# Knihovna ToStringLib

**TXV 003 53.01**
**první vydání**
**září 2009**
**změny vyhrazeny**

## Historie změn

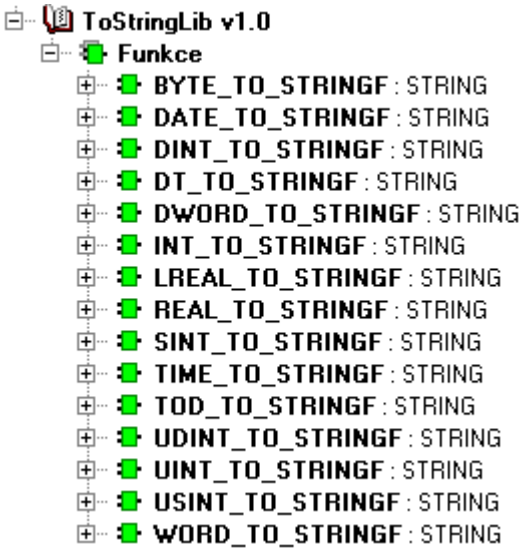
| Datum | Vydání | Popis změn |
|---|---|---|
| Září 2009 | 1 | První vydání |
| | | |

OBSAH

# 1 ÚVOD

Knihovna ToStringLib je standardně dodávaná jako součást programovacího prostředí Mosaic. Knihovna obsahuje konverzní funkce pro převod proměnných na řetězec s možností zadávat výstupní formát.

Následují obrázek ukazuje strukturu knihovny ToStringLib v prostředí Mosaic




Pokud chceme funkce z knihovny ToStringLib použít v aplikačním programu PLC, je třeba nejprve přidat tuto knihovnu do projektu. Knihovna je dodávaná jako součást instalace prostředí Mosaic od verze v2.0.18.0.

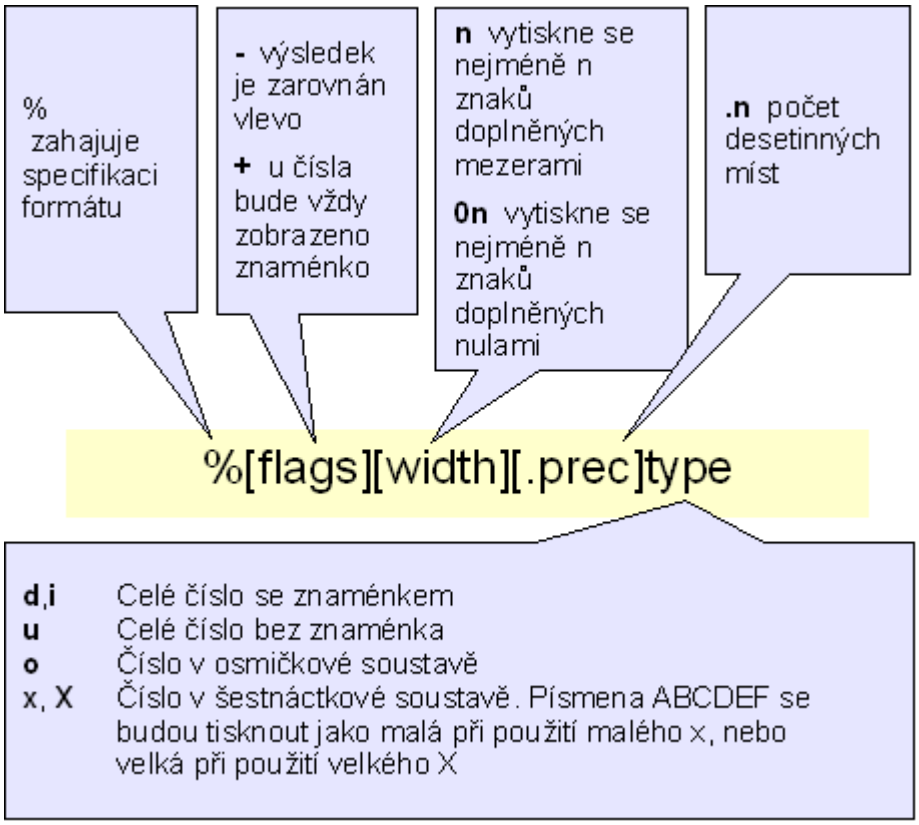
Knihovna ToStringLib není podporovaná na systémech TC-650, u systému TC700 nelze knihovnu použít s procesorovými moduly CP-7002, CP-7003 a CP-7005.

Funkce z knihovny ToStringLib jsou podporovány v centrálních jednotkách řady K (TC700 CP-7000 a CP-7004, všechny varianty systému Foxtrot) od verze v4.3.

## 1.1 Formátovací řetězec pro konverze ANY_INT_TO_STRINGF

Tento formátovací řetězec se používá pro následující konverzní funkce:

- *BYTE_TO_STRINGF*
- *WORD_TO_STRINGF*
- *DWORD_TO_STRINGF*
- *SINT_TO_STRINGF*
- *INT_TO_STRINGF*
- *DINT_TO_STRINGF*
- *USINT_TO_STRINGF*
- *UINT_TO_STRINGF*
- *UDINT_TO_STRINGF*



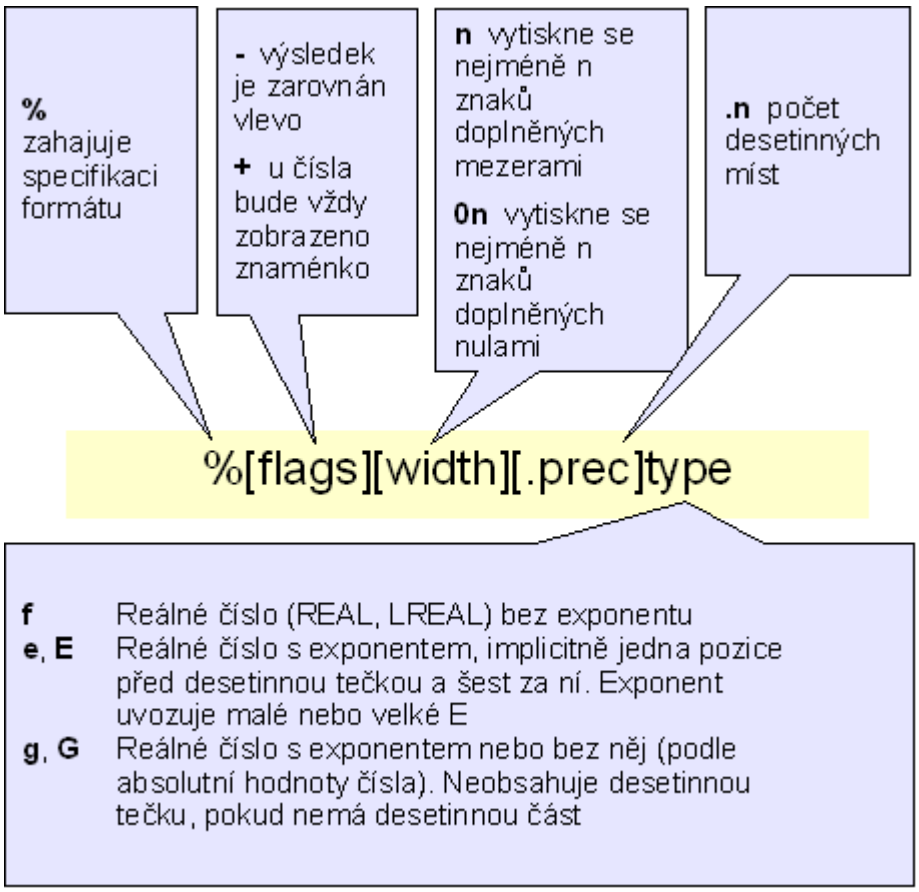

Formátovací řetězec může kromě specifikace formátu obsahovat i další znaky, které pak budou zkopírovány do výstupního řetězce (viz příklady v následující tabulce). Položky [flags], [width] a [.prec] ve specifikaci formátu nejsou povinné. Pokud je použita položka [prec], potom je na hodnotu pohlíženo jako na číslo s pevnou desetinnou čárkou.

| *Hodnota* | *Formátovací řetězec* | *Výstupní string* |
|---|---|---|
| `USINT#65` | `'humidity: %3u%%'` | `'humidity: 65%'` |
| `INT#-105` | `'temperature: %5.1d°C'` | `'temperature: -10.5°C'` |
| `WORD#16#ab7` | `'hex value: 0x%04X'` | `'hex value: 0x0AB7'` |

## 1.2 Formátovací řetězec pro konverze ANY_REAL_TO_STRINGF

Tento formátovací řetězec se používá pro následující konverzní funkce:

