

# **Knihovna SysLib**

**TXV 003 48.01  
sedmnácté vydání  
červenec 2024  
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
Říjen 2013	9	Opraven příklad použití funkčního bloku <i>fbBondRFunit</i>
Červen 2018	10	Doplněn popis bloků fbTPR, fbLoadRemFromFile() a fbSaveRemToFile()
Červenec 2019	11	Doplněny informace o funkcích CIBunitInfo2() a SetAddressCIBunit2() Popis odpovídá knihovně SysLib_v41
Květen 2021	12	Doplněny informace o funkcích GetProgramInfo2(), GetPlcInfo2(), ResetLTE2() a LoadPackage2() Doplněn popis fbStopwatch100us() Popis odpovídá knihovně SysLib_v45
Srpen 2021	13	Doplněny informace o globálních proměnných, které jsou mapované do systémových registrů %S Popis odpovídá knihovně SysLib_v46
Červen 2022	14	Doplněn popis GetVarAbsLoc(), GetVarAbsLocBool(), fbSetConfigWebPSW2(), fbSetAvahiParam2() a GetAvahiHostName2() Popis odpovídá knihovně SysLib_v48
Prosinec 2022	15	Doplněn popis TestHotRestartFailed2() a fbFlashRem2() Popis odpovídá knihovně SysLib_v49

Datum	Vydání	Popis změn
Únor 2023	16	Doplněn popis funkce RandomU Popis odpovídá knihovně SysLib_v50
Červenec 2024	17	Úprava funkce ModuleInfo2 Popis odpovídá knihovně SysLib_v51

## OBSAH

<b>1 Úvod</b>	<b>6</b>
<b>2 Datové typy</b>	<b>6</b>
2.1 Typ TIOSystemInfo	7
2.2 Typ TmoduleInfo	8
2.3 Typ TSYSTEM_S	9
2.4 Typ TTecoDateTime	11
2.5 Typ TCIBunitState	12
2.6 Typ TCIBunitInfo	13
2.7 Typ TRFunitState	14
2.8 Typ TRFunitInfo	15
2.9 Typ TProgramInfo2	16
2.10 Typ TPlcInfo2	17
<b>3 Globální proměnné a konstanty</b>	<b>18</b>
<b>4 Funkce</b>	<b>22</b>
4.1 Funkce SetSummerTime	24
4.2 Funkce IsSummerTime	25
4.3 Funkce SetWinterTime	26
4.4 Funkce IsWinterTime	27
4.5 Funkce GetDate	28
4.6 Funkce GetTime	29
4.7 Funkce GetDateTime	30
4.8 Funkce GetRTC	31
4.9 Funkce SetRTC	32
4.10 Funkce TecoDT_TO_DT	33
4.11 Funkce DT_TO_TecoDT	34
4.12 Funkce IOSystemInfo	35
4.13 Funkce ModuleInfo	36
4.14 Funkce ModuleInfo2	37
4.15 Funkce CIBunitInfo	38
4.16 Funkce CIBunitInfo2	40
4.17 Funkce SetAddressCIBunit	41

4.18	Funkce SetAddressCIBunit2.....	44
4.19	Funkce Memcpy.....	45
4.20	Funkce MemcpyEx.....	47
4.21	Funkce Memset.....	49
4.22	Funkce MemsetEx.....	50
4.23	Funkce Memcmp.....	52
4.24	Funkce MemcmpEx.....	53
4.25	Funkce IncreaseMaxCycleTime.....	55
4.26	Funkce VerifyWebPSW.....	56
4.27	Funkce SetWebPSW.....	58
4.28	Funkce VerifyWebMAC.....	60
4.29	Funkce SetWebMAC.....	62
4.30	Funkce ProgramLock.....	64
4.31	Funkce SystemDisplayBacklightOn.....	65
4.32	Funkce SystemDisplayBacklightOff.....	66
4.33	Funkce RFunitInfo.....	67
4.34	Funkce SetAddressRFunit.....	69
4.35	Funkce GetVarValueByName.....	73
4.36	Funkce GetVarDescByName.....	74
4.37	Funkce GetVarNameByAdr.....	75
4.38	Funkce SetVarValueByName.....	76
4.39	Funkce GetProgramInfo2.....	77
4.40	Funkce GetPlcInfo2.....	78
4.41	Funkce ResetLTE2.....	79
4.42	Funkce LoadNewPackage2.....	80
4.43	Funkce GetVarAbsLoc.....	81
4.44	Funkce GetVarAbsLocBool.....	82
4.45	Funkce GetAvahiHostName2.....	83
4.46	Funkce TestHotRestartFailed2.....	84
4.47	Funkce RandomU.....	85
5	<i>Funkční bloky</i> .....	87
5.1	Funkční blok fbTick.....	88
5.2	Funkční blok fbBondRFunit.....	90
5.3	Funkční blok fbTPR.....	96
5.4	Funkční blok fbSaveRemToFile.....	97
5.5	Funkční blok fbLoadRemFromFile.....	100
5.6	Funkční blok fbStopwatch100us.....	103
5.7	Funkční blok fbSetConfigWebPsw2.....	105

<b>5.8 Funkční blok fbSetAvahiParam2.....</b>	<b>107</b>
<b>5.9 Funkční blok fbFlashRem2.....</b>	<b>109</b>

## 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 obsahuje deklarace funkcí, funkčních bloků, datových typů a konstant. Knihovna SysLib patří mezi standardně dodávané knihovny.

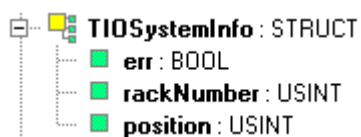
Pokud chceme funkce z knihovny SysLib použít v aplikačním programu PLC, je třeba nejprve přidat tuto knihovnu do projektu. Knihovna je dodávaná jako součást instalace prostředí Mosaic.

## 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)
<i>TProgramInfo2</i>	STRUCT	Informace o uživatelském programu (pouze pro Foxtrot 2 a TC800)
<i>TPlcInfo2</i>	STRUCT	Informace o uživatelském programu (pouze pro Foxtrot 2 a TC800)

## 2.1 Typ *TIOSystemInfo*

Knihovna : *SysLib*

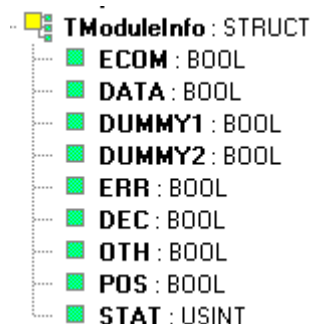
Typ *TIOSystemInfo* je struktura nesoucí informaci o stavu IO systému PLC, kterou vrací funkce *IOSystemInfo*.

Význam jednotlivých položek struktury *TIOSystemInfo* je následující:

<i>TIOSystemInfo</i>	Typ	Význam
<i>.err</i>	BOOL	0 = I/O systém PLC je bez chyby 1 = v I/O systému PLC je nějaký problém
<i>.rackNumber</i>	USINT	při <i>err</i> = 1 udává číslo rámu PLC, kde je umístěn modul signalizující nějaký problém
<i>.position</i>	USINT	při <i>err</i> = 1 udává číslo pozice v rámu PLC, kde je umístěn modul signalizující nějaký problém

Viz také Funkce *IOSystemInfo*

## 2.2 Typ TmoduleInfo



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:

<i>TModuleInfo</i>	<i>Typ</i>	<i>Význam</i>
<i>.ECOM</i>	BOOL	1 = chyba komunikace mezi CPU a I/O modulem
<i>.DATA</i>	BOOL	1 = data poskytovaná modulem jsou platná
<i>.DUMMY1</i>	BOOL	nepoužito
<i>.DUMMY2</i>	BOOL	nepoužito
<i>.ERR</i>	BOOL	1 = chyba I/O modulu
<i>.DEC</i>	BOOL	1 = modul má platnou deklaraci v programu PLC
<i>.OTH</i>	BOOL	1 = typ I/O modulu neodpovídá deklaraci v programu PLC
<i>.POS</i>	BOOL	1 = modul je přítomen v dané pozici
<i>.STAT</i>	USINT	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:

```

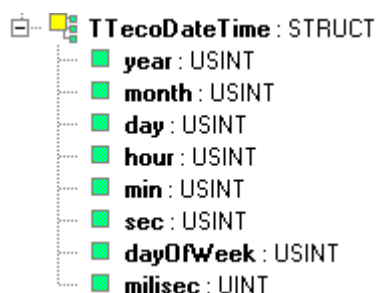
TYPE
  TSYSTEM_S : STRUCT
    S0          : BYTE; // %S0   výsledky aritmetických operací
    S1          : BYTE; // %S1   výsledky logických operací
    S2_0        : BOOL; // %S2.0 stav služebního vstupu SP
    S2_1        : BOOL; // %S2.1 stav služebního vstupu MS
    S2_2        : BOOL; // %S2.2 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
SW56 : WORD; // %SW56 rezerva
CPU_DI : BYTE; // %S58 vstupy obsluhované centrálou
CPU_DO : BYTE; // %S59 výstupy obsluhované centrálou
SL60 : UDINT; // %SL70 rezerva
SIZE_OF_RETAIN_ZONE : UDINT; // %SL64 velikost remanentní zóny v bytech
UTC_OFFSET : INT; // %SW68 offset proti UTC
CRC_OF_USER_PROGRAM : WORD; // %SW70 CRC uživatelského programu
CRC_OF_HEADER_PROGRAM : WORD; // %SW72 CRC hlavičky uživ. programu
S74 : BYTE; // %S74 rezerva
S75 : BYTE; // %S75 rezerva
S76 : BYTE; // %S76 rezerva
S77 : BYTE; // %S77 rezerva
COUNTER_MILLISECONDS : UINT; // %SW78 desetinná část system. času [ms]
PLC_PRIVATE_AREA : ARRAY[0..19] OF BYTE; // %S80..%S99 vyhrazeno CPU
END_STRUCT;
END_TYPE
```

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

## 2.4 Typ *TTecoDateTime*

Knihovna : *SysLib*

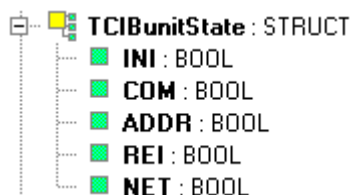
Typ *TTecoDateTime* je struktura s údaji o datumu a času. Tuto strukturu používají konverzní funkce *TecoDT\_TO\_DT* a *DT\_TO\_TecoDT*.

Význam jednotlivých položek struktury *TTecoDateTime* je následující:

<i>TTecoDateTime</i>	Typ	Význam
<i>.year</i>	USINT	rok (poslední dvě číslice letopočtu)
<i>.month</i>	USINT	měsíc (1 .. 12)
<i>.day</i>	USINT	den (1 .. 28/29/30/31)
<i>.hour</i>	USINT	hodina (0 .. 23)
<i>.min</i>	USINT	minuta (0 .. 59)
<i>.sec</i>	USINT	sekunda (0 .. 59)
<i>.dayOfWeek</i>	USINT	den v týdnu (1 = pondělí, 2 = úterý, ..., 7 = neděle)
<i>.milisec</i>	UINT	milisekunda

Viz také Funkce *TecoDT\_TO\_DT*, Funkce *DT\_TO\_TecoDT*

## 2.5 Typ *TCIBunitState*

Knihovna : *SysLib*

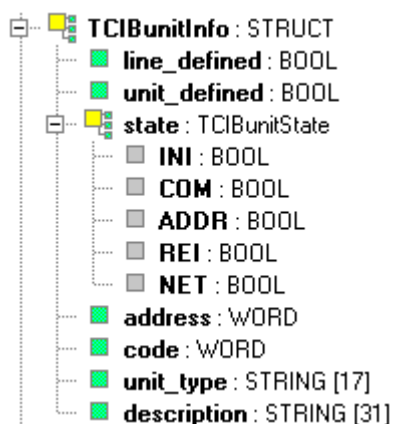
Typ *TCIBunitState* je struktura nesoucí informaci o stavu jednotky na CIB sběrnici, kterou vrací funkce *CIBunitInfo()* jako součást informací o CIB jednotce.

Význam jednotlivých položek struktury *TCIBunitState* je následující:

<i>TCIBunitState</i>	<i>Typ</i>	<i>Význam</i>
<i>.INI</i>	BOOL	CIB jednotka je inicializovaná
<i>.COM</i>	BOOL	komunikace s CIB jednotkou je bez závad
<i>.ADDR</i>	BOOL	adresa CIB jednotky byla akceptovaná
<i>.DUMMY3</i>	BOOL	
<i>.REI</i>	BOOL	příznak reinitializace CIB jednotky
<i>.DUMMY5</i>	BOOL	
<i>.ALT</i>	BOOL	změna tohoto bitu signalizuje nová data (pouze Foxtrot 2)
<i>.NET</i>	BOOL	CIB jednotka je definovaná v programu a obsluhovaná centrální jednotkou

Viz také Typ *TCIBunitInfo*, Funkce *CIBunitInfo*

## 2.6 Typ *TCIBunitInfo*

Knihovna : *SysLib*

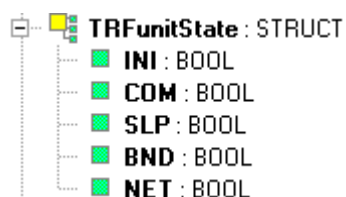
Typ *TCIBunitInfo* je struktura nesoucí informace o jednotce na CIB sběrnici, kterou vrací funkce *CIBunitInfo()*.

Význam jednotlivých položek struktury *TCIBunitInfo* je následující:

<i>TCIBunitInfo</i>	Typ	Význam
<i>.line_defined</i>	BOOL	CIB linka je definovaná v HW konfiguraci PLC
<i>.unit_defined</i>	BOOL	CIB jednotka je definovaná v HW konfiguraci PLC
<i>.state</i>	TCIBunitState	stav CIB jednotky (viz Typ TCIBunitState)
<i>.address</i>	WORD	aktuálně nastavená HW adresa CIB jednotky
<i>.code</i>	WORD	kód CIB jednotky
<i>.unit_type</i>	STRING[17]	typové označení CIB jednotky
<i>.description</i>	STRING[31]	popis CIB jednotky

Viz také Typ *TCIBunitState*, Funkce *CIBunitInfo*

## 2.7 Typ *TRFunitState*

Knihovna : *SysLib*

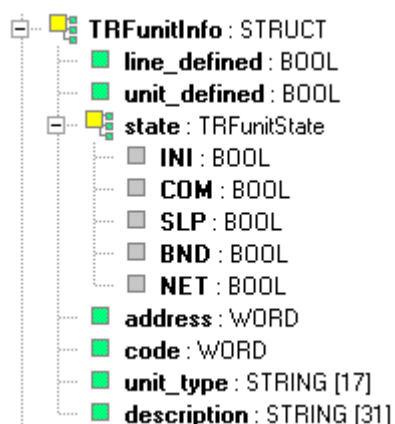
Typ *TRFunitState* je struktura nesoucí informaci o stavu RF jednotky, kterou vrací funkce *RFunitInfo()* jako součást informací o RFox jednotce. Tyto informace platí pouze pro Foxtrot 1.

Význam jednotlivých položek struktury *TRFunitState* je následující:

<i>TRFunitState</i>	<i>Typ</i>	<i>Význam</i>
<i>.INI</i>	BOOL	RF jednotka je inicializovaná
<i>.COM</i>	BOOL	komunikace s RF jednotkou je bez závad
<i>.DUMMY2</i>	BOOL	
<i>.DUMMY3</i>	BOOL	
<i>.DUMMY4</i>	BOOL	
<i>.SLP</i>	BOOL	1 = RF jednotka může přecházet do sleep režimu 0 = RF jednotka je trvale aktivní
<i>.BND</i>	BOOL	RF master považuje jednotku za spárovanou
<i>.NET</i>	BOOL	RF jednotka je definovaná v programu a obsluhovaná centrální jednotkou

Viz také Typ *TRFunitInfo*, Funkce *RFunitInfo*

## 2.8 Typ *TRFunitInfo*

Knihovna : *SysLib*

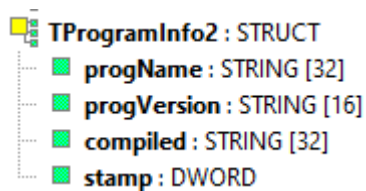
Typ *TRFunitInfo* je struktura nesoucí informace o RF jednotce (RFox), kterou vrací funkce *RfunitInfo()*. Tyto informace platí pouze pro Foxtrot 1.

Význam jednotlivých položek struktury *TRFunitInfo* je následující:

<i>TRFunitInfo</i>	Typ	Význam
<i>.line_defined</i>	BOOL	RF linka je definovaná v HW konfiguraci PLC
<i>.unit_defined</i>	BOOL	RF jednotka je definovaná v HW konfiguraci PLC
<i>.state</i>	TRFunitState	stav RF jednotky (viz Typ TRFunitState)
<i>.address</i>	WORD	aktuálně nastavená HW adresa RF jednotky
<i>.code</i>	WORD	kód RF jednotky
<i>.unit_type</i>	STRING[17]	typové označení RF jednotky
<i>.description</i>	STRING[31]	popis RF jednotky

Viz také Typ TRFunitState, Funkce RFunitInfo

## 2.9 Typ *TProgramInfo2*

Knihovna : *SysLib*

Typ *TProgramInfo2* je struktura s informacemi o uživatelském programu v PLC, kterou vrací funkce *GetProgramInfo2()*. Tyto informace platí pouze pro Foxtrot 2 a TC800.

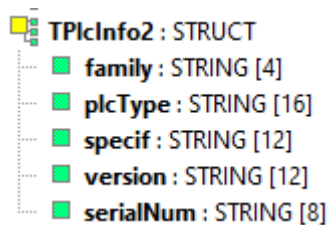
Význam jednotlivých položek struktury *TProgramInfo2* je následující:

<i>TProgramInfo2</i>	Typ	Význam
<i>.progName</i>	STRING[32]	název programu (např. TEST_PLC)
<i>.progVersion</i>	STRING[16]	verze programu (např. v1.0.0.0)
<i>.compiled</i>	STRING[32]	datum a čas překladu programu (např. 2021-05-15 13:21:11)
<i>.stamp</i>	DWORD	CRC razítko programu (např. 16#BCAF4F22)

Viz také Funkce *GetProgramInfo2*



## 2.10 Typ *TPlcInfo2*

Knihovna : *SysLib*

Typ *TPlcInfo2* je struktura obsahující informace o PLC, kterou vrací funkce *GetPlcInfo2()*. Tyto informace platí pouze pro Foxtrot 2 nebo TC800.

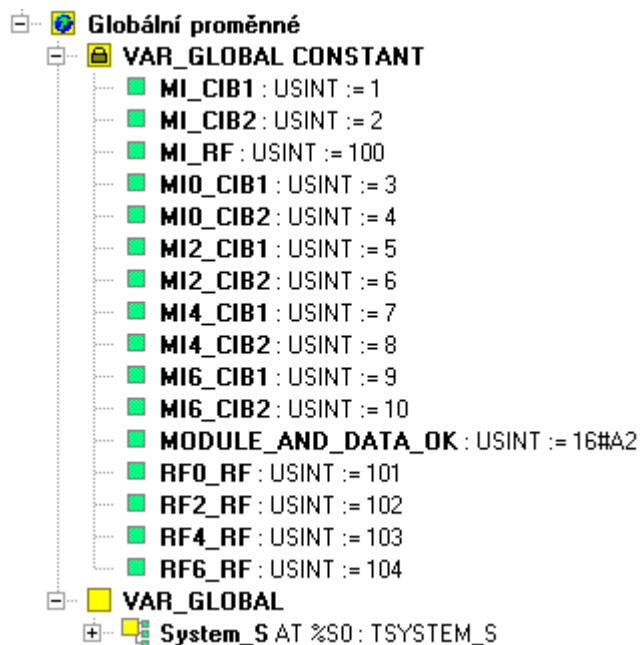
Význam jednotlivých položek struktury *TPlcInfo2* je následující:

<i>TPlcInfo2</i>	<i>Typ</i>	<i>Význam</i>
<i>.family</i>	STRING[4]	rodina PLC (např. F2x pro Foxtrot 2)
<i>.plcType</i>	STRING[16]	typ PLC (např. CP2005)
<i>.specif</i>	STRING[12]	specifikace (např. 11NSNN)
<i>.version</i>	STRING[12]	verze firmwaru PLC (např. 2.1.044)
<i>.serialNum</i>	STRING[8]	výrobní číslo (např. K6 0038)

Viz také Funkce *GetPlcInfo2*

### 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*. Tyto konstanty lze použít pouze pro Foxtrot 1.

Konstanty *MI\_RF*, *RF0\_RF* až *RF6\_RF* slouží ke specifikaci bezdrátové RF sítě ve funkcích *RFunitInfo* a *SetAddressRFunit*. Tyto konstanty lze použít pouze pro Foxtrot 1.

