Knihovna ConvertLib

TXV 003 82.01
první vydání
září 2013
změny vyhrazeny

Historie změn

Datum	Vydání	Popis změn
Září 2013	1	První vydání, popis odpovídá ConvertLib_v16

OBSAH

1 ÚVOD

Knihovna ConvertLib je standardně dodávaná jako součást programovacího prostředí Mosaic. Knihovna obsahuje funkce a funkční bloky pro převod hodnot a formátů.

Objednací číslo dokumentace ke knihovně ConvertLib je TXV 003 82.01

2 DATOVÉ TYPY

V knihovně ConvertLib nejsou definovány žádné datové typy.

3 KONSTANTY

V knihovně ConvertLib nejsou definovány žádné konstanty.

4 GLOBÁLNÍ PROMĚNNÉ

V knihovně ConvertLib je definována globální proměnná *MonthsNamesRFC822*.

Proměnná	***Typ***	***Popis***
MonthsNamesRFC822	ARRAY [1..12] OF STRING[3]	Trojpísmenné anglické zkratky měsíců popsané v RFC822

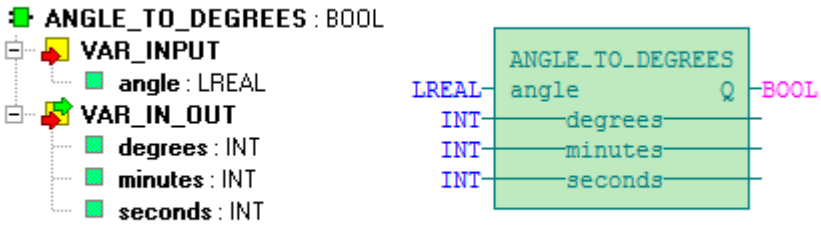
5 FUNKCE

Knihovna ConvertLib obsahuje následující funkce:

Funkce	Popis
ANGLE_TO_DEGREES	Převod úhlu ve stupních na stupně/minuty/sekundy
DEGREES_TO_ANGLE	Převod úhlu ze stupňů, minut a sekund na stupně
CELSIUS_TO_FAHRENHEIT	Převod teploty ze stupňů Celsia na stupně Fahrenheita
FAHRENHEIT_TO_CELSIUS	Převod teploty ze stupňů Fahrenheita na stupně Celsia
DEG_TO_RAD	Převod úhlových stupňů na radiány
RAD_TO_DEG	Převod radiánů na úhlové stupně
DT_RFC822_TO_DT	Převod data a času ve formátu podle RFC822 na IEC formát DATE_AND_TIME
DT_TO_DT_RFC822	Vrací datum a čas dle RFC822
ISO8859_2_TO_CP1250	Převod znaků kódovaných podle ISO 8859-2 na kódování CP 1250
REPLACE_CHAR	Nahradí všechny znaky *C1* znakem *C2*
STRING_HEX_TO_UDINT	Převod STRING (pouze hexadecimální číslice) na UDINT
STRING_TO_LOWER	Převod všech znaků v řetězci na malá písmena. Vrací délku převedeného řetězce
STRING_TO_UPPER	Převod všech znaků v řetězci na velká písmena. Vrací délku převedeného řetězce
WindDirectionT115Deg	Převod odporu měřeného na ukazateli směru větru T115 na úhlové stupně
WindDirectionT115StringCs	Převod odporu měřeného na ukazateli směru větru T115 na české označení směru
WindDirectionT115StringEn	Převod odporu měřeného na ukazateli směru větru T115 na anglické označení směru

5.1 Funkce ANGLE_TO_DEGREES

Knihovna : *ConvertLib*



Funkce *ANGLE_TO_DEGREES* převádí úhel ve stupních na stupně/minuty/vteřiny.

Popis proměnných :

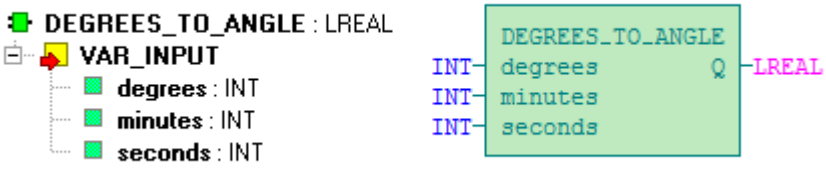
	Proměnná	Typ	Význam
VAR_INPUT			
	angle	LREAL	Úhel ve stupních
VAR_IN_OUT			
	degrees	INT	Stupně
	minutes	INT	Minuty
	seconds	INT	Vteřiny
ANGLE_TO_DEGREES			
	Návratová hodnota	BOOL	Vždy TRUE

Příklad programu s voláním funkce *ANGLE_TO_DEGREES* :

```
PROGRAM prgAngleToDegrees
  VAR
    angle : LREAL := 25.755;
    degrees, minutes, seconds : INT;
    text  : STRING;
  END_VAR

  ANGLE_TO_DEGREES(angle := angle, degrees := degrees,
                   minutes := minutes, seconds := seconds);
  text := INT_TO_STRING(degrees) + '°' +
          INT_TO_STRING(minutes) + '$'' +
          INT_TO_STRING(seconds) + '"';

END_PROGRAM
```

5.2 Funkce DEGREES_TO_ANGLE

Knihovna : *ConvertLib*



Funkce *DEGREES_TO_ANGLE* převádí úhel ve stupních/minutách/vteřinách na stupně.