- *REAL_TO_STRINGF*
- *LREAL_TO_STRINGF*



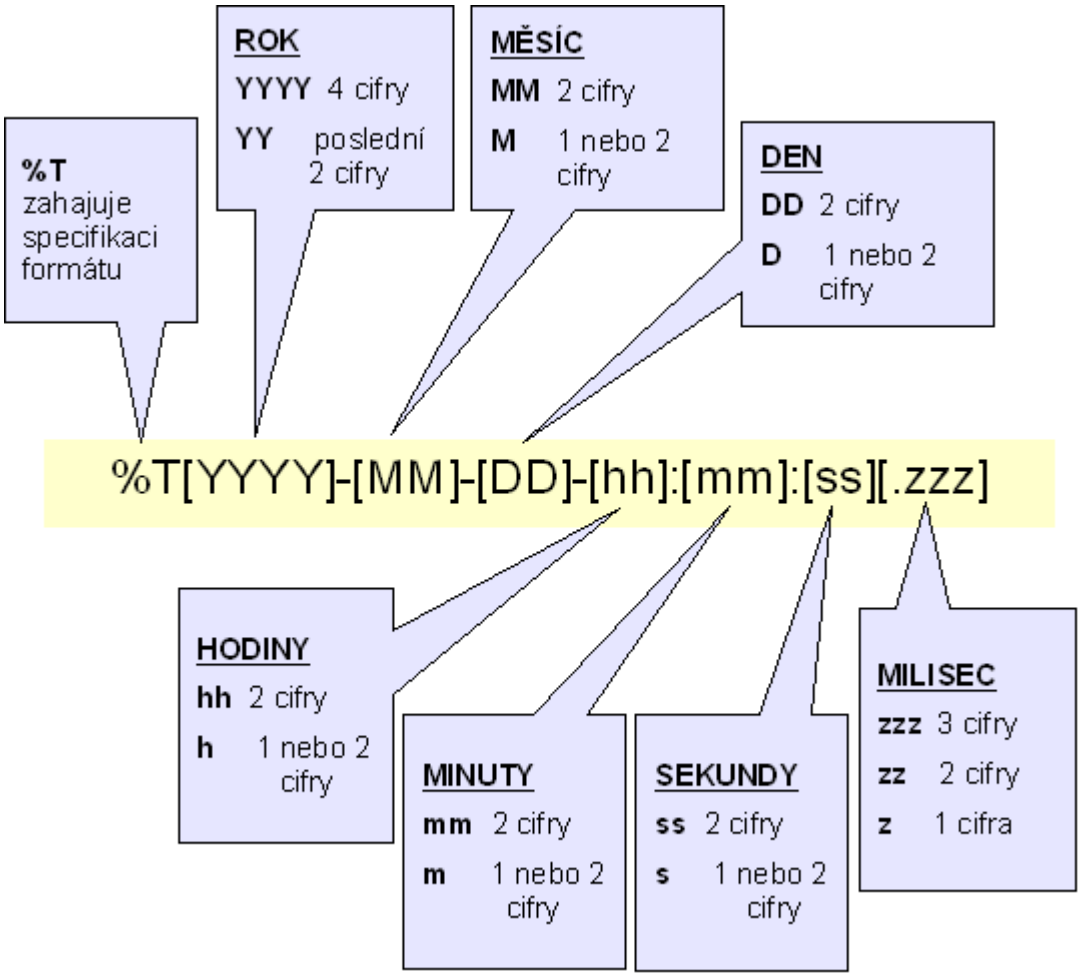

Formátovací řetězec může kromě specifikace formátu obsahovat i další znaky, které pak budou zkopírovány do výstupního řetězce (viz příklady v následující tabulce). Položky [flags], [width] a [.prec] ve specifikaci formátu nejsou povinné. Poslední platné cifry vypisované hodnoty budou zaokrouhleny.

| *Hodnota* | *Formátovací řetězec* | *Výstupní string* |
|---|---|---|
| `REAL#65.235` | `'humidity: %3.0f%%'` | `'humidity:  65%'` |
| `REAL#-10.576` | `'temperature: %5.1f°C'` | `'temperature: -10.6°C'` |
| `REAL#-10.476` | `'temperature: %5.1f°C'` | `'temperature: -10.5°C'` |
| `LREAL#12345678.9123` | `'current: %6.2e [A]'` | `'current: 1.23e+07 [A]'` |
| `LREAL#-12355678.9123` | `'number = %7.3G'` | `'number = -1.24E+07'` |

### *1.3 Formátovací řetězec pro konverze ANY_DATE_TO_STRING*

Tento formátovací řetězec se používá pro následující konverzní funkce:

- *DATE_TO_STRINGF*
- *DT_TO_STRINGF*



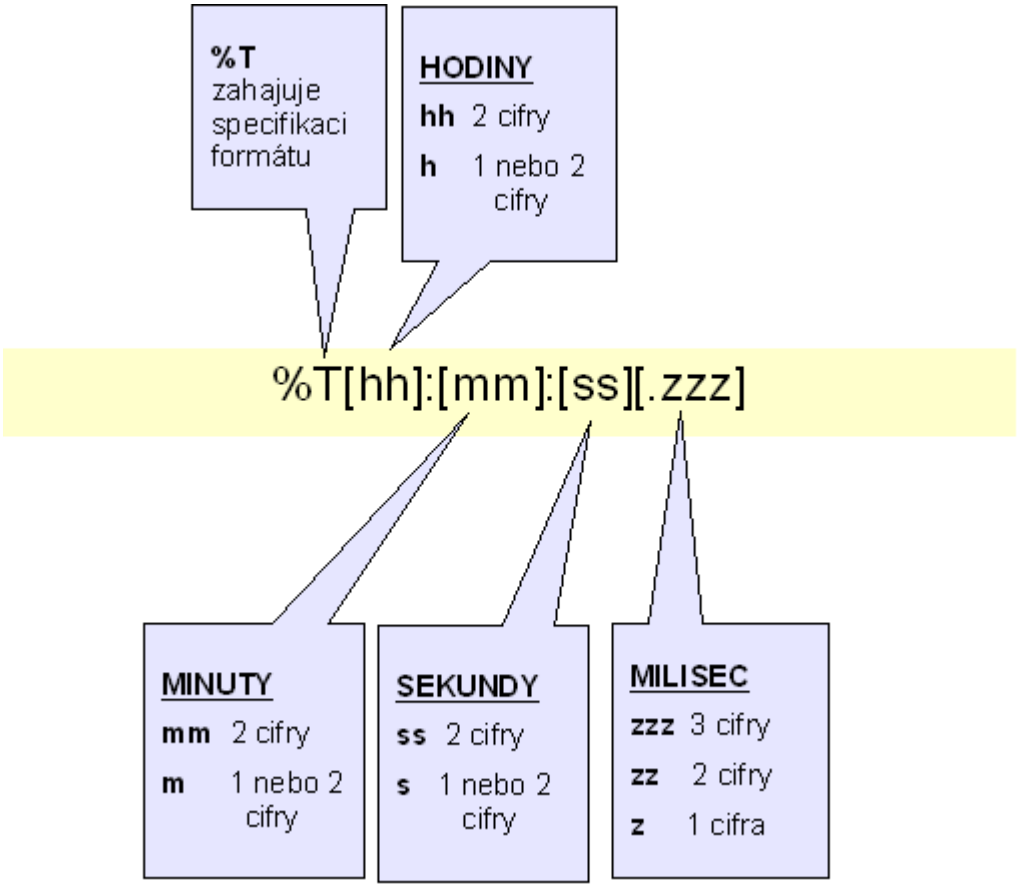

Formátovací řetězec může kromě specifikace formátu obsahovat i další znaky, které pak budou zkopírovány do výstupního řetězce (viz příklady v následující tabulce). Pořadí jednotlivých položek ve specifikaci formátu může být libovolné. Jako oddělovač položek mohou být použity znaky '-' (pomlčka), '.' (tečka), '/' (lomítko), ':' (dvojtečka) nebo pevná mezera (16#A0).

| *Hodnota* | *Formátovací řetězec* | *Výstupní string* |
|---|---|---|
| D#2007-11-05 | 'date: %TYYYY-MM-DD' | 'date: 2007-11-05' |
| D#2007-11-05 | 'date: %TDD.MM.YYYY' | 'date: 05.11.2007' |
| DT#2007-11-05-12:34:56 | '%TYYYY-MM-DD-hh:mm:ss' | '2007-11-05-12:34:56' |
| DT#2007-11-05-12:34:56 | '%Thh:mm:ss$A0DD.MM.YYYY' | '12:34:56 05.11.2007' |
| DT#2007-11-05-12:34:56 | 'month = %TMM' | 'month = 11' |
| DT#2007-11-05-12:34:56 | 'time : %Thh:mm [hh:mm]' | 'time : 12:34 [hh:mm]' |

### 1.4 Formátovací řetězec pro konverze ANY_TIME_TO_STRING

Tento formátovací řetězec se používá pro následující konverzní funkce:

- *TIME_TO_STRINGF*
- *TOD_TO_STRINGF*




Formátovací řetězec může kromě specifikace formátu obsahovat i další znaky, které pak budou zkopírovány do výstupního řetězce (viz příklady v následující tabulce). Pořadí jednotlivých položek ve specifikaci formátu může být libovolné. Jako oddělovač položek mohou být použity znaky '-' (pomlčka), '.' (tečka), '/' (lomítko), ':' (dvojtečka) nebo pevná mezera (16#A0).

| *Hodnota* | *Formátovací řetězec* | *Výstupní string* |
|---|---|---|
| `T#12:34:56.789` | `'time = %Thh:mm:ss.zzz'` | `'time = 12:34:56.789'` |
| `T#12:34:56.789` | `'%Tmm:ss [min:sec]'` | `'34:56 [min:sec]'` |
| `T#12:34:56.789` | `'preset : %Tss [sec]'` | `'preset : 56 [sec]'` |
| `T#12:34:56.789` | `'delay = %Tmm min'` | `'delay = 34 min'` |

# 2 DATOVÉ TYPY

V knihovně ToStringLib nejsoou definovány žádné datové typy.