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>	<b>BYTE</b>	%S0	výsledky aritmetických operací
<i>System_S.S1</i>	<b>BYTE</b>	%S1	výsledky logických operací
<i>System_S.S2_0</i>	<b>BOOL</b>	%S2.0	stav služebního vstupu SP
<i>System_S.S2_1</i>	<b>BOOL</b>	%S2.1	stav služebního vstupu MS
<i>System_S.S2_2</i>	<b>BOOL</b>	%S2.2	režim RUN
<i>System_S.S2_3</i>	<b>BOOL</b>	%S2.3	teplý restart
<i>System_S.S2_4</i>	<b>BOOL</b>	%S2.4	studený restart
<i>System_S.OUTPUTS_ARE_ENABLED</i>	<b>BOOL</b>	%S2.5	výstupy odblokovány
<i>System_S.S2_6</i>	<b>BOOL</b>	%S2.6	první průchod cyklem bez restartu
<i>System_S.CYCLE_TIME_WARNING</i>	<b>BOOL</b>	%S2.7	překročena první mez doby cyklu
<i>System_S.LAST_CYCLE_TIME_10MS</i>	<b>USINT</b>	%S3	doba minulého cyklu v 10 ms
<i>System_S.CYCLE_COUNTER</i>	<b>USINT</b>	%S4	čítač cyklu
<i>System_S.COUNTER_10MS</i>	<b>USINT</b>	%S5	čítač desítek milisekund
<i>System_S.COUNTER_SECONDS</i>	<b>USINT</b>	%S6	čítač sekund systémového času
<i>System_S.COUNTER_MINUTES</i>	<b>USINT</b>	%S7	čítač minut systémového času
<i>System_S.COUNTER_HOURS</i>	<b>USINT</b>	%S8	čítač hodin systémového času
<i>System_S.COUNTER_DAYS_OF_WEEK</i>	<b>USINT</b>	%S9	čítač dnů v týdnu
<i>System_S.COUNTER_DAYS_OF_MONTH</i>	<b>USINT</b>	%S10	čítač dnů v měsíci
<i>System_S.COUNTER_MONTHS</i>	<b>USINT</b>	%S11	čítač měsíců
<i>System_S.COUNTER_YEARS</i>	<b>USINT</b>	%S12	čítač roků
<i>System_S.PERIOD_PULSE_100MS</i>	<b>BOOL</b>	%S13.0	pulz s periodou 100 ms
<i>System_S.PERIOD_PULSE_500MS</i>	<b>BOOL</b>	%S13.1	pulz s periodou 500 ms
<i>System_S.PERIOD_PULSE_1SEC</i>	<b>BOOL</b>	%S13.2	pulz s periodou 1 s
<i>System_S.PERIOD_PULSE_10SEC</i>	<b>BOOL</b>	%S13.3	pulz s periodou 10 s
<i>System_S.PERIOD_PULSE_1MIN</i>	<b>BOOL</b>	%S13.4	pulz s periodou 1 min
<i>System_S.PERIOD_PULSE_10MIN</i>	<b>BOOL</b>	%S13.5	pulz s periodou 10 min
<i>System_S.PERIOD_PULSE_1HOUR</i>	<b>BOOL</b>	%S13.6	pulz s periodou 1 hod
<i>System_S.PERIOD_PULSE_1DAY</i>	<b>BOOL</b>	%S13.7	pulz s periodou 1 den
<i>System_S.COUNTER_100MS</i>	<b>UINT</b>	%SW14	čítač v 100m
<i>System_S.COUNTER_1SEC</i>	<b>UINT</b>	%SW16	čítač v 1s
<i>System_S.COUNTER_10SEC</i>	<b>UINT</b>	%SW18	čítač v 10s
<i>System_S.R_EDGE_100MS</i>	<b>BOOL</b>	%S20.0	náběžná hrana 1x za 100 ms

System_S.R_EDGE_500MS	<b>BOOL</b>	%S20.1	náběžná hrana 1x za 500 ms
System_S.R_EDGE_1SEC	<b>BOOL</b>	%S20.2	náběžná hrana 1x za 1 s
System_S.R_EDGE_10SEC	<b>BOOL</b>	%S20.3	náběžná hrana 1x za 10 s
System_S.R_EDGE_1MIN	<b>BOOL</b>	%S20.4	náběžná hrana 1x za 1 min
System_S.R_EDGE_10MIN	<b>BOOL</b>	%S20.5	náběžná hrana 1x za 10 min
System_S.R_EDGE_1HOUR	<b>BOOL</b>	%S20.6	náběžná hrana 1x za 1 hod
System_S.R_EDGE_1DAY	<b>BOOL</b>	%S20.7	náběžná hrana 1x za 1 den
System_S.F_EDGE_100MS	<b>BOOL</b>	%S21.0	sestupná hrana 1x za 100 ms
System_S.F_EDGE_500MS	<b>BOOL</b>	%S21.1	sestupná hrana 1x za 500 ms
System_S.F_EDGE_1SEC	<b>BOOL</b>	%S21.2	sestupná hrana 1x za 1 s
System_S.F_EDGE_10SEC	<b>BOOL</b>	%S21.3	sestupná hrana 1x za 10 s
System_S.F_EDGE_1MIN	<b>BOOL</b>	%S21.4	sestupná hrana 1x za 1 min
System_S.F_EDGE_10MIN	<b>BOOL</b>	%S21.5	sestupná hrana 1x za 10 min
System_S.F_EDGE_1HOUR	<b>BOOL</b>	%S21.6	sestupná hrana 1x za 1 hod
System_S.F_EDGE_1DAY	<b>BOOL</b>	%S21.7	sestupná hrana 1x za 1 den
System_S.LAST_CYCLE_TIME_100US	<b>UINT</b>	%SW22	doba minulého cyklu v 100 µs
System_S.S24	<b>BYTE</b>	%S24	řídící masky procesů
System_S.S25	<b>BYTE</b>	%S25	řídící masky procesů
System_S.S26	<b>BYTE</b>	%S26	řídící masky procesů
System_S.S27	<b>BYTE</b>	%S27	řídící masky procesů
System_S.S28	<b>BYTE</b>	%S28	řídící masky procesů
System_S.S29	<b>BYTE</b>	%S29	řídící masky procesů
System_S.SL30	<b>UDINT</b>	%SL30	rezerva
System_S.S34	<b>BYTE</b>	%S34	hlavní kód chyby
System_S.BAT_ERR	<b>BOOL</b>	%S35.0	chyba zálohovací baterie
System_S.S35_1	<b>BOOL</b>	%S35.1	rezerva
System_S.S35_2	<b>BOOL</b>	%S35.2	stav tlačítka MODE
System_S.S35_3	<b>BOOL</b>	%S35.3	stav tlačítka SET
System_S.S35_4	<b>BOOL</b>	%S35.4	rezerva
System_S.S35_5	<b>BOOL</b>	%S35.5	rezerva
System_S.IS_SUMMER_TIME	<b>BOOL</b>	%S35.6	indikace letního času
System_S.SUMMER_TIME_REQUEST	<b>BOOL</b>	%S35.7	žádost o přechod na letní čas
System_S.CPU_TEMPERATURE	<b>USINT</b>	%S36	teplota procesorového modulu [°C]
System_S.MOSAIC_IS_CONNECTED	<b>BOOL</b>	%S37.0	Mosaic komunikuje s PLC
System_S.S37_1	<b>BOOL</b>	%S37.1	rezerva
System_S.S37_2	<b>BOOL</b>	%S37.2	rezerva
System_S.IO_IS_FIXED	<b>BOOL</b>	%S37.3	příznak fixace IO (pouze CP-2xxx)
System_S.S37_4	<b>BOOL</b>	%S37.4	rezerva
System_S.USB_DISK_READY	<b>BOOL</b>	%S37.5	USB disk připraven k použití
System_S.IS_USB_DISK	<b>BOOL</b>	%S37.6	USB disk zasunut do konektoru
System_S.EEPROM_IS_ON	<b>BOOL</b>	%S37.7	uživ. program uložen ve Flash
System_S.S38	<b>BYTE</b>	%S38	verze uživatelského programu
System_S.S39	<b>BYTE</b>	%S39	verze uživatelského programu
System_S.S40	<b>BYTE</b>	%S40	verze FW systému
System_S.S41	<b>BYTE</b>	%S41	verze FW systému
System_S.S42	<b>BYTE</b>	%S42	řada CPU
System_S.S43	<b>BYTE</b>	%S43	příznaky chování PLC
System_S.S44	<b>BYTE</b>	%S44	typ překladače
System_S.S45	<b>BYTE</b>	%S45	typ překladače
System_S.S46	<b>BYTE</b>	%S46	varovná mez doby cyklu
System_S.S47	<b>BYTE</b>	%S47	max. mez doby doby cyklu
System_S.S48	<b>BYTE</b>	%S48	úplný kód chyby PLC
System_S.S49	<b>BYTE</b>	%S49	úplný kód chyby PLC
System_S.S50	<b>BYTE</b>	%S50	úplný kód chyby PLC
System_S.S51	<b>BYTE</b>	%S51	úplný kód chyby PLC
System_S.COUNTER_1MS	<b>UDINT</b>	%SL52	čítač po 1 ms
System_S.SW56	<b>WORD</b>	%SW56	rezerva
System_S.CPU_DI	<b>BYTE</b>	%S58	vstupy obsluhované centrálou
System_S.CPU_DO	<b>BYTE</b>	%S59	výstupy obsluhované centrálou
System_S.SL60	<b>UDINT</b>	%SL70	rezerva
System_S.SIZE_OF_RETAIN_ZONE	<b>UDINT</b>	%SL64	velikost remanentní zóny v bytech

System_S.UTC_OFFSET	<b>INT</b>	%SW68	offset proti UTC
System_S.CRC_OF_USER_PROGRAM	<b>WORD</b>	%SW70	CRC uživatelského programu
System_S.CRC_OF_HEADER_PROGRAM	<b>WORD</b>	%SW72	CRC hlavičky uživ. programu
System_S.S74	<b>BYTE</b>	%S74	rezerva
System_S.S75	<b>BYTE</b>	%S75	rezerva
System_S.S76	<b>BYTE</b>	%S76	rezerva
System_S.S77	<b>BYTE</b>	%S77	rezerva
System_S.COUNTER_MILLISECONDS	<b>UINT</b>	%SW78	desetinná část system. času [ms]
System_S.PLC_PRIVATE_AREA	<b>ARRAY</b> [0..19] <b>OF</b> <b>BYTE</b>	%S80..%S99	vyhrazeno CPU

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

Další globální proměnné, které jsou rovněž mapované do systémových registrů, jsou:

**VAR\_GLOBAL**

```
// restarty PLC
IS_HOT_RESTART_PLC AT System_S.S2_3 : BOOL; // %S2.3 teplý restart PLC
IS_COLD_RESTART_PLC AT System_S.S2_4 : BOOL; // %S2.4 studený restart PLC
IS_RESTART_PLC AT System_S.S2_6 : BOOL; // %S2.6 RUN bez restartu

// %SL70 CRC aplikačního programu
CRC_OF_APLIC_PROGRAM AT System_S.CRC_OF_USER_PROGRAM : DWORD;

// %S355.0 potlačit tlačítka DEL v seznamu souborů
DISABLE_WEB_DEL_BUTTONS AT %S355.0 : BOOL;

// %S355.1 pro přihlášení použít uživatelskou web stránku místo systémové
USE_USER_LOGIN_PAGE AT %S355.1 : BOOL;

// %S355.2 povolit CORS pro web server
ENABLE_WEB_CORS AT %S355.2 : BOOL;

// %S355.3 povolit CORS pro TecoApi
ENABLE_API_CORS AT %S355.3 : BOOL;
END_VAR
```

## 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 (pouze pro Foxtrot 1)
<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 na sběrnici TCL2 v systému Foxtrot 1 nebo TC700
<i>ModuleInfo2</i>	Funkce vrátí informace o aktuálním stavu jednoho I/O modulu na sběrnici TCL2/TCL3 v systému Foxtrot 2 nebo TC800
<i>CIBunitInfo</i>	Funkce vrátí informace o aktuálním stavu jedné CIB jednotky v systému Foxtrot 1
<i>CIBunitInfo2</i>	Funkce vrátí informace o aktuálním stavu jedné CIB jednotky v systému Foxtrot 2
<i>SetAddressCIBunit</i>	Funkce nastaví novou HW adresu CIB jednotky v systému Foxtrot 1
<i>SetAddressCIBunit2</i>	Funkce nastaví novou HW adresu CIB jednotky v systému Foxtrot 2
<i>Memcpy, MемcpyEx</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

<b>Funkce</b>	<b>Popis</b>
<i>VerifyWebMAC</i>	Funkce otestuje, zda-li existuje zadaná MAC adresa
<i>ProgramLock</i>	Funkce zapne ochranu aplikačního programu PLC, takže jej nelze dekompileovat (nelze použít pro Foxtrot 2 a TC800)
<i>SystemDisplayBacklightOn</i>	Funkce rozsvítí podsvícení LCD displeje na základním modulu PLC
<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 (pouze Foxtrot 1)
<i>SetAddressRFunit</i>	Funkce nastaví novou HW adresu RF jednotky (pouze Foxtrot 1)
<i>GetVarValueByName</i>	Funkce vrátí hodnotu proměnné podle zadaného jména proměnné
<i>GetVarDescByName</i>	Funkce vrátí popis proměnné podle zadaného jména proměnné
<i>GetVarNameByAdr</i>	Funkce vrátí jméno proměnné podle zadané adresy proměnné
<i>SetVarValueByName</i>	Funkce nastaví hodnotu proměnné podle zadaného jména proměnné
<i>GetProgramInfo2</i>	Informace o uživatelském programu (pouze Foxtrot 2 a TC800)
<i>GetPlcInfo2</i>	Informace o uživatelském programu (pouze Foxtrot 2 a TC800)
<i>ResetLTE2</i>	Reset LTE modemu (pouze Foxtrot 2)
<i>LoadNewPackage2</i>	Restart PLC s načtením nového programu (pouze Foxtrot 2 a TC800)
<i>GetVarAbsLoc</i>	Funkce vrací informace o absolutním umístění proměnné (nelze použít pro proměnné typu BOOL!)
<i>GetVarAbsLocBool</i>	Funkce vrací informace o absolutním umístění proměnné typu BOOL (lze použít pouze pro proměnné typu BOOL!)
<i>GetAvahiHostName2</i>	Funkce vrací aktuální hostname služby Avahi (pouze Foxtrot 2 a TC800)
<i>TestHotRestartFailed2</i>	Funkce vrací TRUE pokud se při teplém restartu nepodaří obnovit hodnoty RETAIN proměnných (pouze Foxtrot 2 a TC800)
<i>RandomU</i>	Funkce vrací pseudonáhodné číslo v rozsahu 0 až 1.

## 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.

Tato funkce je určena pouze pro systémy Foxtrot 1 a TC700, v systémech Foxtrot 2 nebo TC800 nic nedělá.

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

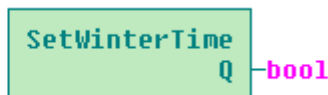
// request for atomatic change between summer and winter time
tmp := SetSummerTime();

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.

Tato funkce je určena pouze pro systémy Foxtrot 1 a TC700, v systémech Foxtrot 2 nebo TC800 nic nedělá.

Příklad programu s voláním funkce *SetWinterTime* :

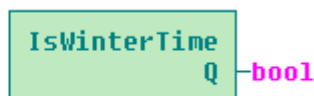
```
PROGRAM ExampleWinterTime
VAR
  tmp      : BOOL;
  message  : STRING;
END_VAR

// 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* :

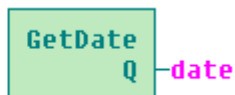
```
PROGRAM ExampleWinterTime
VAR
  tmp      : BOOL;
  message  : STRING;
END_VAR

// no atomatic 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 vrátí, je nastavován na začátku každého cyklu PLC. Během cyklu se jeho hodnota nemění.

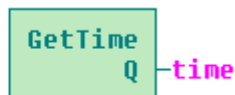
Příklad programu s voláním funkce *GetDate* :

```
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 vrátí, 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 vrací 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* :

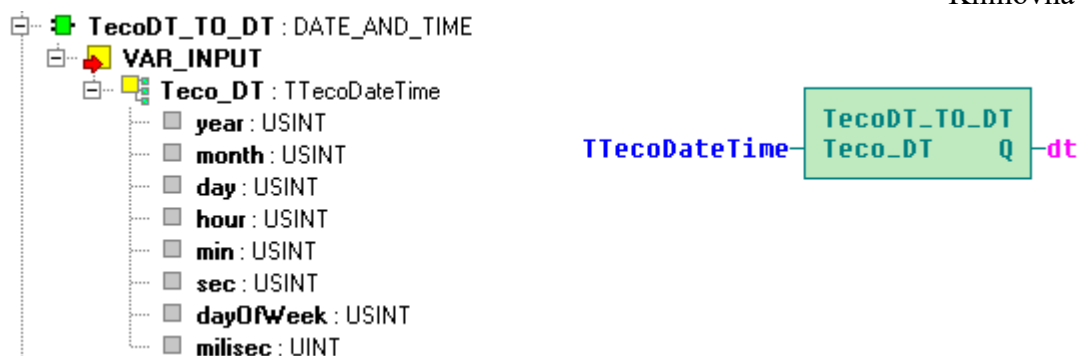
```
PROGRAM SetRTCexample
VAR
  newDateTimeRTC   : DT;
  setTime          : R_TRIG;
  setNewTime       : BOOL;
  tmp              : BOOL;
END_VAR

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

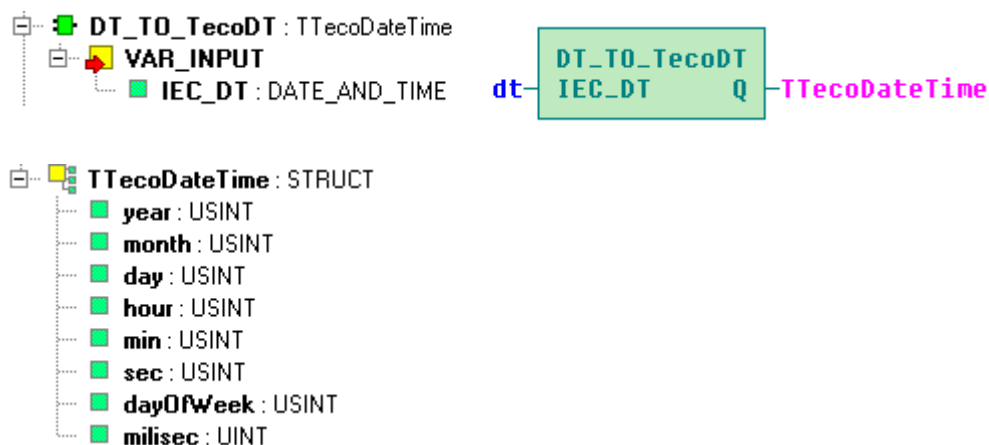
dateTime.year      := 09;
dateTime.month     := 11;
dateTime.day       := 22;
dateTime.hour      := 12;
dateTime.min       := 34;
dateTime.sec       := 56;
dateTime.milisec   := 500;
dateTimePLC       := TecoDT_TO_DT( dateTime);

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*.

Funkce je určena pouze pro systémy Foxtrot 1 a TC700.

Příklad programu s voláním funkce *IOSystemInfo* :

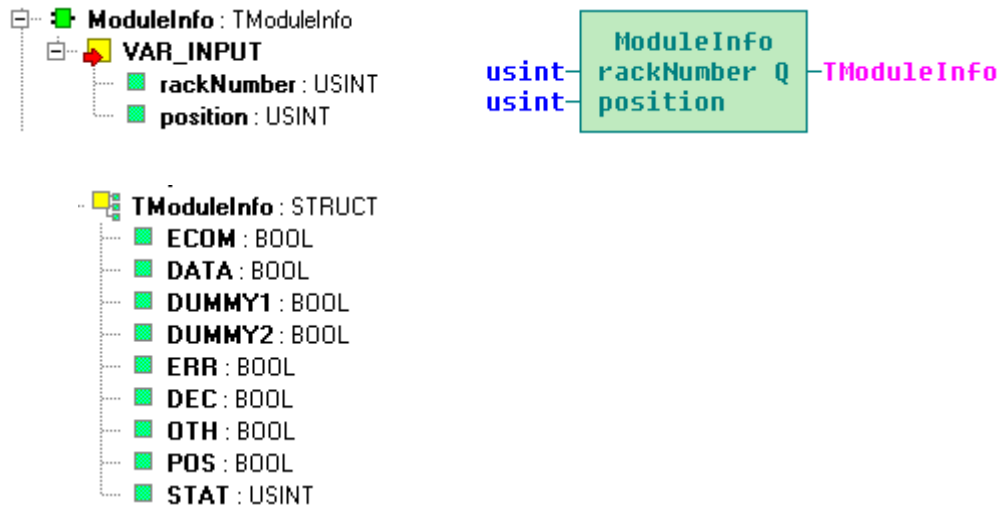
```
PROGRAM IOSystemInfoExample
  VAR
    IO_Status      : TIOSystemInfo;
    ModuleStatus   : TModuleInfo;
  END_VAR

  // 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 na sběrnici TCL2 v systémech Foxtrot 1 a TC700. Vstupní parametr *rackNumber* udává číslo rámu a parametr *position* říká číslo pozice, pro kterou se vrátí informace o modulu. Funkce vrací strukturu *TmoduleInfo*.

Funkce je určena pouze pro systémy Foxtrot 1 a TC700. V systémech Foxtrot 2 a TC800 je potřeba použít funkci *ModuleInfo2*.

Příklad programu s voláním funkce *ModuleInfo* :

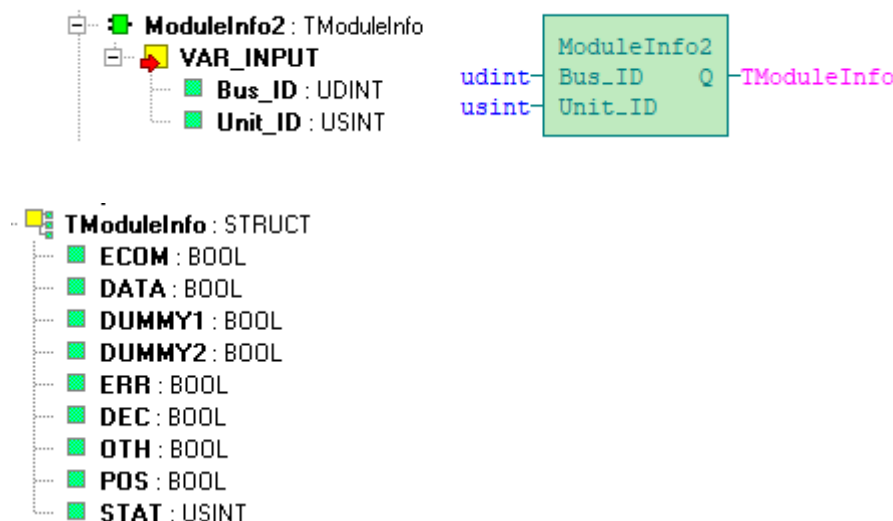
```
PROGRAM ModuleInfoExample
VAR
  IO_Status      : TIOSystemInfo;
  ModuleStatus   : TModuleInfo;
END_VAR

// 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 ModuleInfo2

Knihovna : SysLib



Funkce *ModuleInfo2* vrátí informace o aktuálním stavu jednoho I/O modulu na sběrnici TCL2 nebo TCL3 v systémech Foxtrot 2 a TC800. Vstupní parametr *Bus\_ID* udává identifikační číslo TCL2 nebo TCL3 sběrnice (např. 16#1020 pro TCL2 sběrnici na základním modulu Foxtrot 2, 16#1010 pro TCL3 sběrnici TC800) a parametr *Unit\_ID* určuje číslo pozice, pro kterou se vrátí informace o modulu (např. modul IB-1301 s adresou 0 nastavenou na přepínači ADR má *Unit\_ID* = 16). Funkce vrací strukturu *TmoduleInfo*.