Popis proměnných :

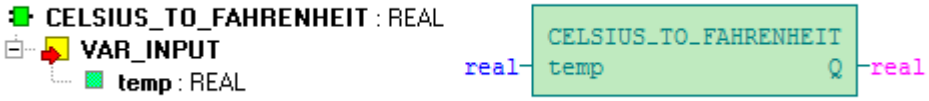
	Proměnná	Typ	Význam
VAR_INPUT			
	degrees	INT	Stupně
	minutes	INT	Minuty
	seconds	INT	Vteřiny
DEGREES_TO_ANGLE			
	Návratová hodnota	LREAL	Úhel ve stupních

Příklad programu s voláním funkce *DEGREES_TO_ANGLE* :

```
PROGRAM prgDegreesToAngle
  VAR
    seconds : INT := 18;
    minutes : INT := 45;
    degrees : INT := 25;
    angle : LREAL;
    text  : STRING;
  END_VAR

  angle := DEGREES_TO_ANGLE(degrees := degrees,
                            minutes := minutes,
                            seconds := seconds);
  text := LREAL_TO_STRING(angle) + '°';

END_PROGRAM
```

5.3 Funkce CELSIUS_TO_FAHRENHEIT

Knihovna : *ConvertLib*



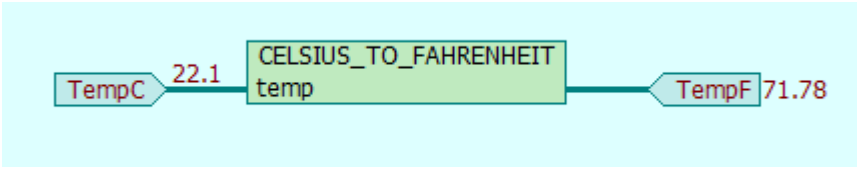
Funkce *CELSIUS_TO_FAHRENHEIT* realizuje převod teploty ze stupňů Celsia na stupně Fahrenheita dle vzorce:

$$\text{CELSIUS\_TO\_FAHRENHEIT} = \frac{9}{5} temp + 32$$

Popis proměnných :

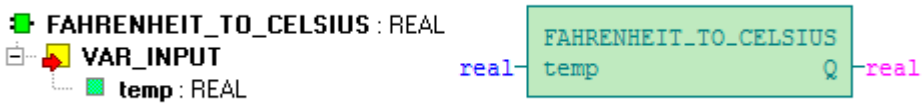
	Proměnná	***Typ***	***Význam***
VAR_INPUT			
	temp	REAL	Teplota ve stupních Celsia
CELSIUS_TO_FAHRENHEIT			
	Návratová hodnota	REAL	Teplota ve stupních Fahrenheita

Příklad programu s voláním funkce *CELSIUS_TO_FAHRENHEIT* :



5.4 Funkce FAHRENHEIT_TO_CELSIUS

Knihovna : *ConvertLib*



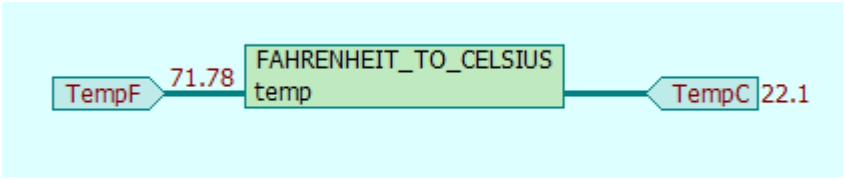
Funkce *FAHRENHEIT_TO_CELSIUS* realizuje převod teploty ze stupňů Fahrenheita na stupně Celsia dle vzorce:

$$\text{CELSIUS\_TO\_FAHRENHEIT} = \frac{5}{9}(temp - 32)$$

Popis proměnných :

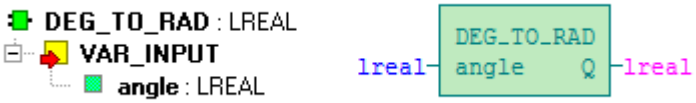
	Proměnná	***Typ***	***Význam***
VAR_INPUT			
	temp	REAL	Teplota ve stupních Fahrenheita
FAHRENHEIT_TO_CELSIUS			
	Návratová hodnota	REAL	Teplota ve stupních Celsia

Příklad programu s voláním funkce *FAHRENHEIT_TO_CELSIUS* :



5.5 Funkce DEG_TO_RAD

Knihovna : *ConvertLib*



Funkce *DEG_TO_RAD* převede úhel ve stupních na radiány.