# 3 KONSTANTY

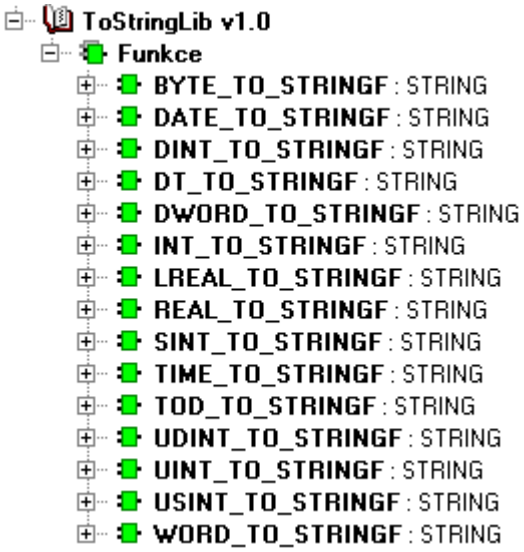
V knihovně ToStringLib nejsou definovány žádné konstanty.

# 4 GLOBÁLNÍ PROMĚNNÉ

V knihovně ToStringLib nejsou definovány žádné globální proměnné.

# 5 FUNKCE

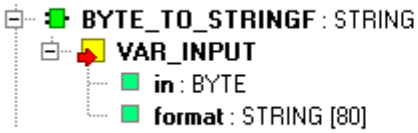
Knihovna ToStringLib obsahuje následující funkce:

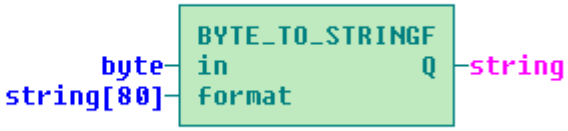



Všechny funkce převádí vstupní proměnnou na proměnnou typu STRING. Jinými slovy se hodnota dané proměnné zapisuje jako řetězec s možností zadávat výstupní formát. Např. pokud chceme převést proměnnou typu LREAL (reálné číslo) na řetězec, použijeme konverzní funkci *LREAL_TO_STRINGF* a máme možnost zvolit, kolik cifer a kolik desetinných míst budeme do řetězce zapisovat. To je rozdíl oproti starndardní funkci *LREAL_TO_STRING*, která neumožňuje ovlivňovat výstupní formát.

## 5.1 Funkce BYTE_TO_STRINGF

Knihovna : *ToStringLib*




Funkce *BYTE_TO_STRINGF* převede vstupní proměnnou typu BYTE do proměnné typu STRING.

Popis proměnných :

| | *Proměnná* | *Typ* | *Význam* |
|---|---|---|---|
| | | **VAR_INPUT** | |
| | *in* | BYTE | Hodnota, která bude převedena na STRING |
| | *format* | STRING | Formátovací řetězec (viz kap 1.1 ANY_INT_TO_STRING) |
| | | **BYTE_TO_STRINGF** | |
| | *Návratová hodnota* | STRING | Formátovaný výstup |

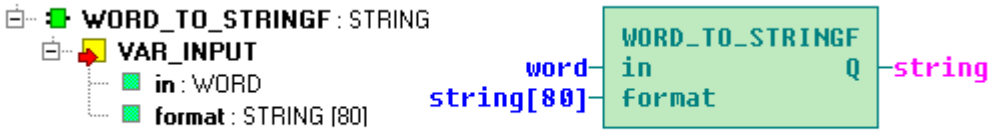
Příklad programu s voláním funkce *BYTE_TO_STRINGF* :

```
FUNCTION_BLOCK Example_BYTE_TO_STRINGF
  VAR_OUTPUT
    Str    : ARRAY[1..20] OF STRING[30];
  END_VAR

  Str[1]  := BYTE_TO_STRINGF( 123, '%d'            );  // '123'
  Str[2]  := BYTE_TO_STRINGF( 123, '%5d'           );  // '  123'
  Str[3]  := BYTE_TO_STRINGF( 123, '%05d'          );  // '00123'
  Str[4]  := BYTE_TO_STRINGF( 123, '%x'            );  // '7b'
  Str[5]  := BYTE_TO_STRINGF( 123, '%X'            );  // '7B'
  Str[6]  := BYTE_TO_STRINGF( 252, '%u'            );  // '252'
  Str[7]  := BYTE_TO_STRINGF( 252, '%5d'           );  // '   -4'
  Str[8]  := BYTE_TO_STRINGF( 252, '%05d'          );  // '-0004'
  Str[9]  := BYTE_TO_STRINGF( 255, '%x'            );  // 'ff'
  Str[10] := BYTE_TO_STRINGF( 255, '%04X'          );  // '00FF'
  Str[11] := BYTE_TO_STRINGF( 123, '%5.1d'         );  // ' 12.3'
  Str[12] := BYTE_TO_STRINGF( 254, '%6.1d'         );  // '  -0.2'
  Str[13] := BYTE_TO_STRINGF( 123, '%5.3d'         );  // '0.123'
  Str[14] := BYTE_TO_STRINGF( 254, '%8.4d'         );  // ' -0.0002'
  Str[15] := BYTE_TO_STRINGF( 0,   '%5.3d'         );  // '0.000'
  Str[16] := BYTE_TO_STRINGF( 0,   '0x%04x'        );  // '0x0000'
  Str[17] := BYTE_TO_STRINGF( 254, '%1d'           );  // '-2'
  Str[18] := BYTE_TO_STRINGF( 123, 'value : %d'    );  // 'value : 123'
  Str[19] := BYTE_TO_STRINGF( 254, '%8.2d'         );  // '   -0.02'
  Str[20] := BYTE_TO_STRINGF( 123, 'hex : 16#%02X' );  // 'hex : 16#7B'
END_FUNCTION_BLOCK
```

## 5.2 Funkce WORD_TO_STRINGF

Knihovna : *ToStringLib*




Funkce *BYTE_TO_STRINGF* převede vstupní proměnnou typu WORD do proměnné typu STRING.

Popis proměnných :

| | *Proměnná* | *Typ* | *Význam* |
|---|---|---|---|
| | | **VAR_INPUT** | |
| | *in* | WORD | Hodnota, která bude převedena na STRING |
| | *format* | STRING | Formátovací řetězec (viz kap 1.1 ANY_INT_TO_STRING) |
| | | **WORD_TO_STRINGF** | |
| | *Návratová hodnota* | STRING | Formátovaný výstup |

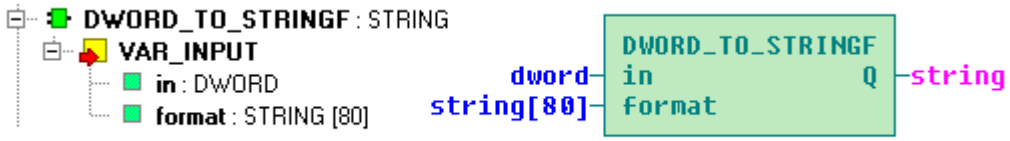
Příklad programu s voláním funkce *WORD_TO_STRINGF* :

```
FUNCTION_BLOCK Example_WORD_TO_STRINGF
  VAR_OUTPUT
    Str    : ARRAY[1..20] OF STRING[30];
  END_VAR

  Str[1]  := WORD_TO_STRINGF( 123,    '%d'          );   // '123'
  Str[2]  := WORD_TO_STRINGF( 123,    '%5d'         );   // '  123'
  Str[3]  := WORD_TO_STRINGF( 123,    '%05d'        );   // '00123'
  Str[4]  := WORD_TO_STRINGF( 123,    '%x'          );   // '7b'
  Str[5]  := WORD_TO_STRINGF( 12346, '%X'           );   // '303A'
  Str[6]  := WORD_TO_STRINGF( 252,    '%u'          );   // '252'
  Str[7]  := WORD_TO_STRINGF( 252,    '%5d'         );   // '  252'
  Str[8]  := WORD_TO_STRINGF( 252,    '%05d'        );   // '00252'
  Str[9]  := WORD_TO_STRINGF( 65535, '%x'           );   // 'ffff'
  Str[10] := WORD_TO_STRINGF( 65535, '%06X'         );   // '00FFFF'
  Str[11] := WORD_TO_STRINGF( 123,    '%5.1d'       );   // ' 12.3'
  Str[12] := WORD_TO_STRINGF( 3254,   '%7.1d'       );   // ' 325.4'
  Str[13] := WORD_TO_STRINGF( 123,    '%5.3d'       );   // '0.123'
  Str[14] := WORD_TO_STRINGF( 254,    '%8.4d'       );   // '  0.0254'
  Str[15] := WORD_TO_STRINGF( 0,      '%5.3d'       );   // '0.000'
  Str[16] := WORD_TO_STRINGF( 0,      '0x%04x'      );   // '0x0000'
  Str[17] := WORD_TO_STRINGF( 35254, '%1d'          );   // '-30282'
  Str[18] := WORD_TO_STRINGF( 123,    'value : %d'  );   // 'value : 123'
  Str[19] := WORD_TO_STRINGF( 35254, '%8.2d'        );   // ' -302.82'
  Str[20] := WORD_TO_STRINGF( 123,    'hex: 16#%04X');   // 'hex: 16#007B'
END_FUNCTION_BLOCK
```

## 5.3 Funkce DWORD_TO_STRINGF

Knihovna : *ToStringLib*




Funkce *DWORD_TO_STRINGF* převede vstupní proměnnou typu DWORD do proměnné typu STRING.