Funkce je určena pouze pro systémy Foxtrot 2 a TC800. V systémech Foxtrot 1 a TC700 je potřeba použít funkci *ModuleInfo*.

Příklad programu s voláním funkce *ModuleInfo2* :

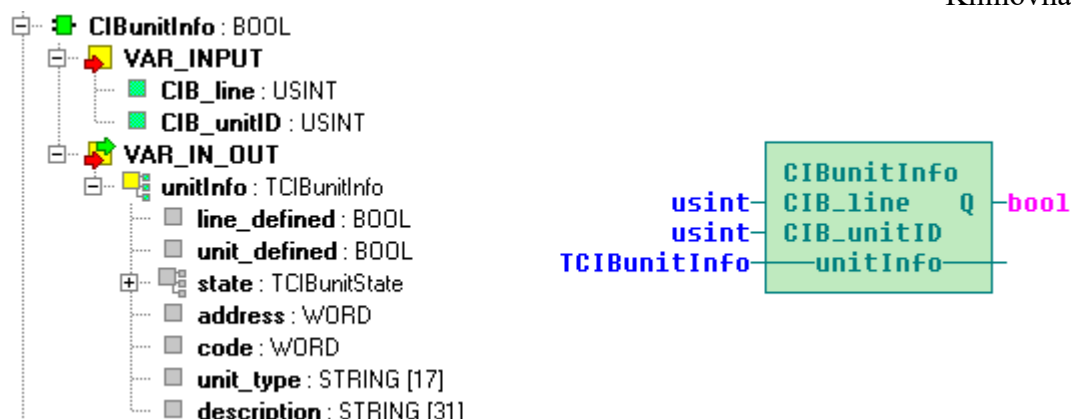
```
PROGRAM ModuleInfo2Example
VAR
  ModuleStatus : TModuleInfo;
END_VAR

// state of module IB-1301, ADR = 0
ModuleStatus := ModuleInfo2( Bus_ID := 16#1020, Unit_ID := 16);
END_IF;
END_PROGRAM
```

Viz také Typ *TmoduleInfo*

## 4.15 Funkce CIBunitInfo

Knihovna : SysLib



Funkce *CIBunitInfo* slouží k získání informace o aktuálním stavu jedné jednotky na CIB sběrnici v systému Foxtrot 1. 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 a je určena pouze pro systémy Foxtrot 1. V systémech Foxtrot 2 je potřeba použít funkci *CIBunitInfo2*.

Příklad programu s voláním funkce *CIBunitInfo* :

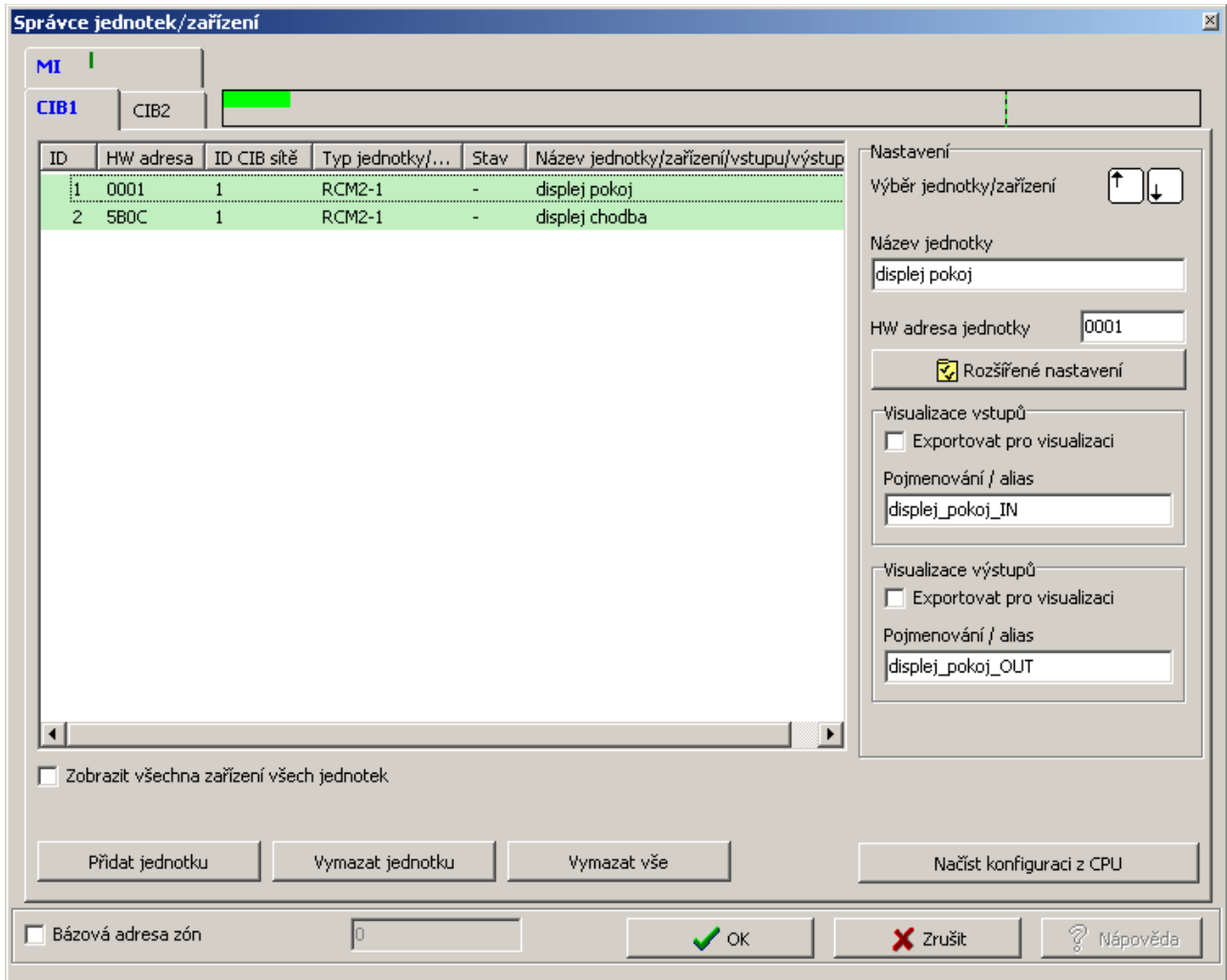
```
PROGRAM CIBunitInfoExample
VAR
  CIB_unit1, CIB_unit2 : TCIBunitInfo;
  result1, result2     : BOOL;
END_VAR

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.



Zprávy 1   Zprávy 2   Symbols   Breakpoint list   Data

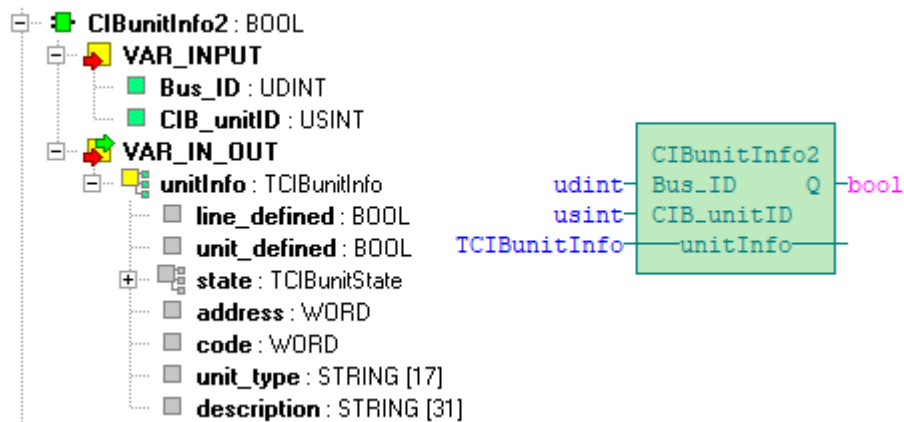
5 Banka3

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

## 4.16 Funkce CIBunitInfo2

Knihovna : SysLib



Funkce *CIBunitInfo2* slouží k získání informace o aktuálním stavu jedné jednotky na CIB sběrnici v systému Foxtrot 2. Vstupní parametr *Bus\_ID* udává identifikační číslo CIB sběrnice (např. 16#1030 pro CIB sběrnici na základním modulu Foxtrot 2) a parametr *Unit\_ID* určuje logickou adresu CIB modulu na sběrnici (0..31). 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 *CIBunitInfo2* vrací FALSE.

Tato funkce je určena pouze pro systémy Foxtrot 2. V systémech Foxtrot 1 je potřeba použít funkci *CIBunitInfo*.

Příklad programu s voláním funkce *CIBunitInfo2* :

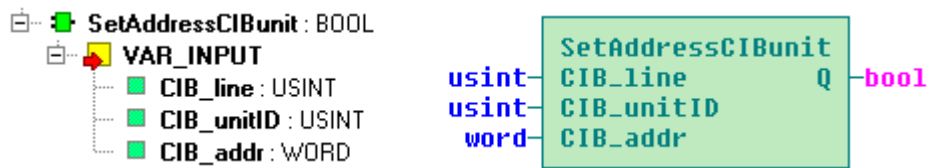
```
PROGRAM CIBunitInfo2Example
VAR
  i : USINT;
  CIB_unit : ARRAY[0..31] OF TCIBunitInfo;
END_VAR

FOR i := 0 TO 31 DO
  CIBunitInfo2( Bus_ID := 16#1030, CIB_unitID := i, unitInfo := CIB_unit[i]);
END_FOR;
END_PROGRAM
```

Uvedený program testuje stav všech CIB jednotek připojených na CIB linku základního modulu systému Foxtrot 2.



## 4.17 Funkce *SetAddressCIBunit*

Knihovna : *SysLib*

Funkce *SetAddressCIBunit* nastaví novou HW adresu CIB jednotky v systému Foxtrot 1. 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 a je určena pouze pro systémy Foxtrot 1. V systémech Foxtrot 2 je potřeba použít funkci *SetAddressCIBunit 2*.

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, 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

```

```

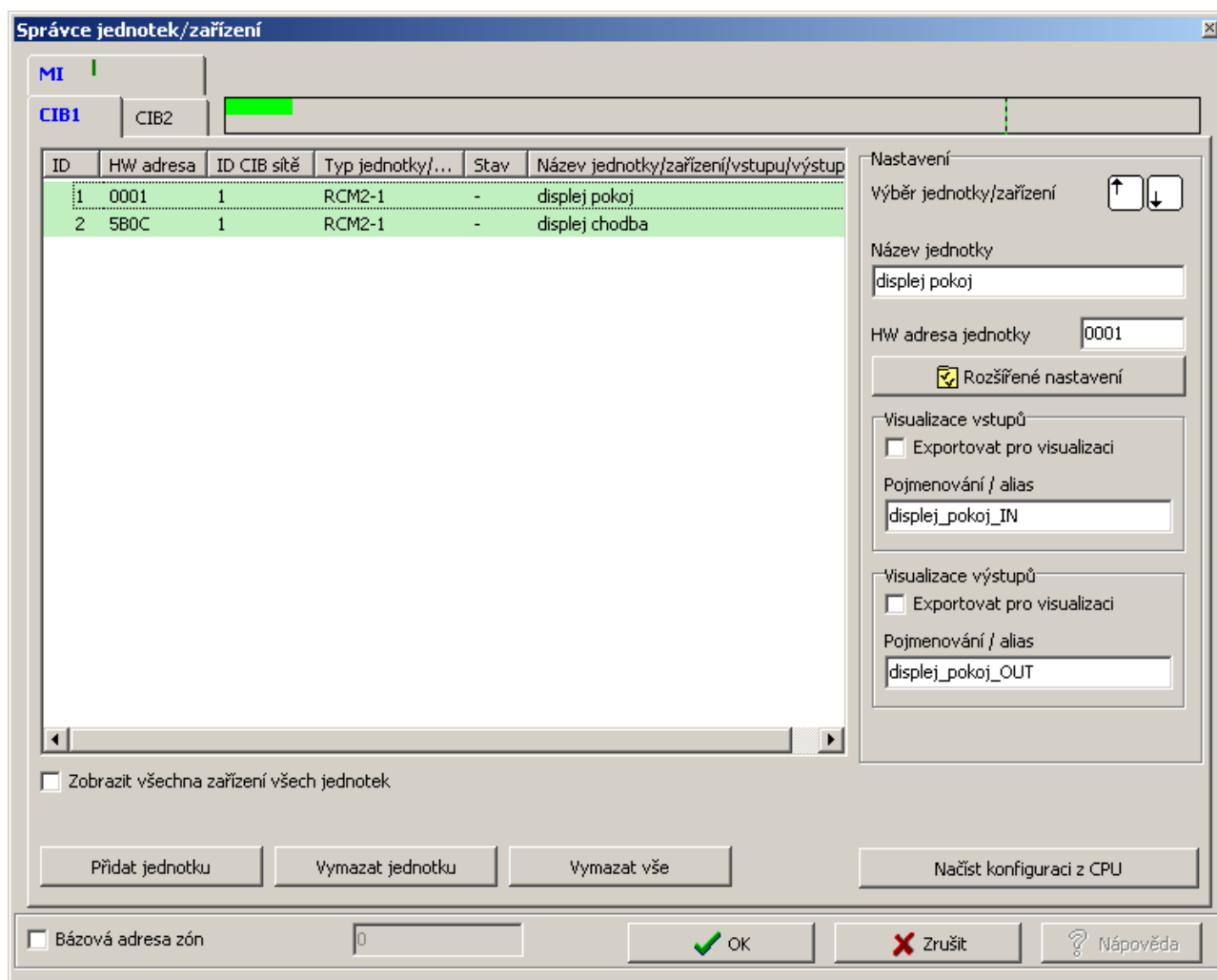
        SetAddressCIBunit(CIB_line := MI_CIB1, CIB_unitID := i,
                        CIB_addr := customCIB[i].address);
    END_IF;
END_IF;
END_IF;
END_FOR;

// 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).



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 CIB 1**

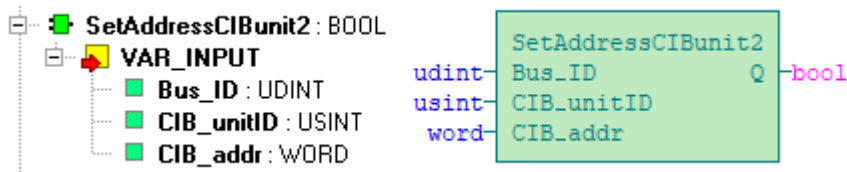
ID	line defined	unit defined	NET	REI	ADR	COM	INI	HW ADDR	TYPE	DESCRIPTION
1	<input type="text" value="1"/>	<input type="text" value="1"/>	<input type="text" value="1"/>	<input type="text" value="1"/>	<input type="text" value="0"/>	<input type="text" value="0"/>	<input type="text" value="0"/>	0001	RCM2-1	displej pokoj
2	<input type="text" value="1"/>	<input type="text" value="1"/>	<input type="text" value="1"/>	<input type="text" value="1"/>	<input type="text" value="0"/>	<input type="text" value="0"/>	<input type="text" value="0"/>	5B0C	RCM2-1	displej chodba
3	<input type="text" value="1"/>	<input type="text" value="0"/>	<input type="text" value="0"/>	<input type="text" value="0"/>	<input type="text" value="0"/>	<input type="text" value="0"/>	<input type="text" value="0"/>	0000		
4	<input type="text" value="1"/>	<input type="text" value="0"/>	<input type="text" value="0"/>	<input type="text" value="0"/>	<input type="text" value="0"/>	<input type="text" value="0"/>	<input type="text" value="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*

4.18 Funkce *SetAddressCIBunit2*Knihovna : *SysLib*

Funkce *SetAddressCIBunit2* nastaví novou HW adresu CIB jednotky v systému Foxtrot 2. To je potřebné například v situaci, kdy chceme používat stejný PLC program pro více aplikací, které se mezi sebou liší HW adresou CIB jednotek.

Vstupní parametr *Bus\_ID* udává identifikační číslo CIB sběrnice (např. 16#1030 pro CIB sběrnici na základním modulu Foxtrot 2) a parametr *CIB\_unitID* říká, pro kterou CIB jednotku bude nastavena nová HW adresa. Ta je daná parametrem *CIB\_addr*. Funkce vrací hodnotu TRUE, pokud se podaří HW adresu CIB jednotky nastavit.

Tato funkce je určena pouze pro systémy Foxtrot 2. V systémech Foxtrot 1 je potřeba použít funkci *SetAddressCIBunit*.

Hodnoty *Bus\_ID*, *CIB\_unitID* (logická adresa CIB jednotky) a *CIB\_addr* (HW adresa CIB jednotky) jsou vidět v okně Konfigurace CIB modulu (v nástroji I/O Configurator):

Viz také Typ TCIBunitState, Typ TCIBunitInfo, Funkce CIBunitInfo2

## 4.19 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
<b>VAR_INPUT</b>			
	<i>length</i>	UINT	Délka kopírovaného bloku v bytech
<b>VAR_IN_OUT</b>			
	<i>source</i>	USINT	Zdroj odkud kopírovat
	<i>dest</i>	USINT	Cíl kam kopírovat
<b>Memcpy</b>			
	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.

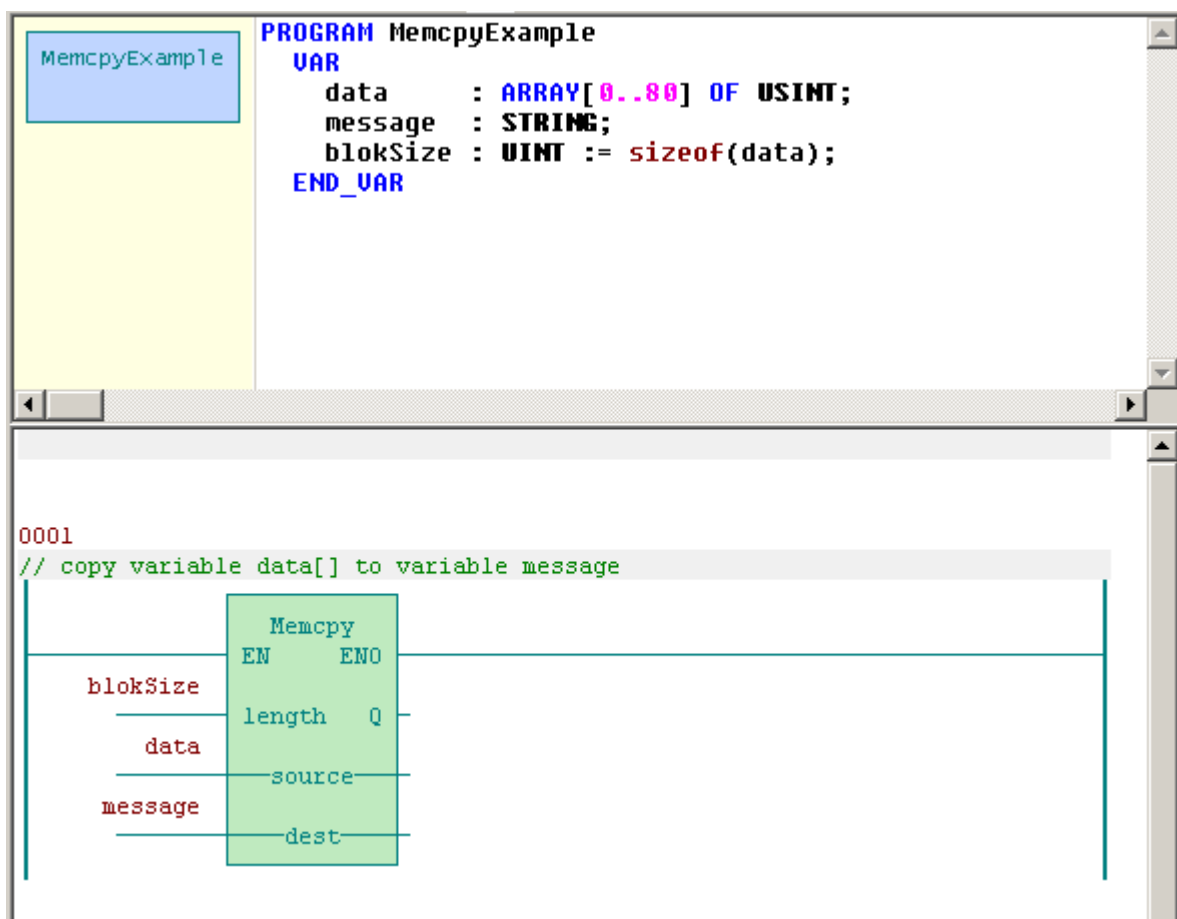
**POZOR !** Proměnné *source* a *dest* nesmí být typu BOOL ! Funkce *Memcpy* nebude fungovat správně, pokud je některý z parametrů *source* resp. *dest* typu BOOL.

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

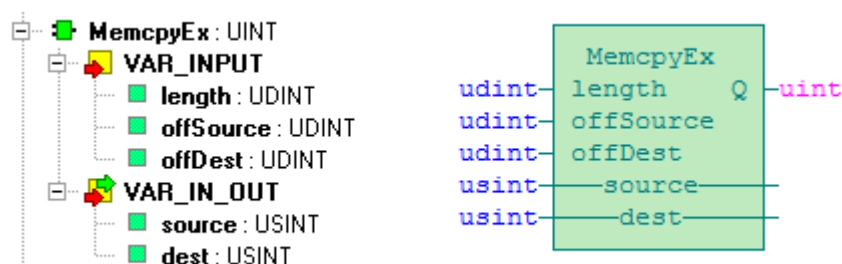
// copy variable data[] to variable message
size := Memcpy( length := sizeof(data),
               source := void(data[0]),
               dest   := void(message));
END_PROGRAM
```

Stejný program bude v jazyce LD vypadat následovně:



## 4.20 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
<b>VAR_INPUT</b>			
	<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
<b>VAR_IN_OUT</b>			
	<i>source</i>	USINT	Zdroj odkud kopírovat
	<i>dest</i>	USINT	Cíl kam kopírovat
<b>MemcpyEx</b>			
	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 4.19).