Popis proměnných :

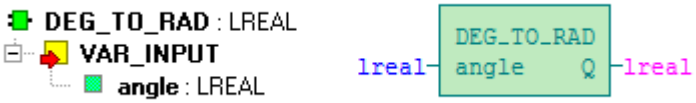
	Proměnná	Typ	Význam
VAR_INPUT			
	angle	LREAL	Úhel ve stupních
DEG_TO_RAD			
	Návratová hodnota	STRING	Úhel v radiánech

Příklad programu s voláním funkce *DEG_TO_RAD* :

```
PROGRAM prgConvertAngle1
  VAR
    angle : LREAL := 90.0;
    sin1  : LREAL;
  END_VAR

  sin1 := SIN(DEG_TO_RAD(angle));

END_PROGRAM
```

5.6 Funkce RAD_TO_DEG

Knihovna : *ConvertLib*



Funkce *RAD_TO_DEG* převede úhel ve stupních na radiány.

Popis proměnných :

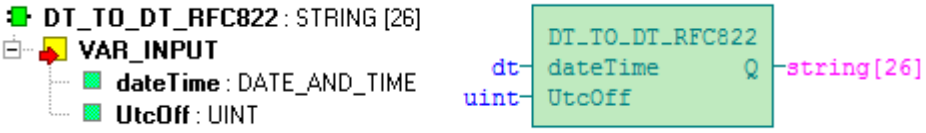
	Proměnná	Typ	Význam
VAR_INPUT			
	angle	LREAL	Úhel v radiánech
RAD_TO_DEG			
	Návratová hodnota	STRING	Úhel ve stupních

Příklad programu s voláním funkce *RAD_TO_DEG* :

```
PROGRAM prgConvertAngle2
  VAR
    sin1  : LREAL := 1.0;
    angle : LREAL;
  END_VAR

  angle := RAD_TO_DEG(ASIN(sin1));

END_PROGRAM
```

5.7 Funkce DT_TO_DT_RFC822

Knihovna : *ConvertLib*



Funkce pro převod časového údaje zapsaného v textovém řetězci ve formátu podle RFC822 na IEC formát DATE_AND_TIME

Datum a čas podle RFC822:

datum a čas	datum čas
datum	den měsíc rok
rok	4 čísla
měsíc	jméno-měsíce
Jméno-měsíce	"Jan" / "Feb" / "Mar" / "Apr" / "May" / "Jun" / "Jul" / "Aug" / "Sep" / "Oct" / "Nov" / "Dec"
den	1-2 čísla
čas	čas-ve-dni zóna
čas-ve-dni	hodina ":" minuta ":" sekunda
hodina	2 čísla
minuta	2 čísla
sekunda	2 čísla
zóna	(("+" / "-") 4 čísla)

Příklad: 15 Jul 2013 08:12:00 +0100

Popis proměnných :

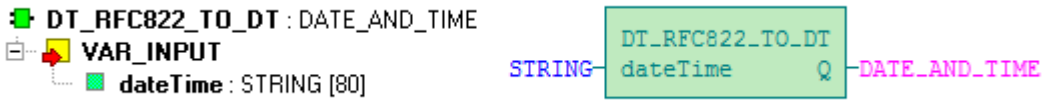
	Proměnná	***Typ***	***Význam***
VAR_INPUT			
	dateTime	DATE_AND_TIME	Datum a čas ve formátu IEC DATE_AND_TIME
DT_TO_DT_RFC822			
	Návratová hodnota	STRING	Datum a čas podle RFC822

Příklad programu s voláním funkce *DT_RFC822_TO_DT* :

```
PROGRAM prgConvertDT_RFC822
  VAR
    DtString : STRING;
  END_VAR

  DtString :=  DT_TO_DT_RFC822(dateTime := GetDateTime(), UtcOff := 60);

END_PROGRAM
```

5.8 Funkce DT_RFC822_TO_DT

Knihovna : *ConvertLib*



Funkce *DT_RFC822_TO_DT* převede datum a čas ve formátu podle RFC822 na IEC formát DATE_AND_TIME.

