18

18. SDÍLENÁ A ZÁLOHOVANÁ PAMĚŤ, PŘÍMÉ SDÍLENÍ PROMĚNNÝCH

18.1 Zálohovaná paměť PLC

18.1.1 Práce se zálohovanou pamětí

PLC program má k dispozici zálohovanou paměťovou oblast: **PLC_MEM_BACKUP** o velikosti 10000 bajtů. Při regulérním ukončení systému se automaticky oblast uloží na disk. Při startu systému se oblast automaticky načte. PLC program má možnost si vyžádat okamžité uložení oblasti na disk.

REQ_BACKUP_MEM ....žádost o uložení zálohovaných oblastí

(bit, čtení/zápis)

PLC program nastavením hodnoty log.1 do bitu **REQ_BACKUP_MEM** způsobí okamžité uložení paměťové oblasti `PLC_MEM_BACKUP[10000]` na disk. Po uložení zálohované oblasti na disk, systém bit `REQ_BACKUP_MEM` vynuluje.

Příklad:
Uložení oblasti PLC_MEM_BACKUP v mechanizmu s čekáním na provedení akce.

```
      FL    1,REQ_BACKUP_MEM        ;žádost o uložení na disk
      EX
      LDR   REQ_BACKUP_MEM          ;čeká na uložení
      EX1
```

Pro orientaci v zálohované paměti PLC je možno využít symbolické offsety, datové struktury nebo indexaci.

Příklady:

```
EQUI  PRVEK_A,50        ;definice symbolického offsetu

      STO   WORD.PLC_MEM_BACKUP+PRVEK_A        ;přístup s offsetem

;...
```

```
ALFA  struc             ;deklarace struktury
      PRVEK_A     DB    ?
      PRVEK_B     DW    ?
ALFA  ends

      STO   WORD.PLC_MEM_BACKUP.PRVEK_B        ;přístup podle struktury

;...

      LOD   MemIdx                             ;načte index (offset)
      MOV   BX,CX
      LOD   WORD.PLC_MEM_BACKUP[BX]            ;přístup podle indexu

;...
```

Pro zápis větších datových oblastí je možno použít instrukci „MV“ (viz „Základní instrukce PLC“).

Příklady:

```
AREA1:      DS    100

      MV    AREA1,PLC_MEM_BACKUP,100      ;kopíruje z AREA1 do zál.paměti
      MV    AREA1,PLC_MEM_BACKUP+20,100
      MV    AREA1+10,PLC_MEM_BACKUP.PRVEK_B,100
      MV    PLC_MEM_BACKUP.PRVEK_B,AREA1,100
```

18.1.2 Zálohování dat při vypínání systému

Pro zálohování dat při vypínání systému možno použít nepovinný modul „MODULE_DONE“ (viz „Struktura PLC programu“). Doporučuje se ale kromě toho provádět i průběžné zálohování pomocí bitu „REQ_BACKUP_MEM“ pro případ výpadku proudu nebo zaběhnutí programu.

Modul „MODULE_DONE“ slouží k činnosti potřebné při stopu PLC, např. k deaktivaci pohonů a k zálohování. Pokud požadujeme zálohování dat při závěrečných operacích PLC které není jednoprůchodové (například se zálohují data ze sdílené paměti), tak se systému musí dát zpráva o ukončení této činnosti pomocí instrukce **MODULE_DONE_FINISHED.** Táto instrukce je nepovinná a pokud nebude použita, tak ke stopu dojde hned po průchodu modulem MODULE_DONE. Pokud ale v daném souboru je instrukce MODULE_DONE_FINISHED použita, systém čeká s pokračováním činnosti při ukončování až do doby průchodu touto instrukcí (až potom se vypne systém). V tomto případě je vhodné v modulu MODULE_DONE aktivovat mechanizmus, který je umístěn standardně v modulu MODULE_MAIN a v něm na konci po ukončení činnosti je použita instrukce MODULE_DONE_FINISHED.

Příklad:

```
EQUI        BSTATE0,200                  ;offset pro STATE0
EQUI        BSTATE1,204                  ;offset pro STATE1
STATE0:     DS    4                      ;zálohovaná buňka STATE0
STATE1:     DS    4                      ;zálohovaná buňka STATE1

MODULE_DONE                              ;Začátek modulu ukončení
      FL          1,MECH_ZALOHA          ;Start mechanizmu zálohování
MODULE_DONE_END
```

```
MODULE_MAIN                              ;umístěno v MODULE_MAIN
      ;....
      MECH_BEGIN  MECH_ZALOHA            ;Mechanizmus zálohování
        SA_READ   0,OFFS_STATE0,4,STATE0       ;načte STATE0 z SA
        EX0
        SA_READ   0,OFFS_STATE1,4,STATE1       ;načte STATE1 z SA
        EX0
        MV        AREA1,PLC_MEM_BACKUP,100     ;kopíruje z AREA1
        LOD       STATE0
        STO       DWRD.PLC_MEM_BACKUP+BSTATE0  ;záloha STATE0
        LOD       STATE1
        STO       DWRD.PLC_MEM_BACKUP+BSTATE1  ;záloha STATE1
        MODULE_DONE_FINISHED             ;Konec čekání na zálohování
      MECH_END    MECH_ZALOHA
```

18.2 Přímé sdílení proměnných

18.2.1 Instrukce pro sdílení PLC proměnných

Systémová část software a PLC program mají možnost poskytnout své proměnné pro přímé sdílení. V některých případech se tak může značně zjednodušit přístup uživatelských obrazovek a dialogů k PLC proměnným, protože se nemusí proměnná přepisovat do sdílené paměti pomocí instrukce SA_WRITE a nemusí se definovat PLC konstanta pro orientaci ve sdílené paměti. Přímo sdílené proměnné je možno také jednoduše sledovat pomocí osciloskopu.

Systémová část software sekundárního procesoru poskytuje pro sdílení pevně stanovený seznam proměnných (viz. „Tabulka systémových sdílených proměnných“).

PLC program si může určit, které proměnné poskytne pro přímé sdílení pomocí instrukcí **SHARE_VAR** a **SHARE_BIT**.

Přímo sdílené proměnné definované pomocí SHARE_VAR a SHARE_BIT mají automaticky nastaven příznak OUTPUT a tak slouží jen pro výstup z PLC programu. Pokud požadujeme zapisovat do sdílené proměnné, například při definici „virtuálního spoje“, nebo ze vstupního HTML dialogu, musíme pro definici sdílení použít instrukci **DEF_IN**, která je popsána v návode: **„Logické vstupy a výstupy modulů systému.“**

Všechny logické a analogové vstupy a výstupy mají automaticky nastaveno přímé sdílení a tak se značně zjednoduší přístup uživatelských obrazovek a dialogů k jejím hodnotám. Na logické vstupy a výstupy není potřeba používat instrukce SHARE_VAR a SHARE_BIT.

instrukce	**SHARE_VAR**

```
funkce      SHARE_VAR   sdílení datové proměnné definované v PLC


syntax      SHARE_VAR   val1,  'TEXT2', type3
            SHARE_VAR   val1,  'TEXT2', type3 [, min4] [, max5]
            SHARE_VAR   poin1, 'TEXT2', type3 [, len4]
            SHARE_VAR   val1,   poin2 , type3

1.parametr  „val1, poin1“     pointer nebo název sdílené proměnné
2.parametr  „poin2,text2“     pointer nebo textový řetězec jména sdílení
3.parametr  „type3“           typ dat
4.parametr  „min4, len4“      název proměnné pro minimum (REAL), nebo délka dat
5.parametr  „max5“            název proměnné pro maximum (REAL)
```

Instrukce má návratové hodnoty:

RLO=1 ... operace dokončena bez chyb
RLO=0 ... operace dokončena, ale proměnná nebyla zařazena pro sdílení

Popis parametrů instrukce SHARE_VAR

parametr	název	význam	typ
1.	**poin1**	Ukazatel na buffer poskytnutý pro sdílení - náveští u řetězce definovaného instrukcí "str". Parametr může mít zadán offset v řetězci (+xx, +BX).	pointer
	val1	název datové proměnné poskytnuté pro sdílení Parametr může mít zadán offset v řetězci (+xx, +BX). - typ BYTE, WORD, DWRD,REAL..	data
2.	**poin2**	Ukazatel na buffer, kde je umístěný text s jménem sdílení - náveští u řetězce definovaného instrukcí "str".	pointer
	text2	Přímé zadání textu s jménem sdílení - text je zadán v apostrofech	řetězec
3.	**type3**	Zadání typu proměnné (viz dále)	typ
4.	**min4** **len4**	název proměnné pro určení minima, musí být typu REAL (nepovinné) nebo délka dat pro binární řetězec	data
5.	**max5**	název proměnné pro určení maxima, musí být typu REAL (nepovinné)	data

Tabulka typů proměnné pro sdílení (3.parametr instrukce SHARE_VAR):

Typ	popis
TYPE_INT_8	celočíselný typ se znaménkem – 8 bitů
TYPE_UNS_8, TYPE_BYTE	celočíselný typ bez znaménka – 8 bitů
TYPE_INT_16	celočíselný typ se znaménkem – 16 bitů
TYPE_UNS_16, TYPE_WORD	celočíselný typ bez znaménka – 16 bitů
TYPE_INT_32	celočíselný typ se znaménkem – 32 bitů
TYPE_UNS_32, TYPE_DWORD	celočíselný typ bez znaménka – 32 bitů
TYPE_INT_64	celočíselný typ se znaménkem – 64 bitů
TYPE_UNS_64, TYPE_QWORD	celočíselný typ bez znaménka – 64 bitů
TYPE_REAL	reálný typ
TYPE_STR	textový řetězec
TYPE_BIN	binární řetězec

Instrukce SHARE_VAR musí být umístěna nebo volána v modulu „MODULE_INIT".
Při ladění PLC programu, v okamžiku načítání nového PLC, všem sdíleným proměnným skončí platnost po okamžik nového načtení okna nebo dialogu se sdílenou proměnnou.