Popis proměnných :

| | *Proměnná* | *Typ* | *Význam* |
|---|---|---|---|
| | | **VAR_INPUT** | |
| | *in* | DWORD | Hodnota, která bude převedena na STRING |
| | *format* | STRING | Formátovací řetězec (viz kap 1.1 ANY_INT_TO_STRING) |
| | | **DWORD_TO_STRINGF** | |
| | *Návratová hodnota* | STRING | Formátovaný výstup |

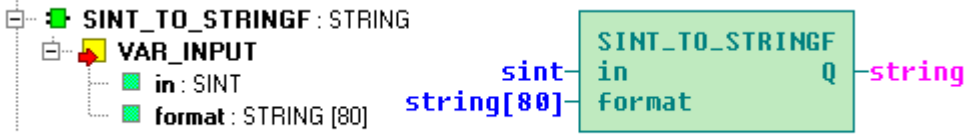
Příklad programu s voláním funkce *DWORD_TO_STRINGF* :

```
FUNCTION_BLOCK Example_DWORD_TO_STRINGF
  VAR_OUTPUT
    Str    : ARRAY[1..20] OF STRING[30];
  END_VAR

  Str[1]  := DWORD_TO_STRINGF( 123,          '%d'          ); // '123'
  Str[2]  := DWORD_TO_STRINGF( 123,          '%5d'         ); // '  123'
  Str[3]  := DWORD_TO_STRINGF( 123,          '%05d'        ); // '00123'
  Str[4]  := DWORD_TO_STRINGF( 123,          '%x'          ); // '7b'
  Str[5]  := DWORD_TO_STRINGF( 12345678,     '%08X'        ); // '00BC614E'
  Str[6]  := DWORD_TO_STRINGF( 252,          '%u'          ); // '252'
  Str[7]  := DWORD_TO_STRINGF( 252,          '%5d'         ); // '  252'
  Str[8]  := DWORD_TO_STRINGF( 252,          '%05d'        ); // '00252'
  Str[9]  := DWORD_TO_STRINGF( 16#ffff_ffff,'%x'           ); // 'ffffffff'
  Str[10] := DWORD_TO_STRINGF( 16#ffff_ffff,'%06X'         ); // 'FFFFFFFF'
  Str[11] := DWORD_TO_STRINGF( 123,          '%5.1d'       ); // ' 12.3'
  Str[12] := DWORD_TO_STRINGF( 3254,         '%7.1d'       ); // '  325.4'
  Str[13] := DWORD_TO_STRINGF( 123,          '%5.3d'       ); // '0.123'
  Str[14] := DWORD_TO_STRINGF( 254,          '%8.4d'       ); // '  0.0254'
  Str[15] := DWORD_TO_STRINGF( 0,            '%5.3d'       ); // '0.000'
  Str[16] := DWORD_TO_STRINGF( 0,            '0x%04x'      ); // '0x0000'
  Str[17] := DWORD_TO_STRINGF( 35254,        '%1d'         ); // '35254'
  Str[18] := DWORD_TO_STRINGF( 123,          'value : %d'  ); // 'value : 123'
  Str[19] := DWORD_TO_STRINGF( 35254,        '%8.2d'       ); // '  352.54';
  Str[20] := DWORD_TO_STRINGF( 16#1234_ABCD,'hex: 16#%08X');
                                                            // 'hex: 16#1234ABCD'
END_FUNCTION_BLOCK
```

## 5.4 Funkce SINT_TO_STRINGF

Knihovna : *ToStringLib*




Funkce *SINT_TO_STRINGF* převede vstupní proměnnou typu SINT do proměnné typu STRING.

Popis proměnných :

| | *Proměnná* | *Typ* | *Význam* |
|---|---|---|---|
| | | **VAR_INPUT** | |
| | *in* | SINT | Hodnota, která bude převedena na STRING |
| | *format* | STRING | Formátovací řetězec (viz kap 1.1 ANY_INT_TO_STRING) |
| | | **SINT_TO_STRINGF** | |
| | *Návratová hodnota* | STRING | Formátovaný výstup |

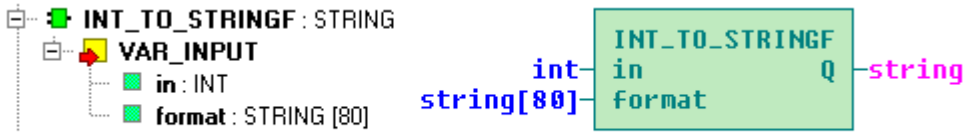
Příklad programu s voláním funkce *SINT_TO_STRINGF* :

```
FUNCTION_BLOCK Example_SINT_TO_STRINGF
  VAR_OUTPUT
    Str  : ARRAY[1..20] OF STRING[30];
  END_VAR

  Str[1]  := SINT_TO_STRINGF( 123, '%d'           );  // '123'
  Str[2]  := SINT_TO_STRINGF( 123, '%5d'          );  // '  123'
  Str[3]  := SINT_TO_STRINGF( 123, '%05d'         );  // '00123'
  Str[4]  := SINT_TO_STRINGF( 123, '%x'           );  // '7b'
  Str[5]  := SINT_TO_STRINGF( 123, '%X'           );  // '7B'
  Str[6]  := SINT_TO_STRINGF(-123, '%d'           );  // '-123'
  Str[7]  := SINT_TO_STRINGF(-123, '%5d'          );  // ' -123'
  Str[8]  := SINT_TO_STRINGF(-123, '%05d'         );  // '-0123'
  Str[9]  := SINT_TO_STRINGF(-1,   '%x'           );  // 'ff'
  Str[10] := SINT_TO_STRINGF(-1,   '%04X'         );  // '00FF'
  Str[11] := SINT_TO_STRINGF( 123, '%5.1d'        );  // ' 12.3'
  Str[12] := SINT_TO_STRINGF(-123, '%6.1d'        );  // ' -12.3'
  Str[13] := SINT_TO_STRINGF( 123, '%5.3d'        );  // '0.123'
  Str[14] := SINT_TO_STRINGF(-123, '%8.4d'        );  // ' -0.0123'
  Str[15] := SINT_TO_STRINGF( 0,   '%5.3d'        );  // '0.000'
  Str[16] := SINT_TO_STRINGF( 0,   '0x%04x'       );  // '0x0000'
  Str[17] := SINT_TO_STRINGF(-123, '%1d'          );  // '-123'
  Str[18] := SINT_TO_STRINGF( 123, 'value : %d'   );  // 'value : 123'
  Str[19] := SINT_TO_STRINGF(-123, '%8.2d'        );  // '   -1.23'
  Str[20] := SINT_TO_STRINGF( 123, 'val: %4.1d mA');  // 'val: 12.3 mA'
END_FUNCTION_BLOCK
```

## 5.5 Funkce INT_TO_STRINGF

Knihovna : *ToStringLib*




Funkce *INT_TO_STRINGF* převede vstupní proměnnou typu INT do proměnné typu STRING.

Popis proměnných :

| | *Proměnná* | *Typ* | *Význam* |
|---|---|---|---|
| | | **VAR_INPUT** | |
| | *in* | INT | Hodnota, která bude převedena na STRING |
| | *format* | STRING | Formátovací řetězec (viz kap 1.1 ANY_INT_TO_STRING) |
| | | **INT_TO_STRINGF** | |
| | *Návratová hodnota* | STRING | Formátovaný výstup |

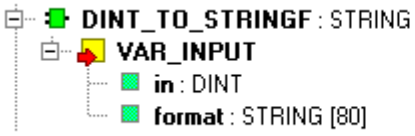
Příklad programu s voláním funkce *INT_TO_STRINGF* :

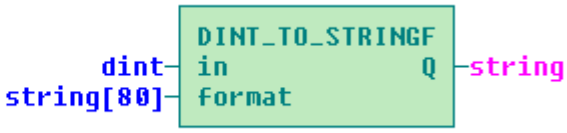
```
FUNCTION_BLOCK Example_INT_TO_STRINGF
  VAR_OUTPUT
    Str    : ARRAY[1..20] OF STRING[30];
  END_VAR

  Str[1]  := INT_TO_STRINGF( 123, '%d'           );  // '123'
  Str[2]  := INT_TO_STRINGF( 123, '%5d'          );  // '  123'
  Str[3]  := INT_TO_STRINGF( 123, '%05d'         );  // '00123'
  Str[4]  := INT_TO_STRINGF( 123, '%x'           );  // '7b'
  Str[5]  := INT_TO_STRINGF( 123, '%X'           );  // '7B'
  Str[6]  := INT_TO_STRINGF(-123, '%d'           );  // '-123'
  Str[7]  := INT_TO_STRINGF(-123, '%5d'          );  // ' -123'
  Str[8]  := INT_TO_STRINGF(-123, '%05d'         );  // '-0123'
  Str[9]  := INT_TO_STRINGF(-1,   '%x'           );  // 'ffff'
  Str[10] := INT_TO_STRINGF(-1,   '%06X'         );  // '00FFFF'
  Str[11] := INT_TO_STRINGF( 123, '%5.1d'        );  // ' 12.3'
  Str[12] := INT_TO_STRINGF(-123, '%6.1d'        );  // ' -12.3'
  Str[13] := INT_TO_STRINGF( 123, '%5.3d'        );  // '0.123'
  Str[14] := INT_TO_STRINGF(-123, '%8.4d'        );  // ' -0.0123'
  Str[15] := INT_TO_STRINGF( 0,   '%5.3d'        );  // '0.000'
  Str[16] := INT_TO_STRINGF( 0,   '0x%04x'       );  // '0x0000'
  Str[17] := INT_TO_STRINGF(-123, '%1d'          );  // '-123'
  Str[18] := INT_TO_STRINGF( 123, 'value : %d'   );  // 'value : 123'
  Str[19] := INT_TO_STRINGF(-123, '%8.2d'        );  // '   -1.23'
  Str[20] := INT_TO_STRINGF( 123, 'val: %4.1d mA');  // 'val: 12.3 mA'
END_FUNCTION_BLOCK
```

## 5.6 Funkce DINT_TO_STRINGF

Knihovna : *ToStringLib*




Funkce *DINT_TO_STRINGF* převede vstupní proměnnou typu DINT do proměnné typu STRING.

