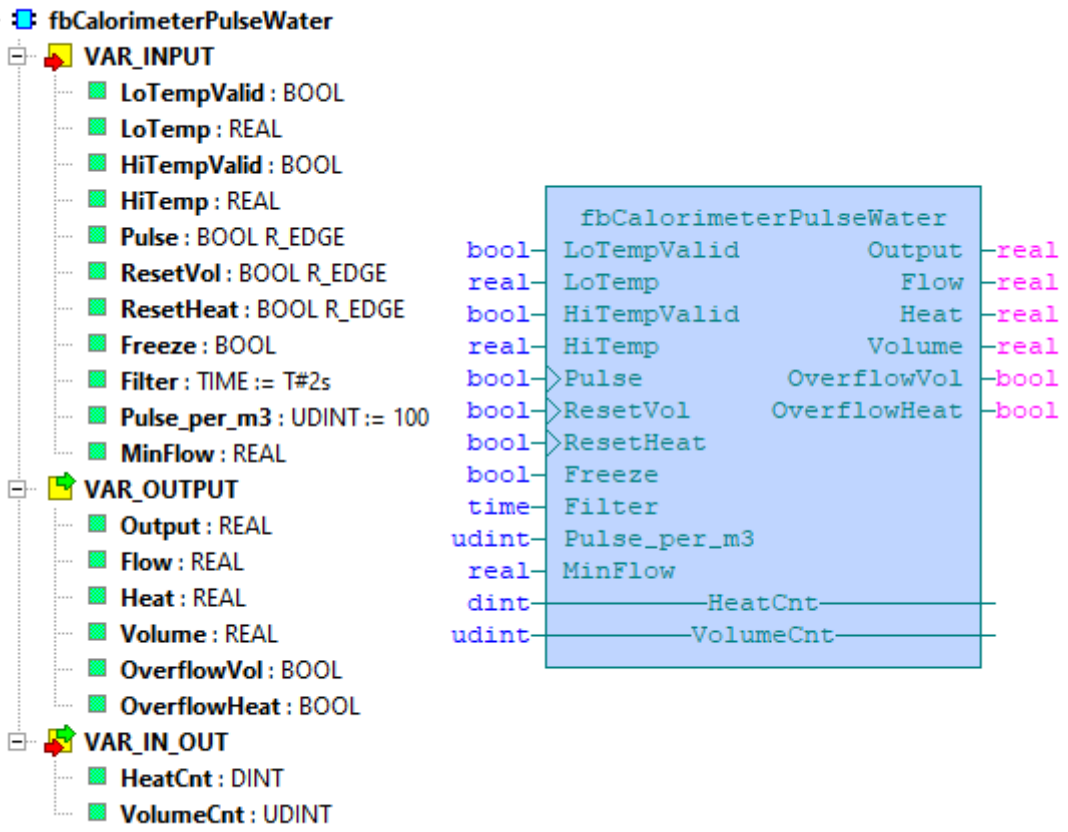
	Proměnná	Typ	Význam
VAR_OUTPUT			
	Output	REAL	Tepelný výkon [kW]
	Heat	REAL	Celkové teplo [kWh]
	Volume	REAL	Celkový objem [m^3]
	OverflowVol	BOOL	Přetečení proteklého objemu
	OverflowHeat	BOOL	Přetečení čítače tepla
VAR_IN_OUT			
	HeatCnt	DINT	Počítadlo teplo [Wh] (musí být RETAIN!)
	VolumeCnt	DINT	Počítadlo proteklého množství [l] (musí být RETAIN!)

V následujícím příkladu je funkční blok *fbCalorimeterWater* použit pro výpočet dodaného tepla a aktuálního výkonu pro vodu jako teplonosné médium.

```
VAR_GLOBAL RETAIN
  HeatCounter : DINT;
  VolCounter  : DINT;
END_VAR

VAR_GLOBAL CONSTANT
  AnalogValid : TAIStat := ( UNF := false, UNR := false,
                             OVR := false, OVF := false,
                             FLS := false);
END_VAR

PROGRAM prgExampleCalorimeterWater
  VAR
    Calorimeter : fbCalorimeterWater;
    aktVykon    : REAL;
    celkTeplo   : REAL;
  END_VAR

  Calorimeter(LoTempValid := r0_p3_AI0.STAT = AnalogValid,
              LoTemp := r0_p3_AI0.ENG,
              HiTempValid := MI_CIB1_IN.ID1_IN.STAT.VLD1 AND
                             NOT MI_CIB1_IN.ID1_IN.STAT.OUF1,
              HiTemp := MI_CIB1_IN.ID1_IN.AI1,
              Flow := MI_CIB1_IN.ID1_IN.AV23.FLOW / 0.06,
              HeatCnt := HeatCounter,
              VolumeCnt := VolCounter,
              Output => aktVykon,
              Heat => celkTeplo);

END_PROGRAM
```