Příklad:

```
dwBUN1:     DS    4     ;buňka - typ DWORD
rBUN2:      DS    8     ;buňka – typ REAL
rMAX:       DS    8     ;buňka pro maximum - typ REAL
rMIN:       DS    8     ;buňka pro minimum – typ REAL
bArray:     DS    20    ;pole 20xByte
sTXShare:   STR   32,'Sdileni text'
```

```
MODULE_INIT

      LOD   CNST.0.0          ;naplní minimum a maximum
      STO   REAL.rMIN         ;jako reálné hodnoty
      LOD   CNST.200.0
      STO   REAL.rMAX

      SHARE_VAR  dwBUN1,    'SHARE_DWORD', TYPE_DWORD
      SHARE_VAR  rBUN2,     'DOBA_MAZANI', TYPE_REAL
      SHARE_VAR  dwBUN1,    'TLAK',        TYPE_DWORD, rMIN, rMAX
      SHARE_VAR  bArray+12,'PRVEK12',     TYPE_INT8

      SHARE_VAR  sTXShare, 'TEXT1', TYPE_STR
      SHARE_VAR  bArray,   'AREA2', TYPE_BIN, 10

MODULE_INIT_END
```

instrukce SHARE_BIT

funkce **SHARE_BIT sdílení bitové proměnné definované v PLC**

syntax **SHARE_BIT val1, 'TEXT2', bit3**
SHARE_VAR val1, poin2 , bit3

1.parametr **„val1"** **název Bajtu, kde je definován sdílený bit**
2.parametr **„poin2,text2"** **pointer nebo textový řetězec jména sdílení**
3.parametr **„bit3"** **jméno sdíleného bitu**

Instrukce má návratové hodnoty:

RLO=1 ... operace dokončena bez chyb
RLO=0 ... operace dokončena, ale bitová proměnná nebyla zařazena pro sdílení

parametr	název	význam	typ
1.	**val1**	Název Bajtové proměnné, ve které je definován sdílený bit (například návěští u instrukce „DFM"). Parametr může mít zadán offset v řetězci (+xx, +BX). - typ BYTE	data
2.	**poin2**	Ukazatel na buffer, kde je umístěný text s jménem sdílení - návěští u řetězce definovaného instrukcí "str".	pointer
	text2	Přímé zadání textu s jménem sdílení - text je zadán v apostrofech	řetězec
3.	**bit3**	Jméno bitové proměnné - definované pomocí instrukce DFM	bit

Instrukce SHARE_BIT musí být umístěna nebo volána v modulu „MODULE_INIT".
Při ladění PLC programu, v okamžiku načítání nového PLC, všem sdíleným proměnným skončí platnost po okamžik nového načtení okna nebo dialogu se sdílenou proměnnou.

První parametr je typicky název Bajtové proměnné, ve které je definován sdílený bit. V tomto případě je to návěští u instrukce „DFM". Při použití složitější adresace bitu nebo u bitových polí, to může být jiné jméno Bajtu, než je to, ve kterém je bit definován. V tomto případě se u prvního parametru může použít také offset (+xx, +BX).

Všechny logické vstupy a výstupy mají automaticky nastaveno přímé sdílení a tak na logické vstupy a výstupy není potřeba používat instrukce SHARE_VAR a SHARE_BIT.

Příklad:

```
PAM100:     DFM   ,,flPressOn,,,,,
IP0:        DFM   ,inLimXMin,,,,inLimXMax,,,

MODULE_INIT

      SHARE_BIT   PAM100,'PRESS_ON',flPressOn
      SHARE_BIT   IP0,'LIMIT_X_MAX',inLimXMax

      SHARE_BIT   CAN_DRIVE_STAT+2,'CAN_Y_READY',CAN_AX_READY
      SHARE_BIT   CAN_DRIVE_STAT+4,'CAN_Z_READY',CAN_AX_READY

MODULE_INIT_END
```

18.2.2 Sdílení systémových proměnných

Systémová část software sekundárního procesoru poskytuje pro sdílení pevně stanovený seznam proměnných. Umožní se tak přístup uživatelských obrazovek a dialogů k systémovým proměnným úplně bez účasti PLC. Přímo sdílené proměnné je možno také jednoduše sledovat pomocí osciloskopu.

Při sestavování názvu pro sdílenou systémovou proměnnou je umožněno v některých případech používat indexaci v poli a orientaci ve struktuře. Dále je uveden seznam možných typu a zápisů sdílených proměnných.

Možnosti pro zápis jména proměnných pro přímé sdílení:

Zápis	Popis
`Name`	Přímé sdílení jednoduché systémové proměnné
`Name[xx]`	a) typy: `UNS8,INT8,UNS16,INT16,UNS32,INT32,UNS64,INT64,REAL` Systémová proměnná je prvek pole. Zadává se vždy index do pole od nuly (xx=0,1,2..). b) typ `BIT` Anonymní přistup k bitu pro proměnné o velkosti 8, 16 nebo 32 bitů. Index udává váhu bitu (xx=0,1,2..).
`Name.Item`	a) typy: `UNS8,INT8,UNS16,INT16,UNS32,INT32,UNS64,INT64,REAL` Systémová proměnná je prvek struktury. b) typ: `BIT` Přístup k pojmenovanému bitu pro proměnné o velkosti 8, 16 nebo 32 bitů.
`Name[xx].Item`	typ `BIT` přístup k pojmenovanému bitu k prvku pole o velikosti 8, 16 nebo 32 bitů. Zadává se vždy index do pole od nuly (xx=0,1,2..).
`Name.Item[xx]`	a) typy: `UNS8,INT8,UNS16,INT16,UNS32,INT32,UNS64,INT64,REAL` Systémová proměnná je prvek struktury, která je pole. Zadává se vždy index do pole od nuly (xx=0,1,2..). b) typ `BIT` Anonymní přistup k bitu pro prvek struktury o velkosti 8, 16 nebo 32 bitů. Index udává váhu bitu (xx=0,1,2..).

Použité zkratky pro typy:

Typ	popis
INT8	celočíselný typ se znaménkem – 8 bitů
UNS8	celočíselný typ bez znaménka – 8 bitů
INT16	celočíselný typ se znaménkem – 16 bitů
UNS16	celočíselný typ bez znaménka – 16 bitů
INT32	celočíselný typ se znaménkem – 32 bitů
UNS32	celočíselný typ bez znaménka – 32 bitů
INT64	celočíselný typ se znaménkem – 64 bitů
UNS64	celočíselný typ bez znaménka – 64 bitů
REAL	reálný typ
BIT	bit
STRUC	struktura

Příklady zápisu:

BlockInProgress	Blok rozpracován, typ BIT
Pos5_Ax[2]	Poloha 3.osy v POS5, typ REAL [mm]
PotControl_PotF.PT_VAL	Hodnota potenciometru %F v promile, typ UNS16
ActualBlock_Flags.SelBlock	První blok po volbě bloku, typ BIT
ActualBlock_RPlcParams[2]	Programovaná reálná hodnota RPLC2, typ REAL
InputOutput_CAN2_0.INOUT_IN[1]	Fyzické vstupy z 2.portu 1.INOUT08, typ UNS8
InputOutput_CAN2_0.INOUT_IN2[1]	2.bit na 3.portu vstupů 1.INOUT08, typ BIT
DriveStatus_CAN1[2]	Status 3.pohonu CAN-BUS, typ UNS16
DriveStatus_CAN1[2].CAN_AX_ENBLD	Stav „Enable“ 3.pohonu CAN-BUS, typ BIT
InputPort[5]	6.port logických vstupů, typ UNS8
InputPort[5].BIT2	3.bit na 6.portu logických vstupů, typ BIT

18.2.3 Tabulka systémových sdílených proměnných

(stav pro revizi 745)

Název		Typ	Popis
Aktuální blok v RTM			
ActualBlock_BlockType		UNS32	Typ aktuálního bloku (btAut/btAutIns/btCAnul/btRef)
ActualBlock_Flags.		UNS32 UNS32.BIT	Příznaky pro aktuální blok (možnost čtení po bitech)
	. FirstBlock	prvek bitové struktury	První blok programu určený pro předání modulu reálného času
	. LastBlock	prvek bitové struktury	Poslední blok programu určený pro předání modulu reálného času
	. StopBlock	prvek bitové struktury	Je v bloku programován stop (M0)?
	. CondStop	prvek bitové struktury	Je v bloku programován podmíněný stop
	. Backward	prvek bitové struktury	Jede se blok pozpátku (couvání)
	. SelBlock	prvek bitové struktury	První blok po volbě bloku
	. Cont	prvek bitové struktury	První blok po CONT?
	. TchlOff	prvek bitové struktury	Má se blok jet bez technologie
	. Partial	prvek bitové struktury	Neúplný blok
ActualBlock_H		REAL	Programovaná hodnota adresy H
ActualBlock_T		INT32	Programovaná hodnota adresy T
ActualBlock_AddrProgr.		UNS32 UNS32.BIT	Změnové bity pro aktuální blok (možnost čtení po bitech)
	. Hprog	prvek bitové struktury	Programována adresa H
	.Tprog	prvek bitové struktury	Programována adresa T
ActualBlock_M[xx]		pole UNS32 (xx = 0,1,..,14)	Hodnoty programovaných M funkce jednotlivých skupin
ActualBlock_MProgr[xx]		pole BIT (xx = 0,1,..,14)	Programované M funkce jednotlivých skupin
ActualBlock_IPlcParams[xx]		pole INT32 (xx = 0,1,..,4)	Programované celočíselné hodnoty pro PLC
ActualBlock_IPlcParamsChanged[xx]		pole BIT (xx = 0,1,..,4)	Programované celočíselné hodnoty pro PLC – změny
ActualBlock_RPlcParams[xx]		pole REAL (xx = 0,1,..,4)	Programované reálné hodnoty pro PLC
ActualBlock_RPlcParamsChanged[xx]		pole BIT (xx = 0,1,..,4)	Programované reálné hodnoty pro PLC– změny
ActualBlock_ModeProgr.		UNS32 UNS32.BIT	Příznaky bloku
	. ContinuousMode	prvek bitové struktury	Plynule navázat blok na předchozí blok
	. RevolutionFeed	prvek bitové struktury	Otáčková rychlost (G95)