**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*, jež 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.

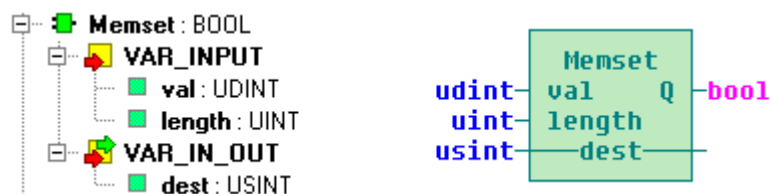
```
PROGRAM ExampleMemcpyEx
VAR
  data      : ARRAY[0..80] OF USINT;
  message   : STRING;
  size      : UINT;
END_VAR

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



## 4.21 Funkce Memset

Knihovna : SysLib



Funkce *Memset* vyplní blok paměti zadanou hodnotou.

Popis proměnných :

	Proměnná	Typ	Význam
<b>VAR_INPUT</b>			
	<i>val</i>	UDINT	Hodnota, která se zapíše do bloku paměti
	<i>length</i>	UINT	Délka bloku paměti v bytech
<b>VAR_IN_OUT</b>			
	<i>dest</i>	USINT	Proměnná, která bude naplněna konstantou
<b>Memset</b>			
	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*.

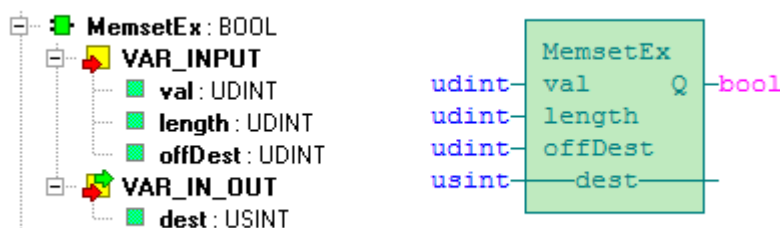
**POZOR !** Proměnná *dest* nesmí být typu *BOOL* ! Funkce *Memset* nebude fungovat správně, pokud je parametr *dest* typu *BOOL*.

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

tmp := Memset( val    := 16#202020,
              length := sizeof(message)-1,
              dest    := void(message) );
END_PROGRAM
```

## 4.22 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
<b>VAR_INPUT</b>			
	<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>
<b>VAR_IN_OUT</b>			
	<i>dest</i>	USINT	Proměnná, která bude naplněna konstantou
<b>MemsetEx</b>			
	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**.

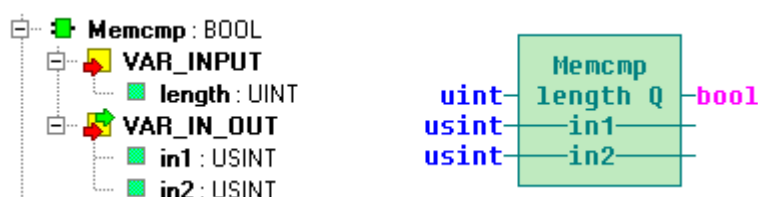
V následujícím příkladu naplní funkce *MemsetEx* proměnnou *message* (typu **STRING**) znakem 16#20 což je kód mezery. Nesoulad datových typů v parametru *dest*, kde je požadován parametr typu **USINT** a je třeba naplnit proměnnou *message*, která je typu **STRING**, řeší použití funkce *void()* při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

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

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

## 4.23 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
<b>VAR_INPUT</b>			
	<i>length</i>	UINT	Délka porovnávaného bloku v bytech
<b>VAR_IN_OUT</b>			
	<i>in1</i>	USINT	První porovnávaný blok
	<i>in2</i>	USINT	Druhý porovnávaný blok
<b>Memcmp</b>			
	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**.

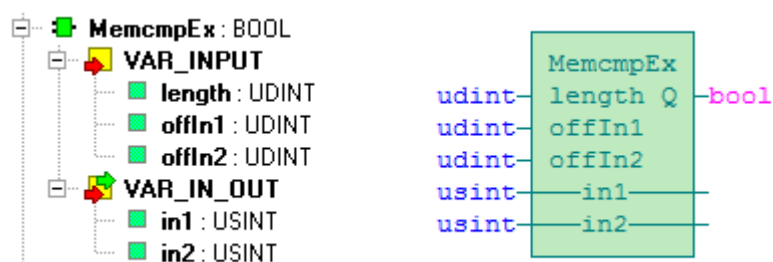
```

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
  
```

## 4.24 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
<b>VAR_INPUT</b>			
	<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
<b>VAR_IN_OUT</b>			
	<i>in1</i>	USINT	První porovnávaný blok
	<i>in2</i>	USINT	Druhý porovnávaný blok
<b>MemcmpEx</b>			
	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
```

## 4.25 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 u systémů Foxtrot 1 resp. 1200 ms u systémů Foxtrot 2 a TC800.

Příklad programu s voláním funkce *IncreaseMaxCycleTime*:

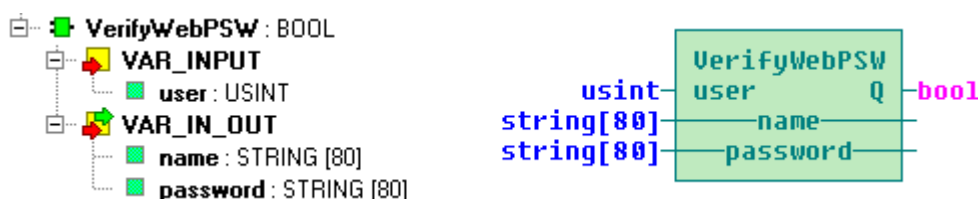
```
PROGRAM RestartExample
  VAR
    tmp : BOOL;
  END_VAR

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

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

## 4.26 Funkce VerifyWebPSW

Knihovna : SysLib



Centrální jednotky systémů Foxtrot 1, Foxtrot 2, TC700 a TC800 mají integrovaný web server. Jinými slovy v těchto centrálních jednotkách mohou být uloženy webové stránky, na kterých lze zobrazit informace o řízení technologie 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
<b>VAR_INPUT</b>			
	<i>user</i>	USINT	Číslo uživatele v seznamu oprávněných uživatelů První uživatel má číslo 0, poslední uživatel má číslo 9
<b>VAR_IN_OUT</b>			
	<i>name</i>	STRING	Jméno uživatele, které bude porovnáno se seznamem
	<i>password</i>	STRING	Heslo, které bude porovnáno se seznamem
<b>VerifyWebPSW</b>			
	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 1) od verze v5.6. U systému Foxtrot 2 a TC800 podporují tuto funkci všechny centrální jednotky.

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



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

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

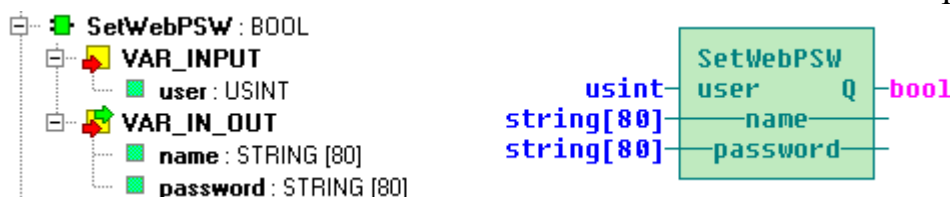
PROGRAM TestWebPass
  VAR
    init, result : BOOL;
  END_VAR

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

```

## 4.27 Funkce SetWebPSW

Knihovna : SysLib



Centrální jednotky systémů Foxtrot 1, Foxtrot 2, TC700 a TC800 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
<b>VAR_INPUT</b>			
	<i>user</i>	USINT	Číslo uživatele v seznamu oprávněných uživatelů První uživatel má číslo 0, poslední uživatel má číslo 9
<b>VAR_IN_OUT</b>			
	<i>name</i>	STRING	Nové jméno uživatele, které bude zapsáno do seznamu
	<i>password</i>	STRING	Nové heslo, které bude zapsáno do seznamu
<b>SetWebPSW</b>			
	Návratová hodnota	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. U systému Foxtrot 2 a TC800 podporují tuto funkci všechny centrální jednotky.

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

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

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

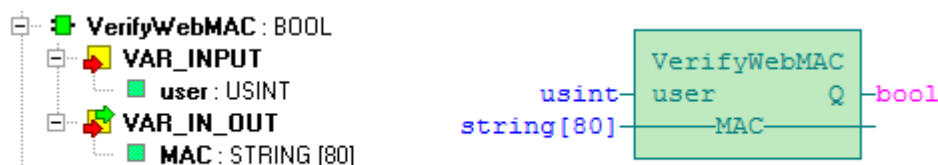
PROGRAM TestWebPass
  VAR
    init   : BOOL;
    result : BOOL;
  END_VAR

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

```

## 4.28 Funkce VerifyWebMAC

Knihovna : SysLib



Centrální jednotky systémů Foxtrot 1, Foxtrot 2, TC700 a TC800 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*.

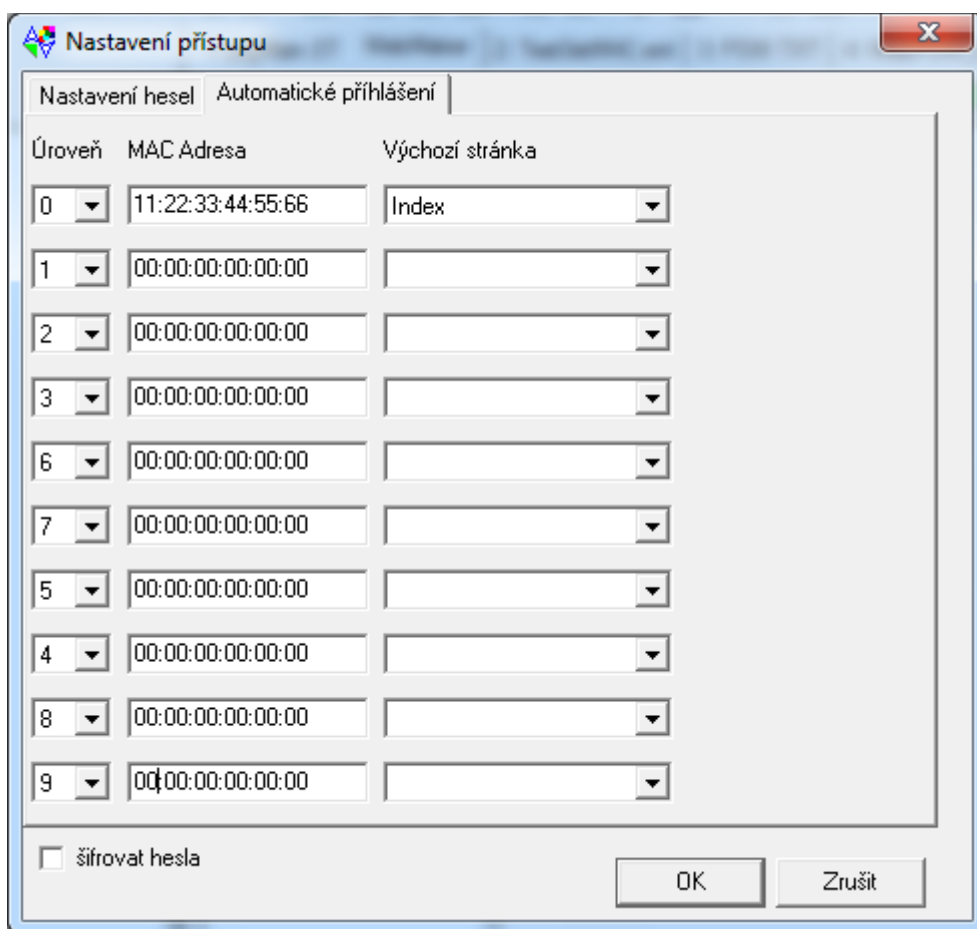
Funkce *VerifyWebMAC* zkontroluje, jestli pro daného uživatele existuje konkrétní MAC adresa v seznamu MAC adres. Seznam může obsahovat až 10 různých uživatelů. První uživatel má číslo 0, druhý 1, poslední uživatel v seznamu má číslo 9. Pokud MAC adresa v seznamu odpovídá zadanému MAC adrese, funkce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

Popis proměnných :

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

Funkce *VerifyWebMAC* 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. U systému Foxtrot 2 a TC800 podporují tuto funkci všechny centrální jednotky.

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



Uživatel na řádce 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

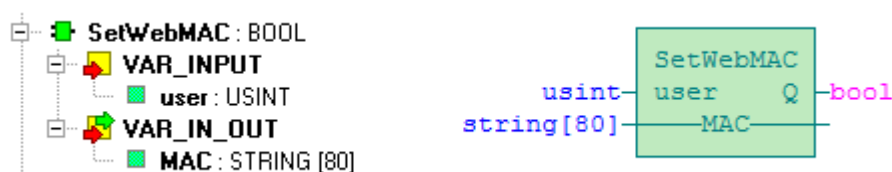
PROGRAM TestWebMAC
  VAR
    init, result : BOOL;
  END_VAR

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

```

## 4.29 Funkce SetWebMAC

Knihovna : SysLib



Centrální jednotky systémů Foxtrot 1, Foxtrot 2, TC700 a TC800 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*.

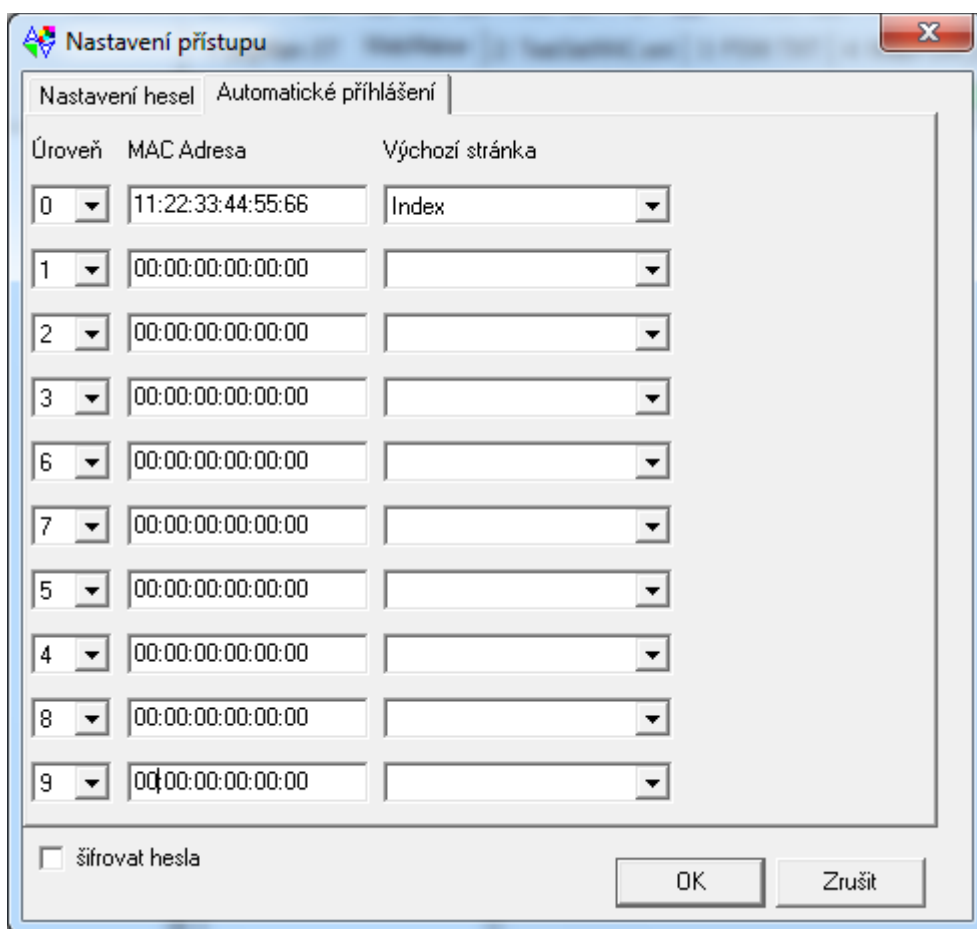
Funkce *SetWebMAC* nejprve zkontroluje, jestli souhlasí zadaná MAC adresa s adresou uvedenou v seznamu MAC adres. Pokud souhlasí, funkce vrátí hodnotu TRUE. Pokud adresy nesouhlasí, funkce *SetWebMAC* nahradí MAC adresu v seznamu za MAC adresu uvedenou v parametrech funkce. Pokud se zápis do seznamu 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
<b>VAR_INPUT</b>			
	user	USINT	Číslo uživatele v seznamu oprávněných uživatelů První uživatel má číslo 0, poslední uživatel má číslo 9
<b>VAR_IN_OUT</b>			
	MAC	STRING	Nová MAC adresa, která bude zapsaná do seznamu (6 hexa cifer oddělených znakem dvojtečka)
<b>SetWebMAC</b>			
	<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. U systému Foxtrot 2 a TC800 podporují tuto funkci všechny centrální jednotky.

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



Uživatel na řádce 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

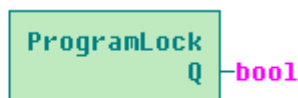
PROGRAM SetWebMAC
  VAR
    init, result : BOOL;
  END_VAR

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

```

### 4.30 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, K a L (všechny procesory systému TC650, TC700 a Foxtrot) ve všech verzích. U systému Foxtrot 2 a TC800 není tato funkce podporována.

