

Knihovna SysLib

TXV 003 48.01
osmé vydání
prosinec 2011
změny vyhrazeny

Historie změn

Datum	Vydání	Popis změn
Únor 2009	1	První vydání, popis odpovídá knihovně SysLib_v17
Březen 2009	2	Vynechány funkce Get_IP_address() a Set_IP_address() Jejich náhradou jsou funkce GetIPAddress() a SetIPAddress() v knihovně ComLib Popis odpovídá knihovně SysLib_v18
Listopad 2009	3	Přidán popis funkcí CIBunitInfo(), SetAddressCIBunit() a ProgramLock() Popis odpovídá knihovně SysLib_v20
Leden 2010	4	Přidán popis funkce Memcmp() a funkčního bloku fbTick() Opraveny konstanty MI2_CIB1 až MI6_CIB2 a doplněny konstanty MI0_CIB1 a MI0_CIB2 Popis odpovídá knihovně SysLib_v22
Květen 2010	5	Přidán popis funkcí SetWebPSW() a VerifyWebPSW() Popis odpovídá knihovně SysLib_v23
Srpen 2010 Únor 2011	6	Přidán popis funkcí SystemDisplayBacklightOn() a SystemDisplayBacklightOff(), které byly přidány do SysLib_v24 Přidán popis funkcí RFunitInfo() a SetAddressRFunit() Popis odpovídá knihovně SysLib_v26
Červenec 2011	7	Přidán popis funkčního bloku fbBondRFunit() Popis odpovídá knihovně SysLib_v27
Prosinec 2011	8	Přidány popisy funkcí MemcpyEx(), MemsetEx(), MemcmpEx(), VerifyWebMAC() a SetWebMAC() Popis odpovídá knihovně SysLib_v29

OBSAH

1 Úvod.....	4
2 Datové typy.....	5
2.1 Typ TIOSystemInfo.....	6
2.2 Typ TmoduleInfo.....	7
2.3 Typ TSYSTEM_S.....	8
2.4 Typ TTecoDateTime.....	10
2.5 Typ TCIBunitState.....	11
2.6 Typ TCIBunitInfo.....	12
2.7 Typ TRFunitState.....	13
2.8 Typ TRFunitInfo.....	14
3 Globální proměnné a konstanty.....	15
4 Funkce	18

4.1 Funkce SetSummerTime.....	19
4.2 Funkce IsSummerTime.....	20
4.3 Funkce SetWinterTime.....	21
4.4 Funkce IsWinterTime.....	22
4.5 Funkce GetDate.....	23
4.6 Funkce GetTime.....	24
4.7 Funkce GetDateTime.....	25
4.8 Funkce GetRTC.....	26
4.9 Funkce SetRTC.....	27
4.10 Funkce TecoDT_TO_DT.....	28
4.11 Funkce DT_TO_TecoDT.....	29
4.12 Funkce IOSystemInfo.....	30
4.13 Funkce ModuleInfo.....	31
4.14 Funkce CIBunitInfo.....	32
4.15 Funkce SetAddressCIBunit.....	34
4.16 Funkce Memcpy.....	37
4.17 Funkce MemcpyEx.....	39
4.18 Funkce Memset.....	41
4.19 Funkce MemsetEx.....	42
4.20 Funkce Memcmp.....	44
4.21 Funkce MemcmpEx.....	45
4.22 Funkce IncreaseMaxCycleTime.....	47
4.23 Funkce VerifyWebPSW.....	48
4.24 Funkce SetWebPSW.....	50
4.25 Funkce VerifyWebMAC.....	52
4.26 Funkce SetWebMAC.....	54
4.27 Funkce ProgramLock.....	56
4.28 Funkce SystemDisplayBacklightOn.....	57
4.29 Funkce SystemDisplayBacklightOff.....	58
4.30 Funkce RFunitInfo.....	59
4.31 Funkce SetAddressRFunit.....	61
5 Funkční bloky.....	65
5.1 Funkční blok fbTick.....	66
5.2 Funkční blok fbBondRFunit.....	68

1 ÚVOD

Knihovny funkcí a funkčních bloků jsou nedílnou součástí instalace programovacího prostředí Mosaic. Z hlediska jejich výstavby je možné knihovny rozdělit na následující typy:

- vestavěné (built-in) knihovny
- standardně dodávané externí knihovny
- uživatelsky definované knihovny

Knihovna může obsahovat deklarace funkcí, funkčních bloků, datových typů a globálních proměnných. Knihovna SysLib patří mezi standardně dodávané knihovny. Následující obrázek ukazuje strukturu knihovny SysLib v prostředí Mosaic



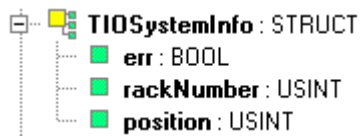
Pokud chceme funkce z knihovny SysLib použít v aplikačním programu PLC, je třeba nejprve přidat tuto knihovnu do projektu. Knihovna je dodávaná jako součást instalace prostředí Mosaic.

2 DATOVÉ TYPY

V knihovně SysLib jsou definovány následující datové typy:

<i>Identifikátor</i>	<i>Typ</i>	<i>Význam</i>
<i>TIOSystemInfo</i>	STRUCT	Struktura nesoucí informaci o stavu IO systému PLC
<i>TmoduleInfo</i>	STRUCT	Struktura s údaji o aktuálním stavu IO modulu
<i>TSYSTEM_S</i>	STRUCT	Struktura umožňující přístup k systémovým registrům (%Sxx)
<i>TTecoDateTime</i>	STRUCT	Struktura s údaji o datumu a času
<i>TCIBunitState</i>	STRUCT	Struktura s údaji o aktuálním stavu CIB jednotky (CFox)
<i>TCIBunitInfo</i>	STRUCT	Struktura s údaji o CIB jednotce (CFox)
<i>TRFunitState</i>	STRUCT	Struktura s údaji o aktuálním stavu RF jednotky (RFox)
<i>TRFunitInfo</i>	STRUCT	Struktura s údaji o RF jednotce (RFox)

2.1 Typ *TIOSystemInfo*

Knihovna : *SysLib*

Typ *TIOSystemInfo* je struktura nesoucí informaci o stavu IO systému PLC, kterou vrací funkce *IOSystemInfo*.

Význam jednotlivých položek struktury *TIOSystemInfo* je následující:

- *err* err = FALSE IO systém PLC je bez chyby
 err = TRUE v IO systému PLC je nějaký problém
- *rackNumber* při err = TRUE udává číslo rámu PLC, kde je umístěn
 modul signalizující nějaký problém
- *position* při err = TRUE udává číslo pozice v rámu PLC, kde je
 umístěn modul signalizující nějaký problém

Viz také Funkce *IOSystemInfo*

2.2 Typ TmoduleInfo

```

TModuleInfo : STRUCT
  ECOM : BOOL
  DATA : BOOL
  DUMMY1 : BOOL
  DUMMY2 : BOOL
  ERR : BOOL
  DEC : BOOL
  OTH : BOOL
  POS : BOOL
  STAT : USINT

```

Knihovna : SysLib

Typ TmoduleInfo je struktura nesoucí informaci o stavu konkrétního IO modulu PLC, kterou vrací funkce ModuleInfo.

Význam jednotlivých položek struktury TmoduleInfo je následující:

- *ECOM* chyba komunikace mezi CPU a IO modulem
FALSE ... komunikace je v pořádku
TRUE ... chyba komunikace
- *DATA* data poskytovaná modulem platná
FALSE ... data nejsou platná
TRUE ... data jsou platná
- *DUMMY1* rezerva
- *DUMMY2* rezerva
- *ERR* chyba IO modulu
FALSE ... IO modul je bez chyby
TRUE ... IO modul hlásí chybu
- *DEC* deklarace IO modulu v programu PLC
FALSE ... modul není v programu PLC deklarován
TRUE ... modul má platnou deklaraci v programu PLC
- *OTH* chyba kontroly typu modulu
FALSE ... typ IO modulu odpovídá deklaraci v programu PLC
TRUE ... typ IO modulu neodpovídá deklaraci v programu PLC
- *POS* přítomnost modulu
FALSE ... v dané pozici není přítomen žádný modul
TRUE ... modul je přítomen
- *STAT* status modulu – výše uvedené proměnné jako 1 byte
(ECOM = STAT.0, ... , POS = STAT.7)

Viz také Funkce ModuleInfo

2.3 Typ *TSYSTEM_S*

Typ *TSYSTEM_S* je struktura umožňující přístup k systémovým registrům PLC. Její definice vypadá následovně:

TYPE

```

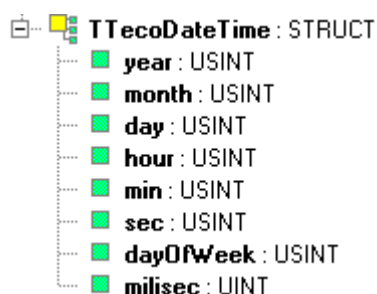
TSYSTEM_S : STRUCT
    S0          : BYTE; // %S0    výsledky aritmetických operací
    S1          : BYTE; // %S1    výsledky logických operací
    S2_0        : BOOL; // %S2.0  stav služebního vstupu SP
    S2_1        : BOOL; // %S2.1  stav služebního vstupu MS
    S2_2        : BOOL; // %S2.2  režim RUN
    S2_3        : BOOL; // %S2.3  teplý restart
    S2_4        : BOOL; // %S2.4  studený restart
    OUTPUTS_ARE_ENABLED : BOOL; // %S2.5  výstupy odblokovány
    S2_6        : BOOL; // %S2.6  první průchod cyklem bez restartu
    CYCLE_TIME_WARNING : BOOL; // %S2.7  překročena první mez doby cyklu
    LAST_CYCLE_TIME_10MS : USINT; // %S3    doba minulého cyklu v 10 ms
    CYCLE_COUNTER : USINT; // %S4    čítač cyklu
    COUNTER_10MS : USINT; // %S5    čítač desítek milisekund
    COUNTER_SECONDS : USINT; // %S6    čítač sekund systémového času
    COUNTER_MINUTES : USINT; // %S7    čítač minut systémového času
    COUNTER_HOURS : USINT; // %S8    čítač hodin systémového času
    COUNTER_DAYS_OF_WEEK : USINT; // %S9    čítač dnů v týdnu
    COUNTER_DAYS_OF_MONTH : USINT; // %S10   čítač dnů v měsíci
    COUNTER_MONTHS : USINT; // %S11   čítač měsíců
    COUNTER_YEARS : USINT; // %S12   čítač roků
    PERIOD_PULSE_100MS : BOOL; // %S13.0 pulz s periodou 100 ms
    PERIOD_PULSE_500MS : BOOL; // %S13.1 pulz s periodou 500 ms
    PERIOD_PULSE_1SEC : BOOL; // %S13.2 pulz s periodou 1 s
    PERIOD_PULSE_10SEC : BOOL; // %S13.3 pulz s periodou 10 s
    PERIOD_PULSE_1MIN : BOOL; // %S13.4 pulz s periodou 1 min
    PERIOD_PULSE_10MIN : BOOL; // %S13.5 pulz s periodou 10 min
    PERIOD_PULSE_1HOUR : BOOL; // %S13.6 pulz s periodou 1 hod
    PERIOD_PULSE_1DAY : BOOL; // %S13.7 pulz s periodou 1 den
    COUNTER_100MS : UINT; // %SW14  čítač v 100m
    COUNTER_1SEC : UINT; // %SW16   čítač v 1s
    COUNTER_10SEC : UINT; // %SW18   čítač v 10s
    R_EDGE_100MS : BOOL; // %S20.0 náběžná hrana 1x za 100 ms
    R_EDGE_500MS : BOOL; // %S20.1 náběžná hrana 1x za 500 ms
    R_EDGE_1SEC : BOOL; // %S20.2 náběžná hrana 1x za 1 s
    R_EDGE_10SEC : BOOL; // %S20.3 náběžná hrana 1x za 10 s
    R_EDGE_1MIN : BOOL; // %S20.4 náběžná hrana 1x za 1 min
    R_EDGE_10MIN : BOOL; // %S20.5 náběžná hrana 1x za 10 min
    R_EDGE_1HOUR : BOOL; // %S20.6 náběžná hrana 1x za 1 hod
    R_EDGE_1DAY : BOOL; // %S20.7 náběžná hrana 1x za 1 den
    F_EDGE_100MS : BOOL; // %S21.0 sestupná hrana 1x za 100 ms
    F_EDGE_500MS : BOOL; // %S21.1 sestupná hrana 1x za 500 ms
    F_EDGE_1SEC : BOOL; // %S21.2 sestupná hrana 1x za 1 s
    F_EDGE_10SEC : BOOL; // %S21.3 sestupná hrana 1x za 10 s
    F_EDGE_1MIN : BOOL; // %S21.4 sestupná hrana 1x za 1 min
    F_EDGE_10MIN : BOOL; // %S21.5 sestupná hrana 1x za 10 min
    F_EDGE_1HOUR : BOOL; // %S21.6 sestupná hrana 1x za 1 hod
    F_EDGE_1DAY : BOOL; // %S21.7 sestupná hrana 1x za 1 den
    LAST_CYCLE_TIME_100US : UINT; // %SW22  doba minulého cyklu v 100 μs
    S24, S25, S26, S27, S28 : BYTE; // %S24,...,%S28 řídicí masky procesů
    S29, S30, S31, S32, S33 : BYTE; // %S29,...,%S33 řídicí masky procesů
    S34 : BYTE; // %S34    interní kód chyby

```

```
BAT_ERR           : BOOL;    // %S35.0 chyba zálohovací baterie
S35_1             : BOOL;    // %S35.1
S35_2             : BOOL;    // %S35.2
S35_3             : BOOL;    // %S35.3
S35_4             : BOOL;    // %S35.4
S35_5             : BOOL;    // %S35.5
IS_SUMMER_TIME    : BOOL;    // %S35.6 indikace letního času
SUMMER_TIME_REQUEST : BOOL;  // %S35.7 žádost o přechod na letní čas
CPU_TEMPERATURE   : USINT;   // %S36 teplota procesorového modulu [°C]
S37              : BYTE;    // %S37 příznaky funkcí systému
S38              : BYTE;    // %S38 verze uživatelského programu
S39              : BYTE;    // %S39 verze uživatelského programu
S40              : BYTE;    // %S40 verze FW systému
S41              : BYTE;    // %S41 verze FW systému
S42              : BYTE;    // %S42 řada CPU
S43              : BYTE;    // %S43 příznaky chování PLC
S44              : BYTE;    // %S44 typ překladače
S45              : BYTE;    // %S45 typ překladače
S46              : BYTE;    // %S46 varovná mez doby cyklu
S47              : BYTE;    // %S47 max. mez doby doby cyklu
S48, S49, S50, S51 : BYTE;  // %S48,...,%S51 úplný kód chyby PLC
COUNTER_1MS       : UDINT;   // %SL52 čítač po 1 ms
END_STRUCT;
END_TYPE
```

Viz také Příručka programátora PLC Tecomat, kap. 5.3 Systémové registry

2.4 Typ *TTecoDateTime*

Knihovna : *SysLib*

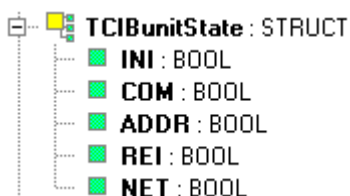
Typ *TTecoDateTime* je struktura s údaji o datumu a času. Tuto strukturu používají konverzní funkce *TecoDT_TO_DT* a *DT_TO_TecoDT*.

Význam jednotlivých položek struktury *TTecoDateTime* je následující:

- *year* rok (poslední dvojčíslí letopočtu)
- *month* měsíc
- *day* den v měsíci
- *hour* hodiny
- *min* minuty
- *sec* sekundy
- *dayOfWeek* den v týdnu (1 = pondělí, 2 = úterý, ... , 7 = neděle)
- *milisec* milisekundy

Viz také Funkce *TecoDT_TO_DT*, Funkce *DT_TO_TecoDT*

2.5 Typ *TCIBunitState*

Knihovna : *SysLib*

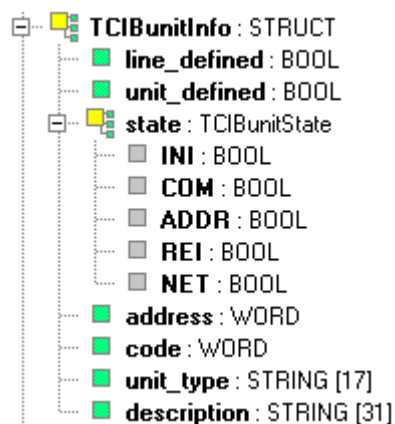
Typ *TCIBunitState* je struktura nesoucí informaci o stavu jednotky na CIB sběrnici, kterou vrací funkce *CIBunitInfo()* jako součást informací o CIB jednotce.

Význam jednotlivých položek struktury *TCIBunitState* je následující:

- *INI* stav inicializace CIB jednotky
 TRUE jednotka CIB je inicializovaná
 FALSE inicializace CIB jednotky se nepodařila
- *COM* stav komunikace s CIB jednotkou
 TRUE komunikace probíhá bez problémů
 FALSE CIB jednotka nekomunikuje
- *ADR* stav adresace CIB jednotky
 TRUE jednotka je úspěšně zaadresovaná
 FALSE přidělení logické adresy se nezdařilo
- *REI* příznak reinicializace CIB jednotky
 TRUE probíhá reinicializace jednotky
 FALSE jednotka je v rutinním provozu
- *NET* stav obsluhy CIB jednotky
 TRUE jednotka je obsluhovaná
 FALSE jednotka není obsluhovaná

Viz také Typ *TCIBunitInfo*, Funkce *CIBunitInfo*

2.6 Typ *TCIBunitInfo*

Knihovna : *SysLib*

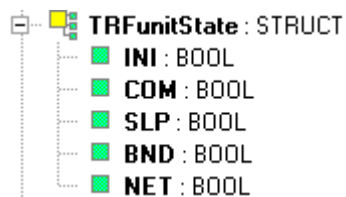
Typ *TCIBunitInfo* je struktura nesoucí informace o jednotce na CIB sběrnici, kterou vrací funkce *CIBunitInfo()*.