	. ConstCutSpeed	prvek bitové struktury	Použít konstantní řeznou rychlost
	. TouchProbeMeas	prvek bitové struktury	Provést v bloku měření dotykovou sondou?
	. TouchProbeFallingEdge	prvek bitové struktury	Reagovat na sestupnou hranu dotykové sondy?
	. TouchProbeCont	prvek bitové struktury	Po příchodu hrany ze sondy pokračovat až do cílového bodu
ActualBlock_InterpolationType		UNS32	Typ interpolace
ActualBlock_Delay		REAL[s]	Časová prodleva programovaná v bloku
ActualBlock_AxNC[xx]		pole REAL[mm] (xx = 0,1,..,15)	Absolutní poloha NC os v POS0
ActualBlock_NCMoveProgr[xx]		pole BIT (xx = 0,1,..,15)	Příznaky pohybu pro jednotlivé NC osy
ActualBlock_AxG[xx]		pole UNS32 (xx = 0,1,2)	Definice geometrických os
ActualBlock_TFlags[xx]		pole BIT (xx = 0,1,..,4)	Příznaky změny transformací / posunutí
ActualBlock_PTMatrix[xx]		pole REAL (xx = 0,1,..,15)	Programová transformace (řazeno po sloupcích)
ActualBlock_PTInvMatrix[xx]		pole REAL (xx = 0,1,..,15)	Inverzní programová transformace
ActualBlock_WTMatrix[xx]		pole REAL (xx = 0,1,..,15)	Transformace polotovaru (řazeno po sloupcích)
ActualBlock_WTInvMatrix[xx]		pole REAL (xx = 0,1,..,15)	Inverzní transformace polotovaru
ActualBlock_LenComp[xx]		pole REAL[mm] (xx = 0,1,2)	Délkové korekce
ActualBlock_Offs1[xx]		pole REAL[mm] (xx = 0,1,..,15)	Vektory posunutí 1
ActualBlock_Offs2[xx]		pole REAL[mm] (xx = 0,1,..,15)	Vektory posunutí 2
ActualBlock_BlockLen		REAL[mm]	Délka dráhy bloku
ActualBlock_VectorC[xx]		pole REAL[mm] (xx = 0,1,2,3)	Informace pro interpolaci kružnice
ActualBlock_HalfRevCount		UNS32	Počet půlotáček pro kruhovou interpolaci
ActualBlock_Axis1		UNS32	Číslo první osy - udává interpolační rovinu
ActualBlock_Axis2		UNS32	Číslo druhé osy - udává interpolační rovinu
ActualBlock_Helix		UNS32	Vytváření šroubovice
ActualBlock_SpiralPitch		REAL[mm/ot]	Stoupání šroubovice
ActualBlock_SpiralAxis		UNS32	Číslo osy pro šroubovici
ActualBlock_ThreadRunInLen		REAL[mm]	Délka nájezdu závitu
ActualBlock_ThreadRunInAngle		REAL[rad]	Úhel nájezdu závitu
ActualBlock_ThreadRunOutLen		REAL[mm]	Délka výjezdu závitu
ActualBlock_ThreadRunOutAngle		REAL[rad]	Úhel výjezdu závitu

ActualBlock_Scale	REAL	Výsledné měřítko
ActualBlock_FeedProgr	REAL [mm/min]	Programovaná rychlost suportu pro pracovní posuv
ActualBlock_FeedMax	REAL [mm/min]	Maximální rychlost suportu
ActualBlock_FeedCriterial	REAL [mm/min]	Kriteriální rychlost suportu na začátku bloku
ActualBlock_AccelerationType	UNS32	Způsob rampování
ActualBlock_SRampJerk	REAL [m/s^3]	Jerk (Parabolické zrychlení)
ActualBlock_LinearAccel	REAL [m/s^2]	Lineární zrychlení
ActualBlock_SpindleSpeed	REAL [ot/min]	Otáčky vřetene
ActualBlock_SpindleSpeedLimit	REAL [ot/min]	Limit otáček vřetene pro konstantní řeznou rychlost
ActualBlock_ConstCuttingSpeed	REAL	Rychlost pro režim konstantní řezné rychlosti
ActualBlock_Reference_Type	UNS32	Typ reference
ActualBlock_Reference_RefAxisMask	UNS32	Maska os pro referenci
ActualBlock_FiveAxTType	UNS32	Typ pětiosé transformace (podle způsobu zadávání a interpolace)
ActualBlock_ToolVect	pole REAL (xx = 0,1,2)	Programovaná orientace nástroje na konci bloku nebo zadání úkosu
ActualBlock_ToolLen	REAL[mm]	Délka nástroje pro pětiosou transformaci
Periferie CAN-BUS		
InputOutput_CAN2_0.	STRUC INOUTS	Systémová struktura pro 1.periferii INOUT
.INOUT_PRESENT	UNS8 prvek struc INOUTS	Je INOUT přítomna ?
.INOUT_MODE	UNS8 prvek struc INOUTS	mód INOUT (0=standard, 1=analog, 2=matice)
.INOUT_VENDORID	UNS32 prvek struc INOUTS	Výrobce INOUT
.INOUT_DEVICENAME	STR prvek struc INOUTS	Název jednotky INOUT
.INOUT_HWVERSION	STR prvek struc INOUTS	HW verze INOUT
.INOUT_SWVERSION	STR prvek struc INOUTS	SW verze INOUT
.INOUT_IN[xx]	pole UNS8 (0,1,2,3) prvek struc INOUTS	Fyzické vstupy INOUT
.INOUT_IN0[xx]	pole BIT (x = 0,1,..,7) prvek struc INOUTS	Fyzické vstupy INOUT, 1.port
.INOUT_IN1[xx]	pole BIT (x = 0,1,..,7) prvek struc INOUTS	Fyzické vstupy INOUT, 2.port
.INOUT_IN2[xx]	pole BIT (x = 0,1,..,7) prvek struc INOUTS	Fyzické vstupy INOUT, 3.port
.INOUT_IN3[xx]	pole BIT (x = 0,1,..,7) prvek struc INOUTS	Fyzické vstupy INOUT, 4.port
.INOUT_OUT[xx]	pole UNS8 (0,1,2) prvek struc INOUTS	Fyzické výstupy INOUT

	.INOUT_OUT0[xx]	pole BIT (x = 0,1,..,7) prvek struc INOUTS	Fyzické výstupy INOUT, 1.port
	.INOUT_OUT1[xx]	pole BIT (x = 0,1,..,7) prvek struc INOUTS	Fyzické výstupy INOUT, 2.port
	.INOUT_OUT2[xx]	pole BIT (x = 0,1,..,7) prvek struc INOUTS	Fyzické výstupy INOUT, 3.port
	.INOUT_ANL[xx]	pole UNS16 (0,1,2,3) prvek struc INOUTS	Analogové vstupy
	.INOUT_ERROR	UNS8 prvek struc INOUTS	Aktuální chybový registr
	.INOUT_ERR_CODE	UNS8 prvek struc INOUTS	Kód chyby
	.INOUT_ERR_STAT	UNS8 prvek struc INOUTS	Chybové hlášení
	.INOUT_CONTROL	UNS8 prvek struc INOUTS	Řízení INOUT
	.INOUT_COUNT	UNS16 prvek struc INOUTS	Čítač pro time-out
	.INOUT_SEND_REQ	UNS8 prvek struc INOUTS	Žádost o vyslání SDO paketu
	.INOUT_RESP_COUNT	UNS8 prvek struc INOUTS	Čítač příjmu
	.INOUT_TL[xx]	UNS8 (x=0,1,2,3) prvek struc INOUTS	Kódy tlačítek (4x)
	.INOUT_TL0[xx]	pole BIT (x = 0,1,..,7) prvek struc INOUTS	Tlačítka po bitech
	.INOUT_TL1[xx]	pole BIT (x = 0,1,..,7) prvek struc INOUTS	Tlačítka po bitech
	.INOUT_RPDO_COUNT	UNS16 prvek struc INOUTS	Čítač zařazených RPDO paketů do fronty na vyslání
	.INOUT_TPDO_COUNT	UNS16 prvek struc INOUTS	Čítač přijmutích TPDO paketů
	.INOUT_SHORT_OUT[xx]	pole UNS8 (0,1,2) prvek struc INOUTS	Informace o zkratu výstupů
	.INOUT_SHORT_OUT0[xx]	pole BIT (x = 0,1,..,7) prvek struc INOUTS	Informace o zkratu výstupů, 1.port
	.INOUT_SHORT_OUT1[xx]	pole BIT (x = 0,1,..,7) prvek struc INOUTS	Informace o zkratu výstupů, 2.port
	.INOUT_SHORT_OUT2[xx]	pole BIT (x = 0,1,..,7) prvek struc INOUTS	Informace o zkratu výstupů, 3.port
	.INOUT_IRC[xx]	UNS16 (x = 0,1) prvek struc INOUTS	Encodery (2x)
InputOutput_CAN2_1.		STRUC INOUTS	Systémová struktura pro 2.periferii INOUT
InputOutput_CAN2_2.		STRUC INOUTS	Systémová struktura pro 3.periferii INOUT
InputOutput_CAN2_3.		STRUC INOUTS	Systémová struktura pro 4.periferii INOUT
InputOutput_CAN2_4.		STRUC INOUTS	Systémová struktura pro 5.periferii INOUT
InputOutput_CAN2_5.		STRUC INOUTS	Systémová struktura pro 6.periferii INOUT
InputOutput_CAN2_6.		STRUC INOUTS	Systémová struktura pro 7.periferii INOUT
InputOutput_CAN2_7.		STRUC INOUTS	Systémová struktura pro 8.periferii INOUT