Příklad programu s voláním funkce *ProgramLock*:

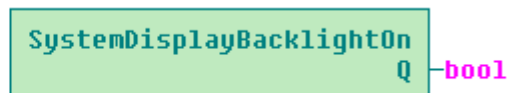
```
PROGRAM prgMainExample
  VAR
    tmp : BOOL;
  END_VAR

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

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



### 4.31 Funkce *SystemDisplayBacklightOn*

Knihovna : *SysLib*
 **SystemDisplayBacklightOn** : 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.

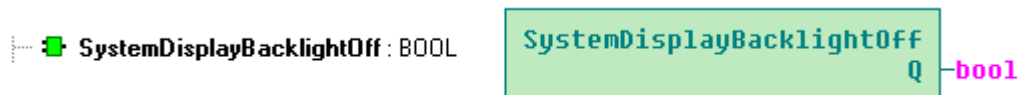
Příklad programu s voláním funkce *SystemDisplayBacklightOn*:

```
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
```

## 4.32 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*:

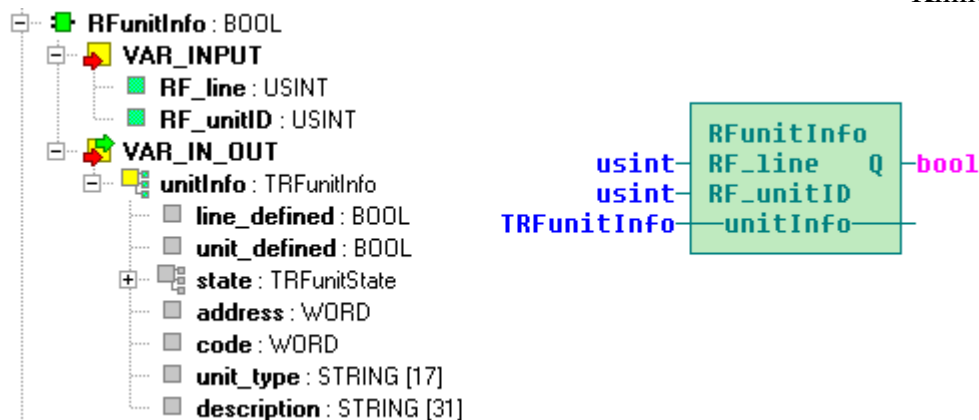
```
PROGRAM TestSystemDisplay
VAR
  errFlg : BOOL;
  errAck : BOOL;
END_VAR

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
```

### 4.33 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* :

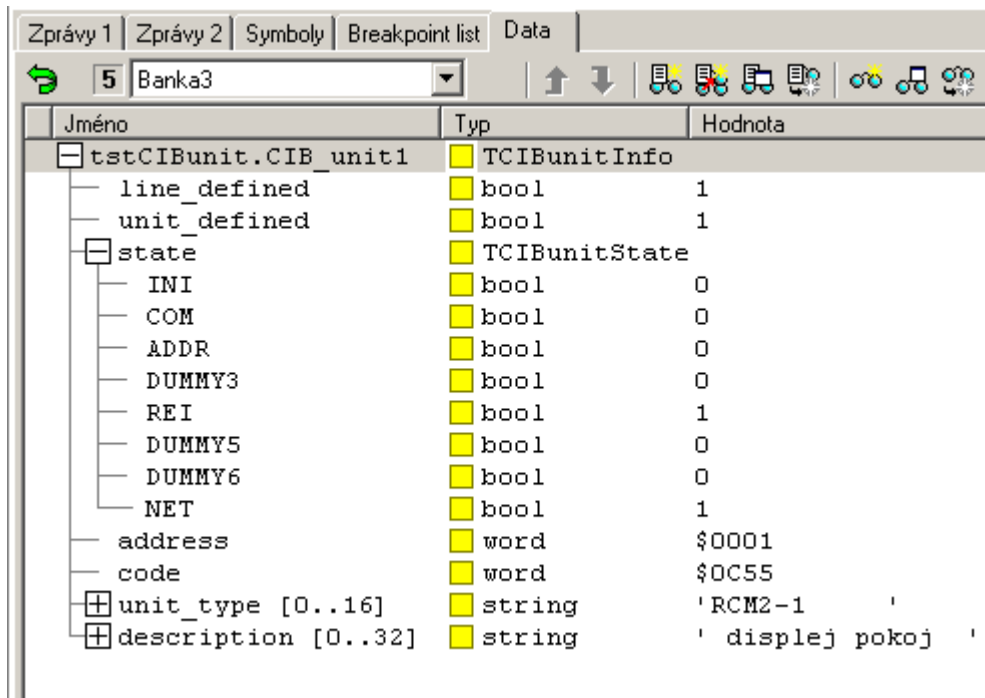
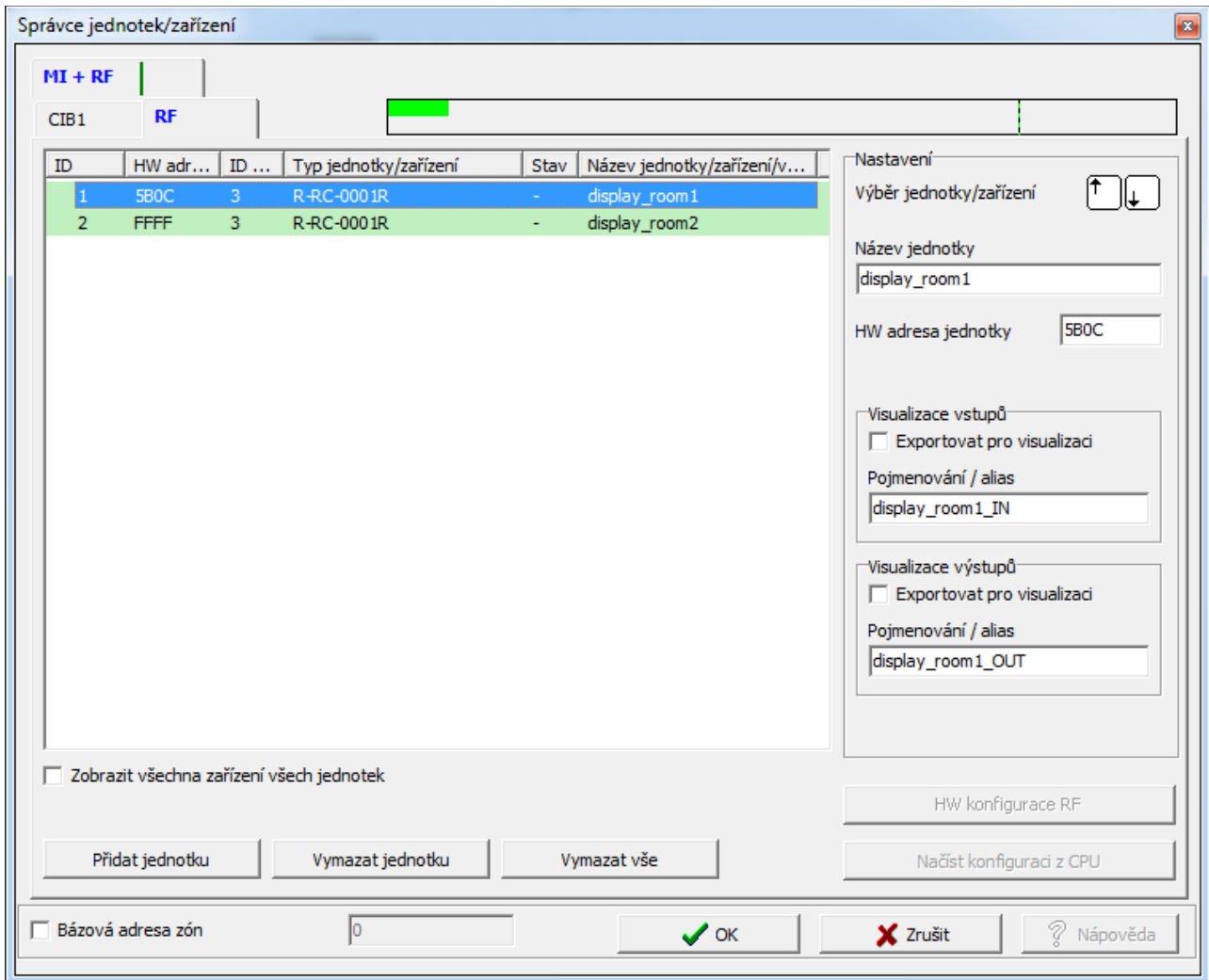
```
PROGRAM RFunitInfoExample
VAR
  RF_unit1 : TRFunitInfo;
  RF_unit2 : TRFunitInfo;
  result   : BOOL;
END_VAR

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.

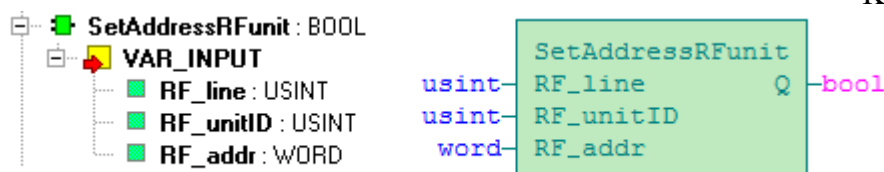
Další obrázek pak ukazuje hodnoty proměnné *RF\_unit1*, které odpovídají stavu první definované RF jednotky. Hodnoty odpovídají situaci, kdy je RF jednotka v programu PLC definovaná, ale nekomunikuje (např. protože není spárována nebo má jinou HW adresu, než je nastavená ve výše uvedeném dialogu).

Pokud by chyběla deklarace RF jednotek v programu, funkce *RFunitInfo* bude vracet FALSE a proměnná *RF\_unit1* bude vynulovaná.



Viz také Typ TRFunitState, Typ TRFunitInfo

### 4.34 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;

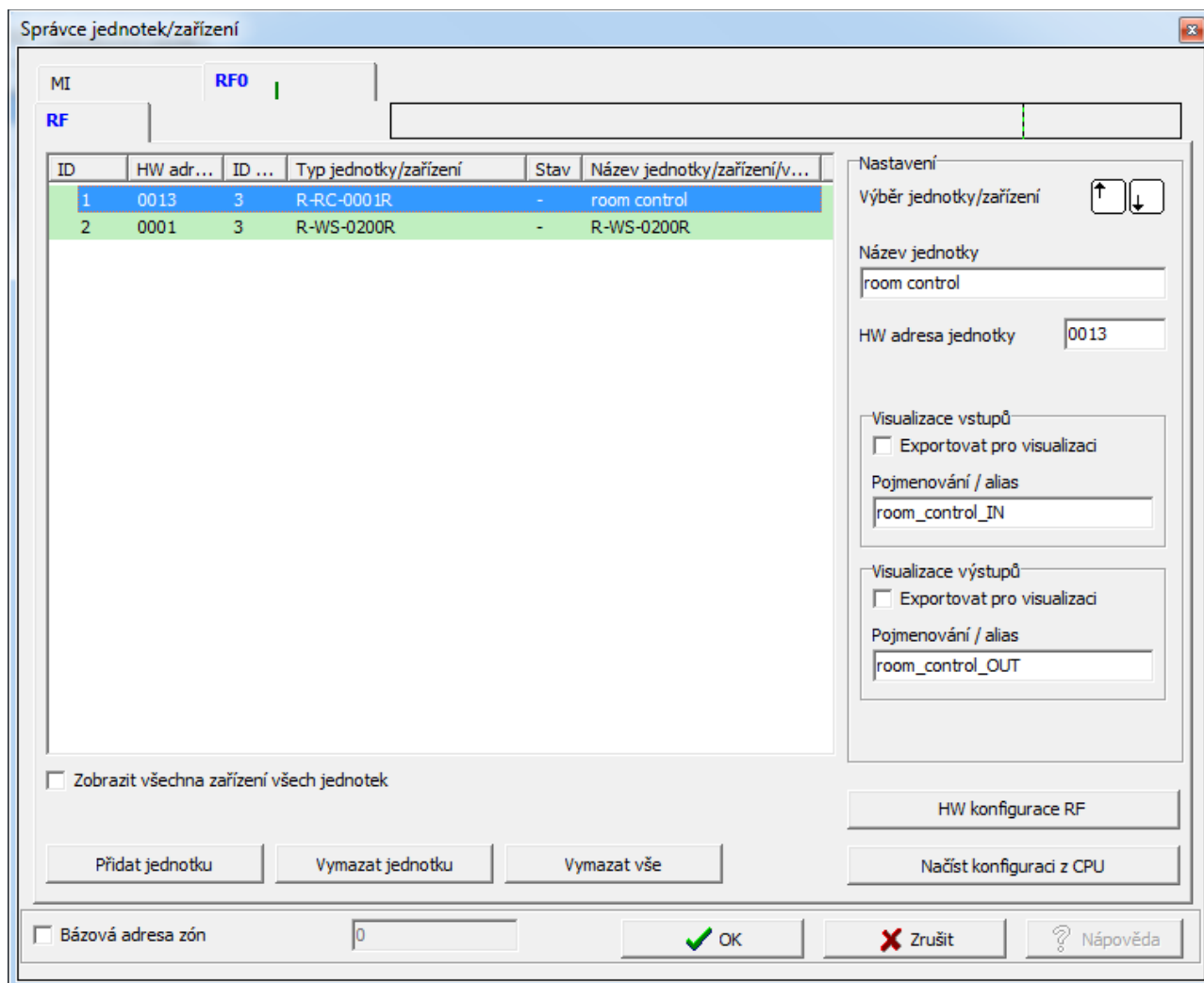
// set new customer address of RF unit
IF setAdr THEN
  IF id > 0 AND id <= NUM_RF_UNIT THEN
    IF newAdr <> RF_unit[id].address THEN
      result := SetAddressRFunit(RF_line := RF0_RF,
                                RF_unitID := id,
                                RF_addr := newAdr);

      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).



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 zapíše 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
			NET	BND	SLP	COM	INI			
1	<input type="text" value="1"/>	<input type="text" value="1"/>	<input type="text" value="1"/>	<input type="text" value="1"/>	<input type="text" value="1"/>	<input type="text" value="0"/>	<input type="text" value="0"/>	0013	R-RC-0001R	room control
2	<input type="text" value="1"/>	<input type="text" value="1"/>	<input type="text" value="1"/>	<input type="text" value="1"/>	<input type="text" value="1"/>	<input type="text" value="0"/>	<input type="text" value="0"/>	0001	R-WS-0200R	R-WS-0200R
3	<input type="text" value="1"/>	<input type="text" value="0"/>	<input type="text" value="0"/>	<input type="text" value="0"/>	<input type="text" value="0"/>	<input type="text" value="0"/>	<input type="text" value="0"/>	0000		
4	<input type="text" value="1"/>	<input type="text" value="0"/>	<input type="text" value="0"/>	<input type="text" value="0"/>	<input type="text" value="0"/>	<input type="text" value="0"/>	<input type="text" value="0"/>	0000		

**NEW SETTINGS**

ID       NEW HW ADDR

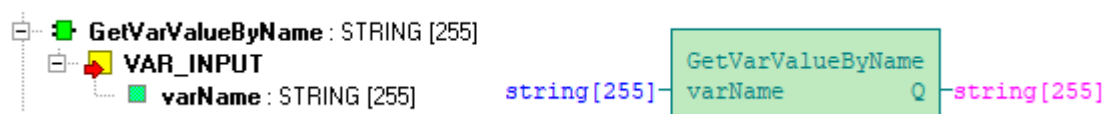
Ve sloupci „line defined“ jsou zobrazeny proměnné *TestRF.RF\_unit[i].line\_defined*, ve sloupci „unit defined“ jsou zobrazeny proměnné *TestRF.RF\_unit[i].unit\_defined*, sloupce „NET“ „BND“ „SLP“ „COM“ a „INI“ obsahují odpovídající položky proměnné *TestRF.RF\_unit[i].state* (tedy *TestRF.RF\_unit[i].state.NET*, 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



### 4.35 Funkce *GetVarValueByName*

Knihovna : *SysLib*

Funkce *GetVarValueByName* vrátí hodnotu proměnné, jejíž jméno je specifikované v parametru *varName*. Hodnota proměnné je vrácena jako text. V definici proměnné musí být uvedena direktiva {OPEN\_UP}. Proměnné deklarované bez této direktivy nejsou pro funkci *GetVarValueByName* dostupné.

Tato funkce je zařazena do knihovny SysLib od v3.6 a je podporována na centrálních jednotkách řady K a L (systém Foxtrot 1) od verze v9.7. V systému Foxtrot 2 a TC800 podporují tuto funkci všechny centrální jednotky.

Popis proměnných :

	Proměnná	Typ	Význam
<b>VAR_INPUT</b>			
	<i>varName</i>	STRING[255]	Název proměnné (úplné jméno instance)
<b>GetVarValueByName</b>			
	<i>Návratová hodnota</i>	STRING[255]	Hodnota proměnné (textově)

V následujícím příkladu je definovaná globální proměnná *g\_varInt* a lokální proměnná *l\_varInt* v instanci programu *prgTestVar* nazvané *Test*. Funkce *GetVarValueByName* je použita pro načtení hodnot uvedených proměnných.

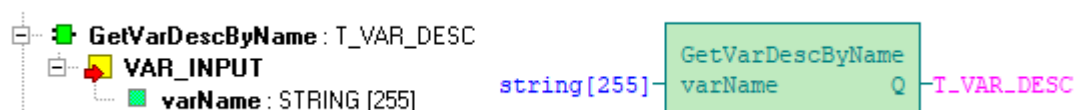
```

VAR_GLOBAL
  g_varInt {OPEN_UP} : INT := 1234;
  g_varIntValue      : STRING;
END_VAR

PROGRAM prgTestVar
  VAR
    l_varInt {OPEN_UP} : INT := 4321;
    l_varIntValue      : STRING;
  END_VAR

  g_varIntValue := GetVarValueByName(varName := 'g_varInt');
  l_varIntValue := GetVarValueByName(varName := 'Test.l_varInt');
END_PROGRAM
  
```

Funkce *GetVarValueByName* se používá například v případě, kdy PLC přijme HTTP dotaz GET na stav proměnných, jejichž jména jsou specifikovaná v parametrech GET dotazu. Funkce *GetVarValueByName* použije jména z GET dotazu pro získání hodnot.

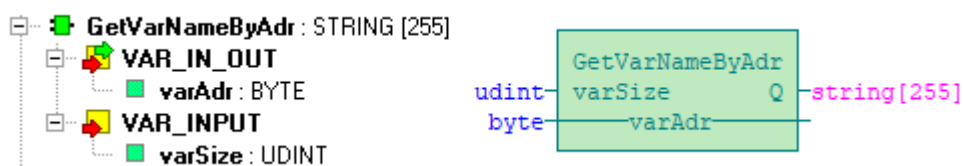
4.36 Funkce *GetVarDescByName*Knihovna : *SysLib*

Funkce *GetVarDescByName* vrátí popis proměnné, jejíž jméno je specifikované v parametru *varName*. Popis proměnné je vrácena jako struktura typu *T\_VAR\_DESC*. V definici proměnné musí být uvedena direktiva `{OPEN_UP}`. Proměnné deklarované bez této direktivy nejsou pro funkci *GetVarDescByName* dostupné.

Tato funkce je zařazena do knihovny *SysLib* od v4.1 a je podporována na centrálních jednotkách řady K a L (systém Foxtrot 1) od verze v10.0. V systému Foxtrot 2 a TC800 podporují tuto funkci všechny centrální jednotky.

Popis proměnných :

	<i>Proměnná</i>	<i>Typ</i>	<i>Význam</i>
<b>VAR_INPUT</b>			
	<i>varName</i>	STRING[255]	Název proměnné (úplné jméno instance)
<b>GetVarDescByName</b>			
	<i>Návratová hodnota</i>	T_VAR_DESC .addr .size .typ  .isArray	Popis proměnné adresa proměnné velikost proměnné (počet bytů) datový typ proměnné 0 ... BIT0 1 ... BIT1 2 ... BIT2 3 ... BIT3 4 ... BIT4 5 ... BIT5 6 ... BIT6 7 ... BIT7 8 ... BYTE, USINT 9 ... SINT 10 ... WORD, UINT 11 ... INT 12 ... DWORD, UDINT 28 ... REAL 30 ... LREAL 48 ... STRING 49 ... DATE 50 ... DATE_AND_TIME 51 ... TIME 52 ... TIME_OF_DAY TRUE pokud je proměnná pole, jinak FALSE

4.37 Funkce *GetVarNameByAdr*Knihovna : *SysLib*

Funkce *GetVarNameByAdr* vrátí jméno proměnné, jejíž adresa je specifikované v parametru *varAdr* a velikost je daná parametrem *varSize*. Pokud neexistuje proměnná, která odpovídá vstupním parametrům, pak funkce vrátí prázdný text. V definici proměnné musí být uvedena direktiva {TRACK\_ADR}. Proměnné deklarované bez této direktivy nejsou pro funkci *GetVarNameByAdr* dostupné. Funkci *GetVarNameByAdr* nelze použít pro proměnné typu BOOL!

Tato funkce je zařazena do knihovny SysLib od v3.6 a je podporována na centrálních jednotkách řady K a L (systém Foxtrot 1) od verze v9.7. V systému Foxtrot 2 a TC800 podporují tuto funkci všechny centrální jednotky.

	Proměnná	Typ	Význam
<b>VAR_INPUT</b>			
	<i>varSize</i>	UDINT	Velikost proměnné (počet bytů)
<b>VAR_IN_OUT</b>			
	<i>varAdr</i>	BYTE	Adresa proměnné
<b>GetVarNameByAdr</b>			
	Návratová hodnota	STRING[255]	Jméno proměnné nebo prázdný řetězec

Příklad použití funkce *GetVarNameByAdr*

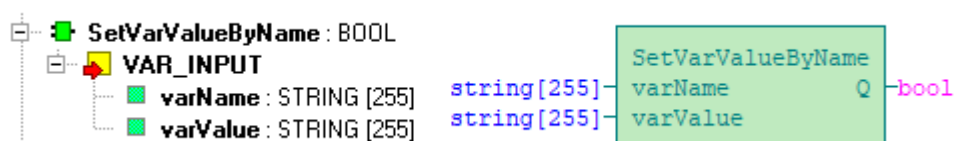
```

VAR_GLOBAL
  g_varInt {TRACK_ADR} : INT;
  g_varIntName       : STRING;
END_VAR

PROGRAM prgTestVar
  VAR
    l_varInt {TRACK_ADR} : INT := 4321;
    l_varIntName       : STRING;
  END_VAR

  g_varIntName := GetVarNameByAdr( varAdr := void(g_varInt),
                                   varSize := sizeof(INT));
  l_varIntName := GetVarNameByAdr( varAdr := void(l_varInt),
                                   varSize := sizeof(INT));
END_PROGRAM
  
```

### 4.38 Funkce *SetVarValueByName*

Knihovna : *SysLib*

Funkce *SetVarValueByName* nastaví hodnotu proměnné, jejíž jméno je specifikované v parametru *varName* a hodnota proměnné je v parametru *varValue*. Oba parametry se zadávají textově. V definici proměnné musí být uvedena direktiva `{OPEN_UP}`. Proměnné deklarované bez této direktivy nejsou pro funkci *SetVarValueByName* dostupné.