Význam jednotlivých položek struktury *TCIBunitState* je následující:

- *line_defined* CIB linka je definovaná v HW konfiguraci PLC
- *unit_defined* CIB jednotka je definovaná v HW konfiguraci PLC
- *state* stav CIB jednotky (viz Typ *TCIBunitState*)
- *address* aktuálně nastavená HW adresa CIB jednotky
- *code* kód CIB jednotky
- *unit_type* typové označení CIB jednotky
- *description* popis CIB jednotky

Viz také Typ *TCIBunitState*, Funkce *CIBunitInfo*

2.7 Typ *TRFunitState*

Knihovna : *SysLib*

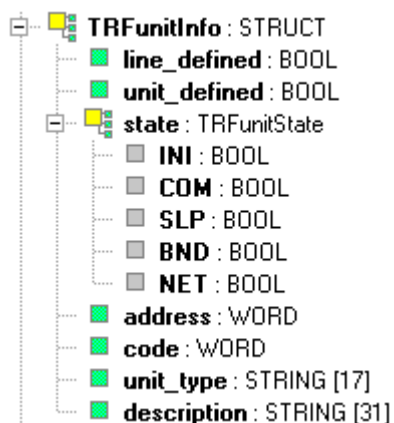
Typ *TRFunitState* je struktura nesoucí informaci o stavu RF jednotky, kterou vrací funkce *RFunitInfo()* jako součást informací o RFox jednotce.

Význam jednotlivých položek struktury *TRFunitState* je následující:

- *INI* stav inicializace RF jednotky
 TRUE jednotka RF je inicializovaná
 FALSE inicializace RF jednotky se nepodařila
- *COM* stav komunikace s RF jednotkou
 TRUE komunikace probíhá bez problémů
 FALSE RF jednotka nekomunikuje
- *SLP* informace o sleep režimu RF jednotky
 TRUE jednotka může přecházet do sleep režimu
 FALSE jednotka je trvale aktivní
- *BND* stav párování RF jednotky
 TRUE RF master považuje jednotku za spárovanou
 FALSE jednotka není spárovaná
- *NET* stav obsluhy RF jednotky
 TRUE RF jednotka je definovaná v aplikačním programu a
 její obsluha je zapnutá
 FALSE jednotka není obsluhovaná

Viz také Typ *TRFunitInfo*, Funkce *RFunitInfo*

2.8 Typ *TRFunitInfo*

Knihovna : *SysLib*

Typ *TRFunitInfo* je struktura nesoucí informace o RF jednotce (RFox), kterou vrací funkce *RFunitInfo()*.

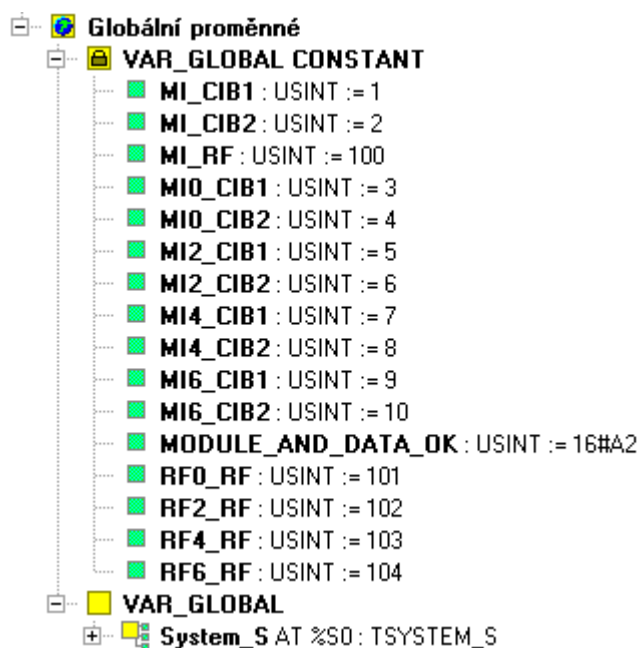
Význam jednotlivých položek struktury *TRFunitState* je následující:

- *line_defined* RF linka je definovaná v HW konfiguraci PLC
- *unit_defined* RF jednotka je definovaná v HW konfiguraci PLC
- *state* stav RF jednotky (viz Typ *TRFunitState*)
- *address* aktuálně nastavená HW adresa RF jednotky
- *code* kód RF jednotky
- *unit_type* typové označení RF jednotky
- *description* popis RF jednotky

Viz také Typ *TRFunitState*, Funkce *RFunitInfo*

3 GLOBÁLNÍ PROMĚNNÉ A KONSTANTY

V knihovně SysLib jsou definovány následující globální proměnné a konstanty:



Konstanta *MODULE_AND_DATA_OK* má hodnotu 16#A2 a lze jí použít ke kontrole stavu IO modulů PLC. Ke každému IO modulu v PLC je přiřazen jeden systémový registr v zóně %S100 až %S227, který říká okamžitý stav IO modulu. Hodnota 16#A2 znamená, že IO modul je v pořádku. Podrobněji viz Příručka programátora PLC Tecomat, kap. 5.3 Systémové registry, stavová zóna periferního systému.

Konstanty *MI_CIB1* až *MI6_CIB2* slouží ke specifikaci CIB linky ve funkcích *CIBunitInfo* a *SetAddressCIBunit*.

Konstanty *MI_RF*, *RF0_RF* až *RF6_RF* slouží ke specifikaci bezdrátové RF sítě ve funkcích *RFunitInfo* a *SetAddressRFunit*.

Význam těchto konstant udává následující tabulka:

Identifikátor	Typ	Hodnota	Význam
<i>MI_CIB1</i>	USINT	1	CIB linka na základním modulu Foxtrot (interní modul MI2-01M nebo CF-1140)
<i>MI_CIB2</i>	USINT	2	Rezerva - v systémech Foxtrot není tato linka použita
<i>MI0_CIB1</i>	USINT	3	Externí CIB master MI2-02M nebo CF-1141, adresa 0, linka CIB1
<i>MI0_CIB2</i>	USINT	4	Externí CIB master MI2-02M nebo CF-1141, adresa 0, linka CIB1
<i>MI2_CIB1</i>	USINT	5	Externí CIB master MI2-02M nebo CF-1141,

			adresa 2, linka CIB1
<i>MI2_CIB2</i>	USINT	6	Externí CIB master MI2-02M nebo CF-1141, adresa 2, linka CIB2
<i>MI4_CIB1</i>	USINT	7	Externí CIB master MI2-02M nebo CF-1141 , adresa 4, linka CIB1
<i>MI4_CIB2</i>	USINT	8	Externí CIB master MI2-02M nebo CF-1141 , adresa 4, linka CIB2
<i>MI6_CIB1</i>	USINT	9	Externí CIB master MI2-02M nebo CF-1141 , adresa 6, linka CIB1
<i>MI6_CIB2</i>	USINT	10	Externí CIB master MI2-02M nebo CF-1141 , adresa 6, linka CIB2
<i>MI_RF</i>	USINT	100	Interní RF master systému Foxtrot (interní modul RF-1130)
<i>RF0_RF</i>	USINT	101	Externí RF master RF-1131, adresa 0
<i>RF2_RF</i>	USINT	102	Externí RF master RF-1131, adresa 2
<i>RF4_RF</i>	USINT	103	Externí RF master RF-1131, adresa 4
<i>RF6_RF</i>	USINT	104	Externí RF master RF-1131, adresa 6

Globální proměnná *System_S* má strukturu *TSYSTEM_S* a umožňuje přístup k systémovým registrům PLC. Význam jednotlivých položek je následující:

<i>System_S.S0</i>	BYTE	%S0	výsledky aritmetických operací
<i>System_S.S1</i>	BYTE	%S1	výsledky logických operací
<i>System_S.S2_0</i>	BOOL	%S2.0	stav služebního vstupu SP
<i>System_S.S2_1</i>	BOOL	%S2.1	stav služebního vstupu MS
<i>System_S.S2_2</i>	BOOL	%S2.2	režim RUN
<i>System_S.S2_3</i>	BOOL	%S2.3	teplý restart
<i>System_S.S2_4</i>	BOOL	%S2.4	studený restart
<i>System_S.OUTPUTS_ARE_ENABLED</i>	BOOL	%S2.5	výstupy odblokovány
<i>System_S.S2_6</i>	BOOL	%S2.6	první průchod cyklem bez restartu
<i>System_S.CYCLE_TIME_WARNING</i>	BOOL	%S2.7	překročena první mez doby cyklu
<i>System_S.LAST_CYCLE_TIME_10MS</i>	USINT	%S3	doba minulého cyklu v 10 ms
<i>System_S.CYCLE_COUNTER</i>	USINT	%S4	čítač cyklu
<i>System_S.COUNTER_10MS</i>	USINT	%S5	čítač desítek milisekund
<i>System_S.COUNTER_SECONDS</i>	USINT	%S6	čítač sekund systémového času
<i>System_S.COUNTER_MINUTES</i>	USINT	%S7	čítač minut systémového času
<i>System_S.COUNTER_HOURS</i>	USINT	%S8	čítač hodin systémového času
<i>System_S.COUNTER_DAYS_OF_WEEK</i>	USINT	%S9	čítač dnů v týdnu
<i>System_S.COUNTER_DAYS_OF_MONTH</i>	USINT	%S10	čítač dnů v měsíci
<i>System_S.COUNTER_MONTHS</i>	USINT	%S11	čítač měsíců
<i>System_S.COUNTER_YEARS</i>	USINT	%S12	čítač roků
<i>System_S.PERIOD_PULSE_100MS</i>	BOOL	%S13.0	pulz s periodou 100 ms
<i>System_S.PERIOD_PULSE_500MS</i>	BOOL	%S13.1	pulz s periodou 500 ms
<i>System_S.PERIOD_PULSE_1SEC</i>	BOOL	%S13.2	pulz s periodou 1 s
<i>System_S.PERIOD_PULSE_10SEC</i>	BOOL	%S13.3	pulz s periodou 10 s
<i>System_S.PERIOD_PULSE_1MIN</i>	BOOL	%S13.4	pulz s periodou 1 min
<i>System_S.PERIOD_PULSE_10MIN</i>	BOOL	%S13.5	pulz s periodou 10 min
<i>System_S.PERIOD_PULSE_1HOUR</i>	BOOL	%S13.6	pulz s periodou 1 hod
<i>System_S.PERIOD_PULSE_1DAY</i>	BOOL	%S13.7	pulz s periodou 1 den
<i>System_S.COUNTER_100MS</i>	UINT	%SW14	čítač v 100m

System_S.COUNTER_1SEC	UINT	%SW16	čítač v 1s
System_S.COUNTER_10SEC	UINT	%SW18	čítač v 10s
System_S.R_EDGE_100MS	BOOL	%S20.0	náběžná hrana 1x za 100 ms
System_S.R_EDGE_500MS	BOOL	%S20.1	náběžná hrana 1x za 500 ms
System_S.R_EDGE_1SEC	BOOL	%S20.2	náběžná hrana 1x za 1 s
System_S.R_EDGE_10SEC	BOOL	%S20.3	náběžná hrana 1x za 10 s
System_S.R_EDGE_1MIN	BOOL	%S20.4	náběžná hrana 1x za 1 min
System_S.R_EDGE_10MIN	BOOL	%S20.5	náběžná hrana 1x za 10 min
System_S.R_EDGE_1HOUR	BOOL	%S20.6	náběžná hrana 1x za 1 hod
System_S.R_EDGE_1DAY	BOOL	%S20.7	náběžná hrana 1x za 1 den
System_S.F_EDGE_100MS	BOOL	%S21.0	sestupná hrana 1x za 100 ms
System_S.F_EDGE_500MS	BOOL	%S21.1	sestupná hrana 1x za 500 ms
System_S.F_EDGE_1SEC	BOOL	%S21.2	sestupná hrana 1x za 1 s
System_S.F_EDGE_10SEC	BOOL	%S21.3	sestupná hrana 1x za 10 s
System_S.F_EDGE_1MIN	BOOL	%S21.4	sestupná hrana 1x za 1 min
System_S.F_EDGE_10MIN	BOOL	%S21.5	sestupná hrana 1x za 10 min
System_S.F_EDGE_1HOUR	BOOL	%S21.6	sestupná hrana 1x za 1 hod
System_S.F_EDGE_1DAY	BOOL	%S21.7	sestupná hrana 1x za 1 den
System_S.LAST_CYCLE_TIME_100US	UINT	%SW22	doba minulého cyklu v 100 μs
System_S.S24	BYTE	%S24	řídící masky procesů
System_S.S25	BYTE	%S25	řídící masky procesů
System_S.S26	BYTE	%S26	řídící masky procesů
System_S.S27	BYTE	%S27	řídící masky procesů
System_S.S28	BYTE	%S28	řídící masky procesů
System_S.S29	BYTE	%S29	řídící masky procesů
System_S.S30	BYTE	%S30	řídící masky procesů
System_S.S31	BYTE	%S31	řídící masky procesů
System_S.S32	BYTE	%S32	řídící masky procesů
System_S.S33	BYTE	%S33	řídící masky procesů
System_S.S34	BYTE	%S34	interní kód chyby
System_S.BAT_ERR	BOOL	%S35.0	chyba zálohovací baterie
System_S.S35_1	BOOL	%S35.1	
System_S.S35_2	BOOL	%S35.2	
System_S.S35_3	BOOL	%S35.3	
System_S.S35_4	BOOL	%S35.4	
System_S.S35_5	BOOL	%S35.5	
System_S.IS_SUMMER_TIME	BOOL	%S35.6	indikace letního času
System_S.SUMMER_TIME_REQUEST	BOOL	%S35.7	žádost o přechod na letní čas
System_S.CPU_TEMPERATURE	USINT	%S36	teplota procesorového modulu [°C]
System_S.S37	BYTE	%S37	příznaky funkcí systému
System_S.S38	BYTE	%S38	verze uživatelského programu
System_S.S39	BYTE	%S39	verze uživatelského programu
System_S.S40	BYTE	%S40	verze FW systému
System_S.S41	BYTE	%S41	verze FW systému
System_S.S42	BYTE	%S42	řada CPU
System_S.S43	BYTE	%S43	příznaky chování PLC
System_S.S44	BYTE	%S44	typ překladače
System_S.S45	BYTE	%S45	typ překladače
System_S.S46	BYTE	%S46	varovná mez doby cyklu
System_S.S47	BYTE	%S47	max. mez doby doby cyklu
System_S.S48	BYTE	%S48	úplný kód chyby PLC
System_S.S49	BYTE	%S49	úplný kód chyby PLC
System_S.S50	BYTE	%S50	úplný kód chyby PLC
System_S.S51	BYTE	%S51	úplný kód chyby PLC
System_S.COUNTER_1MS	UDINT	%SL52	čítač po 1 ms

Viz také Příručka programátora PLC Tecomat, kap. 5.3 Systémové registry

4 FUNKCE

Knihovna SysLib obsahuje následující funkce:

<i>Funkce</i>	<i>Popis</i>
<i>SetSummerTime</i>	Funkce nastaví požadavek na automatický přechod mezi letním a zimním časem
<i>IsSummerTime</i>	Funkce otestuje je-li aktuálně nastaven letní čas
<i>SetWinterTime</i>	Funkce vypne požadavek na automatický přechod mezi letním a zimním časem
<i>IsWinterTime</i>	Funkce otestuje je-li aktuálně nastaven zimní čas
<i>GetDate</i>	Funkce vrátí aktuální datum
<i>GetTime</i>	Funkce vrátí aktuální čas
<i>GetDateTime</i>	Funkce vrátí aktuální datum a čas
<i>GetRTC</i>	Funkce načte datum a čas z RTC obvodu
<i>SetRTC</i>	Funkce nastaví nový datum a čas do RTC obvodu
<i>TecoDT_TO_DT</i>	Funkce převede datum a čas ze struktury TTecoDateTime do IEC formátu DATE_AND_TIME
<i>DT_TO_TecoDT</i>	Funkce převede datum a čas z IEC formátu DATE_AND_TIME do struktury TTecoDateTime
<i>IOSystemInfo</i>	Funkce vrátí celkovou informaci o stavu I/O systému PLC
<i>ModuleInfo</i>	Funkce vrátí informace o aktuálním stavu jednoho I/O modulu
<i>CIBunitInfo</i>	Funkce vrátí informace o aktuálním stavu jedné CIB jednotky
<i>SetAddressCIBunit</i>	Funkce nastaví novou HW adresu CIB jednotky
<i>Memcpy, MemcpyEx</i>	Funkce zkopíruje blok paměti
<i>Memset, MemsetEx</i>	Funkce vyplní blok paměti zadanou konstantou
<i>Memcmp, MemcmpEx</i>	Funkce porovná dva bloky paměti
<i>IncreaseMaxCycleTime</i>	Funkce zvýší jednorázově hlídanou dobu cyklu PLC
<i>SetWebPSW</i>	Funkce nastaví nové jméno a heslo pro přístup na web stránky uložené v PLC
<i>VerifyWebPSW</i>	Funkce otestuje, zda-li existuje zadané jméno a heslo
<i>SetWebMAC</i>	Funkce nastaví novou MAC adresu pro přístup na web stránky uložené v PLC
<i>VerifyWebMAC</i>	Funkce otestuje, zda-li existuje zadaná MAC adresa
<i>ProgramLock</i>	Funkce zapne ochranu aplikačního programu PLC, takže jej nelze dekompileovat
<i>SystemDisplayBacklightOn</i>	Funkce rozsvítí podsvícení LCD displeje na základním modulu PLC

<i>Funkce</i>	<i>Popis</i>
<i>SystemDisplayBacklightOff</i>	Funkce zhasne podsvícení LCD displeje na základním modulu PLC
<i>RFunitInfo</i>	Funkce získá informace o stavu jednotky na rádiové síti RFox
<i>SetAddressRFunit</i>	Funkce nastaví novou HW adresu RF jednotky

4.1 Funkce SetSummerTime

Knihovna : SysLib

 **SetSummerTime** : BOOL


Funkce *SetSummerTime* nastaví požadavek na automatický přechod mezi letním a zimním časem. Funkce *SetSummerTime* nemá žádné vstupní parametry. Výstupem této funkce je hodnota TRUE, pokud se podaří požadavek nastavit. V tom případě začne centrální jednotka hlídat přechod ze zimního času na letní a obráceně. Přechod ze zimního na letní čas je proveden poslední neděli v březnu tak, že se čas posune ve 02:00 o hodinu na 03:00. Přechod z letního na zimní čas se pak provede poslední neděli v říjnu, kdy je čas ve 03:00 posunut zpět na 02:00. Pokud není PLC systém v uvedené hodiny zapnutý, přechod se provede v nejbližší zapínací sekvenci.