Popis proměnných :

| | *Proměnná* | *Typ* | *Význam* |
|---|---|---|---|
| | | **VAR_INPUT** | |
| | *in* | DINT | Hodnota, která bude převedena na STRING |
| | *format* | STRING | Formátovací řetězec (viz kap 1.1 ANY_INT_TO_STRING) |
| | | **DINT_TO_STRINGF** | |
| | *Návratová hodnota* | STRING | Formátovaný výstup |

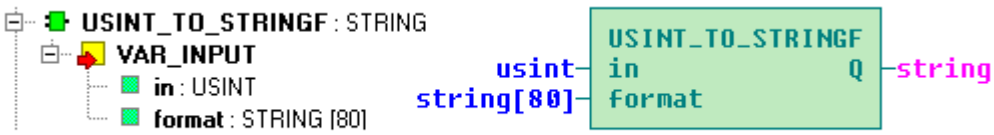
Příklad programu s voláním funkce *DINT_TO_STRINGF* :

```
FUNCTION_BLOCK Example_DINT_TO_STRINGF
  VAR_OUTPUT
    Str    : ARRAY[1..20] OF STRING[30];
  END_VAR

  Str[1]  := DINT_TO_STRINGF( 12345678, '%d'         ); // '12345678'
  Str[2]  := DINT_TO_STRINGF( 12345678, '%5d'        ); // '12345678'
  Str[3]  := DINT_TO_STRINGF( 12345678, '%09d'       ); // '012345678'
  Str[4]  := DINT_TO_STRINGF( 12345678, '%x'         ); // 'bc614e'
  Str[5]  := DINT_TO_STRINGF( 12345678, '%08X'       ); // '00BC614E'
  Str[6]  := DINT_TO_STRINGF(-12345678, '%d'         ); // '-12345678'
  Str[7]  := DINT_TO_STRINGF(-12345678, '%10d'       ); // ' -12345678'
  Str[8]  := DINT_TO_STRINGF(-12345678, '%010d'      ); // '-012345678'
  Str[9]  := DINT_TO_STRINGF(-1,        '%x'         ); // 'ffffffff'
  Str[10] := DINT_TO_STRINGF(-1,        '%08X'       ); // 'FFFFFFFF'
  Str[11] := DINT_TO_STRINGF( 12345678, '%5.1d'      ); // '1234567.8'
  Str[12] := DINT_TO_STRINGF(-12345678, '%6.1d'      ); // '-1234567.8'
  Str[13] := DINT_TO_STRINGF( 12345678, '%5.3d'      ); // '12345.678'
  Str[14] := DINT_TO_STRINGF(-12345678, '%8.4d'      ); // '-1234.5678'
  Str[15] := DINT_TO_STRINGF( 0,        '%5.3d'      ); // '0.000'
  Str[16] := DINT_TO_STRINGF( 0,        '0x%04x'     ); // '0x0000'
  Str[17] := DINT_TO_STRINGF(-12345678, '%1d'        ); // '-12345678'
  Str[18] := DINT_TO_STRINGF( 12345678, 'value : %d' ); // 'value : 12345678'
  Str[19] := DINT_TO_STRINGF(-12345678, '%8.2d'      ); // '-123456.78'
  Str[20] := DINT_TO_STRINGF(12345678,'val: %4.1d mA');//'val: 1234567.8 mA'
END_FUNCTION_BLOCK
```

## 5.7 Funkce USINT_TO_STRINGF

Knihovna : *ToStringLib*




Funkce *USINT_TO_STRINGF* převede vstupní proměnnou typu USINT do proměnné typu STRING.

Popis proměnných :

| | *Proměnná* | *Typ* | *Význam* |
|---|---|---|---|
| | | **VAR_INPUT** | |
| | *in* | USINT | Hodnota, která bude převedena na STRING |
| | *format* | STRING | Formátovací řetězec (viz kap 1.1 ANY_INT_TO_STRING) |
| | | **USINT_TO_STRINGF** | |
| | *Návratová hodnota* | STRING | Formátovaný výstup |

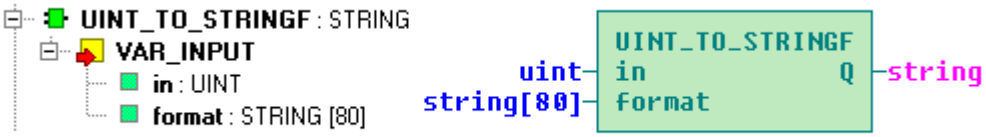
Příklad programu s voláním funkce *USINT_TO_STRINGF* :

```
FUNCTION_BLOCK Example_USINT_TO_STRINGF
  VAR_OUTPUT
    Str    : ARRAY[1..20] OF STRING[30];
  END_VAR
  VAR
    forString     : ARRAY [1..20] OF STRING [20] := [
    '%d'            ,  // 1
    '%5d'           ,  // 2
    '%05d'          ,  // 3
    '%x'            ,  // 4
    '%X'            ,  // 5
    '%u'            ,  // 6
    '%5d'           ,  // 7
    '%05d'          ,  // 8
    '%x'            ,  // 9
    '%04X'          ,  // 10
    '%5.1d'         ,  // 11
    '%6.1d'         ,  // 12
    '%5.3d'         ,  // 13
    '%8.4d'         ,  // 14
    '%5.3d'         ,  // 15
    '0x%04x'        ,  // 16
    '%1d'           ,  // 17
    'value : %d'    ,  // 18
    '%8.2d'         ,  // 19
    'val: %4.1d mA']; // 20
  END_VAR
```

```
  Str[1]  := USINT_TO_STRINGF( 123, forString[1]  );  // '123'
  Str[2]  := USINT_TO_STRINGF( 123, forString[2]  );  // '  123'
  Str[3]  := USINT_TO_STRINGF( 123, forString[3]  );  // '00123'
  Str[4]  := USINT_TO_STRINGF( 123, forString[4]  );  // '7b'
  Str[5]  := USINT_TO_STRINGF( 123, forString[5]  );  // '7B'
  Str[6]  := USINT_TO_STRINGF( 252, forString[6]  );  // '252'
  Str[7]  := USINT_TO_STRINGF( 252, forString[7]  );  // '   -4'
  Str[8]  := USINT_TO_STRINGF( 252, forString[8]  );  // '-0004'
  Str[9]  := USINT_TO_STRINGF( 255, forString[9]  );  // 'ff'
  Str[10] := USINT_TO_STRINGF( 255, forString[10] );  // '00FF'
  Str[11] := USINT_TO_STRINGF( 123, forString[11] );  // ' 12.3'
  Str[12] := USINT_TO_STRINGF( 254, forString[12] );  // ' -0.2'
  Str[13] := USINT_TO_STRINGF( 123, forString[13] );  // '0.123'
  Str[14] := USINT_TO_STRINGF( 254, forString[14] );  // ' -0.0002'
  Str[15] := USINT_TO_STRINGF( 0,   forString[15] );  // '0.000'
  Str[16] := USINT_TO_STRINGF( 0,   forString[16] );  // '0x0000'
  Str[17] := USINT_TO_STRINGF( 254, forString[17] );  // '-2'
  Str[18] := USINT_TO_STRINGF( 123, forString[18] );  // 'value : 123'
  Str[19] := USINT_TO_STRINGF( 254, forString[19] );  // '   -0.02'
  Str[20] := USINT_TO_STRINGF( 123, forString[20]);   // 'val: 12.3 mA'
END_FUNCTION_BLOCK
```

## 5.8 Funkce UINT_TO_STRINGF

Knihovna : *ToStringLib*




Funkce *UINT_TO_STRINGF* převede vstupní proměnnou typu UINT do proměnné typu STRING.

Popis proměnných :

| | *Proměnná* | *Typ* | *Význam* |
|---|---|---|---|
| | | **VAR_INPUT** | |
| | *in* | UINT | Hodnota, která bude převedena na STRING |
| | *format* | STRING | Formátovací řetězec (viz kap 1.1 ANY_INT_TO_STRING) |
| | | **UINT_TO_STRINGF** | |
| | *Návratová hodnota* | STRING | Formátovaný výstup |