Tato funkce je zařazena do knihovny SysLib od v3.6 a je podporována na centrálních jednotkách řady K a L (systém Foxtrot 1) od verze v9.7. V systému Foxtrot 2 a TC800 podporují tuto funkci všechny centrální jednotky.

Popis proměnných :

	<i>Proměnná</i>	<i>Typ</i>	<i>Význam</i>
<b>VAR_INPUT</b>			
	<i>varName</i>	STRING[255]	Název proměnné (úplné jméno instance)
	<i>varValue</i>	STRING[255]	Hodnota proměnné
<b>SetVarValueByName</b>			
	<i>Návratová hodnota</i>	BOOL	TRUE pokud se proměnnou podaří nastavit

Příklad použití funkce *SetVarValueByName*

```

VAR_GLOBAL
  g_varInt {OPEN_UP} : INT := 1234;
END_VAR

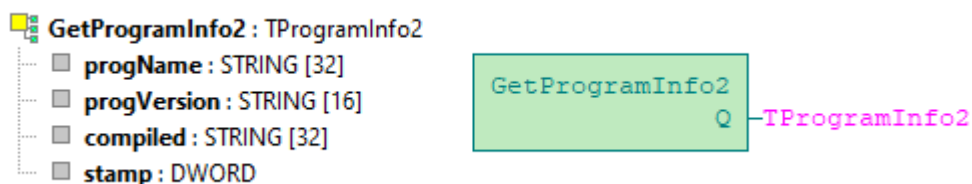
PROGRAM prgTestVar
  VAR
    l_varInt {OPEN_UP} : INT := 4321;
    res : BOOL;
  END_VAR

  res := SetVarValueByName(varName := 'g_varInt', varValue := '1111');
  res := SetVarValueByName(varName := 'Test.l_varInt', varValue := '2222');

END_PROGRAM

```

### 4.39 Funkce *GetProgramInfo2*

Knihovna : *SysLib*

Funkce *GetProgramInfo2* vrátí informace o aktuálním uživatelském programu. Funkce nemá žádné vstupní parametry. Funkce vrací strukturu *TProgramInfo2*.

Tato funkce je podporována pouze na centrálních jednotkách řady I (systémy TC800 a Foxtrot 2). Centrální jednotky řady K a L (systémy TC700 a Foxtrot 1) tuto funkci nepodporují.

Příklad programu s voláním funkce *GetProgramInfo2* :

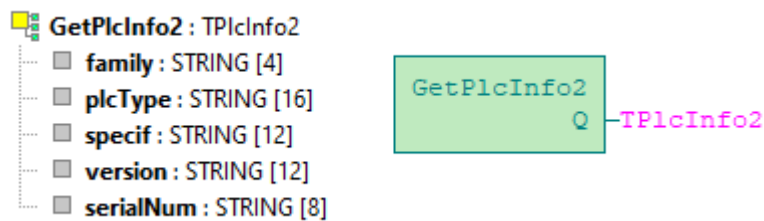
```

PROGRAM ExampleProgInfo
VAR
  progInfo : TProgramInfo2;
END_VAR

progInfo := GetProgramInfo2();
END_PROGRAM
  
```

Viz také Typ *TProgramInfo2*

#### 4.40 Funkce *GetPlcInfo2*

Knihovna : *SysLib*

Funkce *GetPlcInfo2* vrátí informace o PLC systému. Funkce nemá žádné vstupní parametry. Funkce vrací strukturu *TPlcInfo2*.

Tato funkce je podporována pouze na centrálních jednotkách řady I (systémy TC800 a Foxtrot 2). Centrální jednotky řady K a L (systémy TC700 a Foxtrot 1) tuto funkci nepodporují.

Příklad programu s voláním funkce *TPlcInfo2* :

```
PROGRAM ExamplePlcInfo
  VAR
    plcInfo    : TPlcInfo2;
  END_VAR

  plcInfo := GetPlcInfo2();
END_PROGRAM
```

Viz také Typ *TPlcInfo2*

#### 4.41 Funkce ResetLTE2

Knihovna : SysLib



Funkce *ResetLTE2* provede reset LTE modemu. Funkce vrací TRUE pokud se modem podaří resetovat, jinak vrací FALSE.

Tato funkce je podporována pouze na centrálních jednotkách řady I (systémy Foxtrot 2). Centrální jednotky řady K a L (systémy TC700 a Foxtrot 1) tuto funkci nepodporují.

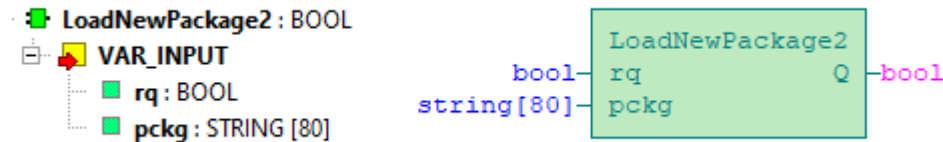
Příklad programu s voláním funkce *ResetLTE2* :

```
PROGRAM ExampleResetLTE
VAR
  rqResetLTE : BOOL;
END_VAR

IF rqResetLTE THEN
  ResetLTE2 ();
  rqResetLTE := 0;
END_IF;
END_PROGRAM
```

## 4.42 Funkce LoadNewPackage2

Knihovna : SysLib



Funkce *LoadNewPackage2* provede restart PLC spojený s načtením nového uživatelského programu z balíčku. Název souboru s balíčkem se udává jako parametr *pckg* při volání funkce. Pokud má parametr *rq* hodnotu TRUE tak funkce zkontroluje soubor s balíčkem a pokud je v pořádku, tak vrátí TRUE a zahájí zpracování balíčku. PLC poté přejde do režimu HALT a výstupy PLC jsou zablokovány. Provození programu (řízení technologie) se zastaví! Dále se načte nový program PLC z balíčku (balíček je soubor vytvořený prostředím Mosaic, který obsahuje kompletní program včetně web stránek a je uložený v souborovém systému PLC). Pak PLC přejde do režimu RUN. Během přechodu do RUN dojde ke kompletní inicializaci IO systému. Typ restartu je zvolen automaticky podle balíčku (pokud jsou v uživatelském programu nějaké RETAIN proměnné tak se provede teplý restart, jinak se provede studený restart). Závěrem dojde ke smazání použitého balíčku z disku PLC.

Funkce vrací *TRUE* pokud soubor s balíčkem existuje. Pokud soubor existuje, ale neobsahuje platný balíček, pak v režimu HALT nevyjde kontrola nového balíčku a dojde ke spuštění původního programu. Pokud soubor s balíčkem neexistuje pak funkce vrátí *FALSE* a k přechodu PLC do režimu HALT nedojde.

Tato funkce je podporována pouze na centrálních jednotkách řady I (systémy TC800 a Foxtrot 2). Centrální jednotky řady K a L (systémy TC700 a Foxtrot 1) tuto funkci nepodporují.

Příklad programu s voláním funkce *LoadNewPackage2* :

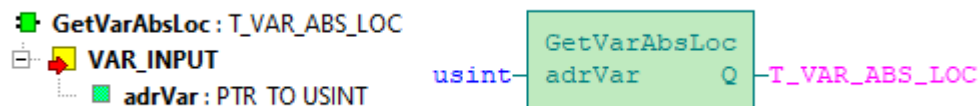
```
PROGRAM ExampleLoadPackage
VAR
  rqNewProg      : BOOL;
  jak            : BOOL;
  newPckgName   : STRING := 'WWW/UPLOAD/NEW_PACK.TTR';
END_VAR

IF rqNewProg THEN
  jak := LoadNewPackage2( rq := rqNewProg, pckg := newPckgName);
  rqNewProg := 0;
END_IF;
END_PROGRAM
```

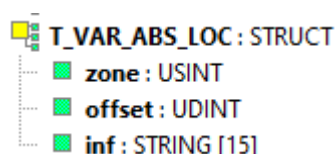


### 4.43 Funkce GetVarAbsLoc

Knihovna : SysLib



Funkce *GetVarAbsLoc* vrací informace o absolutním umístění proměnné, která je specifikovaná vstupem *adrVar*. Struktura informací, kterou funkce vrací, je následující



Význam jednotlivých položek je následující:

- zone zóna paměti PLC ... 0 = %X, 1 = %Y, 2 = %S, 3 = %R
- offset offset proměnné v dané zóně
- inf textová informace (např. %R130)

Popis proměnných :

	Proměnná	Typ	Význam
<b>VAR_INPUT</b>			
	<i>adrVar</i>	PTR_TO USINT	pointer to variable
<b>GetVarAbsLoc</b>			
	Návratová hodnota	T_VAR_ABS_LOC	Funkce vrací informace o absolutním umístění proměnné (nelze použít pro proměnné typu BOOL!)

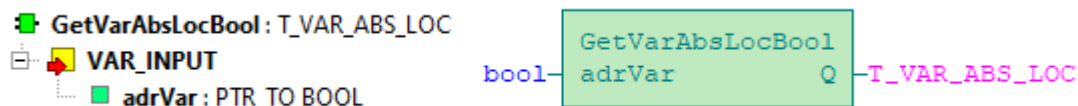
Příklad programu s voláním funkce *GetVarAbsLoc* :

```
PROGRAM prgTestVarAbsLoc
VAR
  test      : USINT;
  testBool  : BOOL;
  varInfo   : T_VAR_ABS_LOC;
  varInfoBool : T_VAR_ABS_LOC;
END_VAR

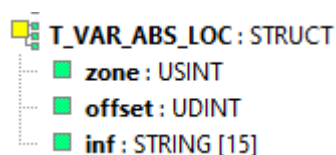
varInfo := GetVarAbsLoc( adrVar := ADR( test));

varInfoBool := GetVarAbsLocBool( adrVar := ADR( testBool));
END_PROGRAM
```

#### 4.44 Funkce *GetVarAbsLocBool*

Knihovna : *SysLib*

Funkce *GetVarAbsLocBool* vrací informace o absolutním umístění proměnné typu **BOOL**, která je specifikovaná vstupem *adrVar*. Struktura informací, kterou funkce vrací, je následující



Význam jednotlivých položek je následující:

- zone zóna paměti PLC ... 0 = %X, 1 = %Y, 2 = %S, 3 = %R
- offset offset proměnné v dané zóně
- inf textová informace (např. %R132.5)

Popis proměnných :

	Proměnná	Typ	Význam
<b>VAR_INPUT</b>			
	<i>adrVar</i>	PTR_TO USINT	pointer to variable
<b>GetVarAbsLocBool</b>			
	Návratová hodnota	T_VAR_ABS_LOC	Funkce vrací informace o absolutním umístění proměnné (pouze pro proměnné typu BOOL!)


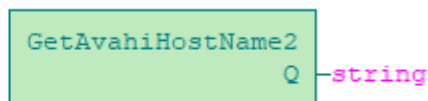
Příklad programu s voláním funkce *GetVarAbsLocBool* :

```
PROGRAM prgTestVarAbsLoc
VAR
  test      : USINT;
  testBool  : BOOL;
  varInfo   : T_VAR_ABS_LOC;
  varInfoBool : T_VAR_ABS_LOC;
END_VAR

varInfo := GetVarAbsLoc( adrVar := ADR( test));

varInfoBool := GetVarAbsLocBool( adrVar := ADR( testBool));
END_PROGRAM
```

#### 4.45 Funkce *GetAvahiHostName2*

Knihovna : *SysLib* **GetAvahiHostName2** : STRING

Funkce *GetAvahiHostName2* vrátí aktuální hostname služby Avahi. Funkce nemá žádné vstupní parametry.

Je-li služba Avahi v PLC zakázána nebo se právě rekonfiguruje tak funkce *GetAvahiHostName2* vrací prázdný string.

Tato funkce je podporována pouze na centrálních jednotkách řady I (systémy TC800 a Foxtrot 2) od verze v2.2.053. Centrální jednotky řady K a L (systémy TC700 a Foxtrot 1) tuto funkci nepodporují. Do knihovny SysLib je funkce zařazena od verze SysLib\_v48.

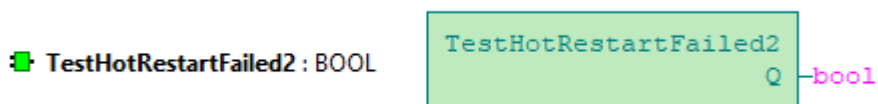
Příklad programu s voláním funkce *GetAvahiHostName2* :

```
PROGRAM prgTestAvahiHostName
VAR
  rqTest          : BOOL := 1;
  avahiHostName  : STRING;
END_VAR

IF rqTest THEN
  avahiHostName := GetAvahiHostName2();
  rqTest := 0;
END_IF;

END_PROGRAM
```

#### 4.46 Funkce *TestHotRestartFailed2*

Knihovna : *SysLib*

Funkce *TestHotRestartFailed2* vrátí TRUE pokud se při teplém restartu PLC nepodaří obnovit hodnoty RETAIN proměnných. Funkce nemá žádné vstupní parametry a vrací výstup pouze na dobu jednoho cyklu.

Tato funkce je podporována pouze na centrálních jednotkách řady I (systémy TC800 a Foxtrot 2) od verze v2.2.056. Centrální jednotky řady K a L (systémy TC700 a Foxtrot 1) tuto funkci nepodporují. Do knihovny SysLib je funkce zařazena od verze SysLib\_v49.

Příklad.

Funkce je užitečná například v případě, že potřebujeme vícekrát zopakovat stejný program pro PLC, ve kterém je použito větší množství RETAIN proměnných. Příkladem může být řízení 10-ti stejných strojů pomocí 10-ti stejných PLC. Na prvním PLC se odladí nastavení RETAIN proměnných pro řízení stroje a to se pomocí web stránky v PLC uloží do souboru např. `MyBackup.RMF` (viz stránka <https://192.168.134.176:8443/cgi-bin/tecomat/retains.cgi>). Tento soubor je třeba přidat do projektu v Mosaicu, nahrát ho do dalších PLC jako jednu ze součástí programu a v další 9-ti strojích ho pak stačí načíst ve chvíli, kdy PLC s daným programem startuje úplně poprvé. Při úplně prvním teplém startu totiž selže načtení hodnot RETAIN proměnných protože dosud nebyly v daném PLC žádné uloženy. Načtení hodnot odladěných v prvním PLC tedy zajistí následující program.

```
PROGRAM prgExampleRetain
VAR
  rqLoad          : BOOL;
  LoadRemFromFile : fbLoadRemFromFile;
END_VAR

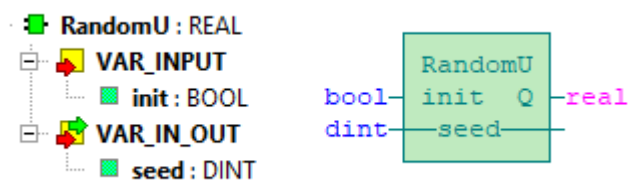
IF TestHotRestartFailed2() THEN
  // když se nepodaří obnovit RETAIN promenne, tak nacist 'MyBackup.RMF'
  rqLoad := 1;
END_IF;

LoadRemFromFile(exec := rqLoad, fileName := 'MyBackup.RMF');
IF LoadRemFromFile.done OR LoadRemFromFile.err THEN
  rqLoad := 0;
END_IF;

END_PROGRAM
```

## 4.47 Funkce RandomU

Knihovna : SysLib



Funkce vrací pseudonáhodné číslo v rozsahu 0 až 1. Výstup je dán hodnotou proměnné na vstupu *seed*, která určuje polohu v pseudonáhodné řadě čísel. Při každém volání je hodnota *seed* aktualizována. V případě, že je nastaven vstup *init*, je hodnota na vstupu *seed* inicializována podle aktuálního data a času.

Funkce je implementována na základě algoritmu *R4\_UNIFORM\_01* jehož autorem je John Burkardt.

	<i>Proměnná</i>	<i>Typ</i>	<i>Význam</i>
<b>VAR_INPUT</b>			
	<i>init</i>	BOOL	Inicializovat <i>seed</i> podle data a času
<b>VAR_IN_OUT</b>			
	<i>seed</i>	DINT	Hodnota určující polohu v pseudonáhodné řadě čísel
<b>RandomU</b>			
	<i>Návratová hodnota</i>	REAL	Pseudonáhodné číslo v rozsahu 0 až 1.

Příklad naplnění pole náhodným obsahem.

```
VAR_GLOBAL
  seed : DINT;
END_VAR

PROGRAM prgRandom8
  VAR
    randomVector : ARRAY [0..7] OF USINT;
    idx          : USINT;
    newVector    : BOOL;
    init         : BOOL;
  END_VAR

  IF NOT init THEN
    RandomU(init := true, seed := seed);
    init := true;
    newVector := true;
  END_IF;

  IF newVector THEN
    FOR idx := 0 TO 7 DO
      randomVector[idx] := REAL_TO_USINT(RandomU(seed := seed)*255);
    END_FOR;
    newVector := false;
  END_IF;
END_PROGRAM
```

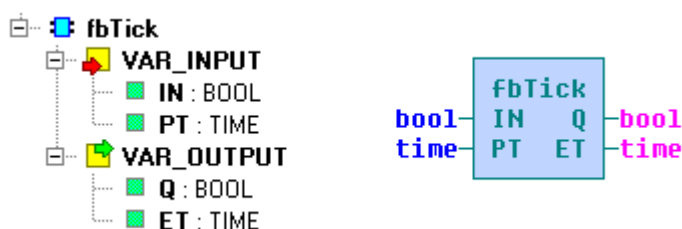
## 5 FUNKČNÍ BLOKY

Knihovna SysLib obsahuje následující funkční bloky:

<b><i>Funkční blok</i></b>	<b><i>Popis</i></b>
<i>fbTick</i>	Časovač periodicky generující pulz na dobu jednoho cyklu
<i>fbBondRFunit</i>	Spárovat RF mastera s RF jednotkou (pouze pro Foxtrot 1)
<i>fbTPR</i>	Časovač generující pulz dané šířky s možností ukončit pulz pomocí vstupu RESET
<i>fbSaveRemToFile</i>	Uložit obsah všech RETAIN proměnných do souboru (pouze pro Foxtrot 2 a TC800)
<i>fbLoadRemFromFile</i>	Obnovit obsah všech RETAIN proměnných ze souboru (pouze pro Foxtrot 2 a TC800)
<i>fbStopwatch100us</i>	Stopky s přesností 100 mikrosekund
<i>fbSetConfigWebPsw2</i>	Nastavit nové jméno a heslo pro přístup ke konfiguračnímu webu PLC (pouze pro Foxtrot 2 a TC800)
<i>fbSetAvahiParam2</i>	Nastavit nové parametry pro službu Avahi v PLC (pouze pro Foxtrot 2 a TC800)
<i>fbFlashRem2</i>	Uložit stav RETAIN proměnných do souboru (pouze pro Foxtrot 2 a TC800)

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

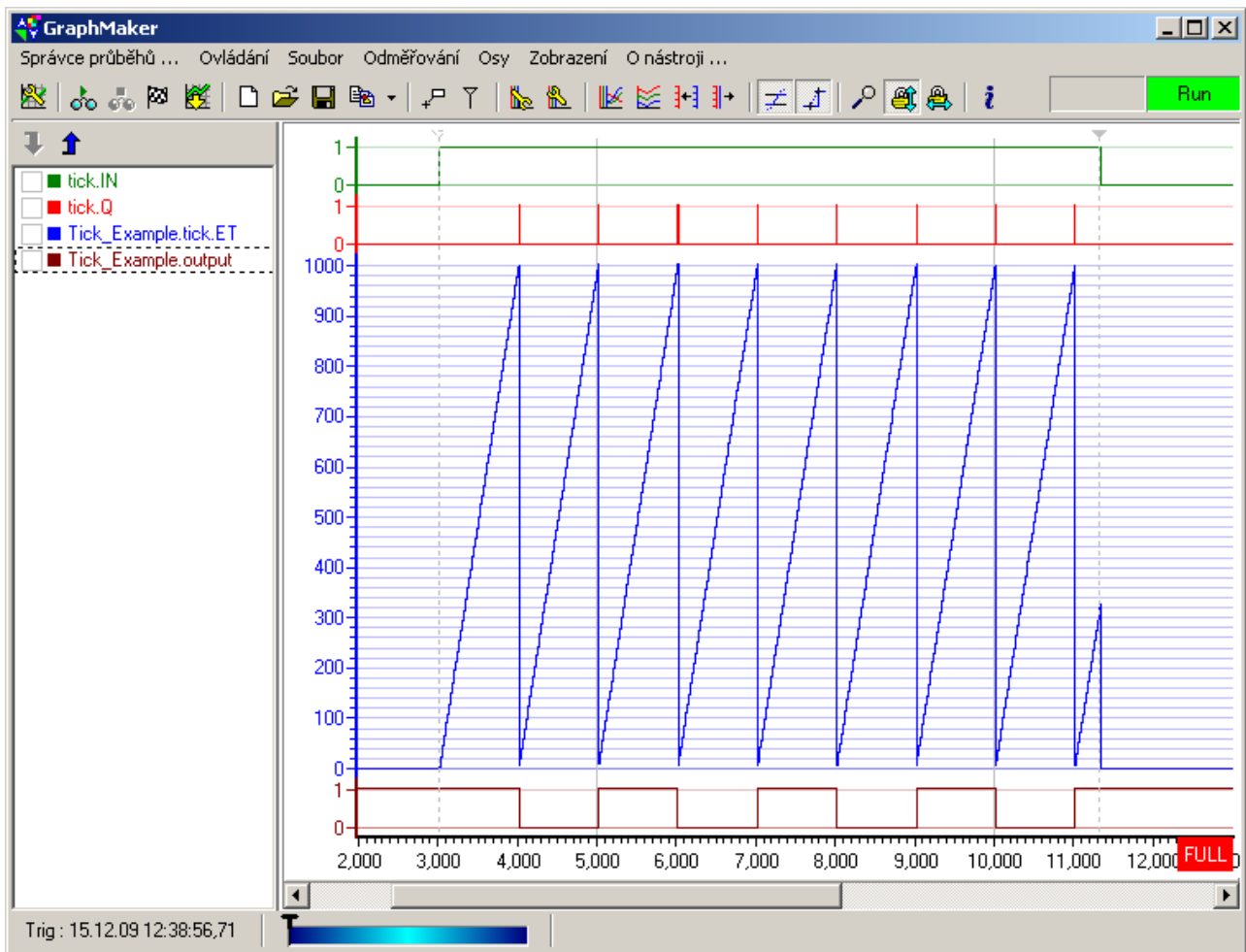
Příklad programu s použitým funkčním blokem *fbTick* :