Příklad programu s voláním funkce SetSummerTime :

```

PROGRAM SummerTimeExample
VAR
    tmp      : BOOL;
    message  : STRING;
END_VAR

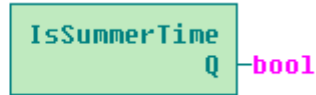
// request for automatic change between summer and winter time
tmp := SetSummerTime();

IF IsSummerTime() THEN
    message := 'Now is summer time';
ELSE
    message := 'Now is winter time';
END_IF;
END_PROGRAM

```

Viz také Funkce IsSummerTime, Funkce SetWinterTime, Funkce IsWinterTime

4.2 Funkce *IsSummerTime*

Knihovna : *SysLib*..  **IsSummerTime** : **BOOL**

Funkce *IsSummerTime* otestuje je-li aktuálně nastaven letní čas. Funkce *IsSummerTime* nemá žádné vstupní parametry. Funkce vrací hodnotu TRUE je-li letní čas, je-li zimní čas tak vrací hodnotu FALSE.

Příklad programu s voláním funkce *IsSummerTime* :

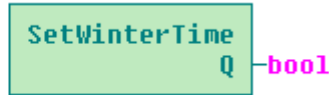
```
PROGRAM SummerTimeExample
VAR
  tmp      : BOOL;
  message  : STRING;
END_VAR

// request for automatic change between summer and winter time
tmp := SetSummerTime();

IF IsSummerTime() THEN
  message := 'Now is summer time';
ELSE
  message := 'Now is winter time';
END_IF;
END_PROGRAM
```

Viz také Funkce *SetSummerTime*, Funkce *SetWinterTime*, Funkce *IsWinterTime*

4.3 Funkce *SetWinterTime*

Knihovna : *SysLib*·  **SetWinterTime** : **BOOL**

Funkce *SetWinterTime* vypne požadavek na automatický přechod mezi letním a zimním časem. Funkce *SetWinterTime* nemá žádné vstupní parametry. Výstupem této funkce je hodnota **TRUE**, pokud se podaří požadavek na automatický přechod času vypnout.

Příklad programu s voláním funkce *SetWinterTime* :

```
PROGRAM ExampleWinterTime
VAR
  tmp      : BOOL;
  message  : STRING;
END_VAR

// no automatic change between summer and winter time
tmp := SetWinterTime();

IF IsWinterTime() THEN
  message := 'Now is winter time';
ELSE
  message := 'Now is summer time';
END_IF;
END_PROGRAM
```

Viz také Funkce *SetSummerTime*, Funkce *IsWinterTime*, Funkce *IsWinterTime*

4.4 Funkce *IsWinterTime*

Knihovna : *SysLib* **IsWinterTime** : **BOOL**

Funkce *IsWinterTime* otestuje je-li aktuálně nastaven zimní čas. Funkce *IsWinterTime* nemá žádné vstupní parametry. Funkce vrací hodnotu **TRUE** je-li zimní čas, je-li letní čas tak vrací hodnotu **FALSE**.

Příklad programu s voláním funkce *IsWinterTime* :

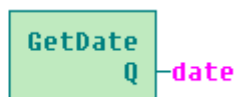
```
PROGRAM ExampleWinterTime
VAR
  tmp      : BOOL;
  message  : STRING;
END_VAR

// no automatic change between summer and winter time
tmp := SetWinterTime();

IF IsWinterTime() THEN
  message := 'Now is winter time';
ELSE
  message := 'Now is summer time';
END_IF;
END_PROGRAM
```

Viz také Funkce *SetWinterTime*, Funkce *SetSummerTime*, Funkce *IsSummerTime*

4.5 Funkce *GetDate*

Knihovna : *SysLib*·  **GetDate** : DATE

Funkce *GetDate* vrátí aktuální datum. Funkce *GetDate* nemá žádné vstupní parametry. Datum, který funkce vrací, je nastavován na začátku každého cyklu PLC. Během cyklu se jeho hodnota nemění.

Příklad programu s voláním funkce *GetDate* :

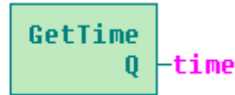
```
PROGRAM GetDateExample
VAR
    presentDate : DATE;
END_VAR

presentDate := GetDate();
IF presentDate = D#2008-12-24 OR
   presentDate = D#2009-12-24 OR
   presentDate = D#2010-12-24
THEN
    message := 'Today is Christmas Eve';
END_IF;
END_PROGRAM
```

Viz také Funkce *GetDateTime*, Funkce *GetTime*

4.6 Funkce GetTime

Knihovna : SysLib

 **GetTime** : TIME


Funkce *GetTime* vrátí aktuální čas PLC. Funkce *GetTime* nemá žádné vstupní parametry. Čas, který funkce vrací, je nastavován na začátku každého cyklu PLC. Během cyklu se jeho hodnota nemění.

Příklad programu s voláním funkce *GetTime* :

```
PROGRAM GetTimeExample
VAR
    timePLC    : TIME;
    greeting   : STRING;
END_VAR

timePLC := GetTime();
IF timePLC > T#06:00:00 AND timePLC < T#12:00:00 THEN
    greeting := 'Good morning';
ELSE
    IF timePLC >= T#12:00:00 AND timePLC < T#18:00:00 THEN
        greeting := 'Good afternoon';
    ELSE
        IF timePLC < T#23:59:59 THEN
            greeting := 'Good evening';
        ELSE
            greeting := 'Good night';
        END_IF;
    END_IF;
END_IF;
END_PROGRAM
```

Viz také Funkce GetDate, Funkce GetDateTime

4.7 Funkce *GetDateTime*

Knihovna : *SysLib*
 **GetDateTime** : DATE_AND_TIME


Funkce *GetDateTime* vrátí aktuální datum a čas PLC. Funkce *GetDateTime* nemá žádné vstupní parametry. Čas a datum, který funkce vrací, je nastavován na začátku každého cyklu PLC. Během cyklu se jeho hodnota nemění.

Příklad programu s voláním funkce *GetDateTime* :

```
PROGRAM GetDateTimeExample
VAR
    dateTimePLC    : DATE_AND_TIME;
    dateTime       : TTecoDateTime;
    dayOfWeek      : STRING;
END_VAR

dateTimePLC := GetDateTime();
// conversion DATE_AND_TIME to struct TTecoDateTime
dateTime := DT_TO_TecoDT(IEC_DT := dateTimePLC);
CASE dateTime.dayOfWeek OF
    1 : dayOfWeek := 'Monday';
    2 : dayOfWeek := 'Tuesday';
    3 : dayOfWeek := 'Wednesday';
    4 : dayOfWeek := 'Thursday';
    5 : dayOfWeek := 'Friday';
    6 : dayOfWeek := 'Saturday';
    7 : dayOfWeek := 'Sunday';
END_CASE;
END_PROGRAM
```

Viz také Funkce *GetDate*, Funkce *GetTime*, Funkce *GetRTC*

4.8 Funkce *GetRTC*

Knihovna : *SysLib* **GetRTC** : DATE_AND_TIME

Funkce *GetRTC* načte datum a čas přímo z RTC obvodu v PLC. Funkce *GetRTC* nemá žádné vstupní parametry. Vzhledem k tomu, že se čas v RTC obvodu mění průběžně, mohou dvě volání funkce *GetRTC* ve stejném cyklu vrátit různou hodnotu. Funkce *GetRTC* tedy vrací datum a čas jaký byl v okamžiku volání této funkce (na rozdíl od funkce *GetDateTime*, která vrací datum a čas jaký byl na začátku cyklu, ve kterém je funkce zavolána).

Příklad programu s voláním funkce *GetRTC* :

```
PROGRAM GetRTCExample
VAR
    dateTimeRTC    : DATE_AND_TIME;
    message         : STRING;
END_VAR

dateTimeRTC := GetRTC();
message     := 'RTC date and time : ' + DT_TO_STRING(dateTimeRTC);
END_PROGRAM
```

Viz také Funkce *GetDateTime*, Funkce *SetRTC*

4.9 Funkce SetRTC

Knihovna : SysLib



Funkce *SetRTC* nastaví nový datum a čas do RTC obvodu. Vstupním parametrem je nový čas a datum. Funkce vrátí TRUE, pokud se podaří nastavit nový čas.

Jak ukazuje následující příklad, funkci *SetRTC* je třeba zavolat pouze jednou, nejlépe na hranu proměnné, která představuje požadavek na zápis nového času a datumu. Hodnota nového času a datumu je pak typicky nastavována z operátorského panelu.

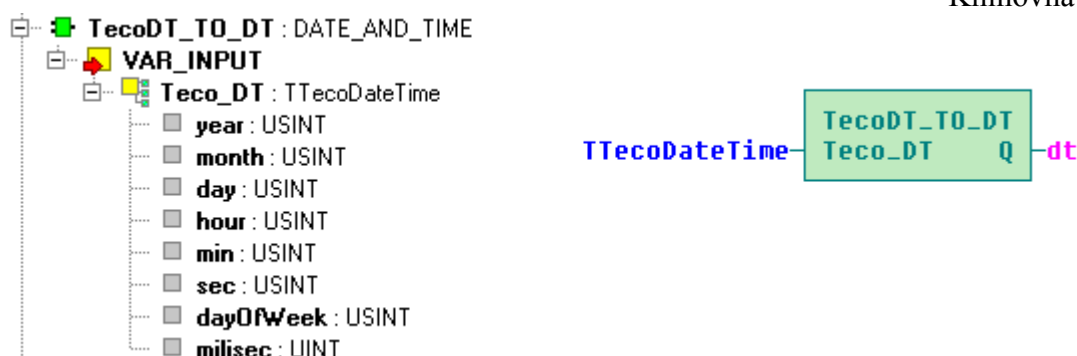
Příklad programu s voláním funkce *SetRTC* :

```
PROGRAM SetRTCExample
VAR
  newDateTimeRTC    : DT;
  setTime           : R_TRIG;
  setNewTime        : BOOL;
  tmp               : BOOL;
END_VAR

setTime( CLK := setNewTime);
IF setTime.Q THEN
  tmp := SetRTC( PDT := newDateTimeRTC);
END_IF;
END_PROGRAM
```

Viz také Funkce *GetRTC*, Funkce *GetDateTime*

4.10 Funkce *TecoDT_TO_DT*

Knihovna : *SysLib*

Funkce *TecoDT_TO_DT* převede datum a čas ze struktury *TTecoDateTime* do IEC formátu *DATE_AND_TIME*.

Příklad programu s voláním funkce *TecoDT_TO_DT* :

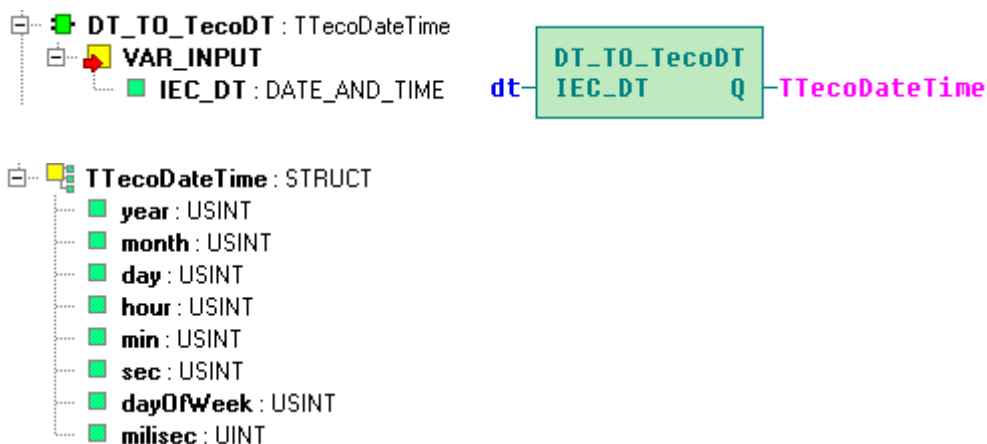
```
PROGRAM TecoDT_TO_DT_example
VAR
    dateTimePLC      : DATE_AND_TIME;
    dateTime          : TTecoDateTime;
END_VAR

dateTime.year       := 09;
dateTime.month      := 11;
dateTime.day        := 22;
dateTime.hour       := 12;
dateTime.min        := 34;
dateTime.sec        := 56;
dateTime.milisec    := 500;
dateTimePLC         := TecoDT_TO_DT( dateTime);
END_PROGRAM
```

Viz také Typ *TTecoDateTime*, Funkce *DT_TO_TecoDT*

4.11 Funkce DT_TO_TecoDT

Knihovna : SysLib



Funkce *DT_TO_TecoDT* převede datum a čas z IEC formátu *DATE_AND_TIME* do struktury *TTecoDateTime*. Výhodou této struktury je, že lze samostatně pracovat pouze s některými položkami času a datumu, jak ukazuje následující příklad.

Příklad programu s voláním funkce *DT_TO_TecoDT*:

```

PROGRAM DT_TO_TecoDT_example
VAR
    dateTimePLC : DATE_AND_TIME;
    dateTime    : TTecoDateTime;
    message     : STRING;
END_VAR

dateTimePLC := GetDateTime();
dateTime    := DT_TO_TecoDT( dateTimePLC);
IF dateTime.month = 05 AND dateTime.day = 30 THEN
    message := 'Birthday of my wife! Do not forget to buy flowers!';
END IF;
END_PROGRAM

```

Viz také Typ *TTecoDateTime*, Funkce *TecoDT_TO_DT*

4.12 Funkce *IOSystemInfo*

Knihovna : *SysLib*

Funkce *IOSystemInfo* vrátí celkovou informaci o stavu I/O systému PLC a je určena pro případy, kdy aplikace využívá možnost vyndávání I/O modulů za chodu resp. možnost ignorovat chybu IO modulu.

Funkce *IOSystemInfo* nemá žádné vstupní parametry. Výstupem funkce je struktura typu *TIOSystemInfo*. Funkce zkontroluje stav IO systému PLC podle toho, co je zadáno v HW konfiguraci PLC. V případě, že je IO systém v pořádku, návratová hodnota výstupu *err* je FALSE a výstupy *rackNumber* a *position* nemají význam. Pokud je v IO systému nějaký problém (např. chybí IO modul požadovaný v HW konfiguraci) tak výstup *err* je nastaven na TRUE, výstupní proměnná *rackNumber* udává číslo rámu a proměnná *position* říká pozici v rámu, na které se našel problém. Pro bližší specifikaci problému lze použít funkci *ModuleInfo*.

Příklad programu s voláním funkce *IOSystemInfo* :

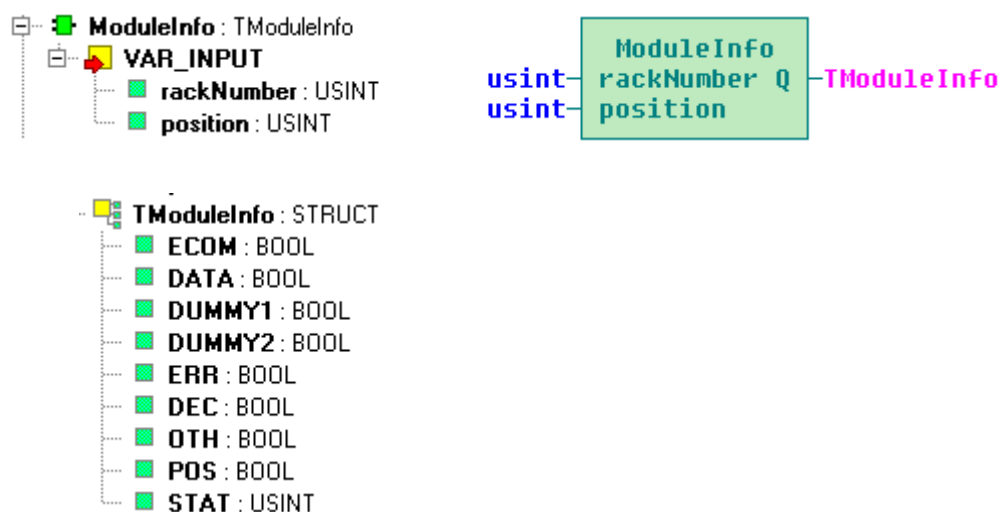
```
PROGRAM IOSystemInfoExample
  VAR
    IO_Status      : TIOSystemInfo;
    ModuleStatus   : TModuleInfo;
  END_VAR

  // check IO system PLC
  IO_Status := IOSystemInfo();
  IF IO_Status.err THEN
    // any modul has problem
    ModuleStatus := ModuleInfo( IO_Status.rackNumber, IO_Status.position);
  END_IF;
END_PROGRAM
```

Viz také Typ *TIOSystemInfo*, Typ *TmoduleInfo*, Funkce *ModuleInfo*

4.13 Funkce ModuleInfo

Knihovna : SysLib



Funkce *ModuleInfo* vrátí informace o aktuálním stavu jednoho I/O modulu. Vstupní parametr *rackNumber* udává číslo rámu a parametr *position* říká číslo pozice, pro kterou se vrátí informace o modulu. Funkce vrací strukturu TModuleInfo.