Popis proměnných :

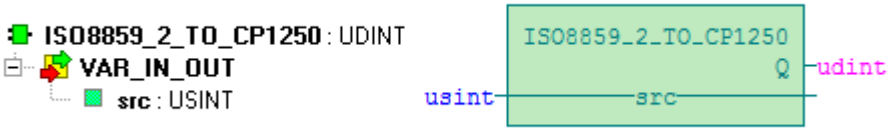
	Proměnná	*Typ*	*Význam*
VAR_INPUT			
	dateTime	STRING	Datum a čas podle RFC822
DT_RFC822_TO_DT			
	Návratová hodnota	DATE_AND_TIME	Datum a čas ve formátu IEC DATE_AND_TIME

Příklad programu s voláním funkce *DT_RFC822_TO_DT* :

```
PROGRAM prgConvertRFC822_DT
  VAR
    DtString : STRING := '15 Jul 2013 08:12:00 +0100';
    Dt1 : DATE_AND_TIME;
  END_VAR

  Dt1 := DT_RFC822_TO_DT(dateTime := DtString);

END_PROGRAM
```

5.9 Funkce ISO8859_2_TO_CP1250

Knihovna : *ConvertLib*



Funkce ISO8859_2_TO_CP1250 slouží pro převod znaků kódovaných podle ISO 8859-2 na kódování CP-1250.

Na vstupu src se očekává první byte řetězce pro převod. Funkce převede všechny znaky až do koncové nuly. Pokud je řetězec uložen v typu STRING je zajištěna koncová nula datovým typem. Pokud je řetěz uložen v jiném datovém typu je nutné koncovou binární nulu doplnit před voláním funkce.

Popis proměnných :

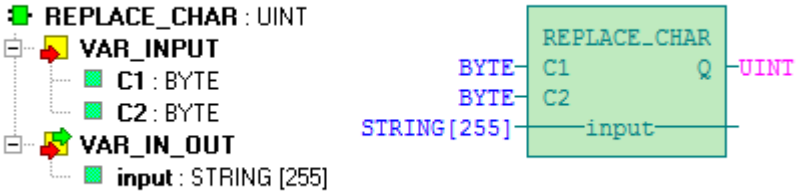
	Proměnná	Typ	Význam
VAR_IN_OUT			
	src	STRING	První byte řetězce pro převod
ISO8859_2_TO_CP1250			
	Návratová hodnota	UDINT	Počet znaků převedeného řetězce

Příklad programu s voláním funkce *ISO8859_2_TO_CP1250*. První volání podmíněné proměnnou *Test*, pouze ověřuje zda funkce pracuje správně. Druhé volání je příklad převodu textu přijatého z komunikace včetně ošetření koncové nuly.

```
PROGRAM prgConvertISO8859_2
  VAR
    Test : BOOL;
    TestString : STRING := 'Přilią ľlu»oučký kůň pěl ďábelské ódy';

    RecvFrom : fbRecvFrom;
    RecvBuff : ARRAY [0..255] OF USINT;
  END_VAR

  IF Test THEN
    ISO8859_2_TO_CP1250(src := void(TestString));
    Test := false;
  END_IF;


  RecvFrom(rq := 1, chanCode := ETH1_uni0, lenRx := 255,
           data := void(RecvBuff));

  IF RecvFrom.lenData <> 0 THEN
    RecvBuff[RecvFrom.lenData] := 0;
    ISO8859_2_TO_CP1250(src := RecvBuff[0]);
  END_IF;


END_PROGRAM
```

5.10 Funkce REPLACE_CHAR

Knihovna : *ConvertLib*



Funkce *REPLACE_CHAR* slouží k nahrazení všech výskytů znaku v řetězci *input* s ASCII hodnotou danou vstupem *C1* za znak daný hodnotu vstupu *C2*. Návratová hodnota je délka řetězce na vstupu *input*.

Popis proměnných :

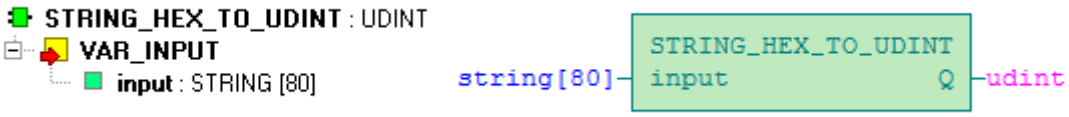
	Proměnná	*Typ*	*Význam*
VAR_INPUT			
	C1	BYTE	ASCII hodnota znaku, který má být nahrazen
	C2	BYTE	ASCII hodnota znaku, kterým bude náhrada provedena
VAR_IN_OUT			
	input	STRING[255]	Řetězec ve kterém bude provedena náhrada
REPLACE_CHAR			
	Návratová hodnota	UDINT	Počet znaků řetězce na vstupu *input*

Příklad programu s voláním funkce *REPLACE_CHAR*, převod desetinné tečky na čárku

```
PROGRAM prgReplaceChar
  VAR
    Message : STRING;
    Temp    : REAL;
  END_VAR

  Message := REAL_TO_STRINGF(in := Temp, format := 'Teplota je %.1f°C');
  REPLACE_CHAR(C1 := 16#2E, C2 := 16#2C, input := Message);

END_PROGRAM
```

5.11 Funkce STRING_HEX_TO_UDINT

Knihovna : *ConvertLib*



Funkce *STRING_HEX_TO_UDINT* slouží k převodu hexadecimálního čísla zapsaného ve STRINGu na UDINT. Ve STRINGu jsou očekávána pouze hexadecimální číslice.