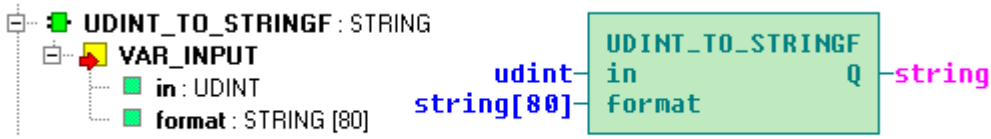
Příklad programu s voláním funkce *UINT_TO_STRINGF* :

```
FUNCTION_BLOCK Example_UINT_TO_STRINGF
  VAR_OUTPUT
    Str    : ARRAY[1..20] OF STRING[30];
  END_VAR

  Str[1]  := UINT_TO_STRINGF( 123,    '%d'            );  // '123'
  Str[2]  := UINT_TO_STRINGF( 123,    '%5d'           );  // '  123'
  Str[3]  := UINT_TO_STRINGF( 123,    '%05d'          );  // '00123'
  Str[4]  := UINT_TO_STRINGF( 123,    '%x'            );  // '7b'
  Str[5]  := UINT_TO_STRINGF( 12346,  '%X'            );  // '303A'
  Str[6]  := UINT_TO_STRINGF( 252,    '%u'            );  // '252'
  Str[7]  := UINT_TO_STRINGF( 252,    '%5d'           );  // '  252'
  Str[8]  := UINT_TO_STRINGF( 252,    '%05d'          );  // '00252'
  Str[9]  := UINT_TO_STRINGF( 65535,  '%x'            );  // 'ffff'
  Str[10] := UINT_TO_STRINGF( 65535,  '%06X'          );  // '00FFFF'
  Str[11] := UINT_TO_STRINGF( 123,    '%5.1d'         );  // ' 12.3'
  Str[12] := UINT_TO_STRINGF( 3254,   '%7.1d'         );  // ' 325.4'
  Str[13] := UINT_TO_STRINGF( 123,    '%5.3d'         );  // '0.123'
  Str[14] := UINT_TO_STRINGF( 254,    '%8.4d'         );  // '  0.0254'
  Str[15] := UINT_TO_STRINGF( 0,      '%5.3d'         );  // '0.000'
  Str[16] := UINT_TO_STRINGF( 0,      '0x%04x'        );  // '0x0000'
  Str[17] := UINT_TO_STRINGF( 35254,  '%1d'           );  // '-30282'
  Str[18] := UINT_TO_STRINGF( 123,    'value : %d'    );  // 'value : 123'
  Str[19] := UINT_TO_STRINGF( 35254,  '%8.2d'         );  // ' -302.82'
  Str[20] := UINT_TO_STRINGF( 123,    'val: %4.1d mA' );  // 'val: 12.3 mA'
END_FUNCTION_BLOCK
```

## 5.9 Funkce UDINT_TO_STRINGF

Knihovna : *ToStringLib*




Funkce *UDINT_TO_STRINGF* převede vstupní proměnnou typu UDINT do proměnné typu STRING.

Popis proměnných :

| | *Proměnná* | *Typ* | *Význam* |
|---|---|---|---|
| | | **VAR_INPUT** | |
| | *in* | UDINT | Hodnota, která bude převedena na STRING |
| | *format* | STRING | Formátovací řetězec (viz kap 1.1 ANY_INT_TO_STRING) |
| | | **UDINT_TO_STRINGF** | |
| | *Návratová hodnota* | STRING | Formátovaný výstup |

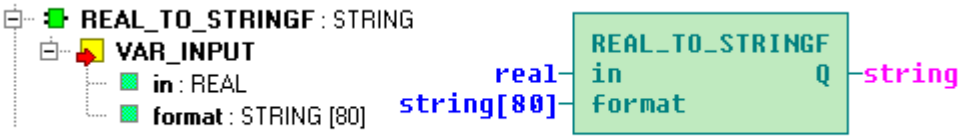
Příklad programu s voláním funkce *UDINT_TO_STRINGF* :

```
FUNCTION_BLOCK Example_UDINT_TO_STRINGF
  VAR_OUTPUT
    Str    : ARRAY[1..20] OF STRING[30];
  END_VAR

  Str[1]  := UDINT_TO_STRINGF( 123,          '%d'           );  // '123'
  Str[2]  := UDINT_TO_STRINGF( 123,          '%5d'          );  // '  123'
  Str[3]  := UDINT_TO_STRINGF( 123,          '%05d'         );  // '00123'
  Str[4]  := UDINT_TO_STRINGF( 123,          '%x'           );  // '7b'
  Str[5]  := UDINT_TO_STRINGF( 12345678,     '%08X'         );  // '00BC614E'
  Str[6]  := UDINT_TO_STRINGF( 252,          '%u'           );  // '252'
  Str[7]  := UDINT_TO_STRINGF( 252,          '%5d'          );  // '  252'
  Str[8]  := UDINT_TO_STRINGF( 252,          '%05d'         );  // '00252'
  Str[9]  := UDINT_TO_STRINGF( 16#ffff_ffff,'%x'            );  // 'ffffffff'
  Str[10] := UDINT_TO_STRINGF( 16#ffff_ffff,'%06X'          );  // 'FFFFFFFF'
  Str[11] := UDINT_TO_STRINGF( 123,          '%5.1d'        );  // ' 12.3'
  Str[12] := UDINT_TO_STRINGF( 3254,         '%7.1d'        );  // '  325.4'
  Str[13] := UDINT_TO_STRINGF( 123,          '%5.3d'        );  // '0.123'
  Str[14] := UDINT_TO_STRINGF( 254,          '%8.4d'        );  // '  0.0254'
  Str[15] := UDINT_TO_STRINGF( 0,            '%5.3d'        );  // '0.000'
  Str[16] := UDINT_TO_STRINGF( 0,            '0x%04x'       );  // '0x0000'
  Str[17] := UDINT_TO_STRINGF( 35254,        '%1d'          );  // '35254'
  Str[18] := UDINT_TO_STRINGF( 123,          'value : %d'   );  // 'value : 123'
  Str[19] := UDINT_TO_STRINGF( 35254,        '%8.2d'        );  // '  352.54';
  Str[20] := UDINT_TO_STRINGF( 123,          'val: %4.1d mA');  // 'val: 12.3 mA'
END_FUNCTION_BLOCK
```

## 5.10 Funkce REAL_TO_STRINGF

Knihovna : *ToStringLib*




Funkce *REAL_TO_STRINGF* převede vstupní proměnnou typu REAL do proměnné typu STRING.

Popis proměnných :

| | *Proměnná* | *Typ* | *Význam* |
|---|---|---|---|
| | | **VAR_INPUT** | |
| | *in* | REAL | Hodnota, která bude převedena na STRING |
| | *format* | STRING | Formátovací řetězec (viz kap 1.2 ANY_REAL_TO_STRING) |
| | | **REAL_TO_STRINGF** | |
| | *Návratová hodnota* | STRING | Formátovaný výstup |

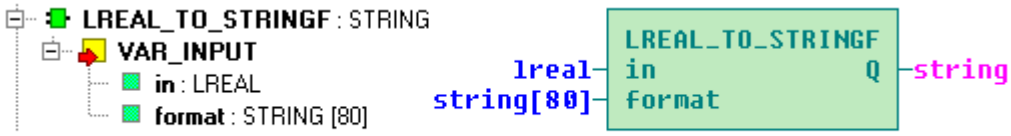
Příklad programu s voláním funkce *REAL_TO_STRINGF* :

```
FUNCTION_BLOCK Example_REAL_TO_STRINGF
  VAR_OUTPUT
    Str  : ARRAY[1..20] OF STRING[30];
  END_VAR

  Str[1]  := REAL_TO_STRINGF( 123.0, '%f'           );  // '123.000000'
  Str[2]  := REAL_TO_STRINGF( 123.0, '%6.1f'        );  // ' 123.0'
  Str[3]  := REAL_TO_STRINGF( 123.0, '%e'           );  // '1.230000e+02'
  Str[4]  := REAL_TO_STRINGF(-123.0, '%6.1e'        );  // '-1.2e+02'
  Str[5]  := REAL_TO_STRINGF( 123.0, '%x'           );  // '42f60000'
  Str[6]  := REAL_TO_STRINGF(-123.0, '%X'           );  // 'C2F60000'
  Str[7]  := REAL_TO_STRINGF(-123.0, '%g'           );  // '-123'
  Str[8]  := REAL_TO_STRINGF(-123.0, '%06g'         );  // '-00123'
  Str[9]  := REAL_TO_STRINGF(-1.099, '%6g'          );  // '-1.099'
  Str[10] := REAL_TO_STRINGF(-1.099, '%4.2g'        );  // '-1.1'
  Str[11] := REAL_TO_STRINGF( 0.99,  '%5.3f'        );  // '0.990'
  Str[12] := REAL_TO_STRINGF( 0.99,  '%5.3E'        );  // '9.900E-01'
  Str[13] := REAL_TO_STRINGF( 0.99,  '%5.3g'        );  // ' 0.99'
  Str[14] := REAL_TO_STRINGF( 0.99,  '%-8.2f'       );  // '0.99    '
  Str[15] := REAL_TO_STRINGF( 0.0,   '%f'           );  // '0.000000'
  Str[16] := REAL_TO_STRINGF( 0.0,   '%e'           );  // '0.000000e+00'
  Str[17] := REAL_TO_STRINGF( 0.0,   '%g'           );  // '0'
  Str[18] := REAL_TO_STRINGF( 123.0, 'value : %5.2f');  // 'value : 123.00'
  Str[19] := REAL_TO_STRINGF( 123.0, 'val: %6g mA'  );  // 'val:    123 mA'
  Str[20] := REAL_TO_STRINGF( 123.0, 'val: %4.1g mA');  // 'val: 1e+02 mA'
END_FUNCTION_BLOCK
```

## 5.11 Funkce LREAL_TO_STRINGF

Knihovna : *ToStringLib*




Funkce *LREAL_TO_STRINGF* převede vstupní proměnnou typu LREAL do proměnné typu STRING.