Panel_CAN2_0.	STRUC INOUTS	Systémová struktura pro 1.ovládací panel
Panel_CAN2_1.	STRUC INOUTS	Systémová struktura pro 2.ovládací panel
Panel_CAN2_2.	STRUC INOUTS	Systémová struktura pro 3.ovládací panel
Panel_CAN2_3.	STRUC INOUTS	Systémová struktura pro 4.ovládací panel
Knob_CAN2_0.	STRUC INOUTS	Systémová struktura pro 1. panel s ručním točítkem
Knob_CAN2_1.	STRUC INOUTS	Systémová struktura pro 2. panel s ručním točítkem
HandPanel_CAN2_0.	STRUC INOUTS	Systémová struktura pro 1. ruční panel
HandPanel_CAN2_1.	STRUC INOUTS	Systémová struktura pro 2. ruční panel
CountErrOverrun_CAN1	UNS32	Čítač chyb příjmu s přepsáním pro CAN1
CountErrOverrun_CAN2	UNS32	Čítač chyb příjmu s přepsáním pro CAN2
CountErrFull_CAN1	UNS32	Čítač chyb plného bufferu pro vysláni v controleru pro CAN1
CountErrFull_CAN2	UNS32	Čítač chyb plného bufferu pro vysláni v controleru pro CAN2
CountErrInt_CAN1	UNS32	Čítač chyb - za periodu SYNC se vše neodvysílalo pro CAN1
CountErrInt_CAN2	UNS32	Čítač chyb - za periodu SYNC se vše neodvysílalo pro CAN2
Keyb1_PLCCommand[xx]	pole UNS32 (xx = 0,1,..255)	Příkaz po stisku tlačítka pro PLC 1. panel
Keyb1_RTMCommand[xx]	pole UNS32 (xx = 0,1,..255)	Příkaz po stisku tlačítka pro RTM 1. panel
Keyb2_PLCCommand[xx]	pole UNS32 (xx = 0,1,..255)	Příkaz po stisku tlačítka pro PLC 2. panel
Keyb2_RTMCommand[xx]	pole UNS32 (xx = 0,1,..255)	Příkaz po stisku tlačítka pro RTM 2. panel
Keyb3_PLCCommand[xx]	pole UNS32 (xx = 0,1,..255)	Příkaz po stisku tlačítka pro PLC 3. panel
Keyb3_RTMCommand[xx]	pole UNS32 (xx = 0,1,..255)	Příkaz po stisku tlačítka pro RTM 3. panel
Keyb4_PLCCommand[xx]	pole UNS32 (xx = 0,1,..255)	Příkaz po stisku tlačítka pro PLC 4. panel
Keyb4_RTMCommand[xx]	pole UNS32 (xx = 0,1,..255)	Příkaz po stisku tlačítka pro RTM 4. panel
Knob1_PLCCommand[xx]	pole UNS32 (xx = 0,1,..255)	Příkaz po stisku tlačítka pro PLC 1. panel s ručním točítkem
Knob1_RTMCommand[xx]	pole UNS32 (xx = 0,1,..255)	Příkaz po stisku tlačítka pro RTM 1. panel s ručním točítkem
HandPanel1_PLCCommand[xx]	pole UNS32 (xx = 0,1,..255)	Příkaz po stisku tlačítka pro PLC 1. ruční panel
HandPanel1_RTMCommand[xx]	pole UNS32 (xx = 0,1,..255)	Příkaz po stisku tlačítka pro RTM 1. ruční panel
HandPanel2_PLCCommand[xx]	pole UNS32 (xx = 0,1,..255)	Příkaz po stisku tlačítka pro PLC 2. ruční panel
HandPanel2_RTMCommand[xx]	pole UNS32 (xx = 0,1,..255)	Příkaz po stisku tlačítka pro RTM 2. ruční panel

NumInputOutput_CAN2	UNS8	Počet periferií INOUT na CAN2
NumInputOutput_CAN3	UNS8	Počet periferií INOUT na CAN3
NumInputOutput_CAN4	UNS8	Počet periferií INOUT na CAN4
NumPanel_CAN2	UNS8	Počet ovládacích panelů na CAN2
NumKnob_CAN2	UNS8	Počet panelů s ručním točítkem na CAN2
NumHandPanel_CAN2	UNS8	Počet ručních panelů na CAN2
CanTPDOLostPanel	UNS8	Sledování ztráty paketu pro panel
CanTPDOLostInout[xx]	pole UNS8 (xx = 0,1,..,31)	Sledování ztráty paketu pro jednotky INOUT
InputPort[xx]	pole UNS8 (0,1,..,19) UNS8[xx].BIT	Logické vstupy
OutputPort[xx]	pole UNS8 (0,1,..,14) UNS8[xx].BIT	Logické výstupy
BIT0,BIT1,..,BIT15	prvek bitové struktury	Anonymní přístup k bitovým prvkům
Pohony CAN-BUS		
DriveStatus_CAN1[xx].	pole UNS16 UNS16[xx].BIT	Drive status z pohonu CAN-BUS (CAN_DRIVE_STAT)
.CAN_AX_READY	prvek bitové struktury	Pohon vpořádku
.CAN_AX_ON	prvek bitové struktury	Pohon má aktivní výkonový můstek
.CAN_AX_ENBLD	prvek bitové struktury	Pohon je ve stavu Enable
.CAN_AX_FAULT	prvek bitové struktury	Pohon v poruše
.CAN_AX_VOLTAGE	prvek bitové struktury	Napětí pohonu
.CAN_AX_QSTOP	prvek bitové struktury	Rychlý stop
.CAN_AX_BRKD	prvek bitové struktury	Brzda pohonu
.CAN_AX_WARN	prvek bitové struktury	Upozornění
ManufactoryStatus_CAN1[xx].	pole UNS16 UNS16[xx].BIT	Manufactory drive status (CAN_MDRIVE_STAT)
DriveCommand_CAN1[xx].	pole UNS16 UNS16[xx].BIT	Drive command pro 1.pohon (CAN_DRIVE_CMD)
.CAN_AX_EN	prvek bitové struktury	Příkaz „Enable“
.CAN_EX_BRK	prvek bitové struktury	Příkaz „Brake“
Jednotka odměřování a analog.výstupů SU05, SU06		
SU5_Counter[xx]	pole UNS16 (xx = 0,1,..15)	Čítač odměřování SU05 (SU_IRC0)
SU5_RefBuff[xx]	pole UNS16 (xx = 0,1,..15)	Buffer reference SU05 (SU_REF0)
SU5_Uout[xx]	pole REAL[10V] (xx = 0,1,..15)	Výstupní napětí pro analogové výstupy SU05

SU5_Status[xx].	pole UNS8 UNS8[xx].BIT	Status jednotky SU05
. SU_ErrIrc	prvek bitové struktury	Zkrat nebo přerušení snímače IRC
.SU_ErrAnl	prvek bitové struktury	Špatná funkce analogových výstupů
.SU_ErrWrite	prvek bitové struktury	Nesprávný počet zápisů
.SU_ErrBoard	prvek bitové struktury	Chyba v připojení na sběrnici
.SU_ErrHver	prvek bitové struktury	Chyba verze hardware jednotky SU05
. SU_ErrSU67	prvek bitové struktury	Chybí subdeska SU67 u jednotky SU06
SU5_Error[xx].	pole UNS8 UNS8[xx].BIT	Chyby jednotky SU05
.SU_ErrPhase	prvek bitové struktury	Chyba fáze signálů snímače IRC
.SU_ErrCheck	prvek bitové struktury	Chyba kontrolního čítače IRC
EncoderChannelType[xx]	pole UNS16 (xx = 0,1,..15)	Konfigurace „ChannelType“ v elementu „EncoderChannel“
EncoderType[xx]	pole UNS16 (xx = 0,1,..15)	Konfigurace „EncoderType“ v elementu „EncoderChannel“
DriveType[xx]	pole UNS16 (xx = 0,1,..15)	Konfigurace „ DriveType“ v elementu „DriveChannel“
SU5_NIPulse[xx].	pole UNS32 UNS32[xx].BIT	Pole pro sledování nulocých pulsů
. SU_NI	prvek bitové struktury	Nulový impuls
SU5_UoutPlc[xx]	pole REAL[10V] (xx = 0,1,..15)	Výstupní napětí pro analogové výstupy z PLC (SU_OUT_PLC)
SU5_CheckCounter[xx]	pole UNS32 (xx = 0,1,..15)	Čítač odměřování pro kontrolní čítač (SU_CHECK_ODM)
SU5_NICounter[xx]	pole UNS32 (xx = 0,1,..15)	Čítač odměřování SU05 po zónách (SU_CHECK_NI)
SU5_NIAbsCounter[xx]	pole UNS32 (xx = 0,1,..15)	Absolutní poloha při nulovém pulsu (SU_CHECK_NI_ABS)
Potenciometry		
TabSensitivity_PotF[xx]	pole UNS16 (xx = 0,1,..999)	Tabulka průběhu potenciometru %F (promile)
TabSensitivity_PotS[xx]	pole UNS16 (xx = 0,1,..999)	Tabulka průběhu potenciometru %S (promile)
TabSensitivity_Pot1[xx]	pole UNS16 (xx = 0,1,..999)	Tabulka průběhu potenciometru P1 (promile)
TabSensitivity_Pot2[xx]	pole UNS16 (xx = 0,1,..999)	Tabulka průběhu potenciometru P2 (promile)
TabSensitivity_Pot3[xx]	pole UNS16 (xx = 0,1,..999)	Tabulka průběhu potenciometru P3 (promile)
TabSensitivity_Pot4[xx]	pole UNS16 (xx = 0,1,..999)	Tabulka průběhu potenciometru P4 (promile)
TabSensitivity_Pot5[xx]	pole UNS16 (xx = 0,1,..999)	Tabulka průběhu potenciometru P5 (promile)
TabSensitivity_Pot6[xx]	pole UNS16 (xx = 0,1,..999)	Tabulka průběhu potenciometru P6 (promile)
PotControl_PotF.	STRUC PRXWR	Systémová struktura pro řízení potenciometru %F

	.PT_VAL	UNS16 prvek struc PRXWR	Hodnota potenciometru v promile
	.PT_INIT	UNS16 prvek struc PRXWR	Řádek v tabulce po zapnutí systému
	.PT_MAX	UNS16 prvek struc PRXWR	Poslední řádek v tabulce
	.PT_IDX	UNS16 prvek struc PRXWR	Aktuální index do tabulky
	.PT_IRC	UNS16 prvek struc PRXWR	Fyzická hodnota čítače
	.PT_SETVAL	UNS16 prvek struc PRXWR	Hodnota pro přímé nastavení z PLC
	.PT_LNK	UNS8 prvek struc PRXWR	Přiřazení fyzického potenciometru
	.PT_LOCK	UNS8 prvek struc PRXWR	Řízení je umožněno jen z panelu
	.PT_PLC	UNS8 prvek struc PRXWR	Řízení potenciometru je umožněno jen z PLC
	.PT_STAT	UNS8 prvek struc PRXWR	Odkud je řízení potenciometru
	.PT_SET	UNS8 prvek struc PRXWR	Přímé nastavení hodnoty z panelu
PotControl_PotS.		STRUC PRXWR	Systémová struktura pro řízení potenciometru %S
PotControl_Pot1.		STRUC PRXWR	Systémová struktura pro řízení potenciometru P1
PotControl_Pot2.		STRUC PRXWR	Systémová struktura pro řízení potenciometru P2
PotControl_Pot3.		STRUC PRXWR	Systémová struktura pro řízení potenciometru P3
PotControl_Pot4.		STRUC PRXWR	Systémová struktura pro řízení potenciometru P4
PotControl_Pot5.		STRUC PRXWR	Systémová struktura pro řízení potenciometru P5
PotControl_Pot6.		STRUC PRXWR	Systémová struktura pro řízení potenciometru P6
PLC			
CANErrorCode		UNS8	Chybový registr pro PLC od CAN2
CANErrorNum		UNS8	Číslo jednotky v poruše na CAN2
CANTimeOutCode		UNS8	Chyba time-out pro PLC od CAN2
CANTimeOutNum		UNS8	Číslo jednotky v poruše time-out na CAN2
PLCMemBackup[xx]		pole UNS8 (xx = 0,1,..9999)	Zálohovaná oblast PLC paměti (PLC_MEM_BACKUP)
BSPProcNumber		UNS32	Číslo procesoru BSP
SECProcNumber		UNS32	Číslo procesoru SEC
MpPlc_Ax[xx]		pole BIT (xx = 0,1,..15)	Povolení pohybu z PLC pro osy
InRef_Ax[xx]		pole BIT (xx = 0,1,..15)	NC osa v referenci
KnobSel_Ax[xx]		pole UNS8 (xx = 0,1,..15)	Volba osy z točítka pro indikaci