6.12 Funkční blok fbCalorimeterPulseWater

Knihovna : *EnergyLib*



Funkční blok *fbCalorimeterPulseWater* je verzí bloku *fbCalorimeterPulse* pro případ, kdy je teplonosným médiem voda. Blok má integrovanou funkcí *Water_WHC*. Tím odpadá nutnost vstupů pro objemovou tepelnou kapacitu. Funkčnost bloku je jinak identická s *fbCalorimeterPulse.*

Popis proměnných :

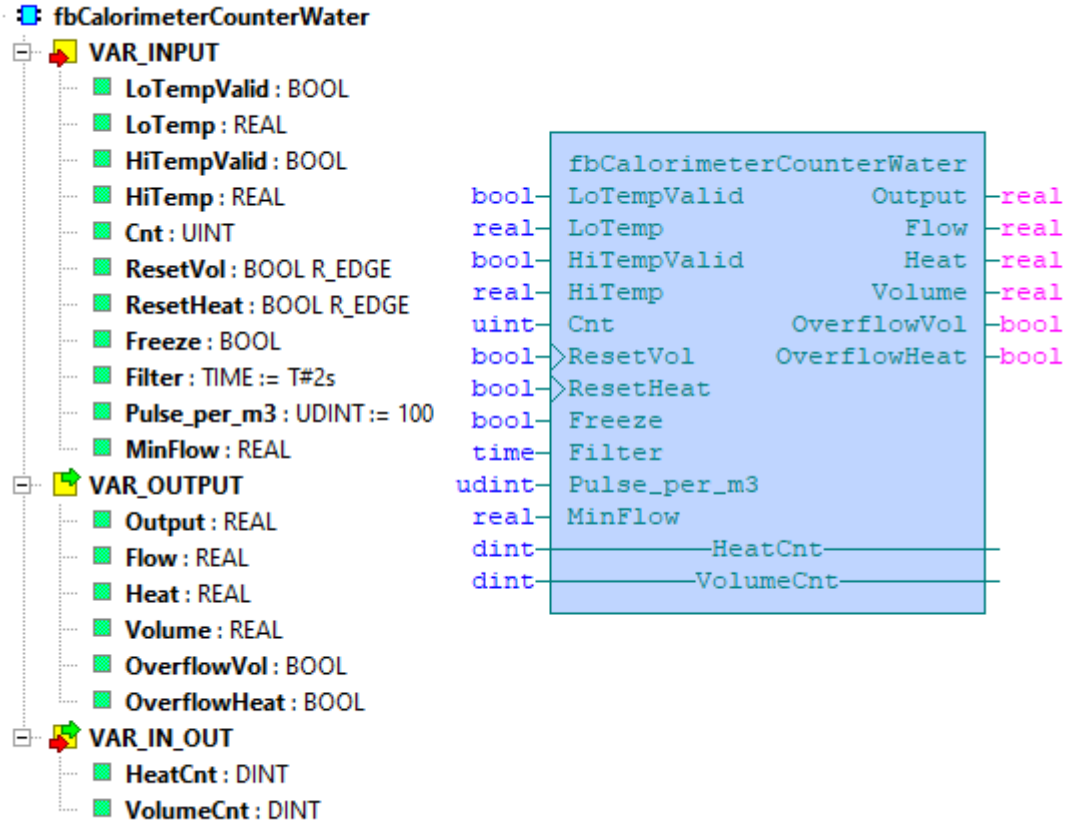
	Proměnná	Typ	Význam
VAR_INPUT			
	LoTempValid	BOOL	Nižší teplota je platná
	LoTemp	REAL	Nižší teplota
	HiTempValid	BOOL	Vyšší teplota je platná
	HiTemp	REAL	Vyšší teplota
	Pulse	BOOL R_EDGE	Pulz z průtokoměru
	ResetVol	BOOL R_EDGE	Nulování počítadla proteklého objemu
	ResetHeat	BOOL R_EDGE	Nulování počítadla tepla
	Freeze	BOOL	Blokování čítání od vypnutého čerpadla
	Filter	TIME	Časová konstanta filtru okamžité spotřeby
	Pulse_per_m3	UDINT	Počet pulzů na jeden kubický metr (1000 l)
	MinFlow	REAL	Minimální měřitelný průtok [m^3/h]
VAR_OUTPUT			
	Output	REAL	Tepelný výkon [kW]
	Heat	REAL	Celkové teplo [kWh]
	Volume	REAL	Celkový objem [m^3]
	OverflowVol	BOOL	Přetečení proteklého objemu
	OverflowHeat	BOOL	Přetečení čítače tepla
VAR_IN_OUT			
	HeatCnt	DINT	Počítadlo teplo [Wh] (musí být RETAIN!)
	VolumeCnt	DINT	Počítadlo proteklého množství [l] (musí být RETAIN!)