Povolené znaky: '0132456789abcdefABCDEF'

Popis proměnných :

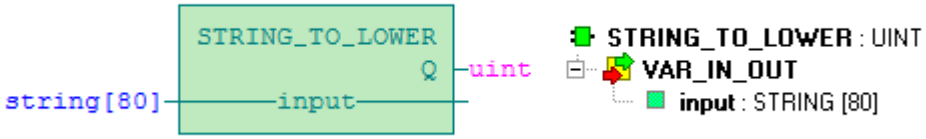
	Proměnná	Typ	Význam
VAR_IN_OUT			
	input	STRING	Řetězec s hexadecimálním číslem
STRING_HEX_TO_UDINT			
	Návratová hodnota	UDINT	Převedená hodnota hexadecimálního čísla

Příklad programu s voláním funkce *STRING_HEX_TO_UDINT*.

```
PROGRAM prgConvertHexNumber
  VAR
    HexString : STRING := 'a0';
    Value     : UDINT;
  END_VAR

  Value := STRING_HEX_TO_UDINT(input := HexString);

END_PROGRAM
```

5.12 Funkce STRING_TO_LOWER

Knihovna : *ConvertLib*



Funkce *STRING_TO_LOWER* slouží k převodu všech znaků v řetězci na malá písmena. Vrací délku převedeného řetězce.

Popis proměnných :

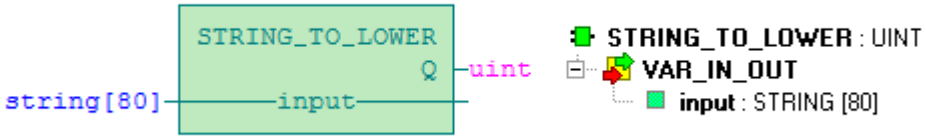
	Proměnná	*Typ*	*Význam*
VAR_IN_OUT			
	input	STRING[255]	Řetězec ve kterém bude provedena náhrada
STRING_TO_LOWER			
	Návratová hodnota	UDINT	Počet znaků řetězce na vstupu *input*

Příklad programu s voláním funkce *STRING_TO_LOWER*. Proměnná *Ok* se nastaví na TRUE v případě, že uživatel vloží do řetězce input slovo 'ok' bez ohledu na velikost písmen.

```
PROGRAM prgTestStringToLower
  VAR
    input : STRING;
    Ok    : BOOL;
  END_VAR

  STRING_TO_LOWER(input);
  Ok := input = 'ok';

END_PROGRAM
```

5.13 Funkce STRING_TO_UPPER

Knihovna : *ConvertLib*



Funkce *STRING_TO_UPPER* slouží k převodu všech znaků v řetězci na velká písmena. Vrací délku převedeného řetězce.

Popis proměnných :

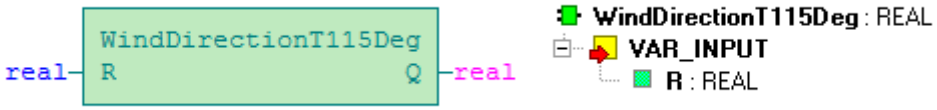
	Proměnná	*Typ*	*Význam*
VAR_IN_OUT			
	input	STRING[255]	Řetězec ve kterém bude provedena náhrada
STRING_TO_UPPER			
	Návratová hodnota	UDINT	Počet znaků řetězce na vstupu *input*

Příklad programu s voláním funkce *STRING_TO_UPPER*. Proměnná *Ok* se nastaví na TRUE v případě, že uživatel vloží do řetězce *input* slovo 'ok' bez ohledu na velikost písmen.

```
PROGRAM prgTestStringToUpper
  VAR
    input : STRING;
    Ok    : BOOL;
  END_VAR

  STRING_TO_UPPER(input);
  Ok := input = 'OK';

END_PROGRAM
```

5.14 Funkce WindDirectionT115Deg

Knihovna : *ConvertLib*



Funkce *WindDirectionT115Deg* slouží k převodu odporu měřeného na ukazateli směru větru T115 na úhlové stupně

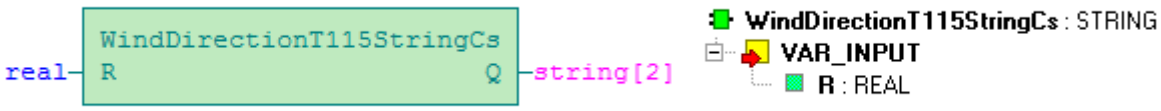
Popis proměnných :

	Proměnná	***Typ***	***Význam***
VAR_INPUT			
	R	REAL	Odpor měřený na ukazateli směru větru T115
STRING_HEX_TO_UDINT			
	Návratová hodnota	UDINT	Směr větru v úhlových stupních

Příklad programu s voláním funkce *WindDirectionT115Deg* viz Funkce WindDirectionT115StringEn.