MpMan_Ax[xx]	pole BIT (xx = 0,1,..15)	Povolení pohybu pro osy pro manuální režim (MPxMan)
Mp_Ax[xx]	pole BIT (xx = 0,1,..15)	Celkové povolení pohybu pro osy (MPx)
Move_Ax[xx]	pole BIT (xx = 0,1,..15)	Pohyb v osách (PO_OSxPI)
Direct_Ax[xx]	pole BIT (xx = 0,1,..15)	Směr pohybu pro osy (SM_POxPI)
Coupling_S[xx]	pole BIT (xx = 0,1,..15)	Vazba polohové servosmyčky (VAZBA_x)
Homing_Ax[xx]	pole BIT (xx = 0,1,..15)	Polohování osy (POLOHV_x)
StartDisable	BIT	Zákaz startu (START_DISABLE)
BlockInProgress	BIT	Blok rozpracován (PO_F)
PosReached	BIT	Programovaná poloha dosažena (PO_POL)
StopAck	BIT	Potvrzení stopu (PO_STOP)
InPos	BIT	Poloha v toleranci (INPOS)
ExtFeedOvr	BIT	Externí řízení FeedOverride (FEED_OVR)
FeedOvrVal	UNS16	Hodnota promile pro externí řízení rychlosti
FeedOvrMultiplier	UNS16	Násobící koeficient pro externí řízení rychlosti
ExtSpeedOvr	UNS16	Externí řízení procenta otáček z PLC (SPEED_OVR)
SpeedOvrVal	UNS16	Hodnota promile pro řízení otáček (SPEED_OVR_EXT)
SpeedOvrMultiplier	UNS16	Násobící koeficient pro otáčky
Delay	BIT	Prodleva G04
LimPlus_Ax[xx]	pole BIT (xx = 0,1,..15)	Limitní spínače v kladném směru (KHx0)
LimMinus_Ax[xx]	pole BIT (xx = 0,1,..15)	Limitní spínače v záporném směru (KHx1)
DragLimPlus_Ax[xx]	pole BIT (xx = 0,1,..5)	Zpomalovací limitní spínače v kladném směru (ZPx0)
DragLimMinus_Ax[xx]	pole BIT (xx = 0,1,..5)	Zpomalovací limitní spínače v záporném směru (ZPx1)
SwRefEnable_Ax[xx]	pole BIT (xx = 0,1,..5)	Povolení softwarových spínačů od reference (SLS_REFER)
HomingPoint_Ax[xx]	pole BIT (xx = 0,1,..5)	Referenční spínače (KRx)
DragHoming_Ax[xx]	pole BIT (xx = 0,1,..5)	Zpomalovací referenční spínače (ZPRx)
ModeAut	BIT	Režim AUT (AUTPI)
ModeRefer	BIT	Ržim REFER (REFPI)
ModeCanul	BIT	Režim CANUL (CAPI)
Thread	BIT	Závitování G33 (G33PI)

RapidTraverse	BIT	Rychloposuv G00 (G00PI)
M00	BIT	Dekódovaná funkce M00 1.skupina M (M00_PID)
M01	BIT	Dekódovaná funkce M01 1.skupina M (M01_PID)
M02	BIT	Dekódovaná funkce M02 1.skupina M (M02_PID)
M30	BIT	Dekódovaná funkce M02 1.skupina M (M30_PID)
M03	BIT	Dekódovaná funkce M03 2.skupina M (M03PI)
M04	BIT	Dekódovaná funkce M04 2.skupina M (M04PI)
M19	BIT	Dekódovaná funkce M19 2.skupina M (M19PI)
M41	BIT	Dekódovaná funkce M41 3.skupina M (M41PI)
M42	BIT	Dekódovaná funkce M42 3.skupina M (M42PI)
M43	BIT	Dekódovaná funkce M43 3.skupina M (M43PI)
M44	BIT	Dekódovaná funkce M44 3.skupina M (M44PI)
M07	BIT	Dekódovaná funkce M07 5.skupina M (M07PI)
M08	BIT	Dekódovaná funkce M08 5.skupina M (M08PI)
M50	BIT	Dekódovaná funkce M50 6.skupina M (M50PI)
M51	BIT	Dekódovaná funkce M51 6.skupina M (M51PI)
M10	BIT	Dekódovaná funkce M10 7.skupina M (M10PID)
M11	BIT	Dekódovaná funkce M11 7.skupina M (M11PID)
M06	BIT	Dekódovaná funkce M06 8.skupina M (M06PID)
M60	BIT	Dekódovaná funkce M60 8.skupina M (M60PI)
SF_Run	BIT	Systém v chodu, stroj jede, nebo jsou technologické funkce
SF_BlockInProgr	BIT	Blok rozpracován
SF_BlockFinished	BIT	Blok ukončen
SF_Stop	BIT	Všechny druhy stopu bloku
SF_NotInpos	BIT	Dosud nebylo dosaženo požadované polohy
SF_DelayInProgr	BIT	Časová prodleva
SF_IndicationMode	BIT	Indikační režim
SF_SimulationMode	BIT	Simulační režim
SF_HighspeedMode	BIT	Zrychlený režim

SF_CondstopMode	BIT	Podmíněny stop aktivní (pro bloky s M01)
SF_ManualMode	BIT	Manuální režim
SF_BlockByBlock	BIT	Režim blok po bloku
SF_LS	BIT	Limitní spínač
SF_SLS	BIT	Softwarový limitní spínač
SF_Backward	BIT	Couvání
SF_StopPosChanged	BIT	Bude nastaven, když při stopu programu bylo ručně popojeto
RtmConfig[xx]	pole UNS32 (xx = 0,1,..799)	RTM Konfigurace (BUKON00)
RtmBeginMem[xx]	pole UNS8	Operační paměť pro sekundární procesor
RtmBeginCom[xx]	pole UNS8	Komunikační paměť pro sekundární procesor
BlockToggle	BIT	Klopní bit pro nový blok
Ruční pohyby		
AckMan	BIT	Pomocné ruční pojezdy jsou aktivní (ACK_AUTMAN)
InposStop	BIT	Systém je v poloze posledního stopu (INPOS_STOP)
ManEnable	BIT	Povolení ručních pojezdů (EN_AUTMAN)
ManControlNC	BIT	Řízení MAN je z panelu NC systému (AUTMAN_CONT_NC)
ManControlKnob	BIT	Řízení MAN je z panýlku točítka (AUTMAN_CONT_TOC)
ManControlSelect	BIT	Předvolba pohybu MAN (AUTMAN_CONT_SELECT)
ManMovePlus_Ax[xx]	pole BIT (x = 0,1,..,5)	Pohyb v režimu MAN v kladném směru (MM_x0)
ManMoveMinus_Ax[xx]	pole BIT (x = 0,1,..,5)	Pohyb v režimu MAN v záporném směru (MM_x1)
ManControlRT	BIT	Řízení MAN rychloposuvem (AUTMAN_CONT_G00)
Shift_Ax[xx]	pole BIT (x = 0,1,..,5)	Pohyb v režimu SHIFT (SHIFT_CONTROL)
ExtManMovePlus_Ax[xx]	pole BIT (x = 0,1,..,5)	Externí pohyb v režimu MAN v kladném směru (EXM_x0)
ExtManMoveMinus_Ax[xx]	pole BIT (x = 0,1,..,5)	Externí pohyb v režimu MAN v záporném směru (EXM_x0)
ReqExtManRT	BIT	Externí požadavek pro G00 MAN (REQ_EXT_G00_AUTMAN)
ReqExtManSelect	BIT	Externí požadavek pro předvolbu REQ_EXT_SELECT_AUTMAN
ReqExtMan	BIT	Externí požadavek na pohyb (REQ_EXT_CONT_AUTMAN)
ReqExtFeedMan	BIT	Externí požadavek na rychlost (REQ_EXT_FEED_AUTMAN)
ExtFeedMan	REAL[mm/min]	Hodnota externí rychlosti v MAN (EXT_FEED_AUTMAN)

ExtFeedRTMan	REAL[mm/min]	Hodnota externí rychlosti v MAN (EXT_FEED_G00_AUTMAN)
ReqG95Man	BIT	Požadavek na otáčkový posuv pro MAN (G95_AUTMAN)
Knob1Disable	BIT	Zákaz ovládání z 1. panýlku (AUTMAN_KNOB1_DIS)
Knob2Disable	BIT	Zákaz ovládání z 2. panýlku (AUTMAN_KNOB2_DIS)
IncManPos1_Ax	pole REAL[mm/min] (x = 0,1,..,15)	Aktuální přírůstek dráhy MAN v POS1
IncManPos2_Ax	pole REAL[mm/min] (x = 0,1,..,15)	Aktuální přírůstek dráhy MAN v POS2
IncManPos3_Ax	pole REAL[mm/min] (x = 0,1,..,15)	Aktuální přírůstek dráhy MAN v POS3
IncManPos4_Ax	pole REAL[mm/min] (x = 0,1,..,15)	Aktuální přírůstek dráhy MAN v POS4
IncManPos5_Ax	pole REAL[mm/min] (x = 0,1,..,15)	Aktuální přírůstek dráhy MAN v POS5
Řízení os a interpolace		
ReqShift	BIT	Požadavek na režim SHIFT
ActShift	BIT	Režim SHIFT aktivní
KnobStep	UNS16	Krok točítka
ShiftEnable_Ax[xx]	pole BIT (x = 0,1,..,15)	Povolení SHIFT z PLC
Pos5Inc	REAL[mm/T]	Prostorový aktuální přírůstek dráhy v GEO osách v POS5
Pos6Inc	REAL[mm/T]	Prostorový aktuální přírůstek dráhy v GEO osách v POS6
Pos5IncPm	REAL[mm/min]	Prostorový aktuální přírůstek dráhy v GEO osách v POS5
Pos6IncPm	REAL[mm/min]	Prostorový aktuální přírůstek dráhy v GEO osách v POS6
Angle5Ax1	REAL[rad]	1.fyzická osa rotace pro naklápění
Angle5Ax2	REAL[rad]	2.fyzická osa rotace pro naklápění
Pos0_Ax[xx]	pole REAL[mm] (x = 0,1,..,15)	Poloha osy v POS0
Pos1_Ax[xx]	pole REAL[mm] (x = 0,1,..,15)	Poloha osy v POS1
Pos2_Ax[xx]	pole REAL[mm] (x = 0,1,..,15)	Poloha osy v POS2
Pos3_Ax[xx]	pole REAL[mm] (x = 0,1,..,15)	Poloha osy v POS3
Pos4_Ax[xx]	pole REAL[mm] (x = 0,1,..,15)	Poloha osy v POS4
Pos5_Ax[xx]	pole REAL[mm] (x = 0,1,..,15)	Poloha osy v POS5
Pos6_Ax[xx]	pole REAL[mm] (x = 0,1,..,15)	Poloha osy v POS6
ToolVectorX	REAL	Jednotkový prostorový vektor orientace nástroje - složka X