Popis proměnných :

| | *Proměnná* | *Typ* | *Význam* |
|---|---|---|---|
| | | **VAR_INPUT** | |
| | *in* | LREAL | Hodnota, která bude převedena na STRING |
| | *format* | STRING | Formátovací řetězec (viz kap 1.2 ANY_REAL_TO_STRING) |
| | | **LREAL_TO_STRINGF** | |
| | *Návratová hodnota* | STRING | Formátovaný výstup |

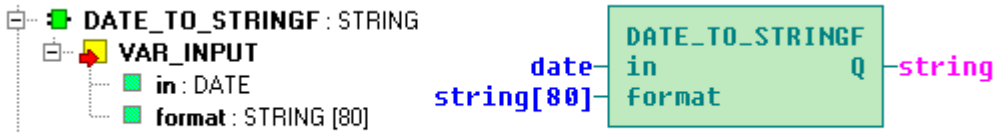
Příklad programu s voláním funkce *LREAL_TO_STRINGF* :

```
FUNCTION_BLOCK Example_LREAL_TO_STRINGF
  VAR_OUTPUT
    Str    : ARRAY[1..20] OF STRING[30];
  END_VAR

  Str[1]  := LREAL_TO_STRINGF( 123.0, '%f'           );  // '123.000000'
  Str[2]  := LREAL_TO_STRINGF( 123.0, '%6.1f'        );  // ' 123.0'
  Str[3]  := LREAL_TO_STRINGF( 123.0, '%e'           );  // '1.230000e+02'
  Str[4]  := LREAL_TO_STRINGF(-123.0, '%6.1e'        );  // '-1.2e+02'
  Str[5]  := LREAL_TO_STRINGF( 123.0, '%G'           );  // '123'
  Str[6]  := LREAL_TO_STRINGF(-123.0, '%06G'         );  // '-00123'
  Str[7]  := LREAL_TO_STRINGF(-123.0, '%g'           );  // '-123'
  Str[8]  := LREAL_TO_STRINGF(-123.0, '%06g'         );  // '-00123'
  Str[9]  := LREAL_TO_STRINGF(-1.099, '%6g'          );  // '-1.099'
  Str[10] := LREAL_TO_STRINGF(-1.099, '%4.2g'        );  // '-1.1'
  Str[11] := LREAL_TO_STRINGF( 0.99,  '%5.3f'        );  // '0.990'
  Str[12] := LREAL_TO_STRINGF( 0.99,  '%5.3E'        );  // '9.900E-01'
  Str[13] := LREAL_TO_STRINGF( 0.99,  '%5.3g'        );  // ' 0.99'
  Str[14] := LREAL_TO_STRINGF( 0.99,  '%-8.2f'       );  // '0.99    '
  Str[15] := LREAL_TO_STRINGF( 0.0,   '%f'           );  // '0.000000'
  Str[16] := LREAL_TO_STRINGF( 0.0,   '%e'           );  // '0.000000e+00'
  Str[17] := LREAL_TO_STRINGF( 0.0,   '%g'           );  // '0'
  Str[18] := LREAL_TO_STRINGF( 123.0, 'value : %5.2f');  // 'value : 123.00'
  Str[19] := LREAL_TO_STRINGF( 123.0, 'val: %6g mA'  );  // 'val:    123 mA'
  Str[20] := LREAL_TO_STRINGF( 123.0, 'val: %4.1g mA');  // 'val: 1e+02 mA'
END_FUNCTION_BLOCK
```

## 5.12 Funkce DATE_TO_STRINGF

Knihovna : *ToStringLib*




Funkce *DATE_TO_STRINGF* převede vstupní proměnnou typu DATE do proměnné typu STRING.

Popis proměnných :

| | *Proměnná* | *Typ* | *Význam* |
|---|---|---|---|
| | | **VAR_INPUT** | |
| | *in* | DATE | Hodnota, která bude převedena na STRING |
| | *format* | STRING | Formátovací řetězec (viz kap 1.3 ANY_DATE_TO_STRING) |
| | | **DATE_TO_STRINGF** | |
| | *Návratová hodnota* | STRING | Formátovaný výstup |

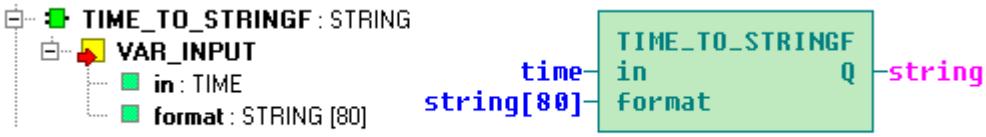
Příklad programu s voláním funkce *DATE_TO_STRINGF* :

```
FUNCTION_BLOCK Example_DATE_TO_STRINGF
  VAR_OUTPUT
    Str : ARRAY[1..10] OF STRING[50];
  END_VAR

  Str[1] := DATE_TO_STRINGF( D#2008-05-30, '%TYYYY-MM-DD'  );  // '2008-05-30'
  Str[2] := DATE_TO_STRINGF( D#2008-05-30, '%TYYYY-M-D'    );  // '2008-5-30'
  Str[3] := DATE_TO_STRINGF( D#2008-05-30, '%TDD.MM.YYYY'  );  // '30.05.2008'
  Str[4] := DATE_TO_STRINGF( D#2008-05-30, '%TYY/MM/DD'    );  // '08/05/30'
  Str[5] := DATE_TO_STRINGF( D#2008-05-30, '%TYYYY'        );  // '2008'
  Str[6] := DATE_TO_STRINGF( D#2008-05-30, 'year : %TYYYY' );  // 'year : 2008'
  Str[7] := DATE_TO_STRINGF( D#2008-05-30, 'month = %TMM'  );  // 'month = 05'
  Str[8] := DATE_TO_STRINGF( D#2008-05-30, 'day ... %TDD'  );  // 'day ... 30'
  Str[9] := DATE_TO_STRINGF( D#2008-05-30, '%TDD-MM-YYYY [day-month-year]');
                                             // '30-05-2008 [day-month-year]'
  Str[10]:= DATE_TO_STRINGF( D#2008-05-30, '$0A$r%TYYYY-MM-DD,');
                                                  // '$l$r2008-05-30,'
END_FUNCTION_BLOCK
```

### 5.13 Funkce TIME_TO_STRINGF

Knihovna : *ToStringLib*




Funkce *TIME_TO_STRINGF* převede vstupní proměnnou typu TIME do proměnné typu STRING.

Popis proměnných :

| | *Proměnná* | *Typ* | *Význam* |
|---|---|---|---|
| | | **VAR_INPUT** | |
| | *in* | TIME | Hodnota, která bude převedena na STRING |
| | *format* | STRING | Formátovací řetězec (viz kap 1.4 ANY_TIME_TO_STRING) |
| | | **TIME_TO_STRINGF** | |
| | *Návratová hodnota* | STRING | Formátovaný výstup |

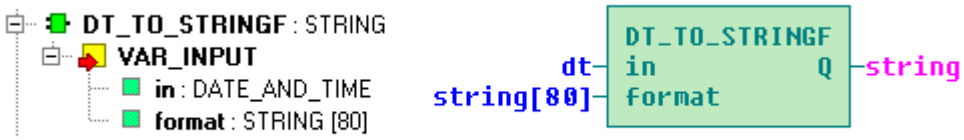
Příklad programu s voláním funkce *TIME_TO_STRINGF* :

```
FUNCTION_BLOCK Example_TIME_TO_STRINGF
  VAR_OUTPUT
    Str    : ARRAY[1..20] OF STRING[50];
  END_VAR

 Str[1] := TIME_TO_STRINGF( T#12:34:56.789, '%Thh:mm:ss.zzz'); // '12:34:56.789'
 Str[2] := TIME_TO_STRINGF( T#12:34:56.789, '%Thh:mm:ss.zz' ); // '12:34:56.78'
 Str[3] := TIME_TO_STRINGF( T#12:34:56.789, '%Thh:mm:ss.z'  ); // '12:34:56.7'
 Str[4] := TIME_TO_STRINGF( T#12:34:56.789, '%Thh:mm:ss'    ); // '12:34:56'
 Str[5] := TIME_TO_STRINGF( T#12:34:56.789, '%Thh:mm'       ); // '12:34'
 Str[6] := TIME_TO_STRINGF( T#12:34:56.789, '%Thh'          ); // '12'
 Str[7] := TIME_TO_STRINGF( T#12:34:56.789, '%Tmm'          ); // '34'
 Str[8] := TIME_TO_STRINGF( T#12:34:56.789, '%Tss'          ); // '56'
 Str[9] := TIME_TO_STRINGF( T#12:34:56.789, '%Tss.z'        ); // '56.7'
 Str[10]:= TIME_TO_STRINGF( T#12:34:56.789, '%Tss.zz'       ); // '56.78'
 Str[11]:= TIME_TO_STRINGF( T#12:34:56.789, '%Tss.zzz'      ); // '56.789'
 Str[12]:= TIME_TO_STRINGF( T#12:34:56.789, '%Tmm:ss.z'     ); // '34:56.7'
 Str[13]:= TIME_TO_STRINGF( T#12:34:56.789, '%Tmm:ss.zz'    ); // '34:56.78'
 Str[14]:= TIME_TO_STRINGF( T#12:34:56.789, '%Tmm:ss.zzz'   ); // '34:56.789'
 Str[15]:= TIME_TO_STRINGF( T#12:34:56.789, '%s'            ); // '12:34:56.789'
 Str[16]:= TIME_TO_STRINGF( T#12:34:56.789, '%Thh:mm:ss.z'  ); // '12:34:56.7'
 Str[17]:= TIME_TO_STRINGF( T#12:34:56.789, '%Thh/mm/ss.z'  ); // '12/34/56.7'
 Str[18]:= TIME_TO_STRINGF( T#12:34:56.789, 'time = %Thh:mm:ss.z [h:m:s.ms]');
                                              // 'time = 12:34:56.7 [h:m:s.ms]'
 Str[19]:= TIME_TO_STRINGF( T#12:34:56.789, '%Thh:mm:ss.z'  ); // '12:34:56.7'
END_FUNCTION_BLOCK
```

## 5.14 Funkce DT_TO_STRINGF

Knihovna : *ToStringLib*




Funkce *DT_TO_STRINGF* převede vstupní proměnnou typu DATE_AND_TIME do proměnné typu STRING.