Příklad programu s voláním funkce *ModuleInfo* :

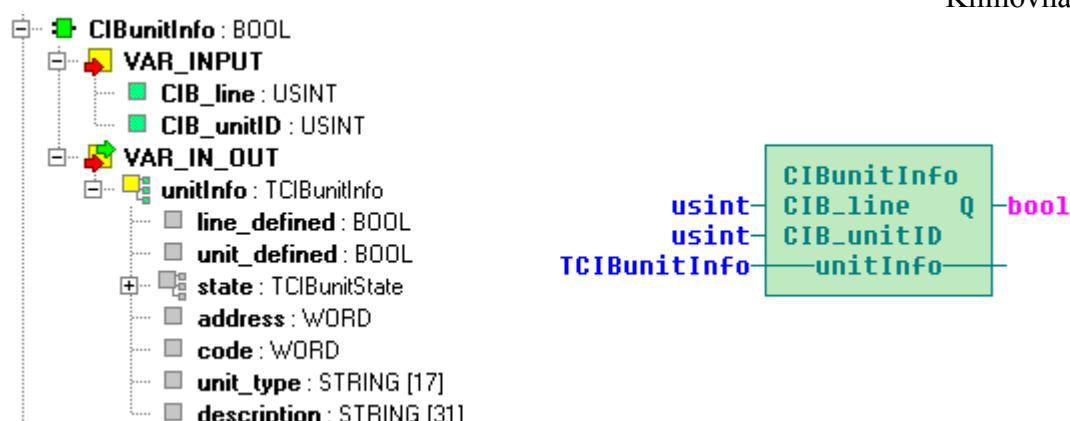
```
PROGRAM ModuleInfoExample
VAR
  IO_Status      : TIOSystemInfo;
  ModuleStatus   : TModuleInfo;
END_VAR

// check IO system PLC
IO_Status := IOSystemInfo();
IF IO_Status.err THEN
  // any modul has problem
  ModuleStatus := ModuleInfo( IO_Status.rackNumber, IO_Status.position);
END_IF;
END_PROGRAM
```

Viz také Typ TIOSystemInfo, Typ TmoduleInfo, Funkce IOSystemInfo

4.14 Funkce CIBunitInfo

Knihovna : SysLib



Funkce *CIBunitInfo* slouží k získání informace o aktuálním stavu jedné jednotky na CIB sběrnici. Vstupní parametr *CIB_line* specifikuje CIB linku (viz konstanty *MI_CIB1* až *MI6_CIB2*) a parametr *CIB_unitID* říká číslo pozice, pro kterou se vrátí informace o CIB jednotce. Funkce vrací hodnotu TRUE, pokud se podaří informace o CIB jednotce získat. Získané informace jsou uloženy v proměnné dané parametrem *unitInfo*. Pokud není CIB jednotka deklarovaná v HW konfiguraci PLC, funkce *CIBunitInfo* vrací FALSE.

Tato funkce je podporovaná v centrálních jednotkách Foxtrot od v5.1. Funkce je zařazena do knihovny SysLib od v19.

Příklad programu s voláním funkce *CIBunitInfo* :

```
PROGRAM CIBunitInfoExample
VAR
  CIB_unit1, CIB_unit2 : TCIBunitInfo;
  result1, result2      : BOOL;
END_VAR

result1 := CIBunitInfo( CIB_line := MI_CIB1, CIB_unitID := 1,
                        unitInfo := CIB_unit1);
result2 := CIBunitInfo( CIB_line := MI_CIB1, CIB_unitID := 2,
                        unitInfo := CIB_unit2);
END_PROGRAM
```

Uvedený program testuje stav prvních dvou CIB jednotek připojených na CIB linku základního modulu systému Foxtrot. Jednotky jsou v programu PLC nastaveny tak, jak ukazuje obrázek dialogu Správce jednotek / zařízení.

Další obrázek pak ukazuje hodnoty proměnné *CIB_unit1*, které odpovídají stavu první definované CIB jednotky. Hodnoty odpovídají situaci, kdy je CIB jednotka v programu PLC definovaná, ale nekomunikuje (např. protože je odpojená nebo má jinou HW adresu, než je nastavená ve výše uvedeném dialogu).

Pokud by chyběla deklarace CIB jednotek v programu, funkce *CIBunitInfo* bude vracet FALSE a proměnná *CIB_unit1* bude vynulovaná.

Správce jednotek/zařízení

MI | CIB1 | CIB2

ID	HW adresa	ID CIB sítě	Typ jednotky/...	Stav	Název jednotky/zařízení/vstupu/výstup
1	0001	1	RCM2-1	-	displej pokoj
2	5B0C	1	RCM2-1	-	displej chodba

Nastavení

Výběr jednotky/zařízení: [↑] [↓]

Název jednotky: [displej pokoj]

HW adresa jednotky: [0001]

☒ Rozšířené nastavení

Visualizace vstupů

☐ Exportovat pro vizualizaci

Pojmenování / alias: [displej_pokoj_IN]

Visualizace výstupů

☐ Exportovat pro vizualizaci

Pojmenování / alias: [displej_pokoj_OUT]

☐ Zobrazit všechna zařízení všech jednotek

Přidat jednotku Vymazat jednotku Vymazat vše Načíst konfiguraci z CPU

☐ Bázová adresa zón [0] OK Zrušit Nápověda

Zprávy 1 | Zprávy 2 | Symbols | Breakpoint list | Data

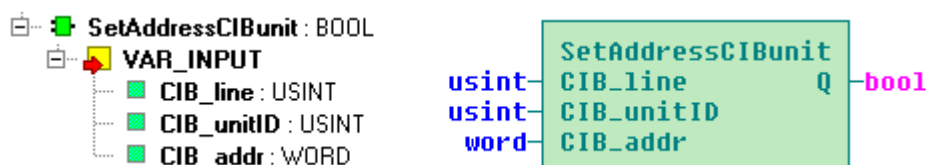
5 Banka3

Jméno	Typ	Hodnota
tstCIBunit.CIB_unit1	TCIBunitInfo	
line_defined	bool	1
unit_defined	bool	1
state	TCIBunitState	
INI	bool	0
COM	bool	0
ADDR	bool	0
DUMMY3	bool	0
REI	bool	1
DUMMY5	bool	0
DUMMY6	bool	0
NET	bool	1
address	word	\$0001
code	word	\$0C55
unit_type [0..16]	string	'RCM2-1'
description [0..32]	string	'displej pokoj'

Viz také Typ TCIBunitState, Typ TCIBunitInfo

4.15 Funkce SetAddressCIBunit

Knihovna : SysLib



Funkce *SetAddressCIBunit* nastaví novou HW adresu CIB jednotky. To je možné využít například v situaci, kdy chceme používat stejný PLC program pro více aplikací, které se mezi sebou liší HW adresou CIB jednotek.

Vstupní parametr *CIB_line* specifikuje CIB linku (viz konstanty *MI_CIB1* až *MI6_CIB2*) a parametr *CIB_unitID* říká, pro kterou CIB jednotku bude nastavena nová adresa. Funkce vrací hodnotu TRUE, pokud se podaří adresu CIB jednotky nastavit.

Tato funkce je podporovaná v centrálních jednotkách Foxtrot od v5.1. Funkce je zařazena do knihovny SysLib od v19.

Příklad programu s voláním funkce *SetAddressCIBunit* :

```

VAR_GLOBAL CONSTANT
    NUM_CIB_UNIT : USINT := 4;
END_VAR

TYPE
    TcustomCIB : STRUCT
        valid    : BOOL;
        address  : WORD;
        code     : WORD;
    END_STRUCT;
END_TYPE

VAR_GLOBAL RETAIN
    customCIB : ARRAY[1..NUM_CIB_UNIT] OF TcustomCIB;
END_VAR

PROGRAM TestCIB
    VAR
        i          : USINT;
        CIB_unit   : ARRAY[1..NUM_CIB_UNIT] OF TCIBunitInfo;
        id         : USINT;
        newAdr     : WORD;
        setAdr     : BOOL;
        result     : BOOL;
    END_VAR

    // read info about CIB units and set customer address
    FOR i := 1 TO NUM_CIB_UNIT DO
        IF CIBunitInfo( CIB_line := MI_CIB1, CIB_unitID := i,
            unitInfo := CIB_unit[i]) THEN
            IF not customCIB[i].valid OR customCIB[i].code <> CIB_unit[i].code THEN
                customCIB[i].address := CIB_unit[i].address;
                customCIB[i].code    := CIB_unit[i].code;
                customCIB[i].valid   := TRUE;
            ELSE

```

```

    IF CIB_unit[i].address <> customCIB[i].address THEN
        SetAddressCIBunit(CIB_line := MI_CIB1, CIB_unitID := i,
            CIB_addr := customCIB[i].address);
    END_IF;
END_IF;
END_IF;
END_FOR;

// set new customer address of CIB unit
IF setAdr THEN
    IF id > 0 AND id < NUM_CIB_UNIT+1 THEN
        IF newAdr <> CIB_unit[id].address THEN
            result := SetAddressCIBunit(CIB_line := MI_CIB1,
                CIB_unitID := id, CIB_addr := newAdr);
            IF result THEN
                customCIB[id].address := newAdr;
            END_IF;
        END_IF;
    END_IF;
    setAdr := FALSE;
END_IF;

END_PROGRAM

```

Předpokládejme, že v programu PLC jsou nadefinovány dvě jednotky RCM2-1 připojené na CIB sběrnici základního modulu systému Foxtrot (viz následující dialog prostředí Mosaic).

Správce jednotek/zařízení

MI | CIB1 | CIB2

ID	HW adresa	ID CIB sítě	Typ jednotky/...	Stav	Název jednotky/zařízení/vstupu/výstup
1	0001	1	RCM2-1	-	displej pokoj
2	5B0C	1	RCM2-1	-	displej chodba

☐ Zobrazit všechna zařízení všech jednotek

☐ Bázová adresa zón

Nastavení

Výběr jednotky/zařízení:

Název jednotky:

HW adresa jednotky:

☒ Rozšířené nastavení

Visualizace vstupů

☐ Exportovat pro vizualizaci

Pojmenování / alias:

Visualizace výstupů

☐ Exportovat pro vizualizaci

Pojmenování / alias:

Úkolem programu v příkladu je umožnit změnu HW adresy CIB jednotek z operátorského panelu nebo z web stránky, aby nebylo nutné při změně adresy znovu překládat program pro PLC. V programu je založena zálohovaná proměnná *customCIB*, která obsahuje informace o aktuálním nastavení čtyřech CIB jednotek. Při studeném restartu se do této proměnné uloží informace o kódu a HW adrese CIB jednotek, které odpovídají nastavení ve výše uvedeném dialogu „Správce zařízení / jednotek“, takže obsah proměnné *customCIB* odpovídá programu PLC. Naopak při teplém restartu se nastaví HW adresy CIB jednotek podle proměnné *customCIB*.

Adresu CIB jednotky lze změnit, pokud z operátorského panelu nastavíme do proměnné *newAdr* novou adresu, číslo CIB jednotky, jejíž adresu chceme změnit, zadáme do proměnné *id* a nastavíme proměnnou *setAdr* na hodnotu TRUE. Pokud je CIB jednotka v programu PLC definovaná, funkce *SetAddressCIBunit* nastaví novou HW adresu podle proměnné *newAdr* a zároveň zapíše novou adresu do proměnné *customCIB*. Takže při následujícím teplém restartu PLC bude použita HW adresa zadaná z operačního panelu, nikoliv adresa daná programem PLC. Takže i při úpravách PLC programu, kdy se do PLC nahrává nový kód, budou pro CIB jednotky použity HW adresy uložené v proměnné *customCIB*.

Uvedený příklad tedy umožňuje nastavovat (měnit) HW adresy CIB jednotek bez nutnosti měnit program PLC. Takže můžeme použít stejný program PLC pro více aplikací, které se liší HW adresami CIB jednotek.

Novou HW adresu CIB jednotky lze také nastavit například z web stránky na následujícím obrázku. Na této web stránce je zobrazen stav čtyřech CIB jednotek na lince MI_CIB1 a nastavovací pole ve žlutém rámečku umožňují nastavit novou HW adresu.

INTERNAL CIB MASTER : MI CIB1

ID	line defined	unit defined	NET	REI	ADR	COM	INI	HW ADDR	TYPE	DESCRIPTION
1	1	1	1	1	0	0	0	0001	RCM2-1	displej pokoj
2	1	1	1	1	0	0	0	5B0C	RCM2-1	displej chodba
3	1	0	0	0	0	0	0	0000		
4	1	0	0	0	0	0	0	0000		

ID > NEW ADDR >

Ve sloupci „line defined“ jsou zobrazeny proměnné *TestCIB.CIB_unit[i].line_defined*, ve sloupci „unit defined“ jsou zobrazeny proměnné *TestCIB.CIB_unit[i].unit_defined*, sloupce „NET“, „REI“, „ADR“, „COM“ a „INI“ obsahují odpovídající položky proměnné *TestCIB.CIB_unit[i].state* (tedy *TestCIB.CIB_unit[i].state.NET*, atd). Sloupec „HW ADDR“ zobrazuje proměnné *TestCIB.CIB_unit[i].address*, sloupec „TYPE“ zobrazuje proměnné *TestCIB.CIB_unit[i].type* a konečně sloupec „DESCRIPTION“ zobrazuje proměnné *TestCIB.CIB_unit[i].description*. Stav těchto proměnných je trvale aktualizován funkcí *CIBunitInfo()*. Všechny dosud uvedené proměnné jsou „Read Only“.

Nastavení nové HW adresy CIB jednotky se provádí přes zadávací pole „ID“, které je navázané na proměnnou *TestCIB.id*, dále přes zadávací pole „NEW ADDR“, které umožňuje měnit proměnnou *TestCIB.newAdr* a konečně přes dvoustavový obrázek „SET“, který umožňuje nastavit proměnnou *TestCIB.setAdr* na hodnotu TRUE.

Viz také Typ *TCIBunitState*, Typ *TCIBunitInfo*, Funkce *CIBunitInfo*

4.16 Funkce Memcpy

Knihovna : SysLib



Funkce *Memcpy* zkopíruje blok paměti. Vstupní proměnná *length* udává délku kopírovaného bloku v bytech, proměnná *source* specifikuje odkud se bude kopírovat a proměnná *dest* udává kam se bude kopírovat.

Popis proměnných :

	Proměnná	Typ	Význam
VAR_INPUT			
	<i>length</i>	UINT	Délka kopírovaného bloku v bytech
VAR_IN_OUT			
	<i>source</i>	USINT	Zdroj odkud kopírovat
	<i>dest</i>	USINT	Cíl kam kopírovat
Memcpy			
	Návratová hodnota	UINT	Počet zkopírovaných bytů

Vzhledem k tomu, že proměnné *source* a *dest* jsou typu USINT, tak se na první pohled může zdát, že funkci *Memcpy* nelze použít na nic jiného, než je zkopírování jednoho bytu paměti, což by prakticky nemělo smysl. Uvedené proměnné jsou však třídy VAR_IN_OUT což znamená, že se při volání funkce *Memcpy* budou předávat adresy proměnných přiřazených do *source* a *dest* (nikoliv hodnoty těchto proměnných). Takže zbývá problém, jak přemluvit ST překladač k tomu, aby nekontroloval datový typ proměnných *source* a *dest*. K tomu slouží funkce *void()*, která zruší kontrolu datového typu u proměnných třídy VAR_IN_OUT. Funkce *void()* tedy umožní kopírovat proměnné libovolného typu pomocí funkce *Memcpy*. Na druhou stranu je nutné pečlivě volit velikost kopírovaného bloku paměti. Pokud bude velikost proměnné *dest* menší než je hodnota předaná v parametru *length*, funkce *Memcpy* přepíše proměnné ležící v paměti za proměnnou *dest*. Překladač tento problém nemůže nahlásit, neboť má vypnutou kontrolu typů pomocí funkce *void()*. Za správnost v tomto případě ručí pouze programátor.

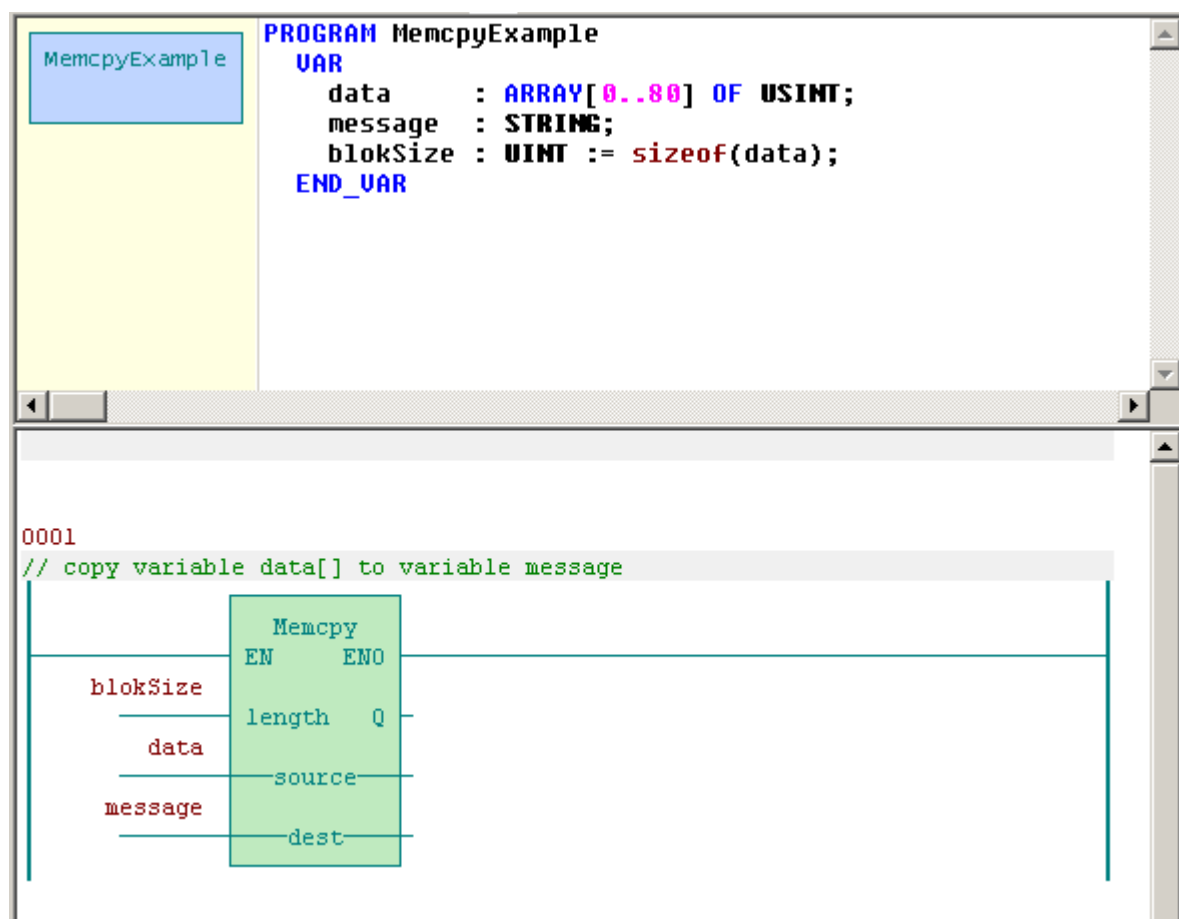
POZOR ! Proměnné *source* a *dest* nesmí být typu BOOL ! Funkce *Memcpy* nebude fungovat správně, pokud je některý z parametrů *source* resp. *dest* typu BOOL.