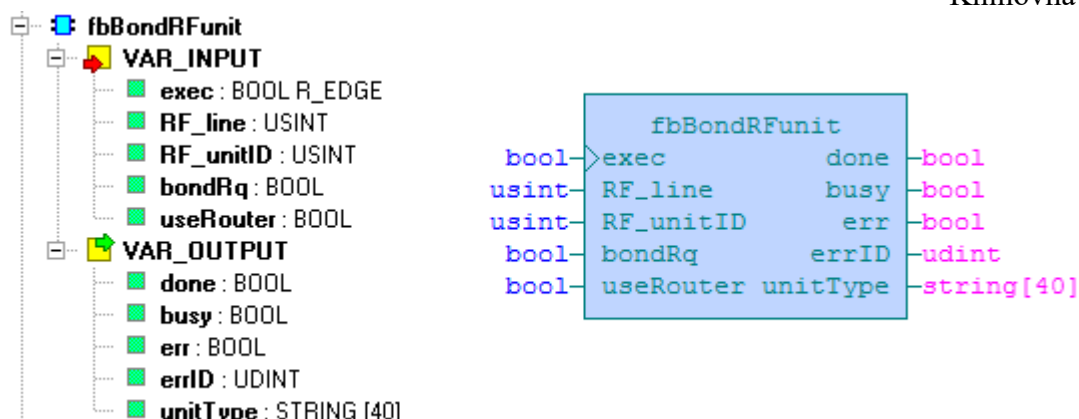
```
PROGRAM TickExample
VAR
    tickEnable : BOOL;
    tick       : fbTick;
    output     : BOOL;
END_VAR

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 napárována, 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 1 od v6.6. Do knihovny SysLib je blok zařazen od verze SysLib\_v27. Blok nelze použít pro řadu Foxtrot 2 a TC800.

Popis proměnných :

	Proměnná	Typ	Význam
<b>VAR_INPUT</b>			
	<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 <i>bondRq=1</i> )

	<i>Proměnná</i>	<i>Typ</i>	<i>Význam</i>
<b>VAR_OUTPUT</b>			
	<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 ( <i>bondRq</i> = 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    := RF0_RF,
                           RF_unitID := i,
                           RF_addr    := customRF[i].address);
        END_IF;
      END_IF;
    END_IF;
  END_FOR;

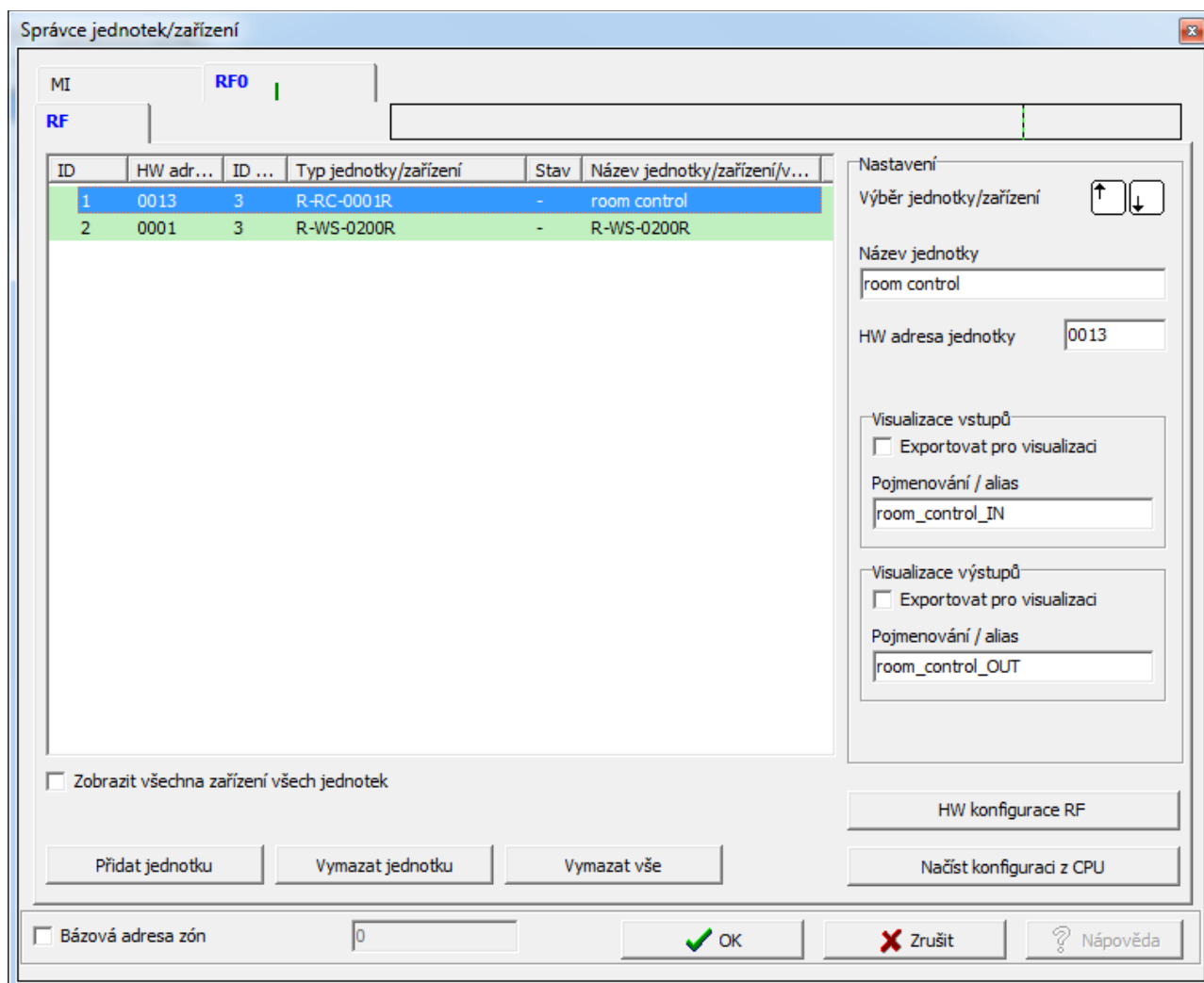
  // set new customer address of RF unit
  IF setAdr THEN
    IF id > 0 AND id <= NUM_RF_UNIT THEN
      IF newAdr <> RF_unit[id].address THEN
        result := SetAddressRFunit( RF_line    := RF0_RF,
                                    RF_unitID := id,
                                    RF_addr    :=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

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



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 : RFO\_RF**

ID	line defined	unit defined	RF UNIT STATE					HW ADDR	TYPE	DESCRIPTION
			NET	BND	SLP	COM	INI			
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  NEW HW ADDR     USE ROUTER

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 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 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

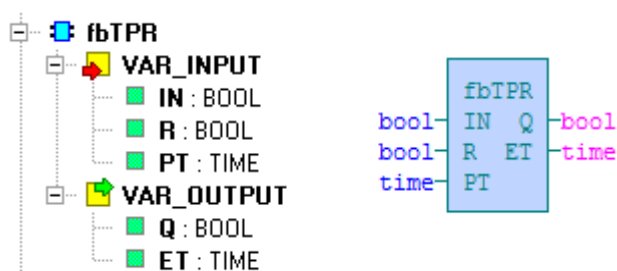
Ve volání instance funkčního bloku *fbBondRFunit* nejprve nastavit proměnnou *exec* na hodnotu TRUE. Na náběžnou hranu proměnné *exec* RF master ukončí běžnou výměnu dat v síti Rfox a pokusí se v několika krocích připárovat novou RF jednotku. Pokud je již na pozici dané parametrem *RF\_unitID* nějaká 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 zavolána 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 párování 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.



### 5.3 Funkční blok fbTPR

Knihovna : SysLib



Funkční blok *fbTPR* je časovač, který na výstupu Q generuje pulz zadané šířky. Náběžná hrana na vstupu IN zahajuje generování pulzu, šířka generovaného pulzu je daná vstupem PT. Vstupem R lze generování pulzu předčasně ukončit.

Tento funkční blok je podporován na všech centrálních jednotkách řady K a L (TC700 CP-7004, Foxtrot) od verze v1.0. Do knihovny SysLib je blok zařazen od verze SysLib\_v21.

Popis proměnných :

	Proměnná	Typ	Význam
<b>VAR_INPUT</b>			
	IN	BOOL	Náběžná hrana na tomto vstupu zahájí generování pulzu
	R	BOOL	Reset časovače (ukončí generování pulzu)
	PT	TIME	Předvolba časovače (šířka generovaného pulzu)
<b>VAR_OUTPUT</b>			
	Q	BOOL	Výstup časovače
	ET	TIME	Aktuální hodnota časovače

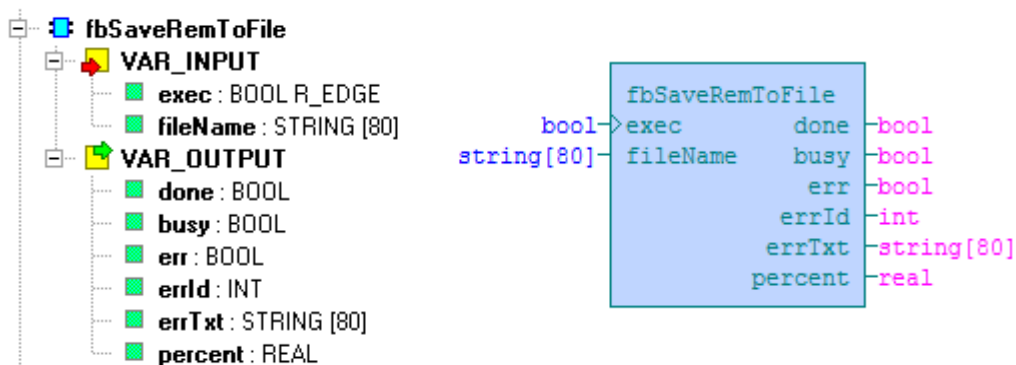
Příklad programu s použitým funkčním blokem *fbTPR* :

```
PROGRAM prgTprExample
  VAR
    rqPulse      : BOOL;
    rqReset      : BOOL;
    timTPR       : fbTPR;
    output       : BOOL;
  END_VAR

  timTPR( IN := rqPulse, IN := rqReset, PT := T#10s, Q => output);
END_PROGRAM
```



## 5.4 Funkční blok *fbSaveRemToFile*

Knihovna : *SysLib*






Funkční blok *fbSaveRemToFile* slouží k uložení hodnot všech RETAIN proměnných do souboru. Ukládání je zahájeno na náběžnou hranu vstupu *EXEC* a může trvat i desítky cyklů PLC v závislosti na množství ukládaných proměnných. Funkční blok si nejprve udělá kopii všech RETAIN proměnných (aby byly ukládané hodnoty konzistentní tak je to kopie stavu, jaký měly proměnné na konci minulého cyklu) a pak zahájí vlastní ukládání hodnot, které se provádí „na pozadí“ s nízkou prioritou, takže prakticky neovlivňuje dobu cyklu PLC. Jméno souboru, do kterého se hodnoty RETAIN proměnných uloží, je dáno vstupem *fileName*. Během ukládání proměnných je výstup *busy* nastaven na TRUE a výstup *percent* udává kolik procent proměnných je aktuálně uloženo. Výstup *done* se nastaví na TRUE v okamžiky kdy jsou všechny hodnoty úspěšně uloženy. V případě nějaké chyby se nastaví na TRUE výstup *err*, výstup *errId* v tom případě obsahuje kód chyby a výstup *errTxt* obsahuje textový popis chyby.

Do souboru se ukládají kompletní informace o proměnných (jméno proměnné, datový typ, aktuální hodnota, inicializační hodnota, ...). To umožňuje hodnoty RETAIN proměnných kdykoliv ze souboru obnovit a to i v případech, že se změnilo mapování RETAIN proměnných nebo když uložený vzorek úplně přesně neodpovídá aktuální definici RETAIN proměnných (např. došlo ke změně datového typu proměnné) apod. Obnovení hodnot RETAIN proměnných se provádí pomocí funkčního bloku *fbLoadRemFromFile*.

Tento funkční blok je podporován pouze na centrálních jednotkách řady I (systémy TC800 a Foxtrot 2). Centrální jednotky řady K a L (systémy TC700 a Foxtrot 1) tento blok nepodporují. Do knihovny SysLib je blok zařazen od verze SysLib\_v39.

Popis proměnných :

	Proměnná	Typ	Význam
<b>VAR_INPUT</b>			
	<i>exec</i>	BOOL R_EDGE	Náběžná hrana na tomto vstupu zahájí ukládání RETAIN proměnných do souboru
	<i>fileName</i>	STRING	Název souboru, do kterého se bude ukládat
<b>VAR_OUTPUT</b>			
	<i>done</i>	BOOL	Hotovo (výstup je nastaven na TRUE po dobu jednoho cyklu PLC)

	<b>Proměnná</b>	<b>Typ</b>	<b>Význam</b>
	<i>busy</i>	BOOL	Probíhá ukládání hodnot RETAIN proměnných
	<i>err</i>	BOOL	Při ukládání hodnot došlo k chybě (výstup je nastaven na TRUE po dobu jednoho cyklu PLC)
	<i>errId</i>	INT	Kód chyby 10 ... v programu nejsou žádné RETAIN proměnné 11 ... není vytvořen vzorek RETAIN dat 12 ... nenalezen soubor PROGRAM.RMF 13 ... chyba CRC hlavičky remanentní zóny 14 ... chyba při zpracování souboru PROGRAM.RMF 15 ... velikost remanentní zóny neodpovídá počtu RETAIN proměnných 16 ... nesouhlasí HASH pro identifikaci RMF souboru 17 ... chyba CRC celé remanentní zóny
	<i>errTxt</i>	STRING	Popis chyby
	<i>percent</i>	REAL	Kolik procent proměnných je již zpracováno

Příklad programu s použitým funkčním blokem *fbSaveRemToFile* :

```
VAR_GLOBAL RETAIN
```

```
arr1 : ARRAY [0..5] OF STRING [10] :=
      ['zero', 'one', 'two', 'three', 'four', 'five'];
```

```
arr2 : ARRAY [0..2, 0..3] OF REAL :=
      [ 0.0, 0.1, 0.2, 0.3,
        1.0, 1.1, 1.2, 1.3,
        2.0, 2.1, 2.2, 2.3];
```

```
arr3 : ARRAY [0..1, 0..2, 0..3] OF BYTE :=
      [ 000, 001, 002, 003,
        010, 011, 012, 013,
        020, 021, 022, 023,
        100, 101, 102, 103,
        110, 111, 112, 113,
        120, 121, 122, 123 ];
```

```
END_VAR
```

```
PROGRAM prgSaveRetainExample
```

```
VAR RETAIN
```

```
retVar1 : REAL; // local RETAIN variable
```

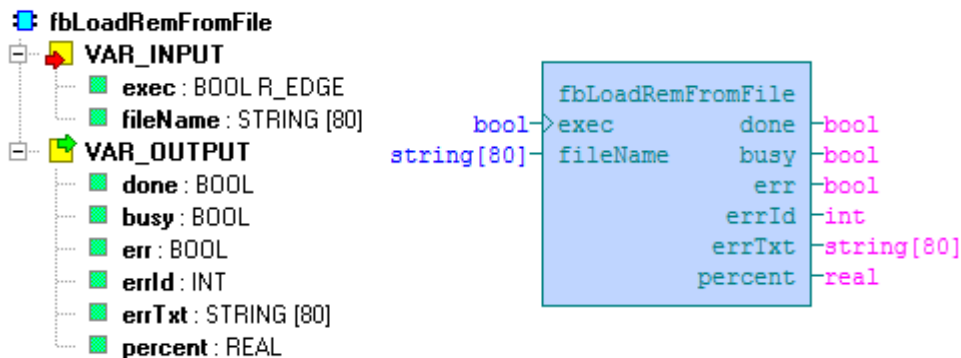
```
END_VAR
```

```
VAR
```

```
rqSave : BOOL;
SaveRemToFile : fbSaveRemToFile;
cntSaveOk : UDINT;
cntSaveErr : UDINT;
SaveTimer : TON;
saveTime : TIME;
```

```
    rqLoad          : BOOL;  
    LoadRemFromFile : fbLoadRemFromFile;  
    cntLoadOk       : UDINT;  
    cntLoadErr      : UDINT;  
    LoadTimer       : TON;  
    loadTime        : TIME;  
END_VAR  
  
SaveRemToFile(exec := rqSave, fileName := 'MyBackup.RMF');  
IF SaveRemToFile.done OR SaveRemToFile.err THEN  
    rqSave := 0;  
    IF SaveRemToFile.done THEN  
        cntSaveOk := cntSaveOk + 1;  
    ELSE  
        cntSaveErr := cntSaveErr + 1;  
    END_IF;  
END_IF;  
  
SaveTimer( IN := rqSave, PT := T#1h);  
IF rqSave THEN  
    saveTime := SaveTimer.ET;    // time of saving  
END_IF;  
  
LoadRemFromFile(exec := rqLoad, fileName := 'MyBackup.RMF');  
IF LoadRemFromFile.done OR LoadRemFromFile.err THEN  
    rqLoad := 0;  
    IF LoadRemFromFile.done THEN  
        cntLoadOk := cntLoadOk + 1;  
    ELSE  
        cntLoadErr := cntLoadErr + 1;  
    END_IF;  
END_IF;  
  
LoadTimer( IN := rqLoad, PT := T#1h);  
IF rqLoad THEN  
    loadTime := LoadTimer.ET;    // time of loading  
END_IF;  
END_PROGRAM
```

## 5.5 Funkční blok *fbLoadRemFromFile*

Knihovna : *SysLib*

Funkční blok *fbLoadRemFromFile* slouží k obnově hodnot všech RETAIN proměnných ze souboru. Předpokládá se, že soubor byl vytvořen pomocí funkčního bloku *fbSaveRemToFile*. Načítání hodnot ze souboru je zahájeno na náběžnou hranu vstupu *EXEC* a může trvat i desítky cyklů PLC v závislosti na množství proměnných. Funkční blok obnovuje postupně hodnoty všech RETAIN proměnných nejprve do vnitřního bufferu (načítání hodnot se provádí „na pozadí“ s nízkou prioritou) a teprve poté, co jsou takto načteny hodnoty všech proměnných tak se provede jejich nastavení v paměti PLC (v jednom cyklu PLC aby byly načtené hodnoty konzistentní). Jméno souboru, ze kterého se hodnoty RETAIN proměnných obnoví, je dáno vstupem *fileName*. Během načítání proměnných je výstup *busy* nastaven na TRUE a výstup *percent* udává kolik procent proměnných je aktuálně načteno. Výstup *done* se nastaví na TRUE v okamžiku kdy jsou všechny hodnoty úspěšně načteny. V případě nějaké chyby se nastaví na TRUE výstup *err*, výstup *errId* v tom případě obsahuje kód chyby a výstup *errTxt* obsahuje textový popis chyby.









Soubor, ze kterého chceme hodnoty RETAIN proměnných obnovit, musí být vytvořen funkčním blokem *fbSaveRemToFile*.

Obnova hodnot RETAIN proměnných probíhá podle následujících pravidel:

- pro obnovu hodnoty je rozhodující jméno proměnné k obnově hodnoty proměnné ze souboru dojde pouze v případě, že se shoduje jméno proměnné v aktuálním programu se jménem proměnné uloženým v souboru
- adresa proměnné v paměti nehraje roli adresa proměnné v okamžiku jejího uložení do souboru a adresa proměnné, jejíž hodnota se obnovuje, se mohou lišit
- pokud se změní datový typ proměnné tak se při načtení provede odpovídající konverze např. pokud byla do souboru uložena hodnota proměnné typu USINT a při obnově je odpovídající proměnná typu UDINT tak se provede konverze USINT\_TO\_UDINT
- stav těch proměnných, které se v souboru nenajdou, se nezmění to se týká proměnných, které v programu přibýly až poté, co bylo provedeno uložení hodnot do souboru funkčním blokem *fbSaveRemToFile*
- proměnné, jejichž hodnoty jsou v souboru uloženy, ale v aktuálním programu neexistují, nehrají roli to se týká proměnných, které byly z programu vypuštěny poté co bylo provedeno uložení hodnot do souboru funkčním blokem *fbSaveRemToFile*

Tento funkční blok je podporován pouze na centrálních jednotkách řady I (systémy TC800 a Foxtrot 2). Centrální jednotky řady K a L (systémy TC700 a Foxtrot 1) tento blok nepodporují. Do knihovny SysLib je blok zařazen od verze SysLib\_v39.

Popis proměnných :

	<i>Proměnná</i>	<i>Typ</i>	<i>Význam</i>
<b>VAR_INPUT</b>			
	<i>exec</i>	BOOL R_EDGE	Náběžná hrana na tomto vstupu zahájí načítání RETAIN proměnných ze souboru
	<i>fileName</i>	STRING	Název souboru, ze kterého se bude číst
<b>VAR_OUTPUT</b>			
	<i>done</i>	BOOL	Hotovo (výstup je nastaven na TRUE po dobu jednoho cyklu PLC)
	<i>busy</i>	BOOL	Probíhá načítání hodnot RETAIN proměnných
	<i>err</i>	BOOL	Při načítání hodnot došlo k chybě (výstup je nastaven na TRUE po dobu jednoho cyklu PLC)
	<i>errId</i>	INT	Kód chyby 20 ... v programu nejsou žádné RETAIN proměnné 21 ... soubor s RETAIN hodnotami nelze otevřít 22 ... nenalezen soubor PROGRAM.RMF 23 ... chyba při zpracování souboru s RETAIN hodnotami 24 ... chyba při zpracování souboru PROGRAM.RMF
	<i>errTxt</i>	STRING	Popis chyby
	<i>percent</i>	REAL	Kolik procent proměnných je již zpracováno

Příklad programu s použitým funkčním blokem *fbLoadRemFromFile* :