ToolVectorY	REAL	Jednotkový prostorový vektor orientace nástroje - složka Y
ToolVectorZ	REAL	Jednotkový prostorový vektor orientace nástroje - složka Z
Ax1BevelAngle	REAL[rad]	Fyzický úhel 1.rotace při úkosové interpolaci (mezivýsledek)
Ax2BevelAngle	REAL[rad]	Fyzický úhel 2.rotace při úkosové interpolaci (mezivýsledek)
ToolTangentialAngleXY	REAL[rad]	Průmět aktuálního tečnového úhlu úkosu TollTangentialAngle
TangentialAngleAct	REAL[rad]	Aktuální tečnový úhel bloku
ToolBevelAngleAct	REAL[rad]	Aktuální úhel úkosu pro úkosovou interpolaci
ToolTangentialAngleAct	REAL[rad]	Aktuální tečnový úhel úkosu v úkosové rovině
IncPt_Ax[xx]	pole REAL[mm/T] (x = 0,1,..,15)	Aktuální přírůstek dráhy v ose
Inc_Ax[xx]	pole REAL[mm/min] (x = 0,1,..,15)	Aktuální přírůstek dráhy v ose
Pos6Shift_Ax[xx]	pole REAL[mm/min] (x = 0,1,..,15)	Posun SHIFT
ICounter[xx]	pole REAL[mm] (xx = 0,1,..,15)	Diferenční čítač polohové servosmyčky
ICounter_S[xx]	pole INT32[µm] (xx = 0,1,..,15)	Diferenční čítač polohové servosmyčky
ServoPosition[xx]	pole REAL[mm] (xx = 0,1,..,15)	Poloha servosmyček podle odměřování
ServoPosition_S[xx]	pole INT32[µm] (xx = 0,1,..,15)	Poloha servosmyček podle odměřování
ServoPosition_Inc[xx]	pole INT32[inc] (xx = 0,1,..,15)	Poloha servosmyček v inkrementech pohonu
ServoReqPosition[xx]	pole REAL[mm] (xx = 0,1,..,15)	Požadovaná poloha servosmyček
ServoReqPosition_S[xx]	pole INT32[µm] (xx = 0,1,..,15)	Požadovaná poloha servosmyček
CircleAngle	REAL[rad]	Aktuální úhel kruhové interpolace
CircleDist	REAL[mm]	Distance pro kruhovou interpolaci
Distanc_Ax[xx]	pole REAL[mm] (x = 0,1,..,15)	Složkové distance pro jednotlivé osy
Distanc	REAL[mm]	Prostorová distance os
TouchPos0_Ax[xx]	pole REAL[mm] (x = 0,1,..,15)	Poloha osy v POS0 naměřena dotykovou sondou
TouchPos1_Ax[xx]	pole REAL[mm] (x = 0,1,..,15)	Poloha osy v POS1 naměřena dotykovou sondou
TouchPos2_Ax[xx]	pole REAL[mm] (x = 0,1,..,15)	Poloha osy v POS2 naměřena dotykovou sondou
TouchPos3_Ax[xx]	pole REAL[mm] (x = 0,1,..,15)	Poloha osy v POS3 naměřena dotykovou sondou
TouchPos4_Ax[xx]	pole REAL[mm] (x = 0,1,..,15)	Poloha osy v POS4 naměřena dotykovou sondou
TouchPos5_Ax[xx]	pole REAL[mm] (x = 0,1,..,15)	Poloha osy v POS5 naměřena dotykovou sondou
TouchPos6_Ax[xx]	pole REAL[mm] (x = 0,1,..,15)	Poloha osy v POS6 naměřena dotykovou sondou

TouchProbe0Stop	BIT	Stop pohybu od dotykové sondy
TouchProbe0	BIT	Kontakt dotykové sondy
IncCan_Ax[xx]	pole INT32[inc] (xx = 0,1,..,15)	Přírůstky dráhy pro CAN-BUS a EtherCAT trajectory
RTM.Servo[xx].Edrv.State	pole UNS8 (xx = 0,1,..,15)	Stav EtherCAT pohonu EDRV (EDRV_INIT, EDRV_DONE,..)
RTM.Servo[xx].Edrv.MoveDisReq	pole BIT (xx = 0,1,..,15)	Žádost o blokování pohybu od pohonů EDRV
RTM.Servo[xx].Edrv.Error	pole BIT (xx = 0,1,..,15)	Chyba EtherCAT pohonu EDRV
RTM.Servo[xx].Edrv.Simul	pole BIT (xx = 0,1,..,15)	EDRV pohon v simulaci
RTM.Servo[xx].Edrv.Init	pole BIT (xx = 0,1,..,15)	Žádost o INIT EtherCAT pohonu typu EDRV
RTM.Servo[xx].Edrv.Done	pole BIT (xx = 0,1,..,15)	Žádost o DONE EtherCAT pohonu typu EDRV
RTM.Servo[xx].Edrv.Enable	pole BIT (xx = 0,1,..,15)	Žádost o ENABLE EtherCAT pohonu typu EDRV
RTM.Servo[xx].Edrv.Disable	pole BIT (xx = 0,1,..,15)	Žádost o DISABLE EtherCAT pohonu typu EDRV
RTM.Servo[xx].Edrv.On	pole BIT (xx = 0,1,..,15)	Žádost o ON (zapnutí) EtherCAT pohonu typu EDRV
RTM.Servo[xx].Edrv.Erase	pole BIT (xx = 0,1,..,15)	Žádost o ERASE EtherCAT pohonu EDRV
RTM.Servo[xx].Edrv.DriveStatus	pole UNS16 (xx = 0,1,..,15)	[in] „Drive status word“ EDRV pohonu (CoE, SoE)
RTM.Servo[xx].Edrv.DrivePos	pole INT32[inc] (xx = 0,1,..,15)	[in] „Position feedback“ EDRV pohonu
RTM.Servo[xx].Edrv.DriveErrReg	pole UNS16 (xx = 0,1,..,15)	[in] „Drive error registr“ EDRV pohonu
RTM.Servo[xx].Edrv.DriveControl	pole UNS16 (xx = 0,1,..,15)	[out] „Master control word“ EDRV pohonu
RTM.Servo[xx].Edrv.DriveVeloReq	pole INT32[inc] (xx = 0,1,..,15)	[out] „Velocity demand value“ EDRV pohonu
RTM.Servo[xx].Edrv.DrivePosReq	pole INT32[inc] (xx = 0,1,..,15)	[out] „Position demand value“ EDRV pohonu
RTM.Servo[xx].Edrv.ScaleSpeed	pole REAL (x = 0,1,..,15)	[in] Měřítko pro výstup rychlosti EDRV pohonu
RTM.Servo[xx].Enc.CounterValue	pole INT32[inc] (xx = 0,1,..,15)	[in] „Encoder counter value“ (EL5101 EDRV)
RTM.Servo[xx].Enc.LatchValue	pole INT32[inc] (xx = 0,1,..,15)	[in] „Encoder latch value“ (EL5101 EDRV)
RTM.Servo[xx].Enc.LatchEn	pole BIT (xx = 0,1,..,15)	[out] „Encoder Enable latch“ (EL5101 EDRV)
RTM.Servo[xx].Enc.LatchValid	pole BIT (xx = 0,1,..,15)	[in] „Encoder Latch C valid“ (EL5101 EDRV)
Vřeteno		
Spindle1Output	REAL[10V]	Výstupní napětí pro 1.vřeteno (bez procenta S)
Spindle2Output	REAL[10V]	Výstupní napětí pro 2.vřeteno (bez procenta S)
Spindle1PrsOutput	REAL[10V]	Výstupní napětí pro 2. vřeteno (s ohledem na procento S)

ReqSpeedPm	REAL[ot/min]	Požadované otáčky vřetene. (s ohledem na procento S)
ReqSpeedPt	REAL[ot/T]	Požadované otáčky vřetene. (s ohledem na procento S)
ActSpeedPm	REAL[ot/min]	Aktuální okamžité otáčky vřetene
ActSpeed2Pm	REAL[ot/min]	Aktuální okamžité otáčky 2.vřetene
ActAvgSpeedPm	REAL[ot/min]	Aktuální průměrované otáčky vřetene
ActAvgSpeed2Pm	REAL[ot/min]	Aktuální průměrované 2.otáčky 2.vřetene [ot/min]
ActSpeedPt	REAL[ot/T]	Aktuální okamžité otáčky vřetene
ActSpeed2Pt	REAL[ot/T]	Aktuální okamžité otáčky 2.vřetene
ActAvgSpeedPt	REAL[ot/T]	Aktuální průměrované otáčky vřetene
ActAvgSpeed2Pt	REAL[ot/T]	Aktuální průměrované otáčky 2.vřetene
SpindleGearAct1	UNS16	Číslo aktuálního převodového stupně pro 1.vřeteno
SpindleGearAct2	UNS16	Číslo aktuálního převodového stupně pro 2.vřeteno
Nelineární korekce		
NoLinComp0_0.	STRUC NOLINCOMPS	Struktura pro 1.nelineární korekci pro servosmyčku 0
. NLC_CONTROLLING	UNS8 prvek NOLINCOMPS	Řídící souřadnice
. NLC_ENABLE	UNS8 prvek NOLINCOMPS	Povolení nelineární korekce
. NLC_STEP	REAL[mm] prvek NOLINCOMPS	Krok
. NLC_BEGIN_TAB	REAL[mm] prvek NOLINCOMPS	Počátek tabulky
. NLC_LENGTH_TAB	REAL[mm] prvek NOLINCOMPS	Délka korigované dráhy
NoLinComp0_1.	STRUC NOLINCOMPS	Struktura pro 1.nelineární korekci pro servosmyčku 1
NoLinComp0_2.	STRUC NOLINCOMPS	Struktura pro 1.nelineární korekci pro servosmyčku 2
NoLinComp0_3.	STRUC NOLINCOMPS	Struktura pro 1.nelineární korekci pro servosmyčku 3
NoLinComp0_4.	STRUC NOLINCOMPS	Struktura pro 1.nelineární korekci pro servosmyčku 4
NoLinComp0_5.	STRUC NOLINCOMPS	Struktura pro 1.nelineární korekci pro servosmyčku 5
NoLinComp0_6.	STRUC NOLINCOMPS	Struktura pro 1.nelineární korekci pro servosmyčku 6
NoLinComp0_7.	STRUC NOLINCOMPS	Struktura pro 1.nelineární korekci pro servosmyčku 7
NoLinComp0_8.	STRUC NOLINCOMPS	Struktura pro 1.nelineární korekci pro servosmyčku 8
NoLinComp0_9.	STRUC NOLINCOMPS	Struktura pro 1.nelineární korekci pro servosmyčku 9
NoLinComp0_10.	STRUC NOLINCOMPS	Struktura pro 1.nelineární korekci pro servosmyčku 10