5.15 Funkce WindDirectionT115StringCs

Knihovna : *ConvertLib*



Funkce *WindDirectionT115StringCs* slouží k převodu odporu měřeného na ukazateli na zkratku vyjadřující směr větru.

Návratové hodnoty	Význam	Hodnota ve stupních
S	sever	0
SV	severovýchod	45
V	východ	90
JV	jihovýchod	135
J	jih	180
JZ	jihozápad	225
Z	západ	270
SZ	severozápad	315

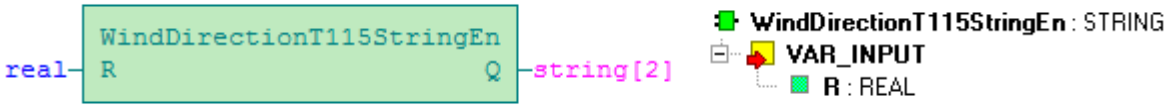
Popis proměnných :

	Proměnná	***Typ***	***Význam***
VAR_INPUT			
	R	REAL	Odpor měřený na ukazateli směru větru T115
WindDirectionT115StringCs			
	Návratová hodnota	STRING[2]	Dvoupísmenná zkratka směru větru

Příklad programu s voláním funkce *WindDirectionT115StringCs* viz Funkce WindDirectionT115StringEn.

5.16 Funkce WindDirectionT115StringEn

Knihovna : *ConvertLib*



Funkce *WindDirectionT115StringCs* slouží k převodu odporu měřeného na ukazateli na anglickou zkratku vyjadřující směr větru.

Návratové hodnoty	Význam	Hodnota ve stupních
N	north	0
NE	northeast	45
E	east	90
SE	southeast	135
S	south	180
SW	southwest	225
W	west	270
NW	northwest	315

Popis proměnných :

	Proměnná	***Typ***	***Význam***
VAR_INPUT			
	R	REAL	Odpor měřený na ukazateli směru větru T115
WindDirectionT115StringEn			
	Návratová hodnota	STRING[2]	Anglická dvoupísmenná zkratka směru větru

Příklad programu s voláním funkcí *WindDirectionT115Deg*, *WindDirectionT115StringCs* a *WindDirectionT115StringEn.*

```
PROGRAM prgWindDirection
  VAR
    Ohms : REAL;
    Direction   : REAL;
    DirectionEn : STRING;
    DirectionCs : STRING;

  END_VAR

  Direction   := WindDirectionT115Deg(R := Ohms);
  DirectionCs := WindDirectionT115StringCs(R := Ohms);
  DirectionEn := WindDirectionT115StringEn(R := Ohms);

END_PROGRAM
```

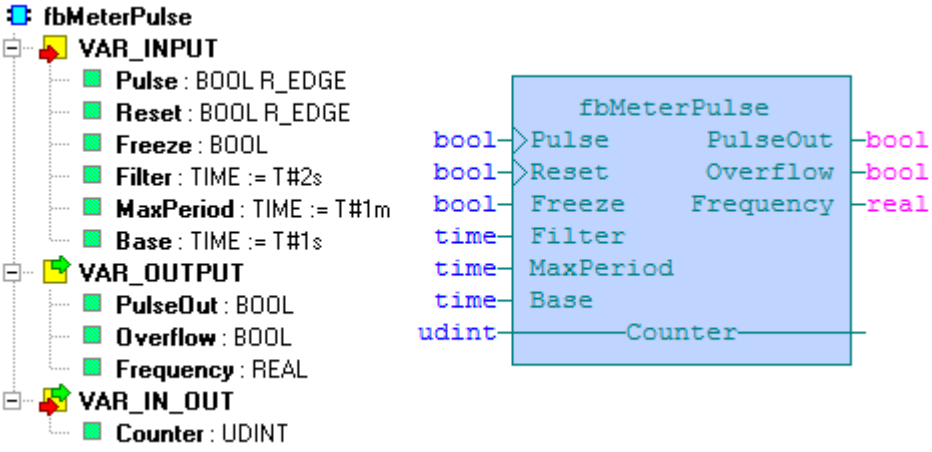
6 FUNKČNÍ BLOKY

V knihovně ConvertLib jsou definovány následující funkční bloky:

Funkční blok	***Popis***
fbMeterPulse	Počítá pulzy a jejich frekvenci
fbMeterCounter	Počítá pulzy a jejich frekvenci (s využitím čítače)

6.1 Funkční blok fbMeterPulse

Knihovna : *ConvertLib*



Funkční blok *fbMeterPulse* slouží k čítání a určování frekvence pulzů přicházejících na binární vstup. Čítány jsou náběžné hrany na vstupu *Pulse*. Na výstupu *PulseOut* je kopie vstupních hran.