Následující příklad ukazuje jak zkopírovat proměnnou *data[]* což je pole USINT, které má 81 položek, do proměnné *message*, jenž je typu STRING (default velikost je 80 znaků plus jeden na koncovou nulu řetězce). Proměnné *data* a *message* musí mít stejnou velikost.

```
PROGRAM MallocExample
VAR
  data      : ARRAY[0..80] OF USINT;
  message   : STRING;
  size      : UINT;
END_VAR

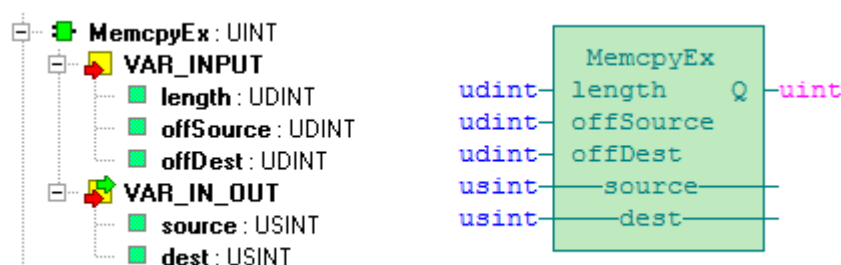
// copy variable data[] to variable message
size := Malloc( length := sizeof(data),
               source := void(data[0]),
               dest    := void(message));
END_PROGRAM
```

Stejný program bude v jazyce LD vypadat následovně:



4.17 Funkce MemcpyEx

Knihovna : SysLib



Funkce *MemcpyEx* zkopíruje blok paměti. Vstupní proměnná *length* udává délku kopírovaného bloku v bytech, proměnná *source* specifikuje odkud se bude kopírovat, proměnná *offSource* umožňuje zadat posunutí od proměnné *source*, proměnná *dest* udává kam se bude kopírovat a proměnná *offDest* umožňuje zadat posunutí oproti proměnné *dest*. Jinak řečeno blok dat bude zkopírován z adresy *source* + *offSource* na adresu *dest* + *offDest*. Návrátová hodnota funkce udává počet skutečně přenesených bytů.

Tato funkce je zařazena do knihovny SysLib od v2.8 a je podporována na centrálních jednotkách řady C,G a K (všechny procesory systému TC650, TC700 a Foxtrot) ve všech verzích.

Popis proměnných :

	Proměnná	Typ	Význam
VAR_INPUT			
	<i>length</i>	UDINT	Délka kopírovaného bloku v bytech
	<i>offSource</i>	UDINT	Posunutí od začátku zdroje
	<i>offDest</i>	UDINT	Posunutí od začátku cíle
VAR_IN_OUT			
	<i>source</i>	USINT	Zdroj odkud kopírovat
	<i>dest</i>	USINT	Cíl kam kopírovat
MemcpyEx			
	Návrátová hodnota	UINT	Počet zkopírovaných bytů

Funkce *MemcpyEx* je rozšířenou variantou funkce *Memcpy*. Pro volání funkce *MemcpyEx* platí stejné zásady jako u funkce *Memcpy* (viz kap 4.16).

POZOR ! Proměnné *source* a *dest* nesmí být typu BOOL ! Funkce *MemcpyEx* nebude fungovat správně, pokud je některý z parametrů *source* resp. *dest* typu BOOL.

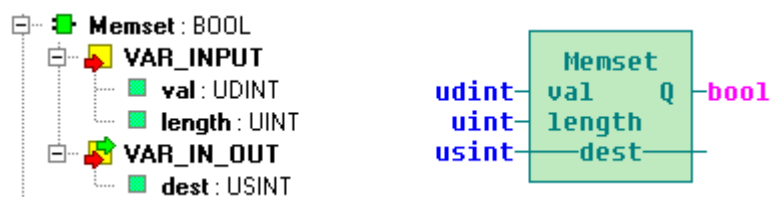
Následující příklad ukazuje jak zkopírovat 20 bytů z pole *data[]* počínajíc bytem *data[10]* do proměnné *message*, jenž je typu *STRING* (default velikost je 80 znaků plus jeden na koncovou nulu řetězce). Proměnná *data* musí mít velikost minimálně 10+20 bytů, proměnná *message* musí mít velikost minimálně 20 znaků.

```
PROGRAM ExampleMemcpyEx
VAR
    data      : ARRAY[0..80] OF USINT;
    message   : STRING;
    size      : UINT;
END_VAR

// copy from variable data[] to variable message
size := MemcpyEx( length   := 20,
                  offSource := 10,
                  offDest  := 0,
                  source    := void(data[0]),
                  dest      := void(message) );
END_PROGRAM
```

4.18 Funkce Memset

Knihovna : SysLib



Funkce *Memset* vyplní blok paměti zadanou hodnotou.

Popis proměnných :

	Proměnná	Typ	Význam
VAR_INPUT			
	<i>val</i>	UDINT	Hodnota, která se zapíše do bloku paměti
	<i>length</i>	UINT	Délka bloku paměti v bytech
VAR_IN_OUT			
	<i>dest</i>	USINT	Proměnná, která bude naplněna konstantou
Memset			
	Návratová hodnota	BOOL	TRUE po úspěšném provedení funkce

V následujícím příkladu naplní funkce *Memset* proměnnou *message* (typu STRING) znakem 16#20 což je kód mezery. Nesoulad datových typů v parametru *dest*, kde je požadován parametr typu USINT a je třeba naplnit proměnnou *message*, která je typu STRING, řeší použití funkce *void()* při předávání parametru. Tato funkce umožní předat v parametru *dest* i jiný datový typ, nejen USINT jak definuje funkce *Memset*.

POZOR ! Proměnná *dest* nesmí být typu BOOL ! Funkce *Memset* nebude fungovat správně, pokud je parametr *dest* typu BOOL.

```

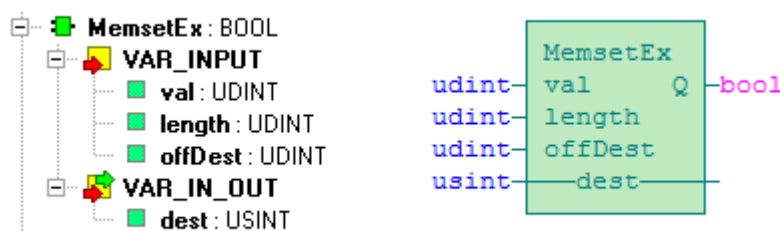
PROGRAM MemsetExample
VAR
  tmp      : BOOL;
  message  : STRING;
END_VAR

tmp := Memset( val    := 16#20202020,
               length := sizeof(message)-1,
               dest   := void(message) );
END_PROGRAM

```

4.19 Funkce MemsetEx

Knihovna : SysLib



Funkce *MemsetEx* vyplní blok paměti zadanou hodnotou. Adresa bloku je daná součtem proměnných *dest* + *offDest*. Blok bude vyplněn hodnotou proměnné *val*. Délku bloku udává proměnná *length*.

Tato funkce je zařazena do knihovny SysLib od v2.8 a je podporována na centrálních jednotkách řady C,G a K (všechny procesory systému TC650, TC700 a Foxtrot) ve všech verzích.

Popis proměnných :

	Proměnná	Typ	Význam
VAR_INPUT			
	<i>val</i>	UDINT	Hodnota, která se zapíše do bloku paměti
	<i>length</i>	UDINT	Délka bloku paměti v bytech
	<i>offDest</i>	UDINT	Posunutí od začátku proměnné <i>dest</i>
VAR_IN_OUT			
	<i>dest</i>	USINT	Proměnná, která bude naplněna konstantou
MemsetEx			
	Návratová hodnota	BOOL	TRUE po úspěšném provedení funkce

POZOR ! Proměnná *dest* nesmí být typu BOOL ! Funkce *MemsetEx* nebude fungovat správně, pokud je parametr *dest* typu BOOL.

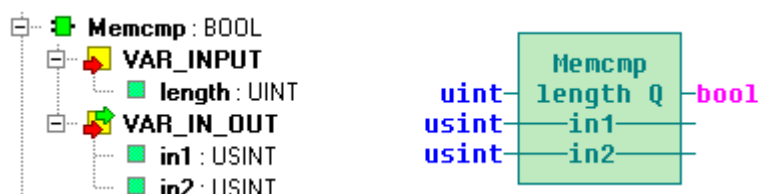
V následujícím příkladu naplní funkce *MemsetEx* proměnnou *message* (typu STRING) znakem 16#20 což je kód mezery. Nesoulad datových typů v parametru *dest*, kde je požadován parametr typu USINT a je třeba naplnit proměnnou *message*, která je typu STRING, řeší použití funkce *void()* při předávání parametru. Tato funkce umožní předat v parametru *dest* i jiný datový typ, nejen USINT jak definuje funkce *MemsetEx*.

```
PROGRAM MemssetExample
VAR
  tmp      : BOOL;
  message  : STRING;
END_VAR

tmp := Memsset( val    := 16#20202020,
                length := sizeof(message)-1,
                dest    := void(message) );
END_PROGRAM
```

4.20 Funkce Memcmp

Knihovna : SysLib



Funkce *Memcmp* porovná vzájemně dva bloky paměti.

Tato funkce je zařazena do knihovny SysLib od v2.1 a je podporována na centrálních jednotkách řady C,G a K (všechny procesory systému TC650, TC700 a Foxtrot) ve všech verzích.

Popis proměnných :

	Proměnná	Typ	Význam
VAR_INPUT			
	<i>length</i>	UINT	Délka porovnávaného bloku v bytech
VAR_IN_OUT			
	<i>in1</i>	USINT	První porovnávaný blok
	<i>in2</i>	USINT	Druhý porovnávaný blok
Memcmp			
	Návratová hodnota	BOOL	TRUE pokud jsou bloky paměti shodné

V následujícím příkladu porovná funkce *Memcmp* proměnnou *message* (typu STRING) s proměnnou *data* (typu ARRAY OF USINT). Nesoulad datových typů v parametrech *in1* a *in2*, kde je požadován parametr typu USINT řeší použití funkce *void()* při předávání parametru.

POZOR ! Proměnné *in1* a *in2* nesmí být typu BOOL ! Funkce *Memcmp* nebude fungovat správně, pokud je některý z parametrů *in1* resp. *in2* typu BOOL.

```

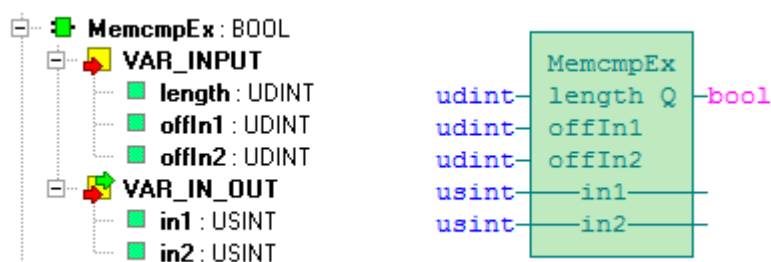
PROGRAM MemcmpExample
VAR
  data      : ARRAY[0..80] OF USINT;
  message   : STRING;
  result    : BOOL;
END_VAR

// compare variable data[] and variable message
result := Memcmp( length := sizeof(data),
                  in1 := void(data[0]), in2 := void(message) );
END_PROGRAM

```

4.21 Funkce MemcmpEx

Knihovna : SysLib



Funkce *MemcmpEx* porovná vzájemně dva bloky paměti. Adresa prvního bloku je daná součtem $in1 + offIn1$. Adresa druhého bloku je daná součtem $in2 + offIn2$. Délku porovnávaného bloku udává proměnná *length*.

Tato funkce je zařazena do knihovny SysLib od v2.8 a je podporována na centrálních jednotkách řady C,G a K (všechny procesory systému TC650, TC700 a Foxtrot) ve všech verzích.

Popis proměnných :

	Proměnná	Typ	Význam
VAR_INPUT			
	<i>length</i>	UDINT	Délka porovnávaného bloku v bytech
	<i>offIn1</i>	UDINT	Posunutí od začátku prvního bloku
	<i>offIn2</i>	UDINT	Posunutí od začátku druhého bloku
VAR_IN_OUT			
	<i>in1</i>	USINT	První porovnávaný blok
	<i>in2</i>	USINT	Druhý porovnávaný blok
MemcmpEx			
	Návratová hodnota	BOOL	TRUE pokud jsou bloky paměti shodné

V následujícím příkladu porovná funkce *MemcmpEx* proměnnou *message* (typu STRING) s proměnnou *data* (typu ARRAY OF USINT). Nesoulad datových typů v parametrech *in1* a *in2*, kde je požadován parametr typu USINT řeší použití funkce *void()* při předávání parametru.

POZOR ! Proměnné *in1* a *in2* nesmí být typu BOOL ! Funkce *MemcmpEx* nebude fungovat správně, pokud je některý z parametrů *in1* resp. *in2* typu BOOL.

PROGRAM MemcmpExample

```
VAR
  data      : ARRAY[0..80] OF USINT;
  message   : STRING;
  result    : BOOL;
END_VAR

// compare variable data[] and variable message
result := Memcmp( length := sizeof(data),
                  in1 := void(data[0]), in2 := void(message));
END_PROGRAM
```

4.22 Funkce *IncreaseMaxCycleTime*

Knihovna : *SysLib*

Funkce *IncreaseMaxCycleTime* zvýší jednorázově hlídanou dobu cyklu PLC. Tato funkce je určena pro případy, kdy např. inicializace aplikačního programu trvá delší dobu než je nastavená doba cyklu PLC a z pohledu aplikace není vhodné prodloužit hlídanou dobu cyklu. Vstupní parametr *addTime* udává o kolik milisekund se má zvýšit hlídaná doba právě prováděného cyklu PLC.

Tato funkce je zařazena do knihovny SysLib od v1.5 a je podporována na centrálních jednotkách řady K (TC700 CP-7004, Foxtrot) od v2.8. Maximální doba cyklu u těchto systémů je 750 ms.

Příklad programu s voláním funkce *IncreaseMaxCycleTime*:

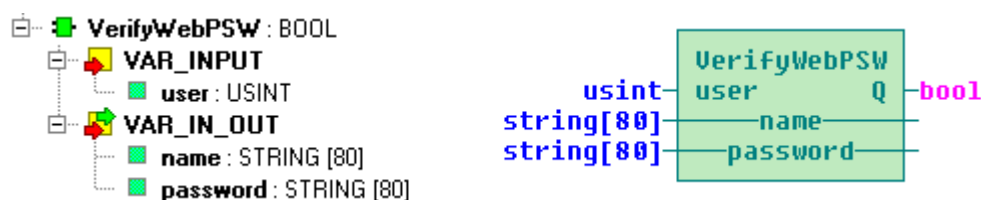
```
PROGRAM RestartExample
VAR
    tmp : BOOL;
END_VAR

// increase max. limit for current PLC cycle time
tmp := IncreaseMaxCycleTime( addTime := 200); // plus 200 ms

// next restart activities
// ...
END_PROGRAM
```

4.23 Funkce VerifyWebPSW

Knihovna : SysLib



Centrální jednotky řady K mají integrovaný web server. Jinými slovy v těchto centrálních jednotkách mohou být uloženy webové stránky, na kterých lze zobrazit informace o řízené technologii případně lze na těchto stránkách nastavovat parametry, podle kterých pak probíhá řízení. K prohlížení web stránek v PLC může být použit libovolný prohlížeč (web browser). Zobrazení stránek v prohlížeči je podmíněno zadáním jména uživatele a hesla. Webové stránky a jména a hesla uživatelů, kterým je povolen přístup na stránky, se připravují nástrojem WebMaker v programovacím prostředí Mosaic. Tento nástroj umožňuje zadat až 10 různých jmen uživatelů a k nim přiřazených hesel. V některých případech je třeba měnit jména uživatelů a jejich hesla bez změny aplikačního programu (bez použití prostředí Mosaic) např. z operátorského panelu. K tomu slouží funkce *VerifyWebPSW* a *SetWebPSW*.

Funkce *VerifyWebPSW* zkontroluje, jestli pro daného uživatele existuje konkrétní jméno a heslo v seznamu jmen oprávněných uživatelů. Seznam může obsahovat až 10 různých uživatelů. První uživatel má číslo 0, druhý 1, poslední uživatel v seznamu má číslo 9. Pokud jméno a heslo v seznamu odpovídá zadanému jménu a heslu, funkce vrátí hodnotu TRUE. Pokud nesouhlasí jméno a heslo, funkce vrátí hodnotu FALSE.