Popis proměnných :

| | *Proměnná* | *Typ* | *Význam* |
|---|---|---|---|
| | | **VAR_INPUT** | |
| | *in* | DT | Hodnota, která bude převedena na STRING |
| | *format* | STRING | Formátovací řetězec (viz kap 1.3 ANY_REAL_TO_STRING) |
| | | **DT_TO_STRINGF** | |
| | *Návratová hodnota* | STRING | Formátovaný výstup |

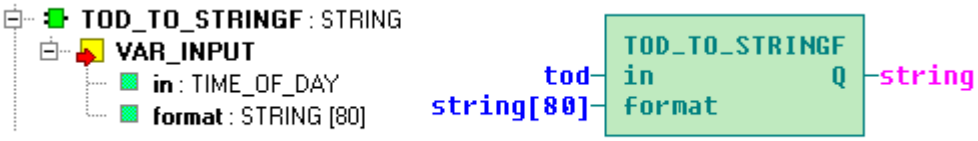
Příklad programu s voláním funkce *DT_TO_STRINGF* :

```
FUNCTION_BLOCK Example_DT_TO_STRINGF
  VAR_OUTPUT
    Str    : ARRAY[1..20] OF STRING[50];
  END_VAR
  VAR
    dateVar      : DATE_AND_TIME := DT#2008-05-30-12:34:56;
  END_VAR

  Str[1]  := DT_TO_STRINGF( dateVar, '%TDD.MM.YYYY$A0hh:mm:ss');
                                              // '30.05.2008 12:34:56'
  Str[2]  := DT_TO_STRINGF( dateVar, '%TYYYY-M-D');          // '2008-5-30'
  Str[3]  := DT_TO_STRINGF( dateVar, '%TYY/MM/DD-hh:mm');    // '08/05/30-12:34'
  Str[4]  := DT_TO_STRINGF( dateVar, '%TYY/MM/DD$A0hh:mm'); // '08/05/30 12:34'
  Str[5]  := DT_TO_STRINGF( dateVar, 'year : %TYYYY');       // 'year : 2008'
  Str[6]  := DT_TO_STRINGF( dateVar, 'month : %TMM');        // 'month : 05'
  Str[7]  := DT_TO_STRINGF( dateVar, 'day : %TDD');          // 'day : 30'
  Str[8]  := DT_TO_STRINGF( dateVar, 'hour : %Thh');         // 'hour : 12'
  Str[9]  := DT_TO_STRINGF( dateVar, 'min : %Tmm');          // 'min : 34'
  Str[10] := DT_TO_STRINGF( dateVar, 'sec : %Tss');          // 'sec : 56'
  Str[11] := DT_TO_STRINGF( dateVar, '%s');             // '2008-05-30-12:34:56'
  Str[12] := DT_TO_STRINGF( dateVar, '%Thh:mm:ss');          // '12:34:56'
  Str[13] := DT_TO_STRINGF( dateVar, 'date : %TYYYY/MM/DD [year/month/day]');
                                    // 'date : 2008/05/30 [year/month/day]'
  Str[14] := DT_TO_STRINGF( dateVar, 'time = %Thh:mm:ss [hour:min:sec]');
                                       // 'time = 12:34:56 [hour:min:sec]'
END_FUNCTION_BLOCK
```

## 5.15 Funkce TOD_TO_STRINGF

Knihovna : *ToStringLib*




Funkce *TOD_TO_STRINGF* převede vstupní proměnnou typu TIME_OF_DAY do proměnné typu STRING.

Popis proměnných :

| | *Proměnná* | *Typ* | *Význam* |
|---|---|---|---|
| | | **VAR_INPUT** | |
| | *in* | TOD | Hodnota, která bude převedena na STRING |
| | *format* | STRING | Formátovací řetězec (viz kap 1.4 ANY_TIME_TO_STRING) |
| | | **TOD_TO_STRINGF** | |
| | *Návratová hodnota* | STRING | Formátovaný výstup |

Příklad programu s voláním funkce *TOD_TO_STRINGF* :

```
FUNCTION_BLOCK Example_TOD_TO_STRINGF
  VAR_OUTPUT
    Str    : ARRAY[1..20] OF STRING[50];
  END_VAR

  Str[1] := TOD_TO_STRINGF( T#12:34:56.789, '%Thh:mm:ss.zzz'); // '12:34:56.789'
  Str[2] := TOD_TO_STRINGF( T#12:34:56.789, '%Thh:mm:ss.zz' ); // '12:34:56.78'
  Str[3] := TOD_TO_STRINGF( T#12:34:56.789, '%Thh:mm:ss.z'  ); // '12:34:56.7'
  Str[4] := TOD_TO_STRINGF( T#12:34:56.789, '%Thh:mm:ss'    ); // '12:34:56'
  Str[5] := TOD_TO_STRINGF( T#12:34:56.789, '%Thh:mm'       ); // '12:34'
  Str[6] := TOD_TO_STRINGF( T#12:34:56.789, '%Thh'          ); // '12'
  Str[7] := TOD_TO_STRINGF( T#12:34:56.789, '%Tmm'          ); // '34'
  Str[8] := TOD_TO_STRINGF( T#12:34:56.789, '%Tss'          ); // '56'
  Str[9] := TOD_TO_STRINGF( T#12:34:56.789, '%Tss.z'        ); // '56.7'
  Str[10]:= TOD_TO_STRINGF( T#12:34:56.789, '%Tss.zz'       ); // '56.78'
  Str[11]:= TOD_TO_STRINGF( T#12:34:56.789, '%Tss.zzz'      ); // '56.789'
  Str[12]:= TOD_TO_STRINGF( T#12:34:56.789, '%Tmm:ss.z'     ); // '34:56.7'
  Str[13]:= TOD_TO_STRINGF( T#12:34:56.789, '%Tmm:ss.zz'    ); // '34:56.78'
  Str[14]:= TOD_TO_STRINGF( T#12:34:56.789, '%Tmm:ss.zzz'   ); // '34:56.789'
  Str[15]:= TOD_TO_STRINGF( T#12:34:56.789, '%s'            ); // '12:34:56.789'
  Str[16]:= TOD_TO_STRINGF( T#12:34:56.789, '%Thh:mm:ss.z'  ); // '12:34:56.7'
  Str[17]:= TOD_TO_STRINGF( T#12:34:56.789, '%Thh/mm/ss.z'  ); // '12/34/56.7'
  Str[18]:= TOD_TO_STRINGF( T#12:34:56.789, 'time = %Thh:mm:ss.z [h:m:s.ms]');
                                             // 'time = 12:34:56.7 [h:m:s.ms]'
  Str[19]:= TOD_TO_STRINGF( T#12:34:56.789, '%Thh:mm:ss.z'  ); // '12:34:56.7'
END_FUNCTION_BLOCK
```

# 6 FUNKČNÍ BLOKY

V knihovně ToStringLib nejsou definovány žádné funkční bloky.

# 7 PŘÍKLAD POUŽITÍ

Předpokládejme, že úkolem PLC je odeslat mailem měřenou teplotu, tlak a vlhkost každý den v 8:00 hod. Potřebujeme tedy sestavit text mailu, do kterého budeme zapisovat aktuální stavy proměnných teploty, tlaku a vlhkosti. Teplota je uložena v proměnné *temp* typu LREAL a program ji bude do textu zapisovat s jedním desetinným místem, tlak je uložen v proměnné *press* typu REAL a program bude do mailu zapisovat hodnotu se dvěmi desetinnými místy a konečně vlhkost je uložena v proměnné *hum* typu UINT a do mailu bude zapsaná jako celé číslo. Dále bude v mailu uveden čas a datum měření, hodnoty proměnných budou doplněny fyzikální jednotkou a textem, ze kterého bude zřejmé o jakou veličinu se jedná. Uvedené zadání lze s použitím funkcí z knihovny ToStringLib naprogramovat například následovně:

```
PROGRAM prgExample
  VAR
    temp        : LREAL;            // temperature
    press       : REAL;             // pressure
    hum         : UINT;             // humidity
    daTi        : DT;               // date and time
    ti          : TIME;             // actual time
    isText      : BOOL;             // flag text is prepared
    isTextEdge  : R_TRIG;
    text        : STRING;           // body of mail
  END_VAR

  ti := GetTime();
  IF ti >= T#08:00:00.0 AND ti < T#08:00:00.5 THEN
    IF NOT isText THEN
      isText := TRUE;
      daTi := GetDateTime();        // actual date and time
      // prepare text of mail
      text := DT_TO_STRINGF(in := daTi, format := '%TYYYY-MM-DD-hh:mm:ss  ') +
              LREAL_TO_STRINGF(in := temp, format := 'temperature: %5.1f°C, ') +
              REAL_TO_STRINGF (in := press, format := 'pressure: %5.2f hPa, ') +
              UINT_TO_STRINGF (in := hum, format := 'humidity: %3u%%');
    END_IF;
  ELSE
    isText := FALSE;
  END_IF;

  isTextEdge(CLK := isText);
  IF isTextEdge.Q THEN
    // send mail (see InternetLib)
    // ...
  END_IF;
END_PROGRAM
```

Výsledkem bude např. následující text:
**'2008-10-02-08:00:00  temperature:  22.5°C, pressure: 1013.25 hPa, humidity: 65%'**