Hodnota na vstupu *MaxPeriod* určuje jaká nejdelší doba mezi pulzy je ještě považována za nenulovou frekvenci. Při zastavení pulzů, určená frekvence klesá k hodnotě dané převrácenou hodnotou *MaxPeriod.* Pokud nepřijde do doby *MaxPeriod* pulz je frekvence prohlášena za nulovou.

Pokud je k dispozici informace o zastavení pulzů, je možné frekvenci vynulovat okamžitě nastavení vstupu *Freeze* na TRUE. Během doby, kdy je Freeze nastaven na TRUE se příchozí pulzy ignorují.

Hodnota vstupu *Base* určuje základ pro určování frekvence. Jednotky výstupu *Frequency* jsou pak dány převrácenou hodnotou hodnoty *Base.* Pro *Base* rovno 1 sekunda jsou jednotky sekundy na mínus první, tedy hertzy.

Pokud je nastaven vstup *Filter* na nenulovou hodnotu, je výstup *Frequency* dále upraven filtrem prvního řádu.

Celkový počet pulzů je uchováván v proměnné na vstupu *Counter* jako počet pulzů. Maximální celkový počet 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 celkového počtu pulzů je možné vynulovat náběžnou hranou na vstupu *Reset*.

Pro zachování hodnoty celkového počtu pulzů 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
	Reset	BOOL R_EDGE	Nulování počítadla pulzů

	Proměnná	Typ	Význam
	Freeze	BOOL	Zastaví čítání a nastaví frekvenci na nulu
	Filter	TIME	Časová konstanta filtru frekvence
	MaxPeriod	TIME	Maximální perioda mezi pulzy
	Base	TIME	Základní časová perioda pro určování frekvence
VAR_IN_OUT			
	Counter	UDINT	Počítadlo pulzů (musí být RETAIN!)
VAR_OUTPUT			
	PulseOut	BOOL	Kopie vstupních pulzů
	Overflow	BOOL	Přetečení čítače pulzů, nastavte Reset do logické 1
	Frequency	REAL	Frekvence [1/Base]

Příklad programu s funkčním blokem *fbMeterPulse* – měření rychlosti větru pro anemometr T114, který dává jeden pulz za sekundu pro rychlost větru 2,4 km/h. Proměnná *Counter* v tomto případě není remanentní, protože celkový počet pulzů není důležitý. *r0_p3_DI.DI0* je vstupem základního modulu. V proměnné *Velocity* bude rychlost v kilometrech za hodinu.

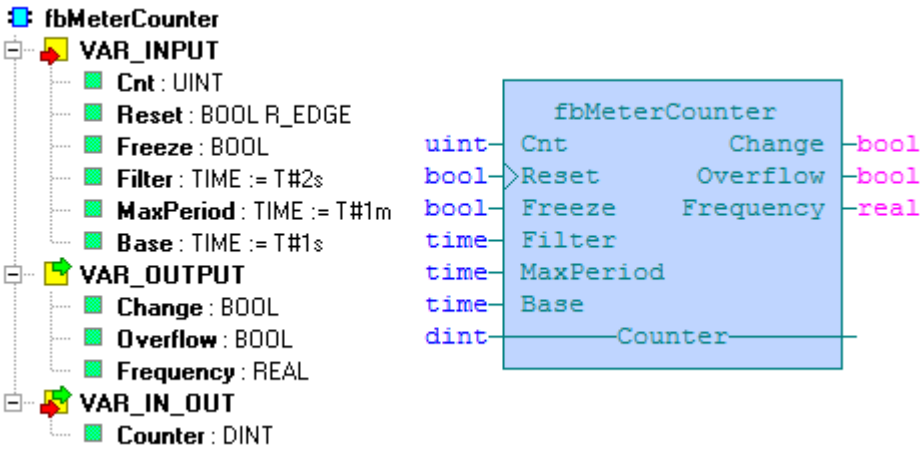
```
PROGRAM prgWindSpeed
  VAR
    MeterPulse : fbMeterPulse;
    Counter : UDINT;
  END_VAR

  MeterPulse(Pulse := r0_p3_DI.DI0,
             MaxPeriod := T#1m,
             Base := T#2.4s,
             Counter := Counter
             Frequency => Velocity);

END_PROGRAM
```