Popis proměnných :

	Proměnná	Typ	Význam
VAR_INPUT			
	<i>user</i>	USINT	Číslo uživatele v seznamu oprávněných uživatelů První uživatel má číslo 0, poslední uživatel má číslo 9
VAR_IN_OUT			
	<i>name</i>	STRING	Jméno uživatele, které bude porovnáno se seznamem
	<i>password</i>	STRING	Heslo, které bude porovnáno se seznamem
VerifyWebPSW			
	Návratová hodnota	BOOL	TRUE pokud jméno a heslo souhlasí se seznamem uživatelů, jinak FALSE

Funkce *VerifyWebPSW* je zařazena do knihovny SysLib od v2.3 a je podporována na centrálních jednotkách řady K (procesor CP-7004 systému TC700 a všechny procesory systému Foxtrot) od verze v5.6.

Seznam oprávněných uživatelů se v prostředí Mosaic nastavuje následujícím dialogem nástroje WebMaker.

Úroveň	Uživatelské jméno	Heslo	Výchozí stránka
0	user0	xxxxx	Index
1	user1	xxxxx	Index
2	user2	xxxxx	Index
-1		x	
-1		x	
-1		x	
-1		x	
-1		x	
-1		x	
-1		x	
9	admin	xxxxx	

Uživatel číslo 0 má nastaveno jméno „user0“, uživatel číslo 1 má nastaveno jméno „user1“, uživatel číslo 2 má nastaveno jméno „user2“, uživatel číslo 9 má nastaveno jméno „admin“. Ostatní uživatelé nejsou nastaveni (viz řádky, kde je nastavena úroveň přístupu -1).

Program v následujícím příkladu zjišťuje, je-li pro uživatele číslo 1 nastaveno jméno „super“ a heslo „user“. Pokud bude seznam oprávněných uživatelů odpovídat výše uvedenému dialogu, funkce *VerifyWebPSW* v příkladu bude vracet FALSE.

```

VAR_GLOBAL RETAIN
  myName   : STRING := 'super';
  myPass   : STRING := 'user';
END_VAR

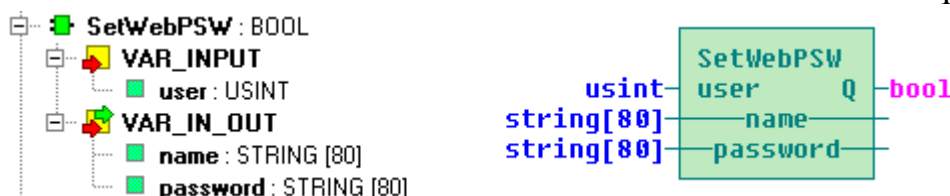
PROGRAM TestWebPass
  VAR
    init, result : BOOL;
  END_VAR

  IF NOT init THEN
    result := VerifyWebPSW(user := 1, name := myName, password := myPass);
    init   := TRUE;
  END_IF;
END_PROGRAM

```

4.24 Funkce SetWebPSW

Knihovna : SysLib



Centrální jednotky řady K mají integrovaný web server. Jinými slovy v těchto centrálních jednotkách mohou být uloženy webové stránky, na kterých lze zobrazit informace o řízené technologii případně lze na těchto stránkách nastavovat parametry, podle kterých pak probíhá řízení. K prohlížení web stránek v PLC může být použit libovolný prohlížeč (web browser). Zobrazení stránek v prohlížeči je podmíněno zadáním jména uživatele a hesla. Webové stránky a jména a hesla uživatelů, kterým je povolen přístup na stránky, se připravují nástrojem WebMaker v programovacím prostředí Mosaic. Tento nástroj umožňuje zadat až 10 různých jmen uživatelů a k nim přiřazených hesel. První uživatel má číslo 0, druhý 1, poslední uživatel v seznamu má číslo 9. V některých případech je třeba měnit jména uživatelů a jejich hesla bez použití prostředí Mosaic např. z operátorského panelu. K tomu slouží funkce *SetWebPSW*.

Funkce *SetWebPSW* nejprve zkontroluje, jestli souhlasí zadané jméno a heslo uživatele se jménem a heslem uvedeným v seznamu oprávněných uživatelů. Pokud souhlasí, funkce vrátí hodnotu TRUE. Pokud jméno nebo heslo v seznamu nesouhlasí, funkce *SetWebPSW* nahradí jméno a heslo v seznamu za jméno a heslo uvedené v parametrech funkce. Pokud se zápis do seznamu podaří, vrátí funkce hodnotu TRUE. Je-li seznam uživatelů nedostupný (například protože aplikační program neobsahuje žádné web stránky), funkce *SetWebPSW* vrací hodnotu FALSE.

Popis proměnných :

	Proměnná	Typ	Význam
VAR_INPUT			
	<i>user</i>	USINT	Číslo uživatele v seznamu oprávněných uživatelů První uživatel má číslo 0, poslední uživatel má číslo 9
VAR_IN_OUT			
	<i>name</i>	STRING	Nové jméno uživatele, které bude zapsáno do seznamu
	<i>password</i>	STRING	Nové heslo, které bude zapsáno do seznamu
SetWebPSW			
	<i>Návratová hodnota</i>	BOOL	TRUE pokud jméno a heslo souhlasí se seznamem uživatelů, jinak FALSE

Tato funkce je zařazena do knihovny SysLib od v2.3 a je podporována na centrálních jednotkách řady K (procesor CP-7004 systému TC700 a všechny procesory systému Foxtrot) od verze v5.6.

Seznam oprávněných uživatelů se v prostředí Mosaic nastavuje následujícím dialogem nástroje WebMaker.

Úroveň	Uživatelské jméno	Heslo	Výchozí stránka
0	user0	xxxxx	Index
1	user1	xxxxx	Index
2	user2	xxxxx	Index
-1		*	
-1		*	
-1		*	
-1		*	
-1		*	
-1		*	
-1		*	
9	admin	xxxxx	

Uživatel číslo 0 má nastaveno jméno „user0“, uživatel číslo 1 má nastaveno jméno „user1“, uživatel číslo 2 má nastaveno jméno „user2“, uživatel číslo 9 má nastaveno jméno „admin“. Ostatní uživatelé nejsou nastaveni (viz řádky, kde je nastavena úroveň přístupu -1).

Program v následujícím příkladu nastavuje pro uživatele číslo 1 jméno „super“ a heslo „user“. Pokud bude seznam oprávněných uživatelů odpovídat výše uvedenému dialogu, funkce *SetWebPSW* v příkladu přepíše původní jméno „user1“ na „super“ a původní heslo změní na „user“.

```

VAR_GLOBAL RETAIN
  myName   : STRING := 'super';
  myPass   : STRING := 'user';
END_VAR

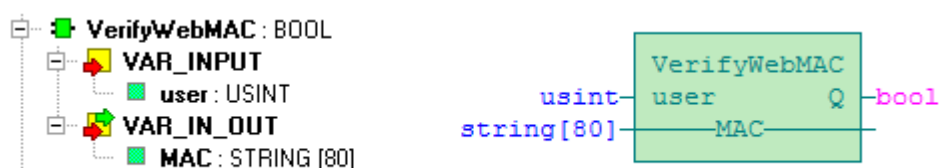
PROGRAM TestWebPass
  VAR
    init    : BOOL;
    result  : BOOL;
  END_VAR

  IF NOT init THEN
    result := SetWebPSW(user := 1, name := myName, password := myPass);
    init   := TRUE;
  END_IF;
END_PROGRAM

```

4.25 Funkce VerifyWebMAC

Knihovna : SysLib



Centrální jednotky řady K mají integrovaný web server. Jinými slovy v těchto centrálních jednotkách mohou být uloženy webové stránky, na kterých lze zobrazit informace o řízené technologii případně lze na těchto stránkách nastavovat parametry, podle kterých pak probíhá řízení. K prohlížení web stránek v PLC může být použit libovolný prohlížeč (web browser). Zobrazení stránek v prohlížeči je podmíněno zadáním jména uživatele a hesla. Webové stránky a jména a hesla uživatelů, kterým je povolen přístup na stránky, se připravují nástrojem WebMaker v programovacím prostředí Mosaic. Tento nástroj umožňuje zadat až 10 různých jmen uživatelů a k nim přiřazených hesel. Zároveň je možné zadat až 10 MAC adres počítačů, na kterých nebude nutné zadávat jména a hesla – budou mít automaticky povolen přístup k web stránkám v PLC. V některých případech je třeba měnit MAC adresy bez změny aplikačního programu (bez použití prostředí Mosaic) např. z operátorského panelu. K tomu slouží funkce *VerifyWebMAC* a *SetWebMAC*.

Funkce *VerifyWebMAC* zkontroluje, jestli pro daného uživatele existuje konkrétní MAC adresa v seznamu MAC adres. Seznam může obsahovat až 10 různých uživatelů. První uživatel má číslo 0, druhý 1, poslední uživatel v seznamu má číslo 9. Pokud MAC adresa v seznamu odpovídá zadanému MAC adrese, funkce vrátí hodnotu TRUE. Pokud MAC adresy nesouhlasí, funkce vrátí hodnotu FALSE.

Popis proměnných :

	Proměnná	Typ	Význam
VAR_INPUT			
	<i>user</i>	USINT	Číslo uživatele v seznamu oprávněných uživatelů První uživatel má číslo 0, poslední uživatel má číslo 9
VAR_IN_OUT			
	<i>MAC</i>	STRING	MAC adresa, která bude porovnána se seznamem
VerifyWebMAC			
	<i>Návratová hodnota</i>	BOOL	TRUE pokud zadaná MAC adresa souhlasí s adresou v seznamu uživatelů, jinak FALSE

Funkce *VerifyWebMACP* je zařazena do knihovny SysLib od v2.9 a je podporována na centrálních jednotkách řady K a L (procesory CP-7004 a CP-7007 systému TC700 a všechny procesory systému Foxtrot) od verze v7.1.

Seznam MAC adres počítačů, jejichž přihlášení k web serveru v PLC bude provedeno automaticky, se v prostředí Mosaic nastavuje následujícím dialogem nástroje WebMaker.

Úroveň	MAC Adresa	Výchozí stránka
0	11:22:33:44:55:66	Index
1	00:00:00:00:00:00	
2	00:00:00:00:00:00	
3	00:00:00:00:00:00	
6	00:00:00:00:00:00	
7	00:00:00:00:00:00	
5	00:00:00:00:00:00	
4	00:00:00:00:00:00	
8	00:00:00:00:00:00	
9	00:00:00:00:00:00	

Uživatel na řádku číslo 0 má nastavenou MAC adresu 11:22:33:44:55:66. Jeho přihlášení proběhne automaticky s úrovní 0 a první zobrazenou web stránkou bude Index. MAC adresy ostatních uživatelů nejsou nastaveny (viz řádky, kde je nastavena MAC adresa 00:00:00:00:00:00).

Program v následujícím příkladu zjišťuje, je-li pro uživatele číslo 0 nastavena MAC adresa 11:22:33:44:55:66. Pokud bude seznam MAC adres odpovídat výše uvedenému dialogu, funkce *VerifyWebMAC* v příkladu bude vracet TRUE.

```

VAR_GLOBAL RETAIN
  myMAC : STRING := '11:22:33:44:55:66';
END_VAR

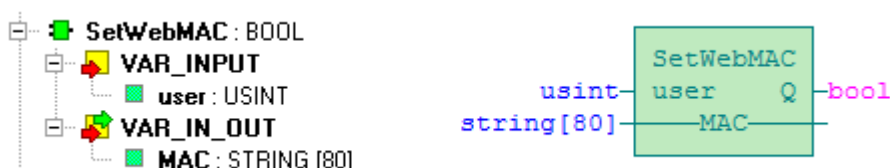
PROGRAM TestWebMAC
  VAR
    init, result : BOOL;
  END_VAR

  IF NOT init THEN
    result := VerifyWebMAC(user := 0, MAC := myMAC);
    init := TRUE;
  END_IF;
END_PROGRAM

```

4.26 Funkce SetWebMAC

Knihovna : SysLib



Centrální jednotky řady K mají integrovaný web server. Jinými slovy v těchto centrálních jednotkách mohou být uloženy webové stránky, na kterých lze zobrazit informace o řízení technologii případně lze na těchto stránkách nastavovat parametry, podle kterých pak probíhá řízení. K prohlížení web stránek v PLC může být použit libovolný prohlížeč (web browser). Zobrazení stránek v prohlížeči je podmíněno zadáním jména uživatele a hesla. Webové stránky a jména a hesla uživatelů, kterým je povolen přístup na stránky, se připravují nástrojem WebMaker v programovacím prostředí Mosaic. Tento nástroj umožňuje zadat až 10 různých jmen uživatelů a k nim přiřazených hesel. Zároveň je možné zadat až 10 MAC adres počítačů, na kterých nebude nutné zadávat jména a hesla – budou mít automaticky povolen přístup k web stránkám v PLC. V některých případech je třeba měnit MAC adresy bez změny aplikačního programu (bez použití prostředí Mosaic) např. z operátorského panelu. K tomu slouží funkce *VerifyWebMAC* a *SetWebMAC*.

Funkce *SetWebMAC* nejprve zkontroluje, jestli souhlasí zadaná MAC adresa s adresou uvedenou v seznamu MAC adres. Pokud souhlasí, funkce vrátí hodnotu TRUE. Pokud adresy nesouhlasí, funkce *SetWebMAC* nahradí MAC adresu v seznamu za MAC adresu uvedenou v parametrech funkce. Pokud se zápis do seznamu podaří, vrátí funkce hodnotu TRUE. Je-li seznam uživatelů nedostupný (například protože aplikační program neobsahuje žádné web stránky), funkce *SetWebMAC* vrací hodnotu FALSE.

Popis proměnných :

	Proměnná	Typ	Význam
VAR_INPUT			
	<i>user</i>	USINT	Číslo uživatele v seznamu oprávněných uživatelů První uživatel má číslo 0, poslední uživatel má číslo 9
VAR_IN_OUT			
	<i>MAC</i>	STRING	Nová MAC adresa, která bude zapsaná do seznamu (6 hexa cifér oddělených znakem dvojtečka)
SetWebMAC			
	<i>Návratová hodnota</i>	BOOL	TRUE pokud MAC adresa souhlasí se seznamem uživatelů, jinak FALSE

Tato funkce je zařazena do knihovny SysLib od v2.9 a je podporována na centrálních jednotkách řady K a L (procesor CP-7004 a CP-7007 systému TC700 a všechny procesory systému Foxtrot) od verze v7.1.

Seznam MAC adres počítačů, jejichž přihlášení k web serveru v PLC bude provedeno automaticky, se v prostředí Mosaic nastavuje následujícím dialogem nástroje WebMaker.

Úroveň	MAC Adresa	Výchozí stránka
0	11:22:33:44:55:66	Index
1	00:00:00:00:00:00	
2	00:00:00:00:00:00	
3	00:00:00:00:00:00	
6	00:00:00:00:00:00	
7	00:00:00:00:00:00	
5	00:00:00:00:00:00	
4	00:00:00:00:00:00	
8	00:00:00:00:00:00	
9	00:00:00:00:00:00	

Uživatel na řádku číslo 0 má nastavenou MAC adresu 11:22:33:44:55:66. Jeho přihlášení proběhne automaticky s úrovní 0 a první zobrazenou web stránkou bude Index. MAC adresy ostatních uživatelů nejsou nastaveny (viz řádky, kde je nastavena MAC adresa 00:00:00:00:00:00).

Program v následujícím příkladu zjišťuje, je-li pro uživatele číslo 0 nastavena MAC adresa 11:11:11:11:11:11. Pokud bude seznam MAC adres odpovídat výše uvedenému dialogu, funkce *SetWebMAC* v příkladu změní MAC adresu pro automatické přihlášení uživatele 0 a vrátí TRUE.

```

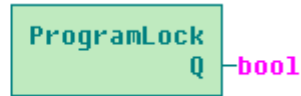
VAR_GLOBAL RETAIN
  myMAC : STRING := '11:11:11:11:11:11';
END_VAR

PROGRAM SetWebMAC
  VAR
    init, result : BOOL;
  END_VAR

  IF NOT init THEN
    result := SetWebMAC (user := 0, MAC := myMAC);
    init := TRUE;
  END_IF;
END_PROGRAM

```

4.27 Funkce ProgramLock

Knihovna : *SysLib* **ProgramLock** : BOOL

Funkce *ProgramLock* uzamkne aplikační program PLC tak, že nelze provést jeho zpětný překlad v prostředí Mosaic. Funkce nemá žádné vstupní parametry. Návrátová hodnota je vždy TRUE.

Tato funkce je zařazena do knihovny SysLib od v2.0 a je podporována na centrálních jednotkách řady C,G a K (všechny procesory systému TC650, TC700 a Foxtrot) ve všech verzích.

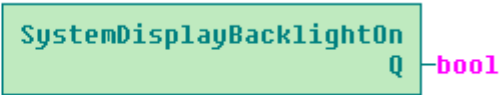
Příklad programu s voláním funkce *ProgramLock*:

```
PROGRAM prgMainExample
VAR
    tmp : BOOL;
END_VAR

// lock application program
tmp := ProgramLock();           // guard program

// next activities
// ...
END_PROGRAM
```

4.28 Funkce *SystemDisplayBacklightOn*

Knihovna : *SysLib* **SystemDisplayBacklightOn** : BOOL**SystemDisplayBacklightOn**
Q — bool

Funkce *SystemDisplayBacklightOn* rozsvítí podsvícení LCD displeje na základním modulu PLC (viz např. displej OI-1074 systému Foxtrot CP-1016). Funkce nemá žádné vstupní parametry. Návrátová hodnota je TRUE pokud je displej dostupný. Pokud je displej nedostupný, funkce vrátí FALSE (např. pokud funkci zavoláme v systému Foxtrot CP-1004, kde není LCD displej).

POZOR !!!

Pokud bude tato funkce volaná trvale, podsvícení displeje bude trvale zapnuté. To nelze doporučit s ohledem na životnost výbojky, kterou je podsvícení realizované.