NoLinComp0_11.	STRUC NOLINCOMPS	Struktura pro 1.nelineární korekci pro servosmyčku 11
NoLinComp0_12.	STRUC NOLINCOMPS	Struktura pro 1.nelineární korekci pro servosmyčku 12
NoLinComp0_13.	STRUC NOLINCOMPS	Struktura pro 1.nelineární korekci pro servosmyčku 13
NoLinComp0_14.	STRUC NOLINCOMPS	Struktura pro 1.nelineární korekci pro servosmyčku 14
NoLinComp0_15.	STRUC NOLINCOMPS	Struktura pro 1.nelineární korekci pro servosmyčku 15
NoLinComp1_0.	STRUC NOLINCOMPS	Struktura pro 2.nelineární korekci pro servosmyčku 0
NoLinComp1_1.	STRUC NOLINCOMPS	Struktura pro 2.nelineární korekci pro servosmyčku 1
NoLinComp1_2.	STRUC NOLINCOMPS	Struktura pro 2.nelineární korekci pro servosmyčku 2
NoLinComp1_3.	STRUC NOLINCOMPS	Struktura pro 2.nelineární korekci pro servosmyčku 3
NoLinComp1_4.	STRUC NOLINCOMPS	Struktura pro 2.nelineární korekci pro servosmyčku 4
NoLinComp1_5.	STRUC NOLINCOMPS	Struktura pro 2.nelineární korekci pro servosmyčku 5
NoLinComp1_6.	STRUC NOLINCOMPS	Struktura pro 2.nelineární korekci pro servosmyčku 6
NoLinComp1_7.	STRUC NOLINCOMPS	Struktura pro 2.nelineární korekci pro servosmyčku 7
NoLinComp1_8.	STRUC NOLINCOMPS	Struktura pro 2.nelineární korekci pro servosmyčku 8
NoLinComp1_9.	STRUC NOLINCOMPS	Struktura pro 2.nelineární korekci pro servosmyčku 9
NoLinComp1_10.	STRUC NOLINCOMPS	Struktura pro 2.nelineární korekci pro servosmyčku 10
NoLinComp1_11.	STRUC NOLINCOMPS	Struktura pro 2.nelineární korekci pro servosmyčku 11
NoLinComp1_12.	STRUC NOLINCOMPS	Struktura pro 2.nelineární korekci pro servosmyčku 12
NoLinComp1_13.	STRUC NOLINCOMPS	Struktura pro 2.nelineární korekci pro servosmyčku 13
NoLinComp1_14.	STRUC NOLINCOMPS	Struktura pro 2.nelineární korekci pro servosmyčku 14
NoLinComp1_15.	STRUC NOLINCOMPS	Struktura pro 2.nelineární korekci pro servosmyčku 15
NoLinComp1ActVal[xx]	pole REAL[mm] (x = 0,1,..,15)	Aktuální hodnota 1.kompenzace
NoLinComp2ActVal[xx]	pole REAL[mm] (x = 0,1,..,15)	Aktuální hodnota 2.kompenzace
NoLinComp1Active[xx]	pole BIT (x = 0,1,..,15)	1.nelineární korekce je aktivní
NoLinComp2Active[xx]	pole BIT (x = 0,1,..,15)	2.nelineární korekce je aktivní
NlcAxisEnable[xx]	pole BIT (x = 0,1,..,15)	Nelineární korekce, povolení pro řídící souřadnici
NlcServoEnable[xx]	pole BIT (x = 0,1,..,15)	Nelineární korekce, povolení pro servosmyčku od PLC

Diagnostické prvky		
FastRunCount_MIN	UNS64	Minimum z naměřeného času průběhu smyčky FAST
FastRunCount_MAX	UNS64	Maximum z naměřeného času průběhu smyčky FAST
FastPeriodCount	UNS64	Definovaná hodnota periody pro rychlou smyčku FAST
FastRunTime_MIN	REAL [s]	Minimum z naměřeného času průběhu smyčky FAST
FastRunTime_MAX	REAL [s]	Maximum z naměřeného času průběhu smyčky FAST
PLCPeriod	REAL [s]	Definovaná hodnota periody pro základní běh PLC (slow .. 20ms)
PLCPeriodFast	REAL [s]	Definovaná hodnota periody pro rychlý běh PLC (1 ms)
PLCPeriodFastest	REAL [s]	Definovaná hodnota periody pro nejrychlejší běh PLC (200 µs)
FastestModuleCount	UNS32	Diagnostický čítač průběhu MODULE_FASTEST
FastestModulePresent	BIT	Modul MODULE_FASTEST je aktivní
FastestInterrupt	REAL [s]	Definovaná hodnota periody pro nejrychlejší přerušení (200 µs)
MeasStart	BIT	Start pro dávková měření
MeasStop	BIT	Stop pro dávková měření
MeasErr	BIT	Chyba dávkového měření
MeasControl	UNS32	Řízení pro dávková měření
EtherCAT		
EcatDiagnostic.	Struc	Základní diagnostické informace pro EtherCAT
.EcatSlaveCount	UNS16	Skutečný počet Slave připojených na EtherCAT
.EcatSlaveCountReq	UNS16	Požadovaný počet Slave
.EcatSlaveCountMaxDiag	UNS16	Maximální počet Slave pro diagnostiku
.EcatSlaveCountReqDiag	UNS16	Požadovaný počet Slave pro diagnostiku
EcatMasterDiagnostic.	Struc	Diagnostické informace Mastra
.EcatMasterDiagState	UNS32	Stav Mastra (bitové pole)
.EcatValidLicense	UNS16	Licence Mastra
EcatSlaveDiagnostic.	Struc	Diagnostické informace 1.Slave
.EcatSlaveDiagState	UNS32	Stav Slave (bitové pole)
EcatSlaveDiagnostic_1.	Struc	Diagnostické informace 2.Slave

EcatSlaveDiagnostic_1.	Struc	Diagnostické informace 2.Slave
EcatSlaveDiagnostic_2.	Struc	Diagnostické informace 3.Slave
EcatSlaveDiagnostic_3.	Struc	Diagnostické informace 4.Slave
EcatSlaveDiagnostic_4.	Struc	Diagnostické informace 5.Slave
EcatMasterCountDgn.	Struc	Čítače změn stavu Mastra
.EcatMasterUpdateCount	UNS32	Diagnostic completed successfully
.EcatMasterSendRecErrCount	UNS32	Error while sending/receiving a frame
.EcatMasterParseErrCount	UNS32	Error while processing the received frame
.EcatMasterLinkDownCount	UNS32	No connection between the NIC adapter and slaves
.EcatMasterWrongConfCount	UNS32	Wrong configuration
.EcatMasterS2STimeoutCount	UNS32	Slave-to-slave timeout
.EcatMasterDefDataSetCount	UNS32	Default data was set
.EcatMasterWatchDogCount	UNS32	WatchDog Time-Out
.EcatMasterIsLockedCount	UNS32	Master Is Locked
.EcatMasterDCIntEstabCount	UNS32	Internal DC synchronisation is established
.EcatMasterDCExtEstabCount	UNS32	External DC synchronisation is established
.EcatMasterDCPropDelCount	UNS32	DC Propagation delay initialized
EcatSlaveCountDgn.	Struc	Čítače změn stavu Slave
.EcatSlaveOffLineCount[xx]	Pole UNS32 (0,1,2,3,…)	Slave doesn't respond to commands
.EcatSlaveErrStateCount[xx]	Pole UNS32 (0,1,2,3,…)	Slave's state is different as the set one
.EcatSlaveNotConfCount[xx]	Pole UNS32 (0,1,2,3,…)	Slave is not configured
.EcatSlaveWrongConfCount[xx]	Pole UNS32 (0,1,2,3,…)	Slave's configuration doesn't match the configuration
.EcatSlaveInitCmdErrCount[xx]	Pole UNS32 (0,1,2,3,…)	Error while Init Cmd
.EcatSlaveMailboxErrCount[xx]	Pole UNS32 (0,1,2,3,…)	Error while Mailbox Init Cmd
EcatStatistic.	Struc	Statistické informace pro EtherCAT
.EcatFramesPerSecond	UNS16	Frames rate (frames per second)
.EcatRxPackets	UNS32	Number of received frames
.EcatTxPackets	UNS32	Number of transmitted frames
.EcatRxBytes	UNS32	Number of received bytes