Příklad programu s funkčním blokem *fbMeterPulse* – měření celkového úhrnu srážek pro srážkoměr, který dává jeden puls na 0,2794 mm srážek. Proměnná *TotalRainFall* udává celkový úhrn srážek v milimetrech a proměnná *LastHour* přibližnou intenzitu srážek v milimetrech za hodinu. Proměnná *MI_CIB1_IN.ID1_IN.DI.DI1* představuje binární vstup CFox jednotky.

```
VAR_GLOBAL RETAIN
  RainfallCounter : UDINT;
END_VAR

PROGRAM prgRainfall
  VAR
    MeterPulse    : fbMeterPulse;
    TotalRainfall : REAL;
    LastHour      : REAL;
  END_VAR
  VAR CONSTANT
    OnePulse : REAL := 0.2794; // mm/pulse
  END_VAR

  MeterPulse(Pulse := MI_CIB1_IN.ID1_IN.DI.DI1,
             MaxPeriod := T#1h,
             Base := T#1h,
             Counter := RainfallCounter);

  LastHour      := MeterPulse.Frequency * OnePulse;
  TotalRainfall := UDINT_TO_REAL(RainfallCounter) * OnePulse;

END_PROGRAM
```

6.2 Funkční blok fbMeterCounter

Knihovna : *ConvertLib*



Funkční blok *fbMeterCounter* slouží k čítání a určování frekvence pulzů přicházejících na binární vstup vybavený čítačem. Čítány jsou rozdíly stavů čítače. Na výstupu *Change* je příznak změny stavu vstupního čítače.

Změna čítače mezi dvěma voláními bloku nesmí být větší než 50. Větší rozdíly jsou považovány za chybu a ignorovány.

Hodnota na vstupu *MaxPeriod* určuje jaká nejdelší doba mezi pulzy je ještě považována za nenulovou frekvenci. Při zastavení pulzů, určená frekvence klesá k hodnotě dané převrácenou hodnotou *MaxPeriod.* Pokud nepřijde do doby *MaxPeriod* pulz je frekvence prohlášena za nulovou.

Pokud je k dispozici informace o zastavení pulzů, je možné frekvenci vynulovat okamžitě nastavení vstupu *Freeze* na TRUE. Během doby, kdy je Freeze nastaven na TRUE se změny čítače ignorují.

Hodnota vstupu *Base* určuje základ pro určování frekvence. Jednotky výstupu *Frequency* jsou pak dány převrácenou hodnotou hodnoty *Base.* Pro *Base* rovno 1 sekunda jsou jednotky sekundy na mínus první, tedy hertzy.

Pokud je nastaven vstup *Filter* na nenulovou hodnotu, je výstup *Frequency* dále upraven filtrem prvního řádu.

Celkový počet pulzů je uchováván v proměnné na vstupu *Counter* jako počet pulzů. Maximální celkový počet 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 celkového počtu pulzů je možné vynulovat náběžnou hranou na vstupu *Reset*.

Pro zachování hodnoty celkového počtu pulzů 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
	Reset	BOOL R_EDGE	Nulování počítadla pulzů
	Freeze	BOOL	Zastaví čítání a nastaví frekvenci na nulu
	Filter	TIME	Časová konstanta filtru frekvence
	MaxPeriod	TIME	Maximální perioda mezi pulzy
	Base	TIME	Základní časová perioda pro určování frekvence
VAR_IN_OUT			
	Counter	DINT	Počítadlo pulzů (musí být RETAIN!)
VAR_OUTPUT			
	Change	BOOL	Čítač pulzů změnil hodnotu
	Overflow	BOOL	Přetečení čítače pulzů, nastavte Reset do logické 1
	Frequency	REAL	Frekvence [1/Base]

Příklad programu s funkčním blokem *fbMeterCounter*, pro měření průtoku. Blok počítá pulzy ze vstupu základního modulu *r0_p3_CNT_IN1.VALA*. Tento vstup má 32 bitový čítač, proto je použita ještě konverze UDINT_TO_UINT. Napočítané pulzy jsou přes konstantu udávající jaké proteklé množství odpovídá jednomu metru krychlovému, převeden na celkový proteklý objem (*celkObjem*). Frekvence udávající pulzy za hodinu je tou samou konstantou převedena na průtok v metrech krychlových za hodinu (*aktPrutok*).

```
VAR_GLOBAL RETAIN
  FCounter3 : DINT;
END_VAR

PROGRAM prgExampleFlowCounter
  VAR
    FMeter       : fbMeterCounter;
    aktPrutok    : REAL;
    celkObjem    : REAL;
  END_VAR

  VAR CONSTANT
    meterPerPulse : REAL := 0.01;
  END_VAR

  FMeter( Cnt           := UDINT_TO_UINT(r0_p3_CNT_IN1.VALA),
          Filter        := T#2s,
          Base          := T#1h,
          MaxPeriod     := T#120s,
          Counter       := FCounter3);

  celkObjem := DINT_TO_REAL(FCounter3) * meterPerPulse;
  aktPrutok := FMeter.Frequency * meterPerPulse;

END_PROGRAM
```