Tato funkce je zařazena do knihovny SysLib od v2.4 a je podporována na centrálních jednotkách řady K (všechny procesory systému Foxtrot) od verze firmwaru v5.7.

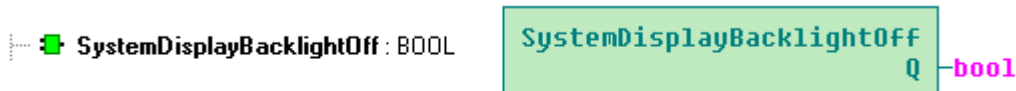
Příklad programu s voláním funkce *SystemDisplayBacklightOn*:

```
PROGRAM TestSystemDisplay
VAR
    errFlg : BOOL;
    errAck : BOOL;
END_VAR

IF errFlg THEN
    SystemDisplayBacklightOn();           // switch on LCD backlight
    errFlg := FALSE;
END_IF;
IF errAck THEN
    SystemDisplayBacklightOff();          // switch off LCD backlight
    errAck := FALSE;
END_IF;

END_PROGRAM
```

4.29 Funkce *SystemDisplayBacklightOff*

Knihovna : *SysLib*

Funkce *SystemDisplayBacklightOff* zhasne podsvícení LCD displeje na základním modulu PLC (viz např. displej OI-1074 systému Foxtrot CP-1016). Funkce nemá žádné vstupní parametry. Návrátová hodnota je TRUE pokud je displej dostupný. Pokud je displej nedostupný, funkce vrátí FALSE (např. pokud funkci zavoláme v systému Foxtrot CP-1004, kde není LCD displej).

POZOR !!!

Pokud bude tato funkce volaná trvale, podsvícení displeje bude trvale vypnuté a nerozsvítí se ani při stisku klávesy pro ovládání displeje.

Tato funkce je zařazena do knihovny SysLib od v2.4 a je podporována na centrálních jednotkách řady K (všechny procesory systému Foxtrot) od verze firmwaru v5.7.

Příklad programu s voláním funkce *SystemDisplayBacklightOff*:

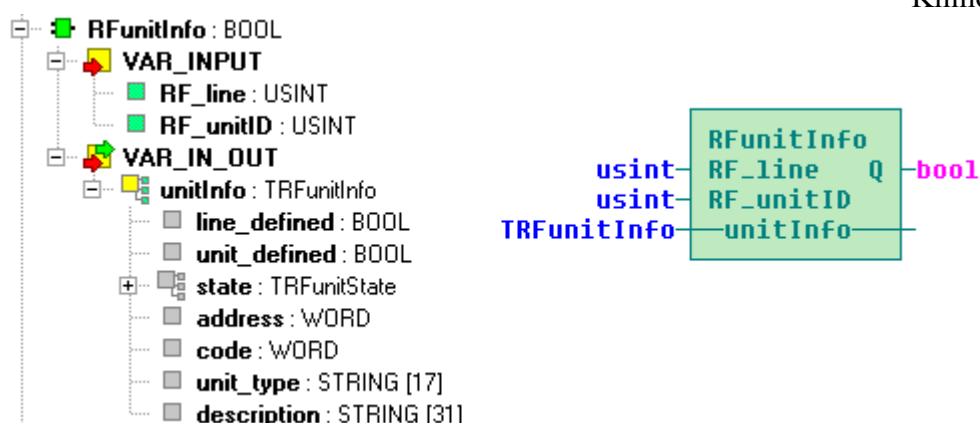
```
PROGRAM TestSystemDisplay
VAR
  errFlg : BOOL;
  errAck : BOOL;
END_VAR

IF errFlg THEN
  SystemDisplayBacklightOn();           // switch on LCD backlight
  errFlg := FALSE;
END_IF;
IF errAck THEN
  SystemDisplayBacklightOff();          // switch off LCD backlight
  errAck := FALSE;
END_IF;

END_PROGRAM
```

4.30 Funkce RFunitInfo

Knihovna : SysLib



Funkce *RFunitInfo* slouží k získání informace o aktuálním stavu jedné jednotky na rádiové síti RFox. Vstupní parametr *RF_line* specifikuje přes kterou jednotku RF master daná jednotka komunikuje se systémem (viz konstanty *MI_RF*, *RF0_RF* až *RF6_RF*) a parametr *RF_unitID* říká číslo pozice, pro kterou se vrátí informace o RF jednotce. Funkce vrací hodnotu TRUE, pokud se podaří informace o RF jednotce získat. Získané informace jsou uloženy v proměnné dané parametrem *unitInfo*. Pokud není RF jednotka deklarovaná v HW konfiguraci PLC, funkce *RFunitInfo* vrací FALSE.

Tato funkce je podporovaná v centrálních jednotkách Foxtrot od v6.6. Funkce je zařazena do knihovny SysLib od v2.6.

Příklad programu s voláním funkce *RFunitInfo* :

```
PROGRAM RFunitInfoExample
VAR
  RF_unit1 : TRFunitInfo;
  RF_unit2 : TRFunitInfo;
  result   : BOOL;
END_VAR

result := RFunitInfo( RF_line := MI_RF, RF_unitID := 1, unitInfo := RF_unit1);
result := RFunitInfo( RF_line := MI_RF, RF_unitID := 2, unitInfo := RF_unit2);
END_PROGRAM
```

Uvedený program testuje stav prvních dvou RF jednotek připojených na RF mastera základního modulu systému Foxtrot. Jednotky jsou v programu PLC nastaveny tak, jak ukazuje obrázek dialogu Správce jednotek / zařízení.

Další obrázek pak ukazuje hodnoty proměnné *RF_unit1*, které odpovídají stavu první definované RF jednotky. Hodnoty odpovídají situaci, kdy je RF jednotka v programu PLC definovaná, ale nekomunikuje (např. protože není spárována nebo má jinou HW adresu, než je nastavená ve výše uvedeném dialogu).

Pokud by chyběla deklarace RF jednotek v programu, funkce *RFunitInfo* bude vracet FALSE a proměnná *RF_unit1* bude vynulovaná.

Správce jednotek/zařízení

MI + RF | CIB1 | RF

ID	HW adr...	ID ...	Typ jednotky/zařízení	Stav	Název jednotky/zařízení/v...
1	5B0C	3	R-RC-0001R	-	display_room1
2	FFFF	3	R-RC-0001R	-	display_room2

Nastavení

Výběr jednotky/zařízení: ↑ ↓

Název jednotky: display_room1

HW adresa jednotky: 5B0C

Visualizace vstupů

☐ Exportovat pro vizualizaci

Pojmenování / alias: display_room1_IN

Visualizace výstupů

☐ Exportovat pro vizualizaci

Pojmenování / alias: display_room1_OUT

☐ Zobrazit všechna zařízení všech jednotek

Přidat jednotku | Vymazat jednotku | Vymazat vše

HW konfigurace RF

Načíst konfiguraci z CPU

☐ Bázová adresa zón: 0

OK Zrušit nápověda

Zprávy 1 | Zprávy 2 | Symbols | Breakpoint list | Data

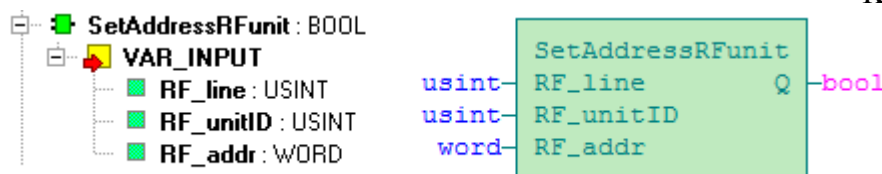
5 Banka3

Jméno	Typ	Hodnota
tstCIBunit.CIB_unit1	TCIBunitInfo	
line_defined	bool	1
unit_defined	bool	1
state	TCIBunitState	
INI	bool	0
COM	bool	0
ADDR	bool	0
DUMMY3	bool	0
REI	bool	1
DUMMY5	bool	0
DUMMY6	bool	0
NET	bool	1
address	word	\$0001
code	word	\$0C55
unit_type [0..16]	string	'RCM2-1'
description [0..32]	string	' displej pokoj '

Viz také Typ TRFunitState, Typ TRFunitInfo

4.31 Funkce SetAddressRFunit

Knihovna : SysLib



Funkce *SetAddressRFunit* nastaví novou HW adresu RF jednotky. To je možné využít například v situaci, kdy chceme používat stejný PLC program pro více aplikací, které se mezi sebou liší HW adresou RF jednotek.

Vstupní parametr *RF_line* specifikuje RF mastera se kterým RF jednotka komunikuje (viz konstanty *MI_RF*, *RF0_RF* až *RF6_RF*) a parametr *RF_unitID* říká, pro kterou RF jednotku bude nastavena nová adresa. Funkce vrací hodnotu TRUE, pokud se podaří adresu RF jednotky nastavit.

POZOR !!!

Po změně adresy RF jednotky funkcí *SetAddressRFunit* bude nutné provést spárování RF jednotky s novou adresou a mastera Rfox sítě (jednotky RF-1130 nebo RF-1131). Bez spárování nebudou RF jednotky komunikovat a data z těchto jednotek nebudou v aplikačním programu dostupná! Párování se provádí pomocí funkčního bloku *fbBondRFunit* (viz Funkční blok *fbBondRFunit*). Naopak pro úspěšné spárování nové bezdrátové jednotky v síti Rfox je nezbytné nastavit hw adresu nové jednotky předtím, než se zahájí párování. Pokud hw adresa jednotky neodpovídá adrese jednotky, kterou párujeme, párování nebude úspěšné.

Funkce *SetAddressRFunit* je podporovaná v centrálních jednotkách Foxtrot od v6.6. Funkce je zařazena do knihovny SysLib od v26.

Příklad programu s voláním funkce *SetAddressRFunit* :

```

VAR_GLOBAL CONSTANT
    NUM_RF_UNIT : USINT := 4;
END_VAR

TYPE
    TcustomRF : STRUCT
        valid      : BOOL;
        address    : WORD;
        code       : WORD;
    END_STRUCT;
END_TYPE

VAR_GLOBAL RETAIN
    customRF : ARRAY[1..NUM_RF_UNIT] OF TcustomRF;
END_VAR

PROGRAM TestRF
    VAR
        i          : USINT;
        RF_unit    : ARRAY[1..NUM_RF_UNIT] OF TRFunitInfo;
        id         : USINT;
        newAdr     : WORD;
    
```

```

    setAdr      : BOOL;
    result      : BOOL;
END_VAR

// read info about RF units and set customer address
FOR i := 1 TO NUM_RF_UNIT DO
    IF RFunitInfo( RF_line := RF0_RF,
                  RF_unitID := i,
                  unitInfo := RF_unit[i]) THEN
        IF not customRF[i].valid OR customRF[i].code <> RF_unit[i].code THEN
            customRF[i].address := RF_unit[i].address;
            customRF[i].code    := RF_unit[i].code;
            customRF[i].valid   := TRUE;
        ELSE
            IF RF_unit[i].address <> customRF[i].address THEN
                SetAddressRFunit(RF_line := MI_CIB1,
                                RF_unitID := i,
                                RF_addr := customRF[i].address);
            END IF;
        END IF;
    END IF;
END_FOR;

// set new customer address of RF unit
IF setAdr THEN
    IF id > 0 AND id <= NUM_RF_UNIT THEN
        IF newAdr <> RF_unit[id].address THEN
            result := SetAddressRFunit(RF_line := RF0_RF,
                                      RF_unitID := id,
                                      RF_addr := newAdr);

            IF result THEN
                customRF[id].address := newAdr;
            END IF;
        END IF;
    END IF;
    setAdr := FALSE;
END_IF;
END_PROGRAM

```

Výše uvedený příklad je pouze ilustrací, jak použít funkci *SetAddressRFunit*. Pro kompletně funkční program je nezbytné po změně adresy RF jednotky provést nové napárování RF jednotky a RF mastera.

Předpokládejme, že v programu PLC jsou nadefinovány dvě bezdrátově komunikující RF jednotky (R-RC-0001R a R-WS-0200R) spárované na jednotku externího RF mastera RF-1131 (viz následující dialog prostředí Mosaic).

Správce jednotek/zařízení

MI RFO

RF

ID	HW adr...	ID ...	Typ jednotky/zařízení	Stav	Název jednotky/zařízení/v...
1	0013	3	R-RC-0001R	-	room control
2	0001	3	R-WS-0200R	-	R-WS-0200R

Nastavení

Výběr jednotky/zařízení ↑ ↓

Název jednotky
room control

HW adresa jednotky
0013

Visualizace vstupů
☐ Exportovat pro vizualizaci
Pojmenování / alias
room_control_IN

Visualizace výstupů
☐ Exportovat pro vizualizaci
Pojmenování / alias
room_control_OUT

☐ Zobrazit všechna zařízení všech jednotek

Přidat jednotku Vymazat jednotku Vymazat vše

HW konfigurace RF

Načíst konfiguraci z CPU

☐ Bázová adresa zón 0

OK Zrušit Nápověda

Úkolem programu v příkladu je umožnit změnu HW adresy RF jednotek z operátorského panelu nebo z web stránky, aby nebylo nutné při změně adresy znovu překládat program pro PLC. V programu je založena zálohovaná proměnná *customRF*, která obsahuje informace o aktuálním nastavení čtyřech RF jednotek. Při studeném restartu se do této proměnné uloží informace o kódu a HW adrese RF jednotek, které odpovídají nastavení ve výše uvedeném dialogu „Správce zařízení / jednotek“, takže obsah proměnné *customRF* odpovídá programu PLC. Naopak při teplém restartu se nastaví HW adresy RF jednotek podle proměnné *customRF*.

Adresu RF jednotky lze změnit, pokud z operátorského panelu nastavíme do proměnné *newAdr* novou adresu, ID číslo RF jednotky, jejíž adresu chceme změnit, zadáme do proměnné *id* a nastavíme proměnnou *setAdr* na hodnotu TRUE. Pokud je RF jednotka v programu PLC definovaná, funkce *SetAddressRFunit* nastaví novou HW adresu podle proměnné *newAdr* a zároveň запиše novou adresu do proměnné *customRF*. Takže při následujícím teplém restartu PLC bude použita HW adresa zadaná z operačního panelu, nikoliv adresa daná programem PLC. Takže i při úpravách PLC programu, kdy se do PLC nahrává nový kód, budou pro RF jednotky použity HW adresy uložené v proměnné *customRF*.

Uvedený příklad tedy umožňuje nastavovat (měnit) HW adresy RF jednotek bez nutnosti měnit program PLC. Takže můžeme použít stejný program PLC pro více aplikací, které se liší HW adresami RF jednotek.

Novou HW adresu RF jednotky lze také nastavit například z web stránky na následujícím obrázku. Na této web stránce je zobrazen stav čtyřech RF jednotek spolupracujících s masterem RF-1131 v síti RF0_RF a nastavovací pole ve žlutém rámečku umožňují nastavit novou HW adresu.

EXTENAL RF MASTER : RF0_RF

ID	line defined	unit defined	NET	BND	SLP	COM	INI	HW ADDR	TYPE	DESCRIPTION
1	1	1	1	1	1	0	0	0013	R-RC-0001R	room control
2	1	1	1	1	1	0	0	0001	R-WS-0200R	R-WS-0200R
3	1	0	0	0	0	0	0	0000		
4	1	0	0	0	0	0	0	0000		

NEW SETTINGS

ID NEW HW ADDR

Ve sloupci „line defined“ jsou zobrazeny proměnné *TestRF.RF_unit[i].line_defined*, ve sloupci „unit defined“ jsou zobrazeny proměnné *TestRF.RF_unit[i].unit_defined*, sloupce „NET“, „BND“, „SLP“, „COM“ a „INI“ obsahují odpovídající položky proměnné *TestRF.RF_unit[i].state* (tedy *TestRF.RF_unit[i].state.NET*, atd). Sloupec „HW ADDR“ zobrazuje proměnné *TestRF.RF_unit[i].address*, sloupec „TYPE“ zobrazuje proměnné *TestRF.RF_unit[i].type* a konečně sloupec „DESCRIPTION“ zobrazuje proměnné *TestRF.RF_unit[i].description*. Stav těchto proměnných je trvale aktualizován funkcí *RFunitInfo()*. Všechny dosud uvedené proměnné jsou „Read Only“.

Nastavení nové HW adresy RF jednotky se provádí přes zadávací pole „ID“, které je navázané na proměnnou *TestRF.id*, dále přes zadávací pole „NEW ADR“, které umožňuje měnit proměnnou *TestRF.newAdr* a konečně přes dvoustavový obrázek „SET“, který umožňuje nastavit proměnnou *TestRF.setAdr* na hodnotu TRUE.

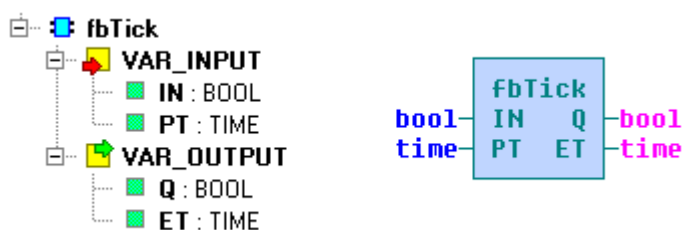
Viz také Typ *TRFunitState*, Typ *TRFunitInfo*, Funkce *RFunitInfo*

5 FUNKČNÍ BLOKY

Knihovna SysLib obsahuje následující funkční bloky:

<i>Funkční blok</i>	<i>Popis</i>
<i>fbTick</i>	Časovač periodicky generující pulz na dobu jednoho cyklu
<i>fbBondRFunit</i>	Spárovat RF mastera s RF jednotkou

5.1 Funkční blok *fbTick*

Knihovna : *SysLib*

Funkční blok *fbTick* je časovač, který generuje periodicky pulzy na dobu jednoho cyklu. Vstup IN povoluje generování pulzů, perioda generovaných pulzů je daná vstupem PT.

Tento funkční blok je podporován na všech centrálních jednotkách řady K (TC700 CP-7004, Foxtrot) od verze v1.0. Do knihovny SysLib je blok zařazen od verze SysLib_v21.