```

VAR_GLOBAL RETAIN

arr1 : ARRAY [0..5] OF STRING [10] :=
    ['zero', 'one', 'two', 'three', 'four', 'five'];

arr2 : ARRAY [0..2, 0..3] OF REAL :=
    [ 0.0, 0.1, 0.2, 0.3,
      1.0, 1.1, 1.2, 1.3,
      2.0, 2.1, 2.2, 2.3];

arr3 : ARRAY [0..1, 0..2, 0..3] OF BYTE :=
    [ 000, 001, 002, 003,
      010, 011, 012, 013,
      020, 021, 022, 023,
      100, 101, 102, 103,
      110, 111, 112, 113,
      120, 121, 122, 123 ];

END_VAR

```

```
PROGRAM prgLoadRetainExample
VAR RETAIN
  retVar1      : REAL;      // local RETAIN variable
END_VAR

VAR
  rqSave      : BOOL;
  SaveRemToFile : fbSaveRemToFile;
  cntSaveOk   : UDINT;
  cntSaveErr  : UDINT;
  SaveTimer   : TON;
  saveTime    : TIME;

  rqLoad      : BOOL;
  LoadRemFromFile : fbLoadRemFromFile;
  cntLoadOk   : UDINT;
  cntLoadErr  : UDINT;
  LoadTimer   : TON;
  loadTime    : TIME;
END_VAR

SaveRemToFile(exec := rqSave, fileName := 'MyBackup.RMF');
IF SaveRemToFile.done OR SaveRemToFile.err THEN
  rqSave := 0;
  IF SaveRemToFile.done THEN
    cntSaveOk := cntSaveOk + 1;
  ELSE
    cntSaveErr := cntSaveErr + 1;
  END_IF;
END_IF;

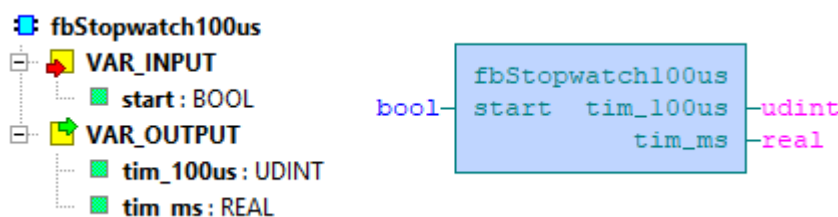
SaveTimer( IN := rqSave, PT := T#1h);
IF rqSave THEN
  saveTime := SaveTimer.ET;  // time of saving
END_IF;

LoadRemFromFile(exec := rqLoad, fileName := 'MyBackup.RMF');
IF LoadRemFromFile.done OR LoadRemFromFile.err THEN
  rqLoad := 0;
  IF LoadRemFromFile.done THEN
    cntLoadOk := cntLoadOk + 1;
  ELSE
    cntLoadErr := cntLoadErr + 1;
  END_IF;
END_IF;

LoadTimer( IN := rqLoad, PT := T#1h);
IF rqLoad THEN
  loadTime := LoadTimer.ET;  // time of loading
END_IF;

END_PROGRAM
```

## 5.6 Funkční blok *fbStopwatch100us*

Knihovna : *SysLib*

Funkční blok *fbStopwatch100us* funguje jako stopky. Blok odměřuje čas mezi dvěma voláními *fbStopwatch100us*. Měření se zahájí zavoláním bloku s parametrem *start = 1*. Další volání bloku s parametrem *start = 0* vrátí čas, který uběhl od prvního zavolání bloku. Měřit čas je možné jak během jednoho cyklu PLC tak v průběhu několika cyklů.

Funkční blok *fbStopwatch100us* vrací čas s přesností 100 mikrosekund. Minimální hodnota je 0,1 ms, maximální hodnota je 4d 23h 18m 16s 729,6ms. Blok vrací naměřenou hodnotu ve dvou formátech:

- *tim\_100us* : UDINT                      naměřený čas ve stovkách mikrosekund
- *tim\_ms*        : REAL                      naměřený čas v milisekundách

Tento funkční blok je podporován na všech centrálních jednotkách řady K a L (TC700 CP-7004, Foxtrot 1 CP-1xxx) od verze v5.0. Na centrálních jednotkách řady I (systémy TC800 a Foxtrot 2) je blok podporován ve všech verzích. Do knihovny SysLib je blok zařazen od verze SysLib\_v45.

Popis proměnných :

	Proměnná	Typ	Význam
<b>VAR_INPUT</b>			
	<i>start</i>	BOOL	1 = zahájit měření 0 = vrátit čas od začátku měření
<b>VAR_OUTPUT</b>			
	<i>tim_100us</i>	UDINT	naměřený čas ve stovkách mikrosekund
	<i>tim_ms</i>	REAL	naměřený čas v milisekundách

Následující příklad ukazuje použití funkčního bloku *fbStopwatch100us* pro měření doby, kterou zabere výpočet obsazeného místa v adresáři WWW/LOGS/.

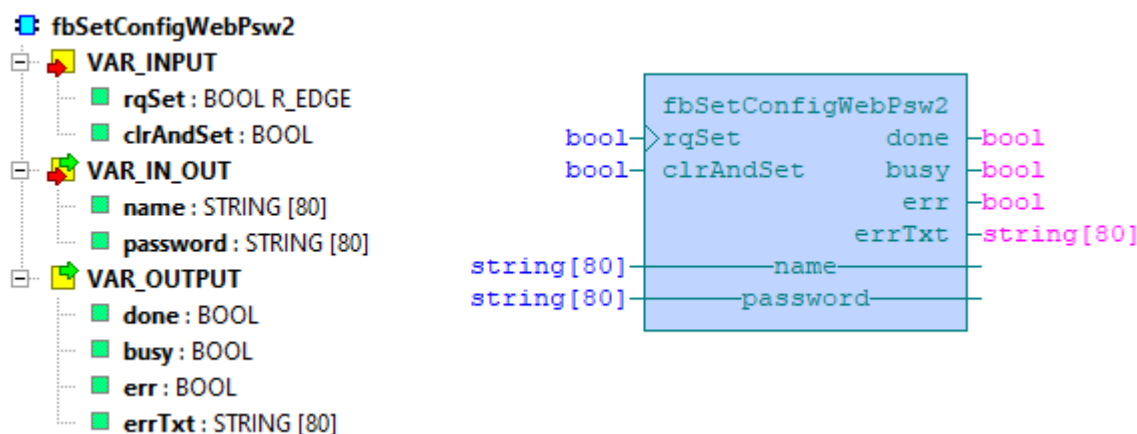
```
PROGRAM prgExampleStopwatch
VAR
  enable      : BOOL := 1;
  Stopwatch   : fbStopwatch100us;
  ListDir     : ListDirectories;
  execTime    : REAL;
  usedSpace   : UDINT;
```

```
END_VAR

IF enable THEN
  Stopwatch(start := 1);
  usedSpace := 0;
  REPEAT
    ListDir(exec := 1, dirName := 'WWW/LOGS/');
    IF ListDir.found AND NOT ListDir.isDir THEN
      usedSpace := usedSpace + ListDir.fileInfo.fileSize;
    END_IF;
  UNTIL ListDir.done or ListDir.err END_REPEAT;
  Stopwatch(start := 0, tim_ms => execTime);
  enable := 0;
END_IF;

END_PROGRAM
```



5.7 Funkční blok *fbSetConfigWebPsw2*Knihovna : *SysLib*

Funkční blok *fbSetConfigWebPsw2* nastaví jméno a heslo pro přístup na konfigurační web stránky PLC (Merecat web server). Nastavení je zahájeno na náběžnou hranu proměnné *rqSet* a bude trvat několik cyklů PLC. Výstup *busy* má během nastavování hodnotu TRUE. Pokud se nastavení podaří úspěšně dokončit, pak se výstup *busy* nastaví na FALSE a dále je na dobu jednoho cyklu nastaven výstup *done*. Naopak pokud při nastavení nové jména a hesla dojde k chybě, pak je nastaven výstup *err* na TRUE a v proměnné *errTxt* je uveden popis chyby.

Tento funkční blok je podporován pouze na centrálních jednotkách řady I (systémy TC800 a Foxtrot 2) od verze v2.2.053. Centrální jednotky řady K a L (systémy TC700 a Foxtrot 1) tento blok nepodporují. Do knihovny SysLib je blok zařazen od verze SysLib\_v48.

Popis proměnných :

	<b>Proměnná</b>	<b>Typ</b>	<b>Význam</b>
<b>VAR_INPUT</b>			
	<i>rqSet</i>	BOOL R_EDGE	žádost o nastavení nového jména a hesla
	<i>clrAndSet</i>	BOOL	TRUE znamená smazat stará jména a hesla a založit jedno nové FALSE znamená ponechat stávající jména a hesla a přidat k nim další
<b>VAR_IN_OUT</b>			
	<i>name</i>	STRING	nové přihlašovací jméno
	<i>password</i>	STRING	nové heslo
<b>VAR_OUTPUT</b>			
	<i>done</i>	BOOL	jméno a heslo nastaveno
	<i>busy</i>	BOOL	blok pracuje
	<i>err</i>	BOOL	došlo k chybě
	<i>errTxt</i>	STRING	popis chyby

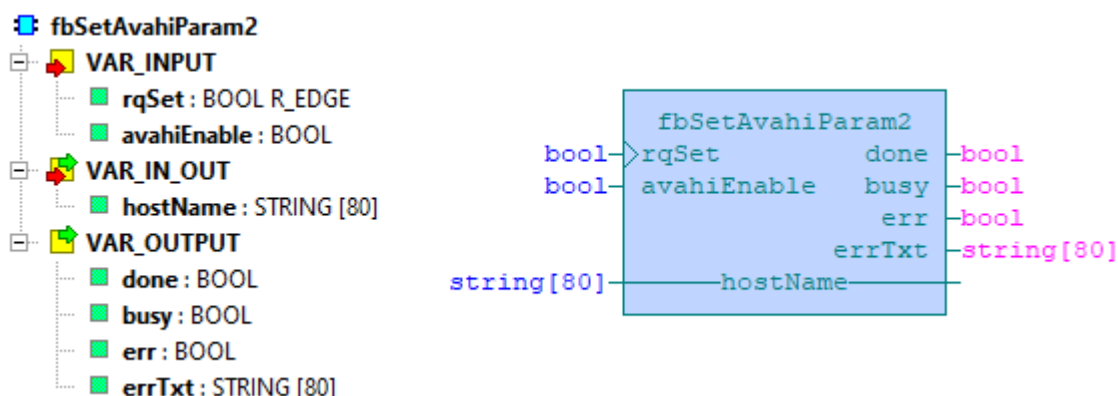
Následující příklad ukazuje použití funkčního bloku *fbSetConfigWebPsw2*. Po nastavení proměnné *rqSet* se přidá další přihlášení (jméno + heslo) do seznamu pro přístup na konfigurační web stránky PLC.

```
PROGRAM prgMain
VAR
  rqSet          : BOOL;
  name           : STRING := 'jmeno';
  pass          : STRING := 'heslo';
  SetConfigWebPSW : fbSetConfigWebPsw2;
  cntOK         : UDINT;
  cntERR        : UDINT;
  errTxt        : STRING
END_VAR

SetConfigWebPSW( rqSet      := rqSet,
                 clrAndSet := 0,
                 name      := name,
                 password  := pass);

IF rqSet THEN
  IF SetConfigWebPSW.done THEN
    cntOK := cntOK + 1;
    rqSet := 0;
  END_IF;
  IF SetConfigWebPSW.err THEN
    errTxt := SetConfigWebPSW.errTxt;
    cntERR := cntERR + 1;
    rqSet := 0;
  END_IF;
END_IF;

END_PROGRAM
```

5.8 Funkční blok *fbSetAvahiParam2*Knihovna : *SysLib*

Funkční blok *fbSetAvahiParam2* nastaví nové parametry pro službu Avahi v PLC. Pomocí tohoto bloku je možné povolit službu Avahi v PLC a nastavit jí název hostitele (hostname) nebo lze službu Avahi zakázat. Název hostitele musí být bez mezer a nesmí obsahovat národní znaky. Nastavení parametrů je zahájeno na náběžnou hranu proměnné *rqSet* a bude trvat několik cyklů PLC. Výstup *busy* má během nastavování hodnotu TRUE. Pokud se nastavení podaří úspěšně dokončit, pak se výstup *busy* nastaví na FALSE a dále je na dobu jednoho cyklu nastaven výstup *done*. Naopak pokud při nastavení nové jména a hesla dojde k chybě, pak je nastaven výstup *err* na TRUE a v proměnné *errTxt* je uveden popis chyby.

Tento funkční blok je podporován pouze na centrálních jednotkách řady I (systémy TC800 a Foxtrot 2) od verze v2.2.053. Centrální jednotky řady K a L (systémy TC700 a Foxtrot 1) tento blok nepodporují. Do knihovny SysLib je blok zařazen od verze SysLib\_v48.

Popis proměnných :

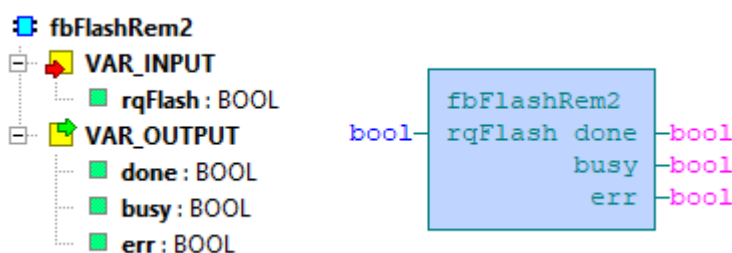
	<i>Proměnná</i>	<i>Typ</i>	<i>Význam</i>
<b>VAR_INPUT</b>			
	<i>rqSet</i>	BOOL R_EDGE	žádost o nastavení nových parametrů pro službu Avahi
	<i>avahiEnable</i>	BOOL	povolit nebo zakázat službu Avahi
<b>VAR_IN_OUT</b>			
	<i>hostName</i>	STRING	Nový název hostitele (Avahi hostname)
<b>VAR_OUTPUT</b>			
	<i>done</i>	BOOL	nové jméno nastaveno
	<i>busy</i>	BOOL	blok pracuje
	<i>err</i>	BOOL	došlo k chybě
	<i>errTxt</i>	STRING	popis chyby

Následující příklad ukazuje použití funkčního bloku *fbSetAvahiParam2*. Po nastavení proměnné *rqSet* se zapne služba Avahi a nastaví se název hostitele. Pokud je již služba zapnutá a název hostitele souhlasí s požadovaným názvem, tak funkční blok pouze nastaví výstup *done* a dál nic nedělá.

```
PROGRAM prgTestAvahiParam
VAR
  rqSet          : BOOL := 1;
  name           : STRING := 'myFox';
  SetAvahiHostName : fbSetAvahiParam2;
  cntOK          : UDINT;
  cntERR         : UDINT;
  actAvahiHostName : STRING;
END_VAR

SetAvahiHostName( rqSet := rqSet, avahiEnable := 1, hostName := name);
IF rqSet THEN
IF SetAvahiHostName.done THEN
  cntOK := cntOK + 1;
  rqSet := 0;
END_IF;
IF SetAvahiHostName.err THEN
  cntERR := cntERR + 1;
  rqSet := 0;
END_IF;
END_IF;

actAvahiHostName := GetAvahiHostName2();
END_PROGRAM
```

5.9 Funkční blok *fbFlashRem2*Knihovna : *SysLib*

Funkční blok *fbFlashRem2* slouží k uložení stavu RETAIN proměnných do souboru. Pokud je vstup *rqFlash* TRUE, tak se nastaví požadavek na uložení RETAIN proměnných. K uložení dojde v následující otočce cyklu PLC. Hodnoty proměnných jsou uloženy do souboru, ze kterého se při teplém restartu PLC automaticky obnoví. Takto lze uložit RETAIN proměnné až 10x za poslední hodinu. Případné další požadavky pak *fbFlashRem2* ignoruje až do chvíle, kdy počet zápisů RETAIN proměnných do souboru za poslední hodinu klesne pod 10 (uplyne alespoň jedna hodina od nejstaršího uložení souboru). To je ochrana PLC systému proti příliš četnému přepisování souboru s hodnotami RETAIN proměnných, který je realizován v paměti typu FLASH a má tedy počet zápisů paměti omezen.

Pokud se podaří zaregistrovat požadavek na uložení RETAIN proměnných tak je nastaven výstup *done*. Když je překročen max. počet zápisů za jednu hodinu, tak se nastaví výstup *busy* a k uložení nedojde. Pokud se funkční blok *fbFlashRem2* zavolá v programu, kde nejsou definovány žádné RETAIN proměnné, pak je nastaven výstup *err*.

Tento funkční blok je podporován pouze na centrálních jednotkách řady I (systémy TC800 a Foxtrot 2) od verze v2.2.056. Centrální jednotky řady K a L (systémy TC700 a Foxtrot 1) tento blok nepodporují. Do knihovny SysLib je blok zařazen od verze SysLib\_v49.

Popis proměnných :

	<i>Proměnná</i>	<i>Typ</i>	<i>Význam</i>
<b>VAR_INPUT</b>			
	<i>rqFlash</i>	BOOL	žádost o jednorázové uložení RETAIN proměnných
<b>VAR_OUTPUT</b>			
	<i>done</i>	BOOL	RETAIN proměnné budou uloženy na konci cyklu PLC
	<i>busy</i>	BOOL	Překročena četnost 10 zápisů za poslední hodinu, RETAIN proměnné se neuloží
	<i>err</i>	BOOL	Nejsou definovány žádné RETAIN proměnné

Následující příklad ukazuje použití funkčního bloku *fbFlashRem2*. Předpokládejme, že RETAIN proměnné v PLC programu obsahují sadu parametrů, jejichž hodnoty může uživatel zadávat přes web stránky, vytvořené v nástroji WebMaker. Pokud chce uživatel změnit hodnoty uložit, pak nastaví TRUE do proměnné *rqSaveParam*. Pokud se RETAIN proměnné podaří uložit, tak se výstup *fbFlashRem2.done* nastaví na TRUE a je třeba zapsat do požadavku *rqSaveParam* hodnotu FALSE (ukončit požadavek na zápis). Pokud by se tímto způsobem vygeneroval během jedné hodiny více než 10 požadavků na zápis, tak se nastaví výstup *busy* a proměnná *rqSaveParam*

zůstane nastavená na TRUE až do okamžiku, kdy uběhne 1 hodina od nejstaršího zápisu. Pak se místo výstupu *busy* nastaví TRUE do výstupu *done* a dojde k uložení RETAIN proměnných.

```
VAR_GLOBAL RETAIN
  grParam1 : UDINT;
  grParam2 : UDINT;
  grParam3 : UDINT;
  grParam4 : REAL;
  grParam5 : REAL;
END_VAR

PROGRAM prgMain
  VAR
    rqSaveParam : BOOL;
    cntSaveParam : UDINT;
    FlashRem : fbFlashRem2;
  END_VAR
  VAR RETAIN
    lrParam1 : UDINT;
    lrParam2 : UDINT;
    lrParam3 : UDINT;
    lrParam4 : REAL;
    lrParam5 : REAL;
  END_VAR

  FlashRem( rqFlash := rqSaveParam);
  IF FlashRem.done THEN
    rqSaveParam := FALSE;
    cntSaveParam := cntSaveParam + 1;
  END_IF;
END_PROGRAM
```

#### Poznámka.

Každé uložení RETAIN proměnných je zaznamenáno v log souboru WWW/LOGS/EVENT.LOG. Tento soubor je dostupný buď z programovacího prostředí Mosaic (viz menu PLC | Souborový systém | Otevřít soubor v externí aplikaci) nebo prostřednictvím konfiguračního webu PLC (viz odkaz event.log na stránce <https://192.168.134.176:8443/cgi-bin/tecomat/log.cgi>).



Teco, a.s., Průmyslová zóna Štářalka 984, 280 02 Kolín, tel. 321 401 111

TXV 003 48.01

Výrobce si vyhrazuje právo na změny dokumentace. Poslední aktuální vydání je k dispozici na stránkách  
[www.tecomat.cz](http://www.tecomat.cz)