V následujícím příkladu je funkční blok *fbCalorimeterPulseWater* použit pro výpočet dodaného tepla a aktuálního výkonu pro vodu jako teplonosné médium.

```
VAR_GLOBAL RETAIN
  HeatCounter : DINT;
  VolCounter2 : UDINT;
END_VAR

VAR_GLOBAL CONSTANT
  AnalogValid : TAIStat := ( UNF := false, UNR := false,
                             OVR := false, OVF := false,
                             FLS := false);
END_VAR

PROGRAM prgExampleCalorimeterPulseWater
  VAR
    Calorimeter : fbCalorimeterPulseWater;
    aktVykon    : REAL;
    celkTeplo   : REAL;
  END_VAR

  Calorimeter(LoTempValid := r0_p3_AI0.STAT = AnalogValid,
              LoTemp := r0_p3_AI0.ENG,
              HiTempValid := MI_CIB1_IN.ID1_IN.STAT.VLD1 AND
                             NOT MI_CIB1_IN.ID1_IN.STAT.OUF1,
              HiTemp := MI_CIB1_IN.ID1_IN.AI1,
              Pulse  := r0_p3_DI.DI0,
              Pulse_per_m3 := 1000,
              HeatCnt := HeatCounter,
              VolumeCnt := VolCounter2,
              Output => aktVykon,
              Heat => celkTeplo);

END_PROGRAM
```

6.13 Funkční blok fbCalorimeterCounterWater

Knihovna : *EnergyLib*



Funkční blok *fbCalorimeterCounterWater* je verzí bloku *fbCalorimeterCounter* pro případ, kdy je teplonosným médiem voda. Blok má integrovanou funkcí *Water_WHC*. Tím odpadá nutnost vstupů pro objemovou tepelnou kapacitu. Funkčnost bloku je jinak identická s *fbCalorimeterCounter.*

Popis proměnných :

	Proměnná	Typ	Význam
VAR_INPUT			
	LoTempValid	BOOL	Nižší teplota je platná
	LoTemp	REAL	Nižší teplota
	HiTempValid	BOOL	Vyšší teplota je platná
	HiTemp	REAL	Vyšší teplota
	Cnt	UINT	Čítač pulzů z průtokoměru
	ResetVol	BOOL R_EDGE	Nulování počítadla proteklého objemu
	ResetHeat	BOOL R_EDGE	Nulování počítadla tepla
	Freeze	BOOL	Blokování čítání od vypnutého čerpadla
	Filter	TIME	Časová konstanta filtru okamžité spotřeby
	Pulse_per_m3	UDINT	Počet pulzů na jeden kubický metr (1000 l)
	MinFlow	REAL	Minimální měřitelný průtok [m^3/h]
VAR_OUTPUT			
	Output	REAL	Tepelný výkon [kW]
	Heat	REAL	Celkové teplo [kWh]
	Volume	REAL	Celkový objem [m^3]
	OverflowVol	BOOL	Přetečení proteklého objemu
	OverflowHeat	BOOL	Přetečení čítače tepla
VAR_IN_OUT			
	HeatCnt	DINT	Počítadlo teplo [Wh] (musí být RETAIN!)
	VolumeCnt	DINT	Počítadlo proteklého množství [l] (musí být RETAIN!)

V následujícím příkladu je funkční blok *fbCalorimeterCounterWater* použit pro výpočet dodaného tepla a aktuálního výkonu pro vodu jako teplonosné médium.

```
VAR_GLOBAL RETAIN
  HeatCounter : DINT;
  VolCounter  : DINT;
END_VAR

VAR_GLOBAL CONSTANT
  AnalogValid : TAIStat := ( UNF := false, UNR := false,
                             OVR := false, OVF := false,
                             FLS := false);
END_VAR

PROGRAM prgExampleCalorimeterCounterWater
  VAR
    Calorimeter : fbCalorimeterCounterWater;
    aktVykon    : REAL;
    celkTeplo   : REAL;
  END_VAR

  Calorimeter(LoTempValid := r0_p3_AI0.STAT = AnalogValid,
              LoTemp := r0_p3_AI0.ENG,
              HiTempValid := MI_CIB1_IN.ID1_IN.STAT.VLD1 AND
                             NOT MI_CIB1_IN.ID1_IN.STAT.OUF1,
              HiTemp := MI_CIB1_IN.ID1_IN.AI1,
              Cnt    := MI_CIB1_IN.ID1_IN.AI5,
              Pulse_per_m3 := 1000,
              HeatCnt := HeatCounter,
              VolumeCnt := VolCounter,
              Output => aktVykon,
              Heat => celkTeplo);

END_PROGRAM
```

TXV 003 65.01