Popis proměnných :

	Proměnná	Typ	Význam
VAR_INPUT			
	IN	BOOL	Pulzy na výstupu Q jsou generovány pokud má vstup IN hodnotu TRUE
	PT	TIME	Perioda výstupních pulzů
VAR_OUTPUT			
	Q	BOOL	Výstupní pulz s dobou trvání jeden cyklus (scan) PLC
	ET	TIME	Průběžný čas v rámci periody

Příklad programu s použitým funkčním blokem *fbTick* :

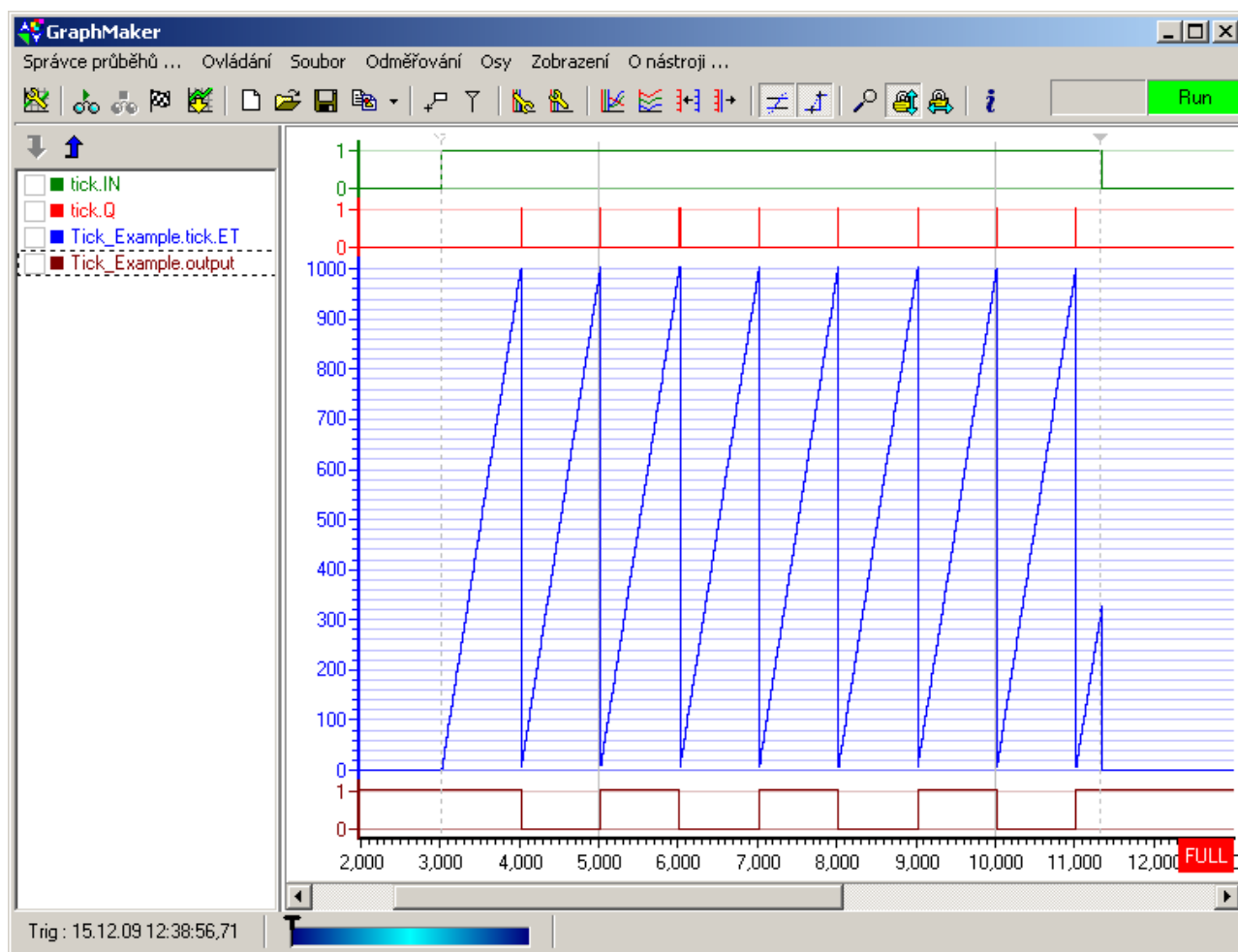
```

PROGRAM TickExample
VAR
    tickEnable : BOOL;
    tick       : fbTick;
    output     : BOOL;
END_VAR

tick( IN := tickEnable, PT := T#1s);
IF tick.Q THEN
    output := NOT output;
END_IF;
END_PROGRAM

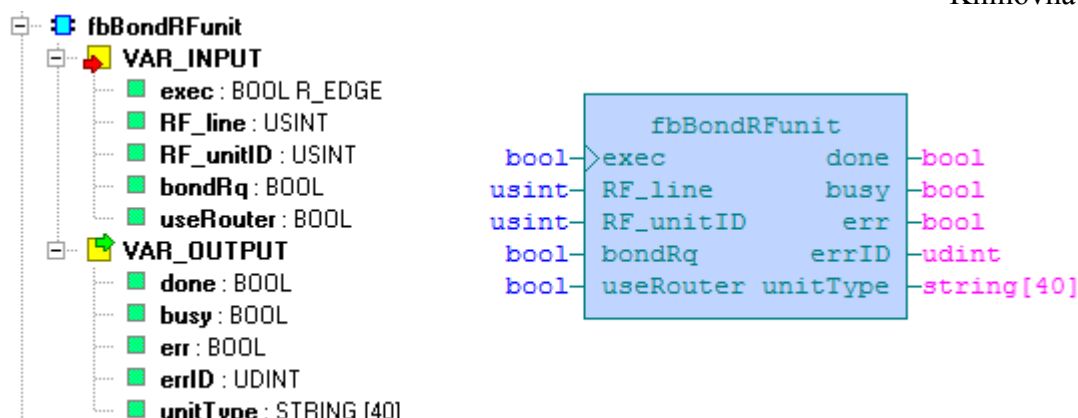
```

Následující obrázek ukazuje chování funkčního bloku fbTick v grafické podobě.



5.2 Funkční blok *fbBondRFunit*

Knihovna : SysLib



Funkční blok *fbBondRFunit* slouží pro spárování RF jednotky s RF masterem v síti Rfox. Tento funkční blok spustí párování v RF masterovi, který zastaví výměnu dat s jednotkami v Rfox síti a zahájí vysílání párovacích informací. Aby došlo k úspěšnému spárování je třeba, aby i párovaná jednotka byla v párovacím režimu. Dále musí souhlasit nastavená adresa RF jednotky (viz funkce *SetAddressRFunit*). Během párování je zastavena výměna dat v Rfox síti což znamená, že v aplikačním programu nebudou k dispozici data poskytovaná jednotkami v Rfox síti. Spárování trvá typicky 12 sekund. Pokud se párování nezdaří do 20 sekund, funkční blok vyhlásí chybu timeoutu.

Funkční blok *fbBondRFunit* umožňuje spárovat RF jednotku s RF masterem a nebo zrušit stávající párování v RF masterovi. Pokud párujeme na RF jednotku na pozici, na které už je nějaká jednotka napárovaná, funkční blok nejprve automaticky zruší stávající párování a pak se pokusí napárovat novou RF jednotku. Pokud budeme pouze rušit párování některé existující RF jednotky je třeba si uvědomit, že volání bloku *fbBondRFunit* s parametrem *RF_bond* := 0 zruší párování pouze v RF masterovi – příslušnou RF jednotku je třeba také „odpárovat“ vyjmutím napájecí baterie (aby se nesnažila vysílat data do Rfox sítě). Parametr *useRouter* říká, jestli po napárování bude RF master komunikovat s RF jednotkou přímo nebo přes router.

Tento funkční blok je podporován v centrálních jednotkách řady Foxtrot od v6.6. Do knihovny SysLib je blok zařazen od verze SysLib_v27.

Popis proměnných :

	Proměnná	Typ	Význam
VAR_INPUT			
	<i>exec</i>	BOOL R_EDGE	Žádost o spárování, náběžná hrana zahájí akci
	<i>RF_line</i>	USINT	Číslo RF linky (viz konstanty MI_RF, RF0_RF,...,RF6_RF)
	<i>RF_unitID</i>	USINT	Číslo pozice RF jednotky (1,...,64)
	<i>bondRq</i>	BOOL	0 = pouze zrušit párování, 1 = napárovat novou jednotku
	<i>useRouter</i>	BOOL	Po napárování bude komunikace mezi RF jednotkou a RF masterem probíhat přes router (má smysl pouze při párování, tj. při bondRq=1)

	<i>Proměnná</i>	<i>Typ</i>	<i>Význam</i>
VAR_OUTPUT			
	<i>done</i>	BOOL	Pokud je párování úspěšně ukončeno má tato proměnná hodnotu TRUE, během párování má hodnotu FALSE
	<i>busy</i>	BOOL	Pokud probíhá párování má tato proměnná hodnotu TRUE
	<i>err</i>	BOOL	Pokud je párování ukončeno s chybou má tato proměnná hodnotu TRUE, během párování má hodnotu FALSE
	<i>errID</i>	UDINT	Kód chyby 0 akce proběhla úspěšně (bez chyby) 1 neznámá chyba 2 nepodařilo se zrušit párování 3 nepodařilo se spárovat novou jednotku 4 překročen max. povolený čas na párování (cca 20 sec)
	<i>unitType</i>	STRING	Typ spárované jednotky, např. “R-RC-0001R 02H0102 AN 0013” kde “R-RC-0001R” je typ RF jednotky “02H0102” je verze HW a SW RF jednotky “AN 0013” je výrobní číslo RF jednotky (poslední 4 číslice jsou zároveň Hw adresa RF jednotky) Pokud se párování nepodaří, <i>unitType</i> je prázdný řetězec Pokud se ruší párování (bondRq = 0) <i>unitType</i> je prázdný řetězec

Příklad programu s použitým funkčním blokem *fbBondRFunit* :

```

VAR_GLOBAL CONSTANT
  NUM_RF_UNIT : USINT := 4;
END_VAR

TYPE
  TcustomRF : STRUCT
    valid    : BOOL;
    address  : WORD;
    code     : WORD;
  END_STRUCT;
END_TYPE

VAR_GLOBAL RETAIN
  customRF : ARRAY[1..NUM_RF_UNIT] OF TcustomRF;
END_VAR

PROGRAM Test_RF_net
  VAR
    i          : USINT;
    RF_unit    : ARRAY[1..NUM_RF_UNIT] OF TRFunitInfo;
    id         : USINT;
    newAdr     : WORD;
    setAdr     : BOOL;
    result     : BOOL;
    bondRq     : BOOL;
    unBondRq   : BOOL;
    bond       : BOOL;
    router     : BOOL;
    start      : SR;
    BondRFunit : fbBondRFunit;
  
```

```

END_VAR

start(S1 := bondRq OR unBondRq);
IF start.Q1 THEN
  // select action (bond / unbond)
  IF bondRq THEN bond := 1; ELSE bond := 0; END_IF;
  // check ID of RF unit
  IF id > 0 AND id <= NUM_RF_UNIT THEN
    BondRFunit( exec := 1,
                RF_line := RF0_RF, RF_unitID := id,
                bondRq := bond, useRouter := router);
    // wait for end of action
    IF NOT BondRFunit.busy THEN
      IF BondRFunit.done OR BondRFunit.err THEN
        bondRq := 0; unBondRq := 0; start(R := 1);
      END_IF;
    END_IF;
  ELSE
    bondRq := 0; unBondRq := 0; start(R := 1);
  END_IF;
ELSE
  BondRFunit(exec := 0);

  // read info about RF units and set customer address
  FOR i := 1 TO NUM_RF_UNIT DO
    IF RFunitInfo( RF_line := RF0_RF, RF_unitID := i, unitInfo := RF_unit[i])
    THEN
      IF not customRF[i].valid OR customRF[i].code <> RF_unit[i].code THEN
        customRF[i].address := RF_unit[i].address;
        customRF[i].code    := RF_unit[i].code;
        customRF[i].valid   := TRUE;
      ELSE
        IF RF_unit[i].address <> customRF[i].address THEN
          SetAddressRFunit( RF_line    := MI_CIB1,
                           RF_unitID  := i,
                           RF_addr    := customRF[i].address);

          END_IF;
        END_IF;
      END_IF;
    END_FOR;

    // set new customer address of RF unit
    IF setAdr THEN
      IF id > 0 AND id <= NUM_RF_UNIT THEN
        IF newAdr <> RF_unit[id].address THEN
          result := SetAddressRFunit( RF_line    := RF0_RF,
                                     RF_unitID  := id,
                                     RF_addr    := newAdr);

          IF result THEN
            customRF[id].address := newAdr;
          END_IF;
        END_IF;
      END_IF;
      setAdr := FALSE;
    END_IF;
  END_IF;
END_PROGRAM

```

Uvedený program řeší spárování RF jednotky s RF masterem a také umožňuje změnu HW adresy RF jednotky.

Příklad konfigurace sítě Rfox je vidět na následujícím obrázku.

Správce jednotek/zařízení

MI **RF0**

RF

ID	HW adr...	ID ...	Typ jednotky/zařízení	Stav	Název jednotky/zařízení/v...
1	0013	3	R-RC-0001R	-	room control
2	0001	3	R-WS-0200R	-	R-WS-0200R

Nastavení

Výběr jednotky/zařízení

Název jednotky

room control

HW adresa jednotky

0013

Visualizace vstupů

☐ Exportovat pro vizualizaci

Pojmenování / alias

room_control_IN

Visualizace výstupů

☐ Exportovat pro vizualizaci

Pojmenování / alias

room_control_OUT

☐ Zobrazit všechna zařízení všech jednotek

Přidat jednotku Vymazat jednotku Vymazat vše

HW konfigurace RF

Načíst konfiguraci z CPU

☐ Bázová adresa zón 0

OK Zrušit Nápověda

Na následující web stránce je zobrazen stav čtyřech RF jednotek spolupracujících s masterem RF-1131 v síti RF0_RF a nastavovací pole ve žlutém rámečku umožňují nastavit novou HW adresu RF jednotky a spárovat novou jednotku s RF masterem např. v případě výměny jednotky za jiný kus.

EXTENAL RF MASTER : RF0_RF

ID	line defined	unit defined	NET	BND	SLP	COM	INI	HW ADDR	TYPE	DESCRIPTION
1	1	1	1	1	1	1	0	0013	R-RC-0001R	room control
2	1	1	1	1	1	0	0	0001	R-WS-0200R	R-WS-0200R
3	1	0	0	0	0	0	0	0000		
4	1	0	0	0	0	0	0	0000		

NEW SETTINGS

ID 1 NEW HW ADDR 0013

SET NEW ADR UNBOND

☐ USE ROUTER BOND

Z uvedeného obrázku lze vyčíst, že v tomto konkrétním případě je v aplikačním programu PLC definována jedna bezdrátová síť Rfox se dvěma RF jednotkami. První RF jednotka je typu R-RC-0001R, druhá jednotka je typu R-WS-0200R. Obě jednotky jsou spárované s RF masterem a obě jsou napájené z baterie (pracují ve spícím módu).

Ve sloupci „line defined“ jsou zobrazeny proměnné *TestRF.RF_unit[i].line_defined*, ve sloupci „unit defined“ jsou zobrazeny proměnné *TestRF.RF_unit[i].unit_defined*, sloupce „NET“, „BND“, „SLP“, „COM“ a „INI“ obsahují odpovídající položky proměnné *TestRF.RF_unit[i].state* (tedy *TestRF.RF_unit[i].state.NET*, atd). Sloupec „HW ADR“ zobrazuje proměnné *TestRF.RF_unit[i].address*, sloupec „TYPE“ zobrazuje proměnné *TestRF.RF_unit[i].type* a konečně sloupec „DESCRIPTION“ zobrazuje proměnné *TestRF.RF_unit[i].description*. Stav těchto proměnných je trvale aktualizován funkcí *RFunitInfo()*. Všechny dosud uvedené proměnné jsou „Read Only“.

Nastavení nové HW adresy RF jednotky se provádí přes zadávací pole „ID“, které je navázané na proměnnou *TestRF.id*, dále přes zadávací pole „NEW ADR“, které umožňuje měnit proměnnou *TestRF.newAdr* a konečně přes dvoustavový obrázek „SET NEW ADR“, který umožňuje nastavit proměnnou *TestRF.setAdr* na hodnotu TRUE. Tlačítka „BOND“ (spárovat) a „UNBOND“ (zrušit párování) jsou navázány na proměnné *TestRF.bondRq* a *TestRF.unBondRq*. Zatrhávací box „USE ROUTER“ je spojen s proměnnou *TestRF.router*.

Obecný postup spárování jedné RF jednotky

Ve volání instance funkčního bloku *fbBondRFunit* nejprve nastavit proměnnou *exec* na hodnotu TRUE. Na náběžnou hranu proměnné *exec* RF master ukončí běžnou výměnu dat v síti Rfox a pokusí se v několika krocích připárovat novou RF jednotku. Pokud je již na pozici dané parametrem *RF_unitID* nějaká připárovaná RF jednotka, tak RF master nejprve automaticky zruší párování s touto jednotkou. Poté se na tuto pozici pokusí připárovat novou RF jednotku. Aby párování proběhlo úspěšně je nutné aby odpovídala nastavená HW adresa RF jednotky. Tu lze v případě potřeby změnit funkcí *SetAddressRFunit()*, která musí být zavolaná před zahájením párování. Dále je nezbytné uvést párovanou RF jednotku do párovacího módu a stisknout tlačítko pro připárování na RF jednotce. Během párování nastavuje funkční blok *fbBondRFunit* výstup *busy*, úspěšné spárování signalizuje výstup *done*, pokud se spárování nepodaří je nastaven výstup *err* a výstup *errID* obsahuje kód chyby.

Další podrobnosti o RF jednotkách lze najít v dokumentaci **Bezdrátové periferní moduly řady Rfox**, obj. číslo TXV 004 14.01.