	.EcatTxBytes	UNS32	Number of transmitted bytes
	.EcatRxErrors	UNS32	Number of wrongly received frames
	.EcatTxErrors	UNS32	Number of wrongly transmitted frames
	.EcatRxDropped	UNS32	Number of dropped received frames
	.EcatTxDropped	UNS32	Number of dropped transmitted frames
	.EcatMulticast	UNS32	Number of multicast frames
	.EcatCollisions	UNS32	Number of collisions
	.EcatAvgCycleTime	UNS32	Average observed cycle time
	.EcatAvgCycleJitter	UNS32	Average observed cycle jitter
	.EcatMinCycleTime	UNS32	Minimal observed cycle time
	.EcatMaxCycleTime	UNS32	Maximal observed cycle time
	.EcatAvgSubCycleTime	UNS32	Average observed subcycle time
	.EcatAvgSubCycleJitter	UNS32	Average observed subcycle jitter
	.EcatMinSubCycleTime	UNS32	Minimal observed subcycle time
	.EcatMaxSubCycleTime	UNS32	Maximal observed subcycle time
	.EcatSendErrors	UNS32	Number of send-errors
	.EcatReceiveErrors	UNS32	Number of receive-errors
	.EcatWrongWC	UNS32	Number of received frames with wrong working counter
	.EcatParseErrors	UNS32	Number of parse errors
	.EcatCPULoad	UNS32	CPU load (percents multiplied by 10)
	.EcatBusLoad	UNS32	Bus load (Bytes per second)
	.EcatRespondTime	UNS32	Slave response time

18.2.4 Zobrazování a zápis do sdílených proměnných

Uvedeme příklad pro zobrazení:

1. systémová sdílená proměnná `ICounter[1] [REAL mm]` (zobrazí `DIFCIT_Y`)
2. PLC sdílená proměnná pro výstup `SHARE_DWORD (DWORD)`
3. PLC sdílený bit pro výstup `PRESS_ON (BIT)`

Příklad pro zápis:

1. PLC sdílená proměnná pro vstup (viz. DEF_IN) `BunDiag1 (REAL)`
2. PLC sdílená proměnná pro vstup (viz. DEF_IN) `BitIn1 (BIT)`

Části HTML souboru pro PLC diagnostiku „`PlcDiagnostics.html`". Jsou použity kaskádové styly. Na tomto místě nebudeme detailně rozepisovat umístění jednotlivých prvků.

Zobrazení hodnot se provede například v elementu `DIV` s atributem:
`CNCType="AsyncIndicatorPainter" a AIPType="PlcVar"`

Zápis hodnot se provede například v elementu `INPUT` s atributem:
`CNCType="PlcVarEdit" a Variable="Name: 'xxxx';"`

Část HTML kódu `BODY`:

```
<BODY scroll="no">
  <DIV ID="Background">

    <DIV class="WindowTitle"> <!-- nadpis -->
      Diagnostika PLC
    </DIV>

    <DIV id="Values">
      <DIV id="Diag0_Lbl" class="LabelMedium">DIFCIT_Y:</DIV>
      <DIV id="Diag0_Val" class="ValueMedium"
       CNCType="AsyncIndicatorPainter" AIPLibrary="StdPlugins"
       AIPType="RtmVar" AIPOptions="Channel: 0; Variable: 'ICounter[1]';
       NumberPrecision:3;">+000</DIV>

      <DIV id="Diag1_Lbl" class="LabelMedium">PLC_DWORD:</DIV>
      <DIV id="Diag1_Val" class="ValueMedium"
       CNCType="AsyncIndicatorPainter" AIPLibrary="StdPlugins"
       AIPType="PlcVar" AIPOptions="Channel: 0; Variable: 'SHARE_DWORD';
       NumberPrecision:0;">+000</DIV>

      <DIV id="Diag2_Lbl" class="LabelMedium">TLAK:</DIV>
      <DIV id="Diag2_Val" class="ValueMedium"
       CNCType="AsyncIndicatorPainter" AIPLibrary="StdPlugins"
       AIPType="PlcVar" AIPOptions="Channel: 0; Variable: 'PRESS_ON';
       NumberPrecision:0;">+000</DIV>

      <DIV id="Diag3_Lbl" class="LabelMedium">Zadat BunDiag1:</DIV>
      <INPUT id="Diag3_Val" class="EditMedium"  type="text"
       CNCType="PlcVarEdit" Variable="Name: 'PLC.Input.BunDiag1';">


      <DIV id="Diag4_Lbl" class="LabelMedium">Zadat BitIn1:</DIV>
      <INPUT id="Diag4_Val" class="EditMedium"  type="text"
       CNCType="PlcVarEdit" Variable="Name: 'PLC.Input.BitIn1';">
```

```
    </DIV>
  </DIV>
</BODY>
```

Vybrané prvky "AsyncIndicatorPainter" nabízené knihovnou StdPlugins pro sdílení:

AIPLibrary="StdPlugins" **CNCType="AsyncIndicatorPainter"**				
AIPType="RtmVar"			**Zobrazení systémové (RTM) sdílené proměnné**	
	AIPOptions		Vlastnosti prvku	
		Channel	0,1,..	číslo suportu (0,1,2..)
		Variable	'text'	textový řetězec se jménem sdílení
AIPType="PlcVar"			**Zobrazení sdílené proměnné PLC**	
	AIPOptions		Vlastnosti prvku	
		Channel	0,1,..	číslo suportu (0,1,2..)
		Variable	'text'	textový řetězec se jménem sdílení
AIPType="PlcVarText"			**Zobrazení sdílené textové proměnné PLC**	
	AIPOptions		Vlastnosti prvku	
		Channel	0,1,..	číslo suportu (0,1,2..)
		Variable	'text'	textový řetězec se jménem sdílení

18.3 Sdílená paměť

18.3.1 Instrukce pro práci se sdílenou pamětí

Sdílená paměť „SA" (Shared Area) je úsek paměti sdílené mezi PLC a primárním procesorem. V současné verzi jsou k dispozici 2 sdílené oblasti, každá o velikosti 32000 bajtů. Pro zabezpečení synchronizace přenosu dat je pro každou oblast definován vlastní mutex. Sdílená oblast je společná pro všechny kanály systému, kdy běží najednou více modulů reálného času (více suportové řízení).

PLC program má k dispozici pro přístup ke sdílené oblasti dvě nové instrukce **SA_READ** a **SA_WRITE**.

instrukce	**SA_READ** **SA_WRITE**

```
funkce      SA_READ     čtení dat ze sdílené oblasti PLC paměti
            SA_WRITE    zápis dat do sdílené oblasti PLC paměti


syntax      SA_READ (SA_WRITE)      SAIdx, Offset, Size, Poin
            SA_READ (SA_WRITE)      SAIdx, Offset, Size, Val


1.parametr  „SAIdx"           index oblasti SA
2.parametr  „Offset"          offset dat od začátku oblasti
3.parametr  „Size"            velkost dat
4.parametr  „Poin, Val"       pointer nebo název proměnné
```

Instrukce **SA_READ** slouží pro čtení dat ze sdílené oblasti PLC. V případě úspěchu vrací RLO=1, jinak RLO=0. Aby čtení dat proběhlo úspěšně, musí zadaná sdílená oblast existovat a všechna požadovaná data musí ležet uvnitř této oblasti (tj. „Offset" musí být větší nebo rovno nule a „Offset" + „Size" musí být menší než velikost dané oblasti). Funkce zajišťuje synchronizaci přístupu k dané paměťové oblasti, tzn. v průběhu jejího provádění nemůže do dané oblasti současně přistupovat nikdo jiný.

Instrukce **SA_WRITE** slouží pro zápis dat do sdílené oblasti PLC. V případě úspěchu vrací RLO=1, jinak RLO=0. Aby zápis dat proběhl úspěšně, musí zadaná sdílená oblast existovat a data, která se mají zapsat se musí do této oblasti vejít V případě úspěchu vrací RLO=1, jinak RLO=0. Aby zápis dat proběhl úspěšně, musí zadaná sdílená oblast existovat a všechna zapisovaná data musí ležet uvnitř této oblasti. Funkce zajišťuje synchronizaci přístupu k dané paměťové oblasti, tzn. v průběhu jejího provádění nemůže do dané oblasti současně přistupovat nikdo jiný.

Návratové hodnoty instrukcí:

Instrukce se musí používat v mechanizmech a jsou typu „EX".
Vrácené datové hodnoty z tabulky se zapisují do řetězce na který ukazuje parametr **„Poin",** nebo přímo do datové proměnné **„Val"**.
Všechny instrukce se mohou volat průchodově a mají návratové hodnoty:

```
RLO=0,      DR=0  ....... stav čekání na dokončení operace
RLO=1,      DR=0  ....... operace dokončena bez chyb
RLO=0,      DR<>0 ....... operace dokončena, ale při výkonu vznikla chyba
```

Instrukce v případě úspěchu vrací RLO nastaveno na 1. Pokud instrukce nastaví RLO na hodnotu 0 (a současně je nulový datový DR registr), je v daném okamžiku přístup do sdílené oblasti zakázán a přenos je potřeba opakovat například v příštím cyklu PLC programu.

Pokud instrukce skončí s chybou (například offset s délkou přenášených dat je větší něž délka sdílené oblasti) instrukce vrátí registr RLO nastaven na hodnotu 0 a nenulový datový DR registr.

Popis parametrů instrukcí:

parametr	název	význam	typ
1.	**SAIdx**	číslo sdílené oblasti (od nuly)	word
2.	**Offset**	Ofset dat, kde se bude zapisovat, od začátku sdílené oblasti	word
3.	**Size**	Délka dat, které se budou kopírovat	word
4.	**Poin**	Ukazatel na buffer , z kterého se zkopírují požadovaná data - náveští u řetězce definovaného instrukcí "str". Parametr může mít zadán offset v řetězci (+xx).	pointr
	Val	Název datové proměnné. - typ BYTE, WORD, DWRD,..	data

18.3.2 Orientace ve sílené paměti

Sdílenou paměť využívá pro komunikaci PLC program, systémová část, uživatelské obrazovky a pod. Offsety pro orientaci ve sdílené paměti se proto nezadávají přímo hodnotou, ale symbolicky a jsou společné pro všechny části, které sdílenou oblast potřebují. Pro tvorbu symbolických offsetů se proto používají PLC konstanty (viz „PLC konfigurace a konstanty“).

PLC konstanty pro definici offsetů ve sdílené paměti jsou zadány v souboru „`SAVars.PlcConstans`“. Všechny PLC konstanty se zadávají v XML tvaru pomocí elementu `PlcConstants`.

Příklad:

Příklad pro konstanty offsetů ve sdílené paměti:

```
<PlcConstants>

  <Constant ID="OFFS_R_FEED"  Value="SA0REAL"></Constant>  <!-- Feed -->
  <Constant ID="OFFS_I_STATE" Value="SA0INT"></Constant>   <!-- Stav -->

</PlcConstants>
```

Načtení PLC konstant pro offsety v modulu inicializace:

```
OFFS_R_FEED:      DS    2     ;jsem se načte offset pro FEED
OFFS_I_STATE:     DS    2     ;jsem se načte offset pro STATE
R_FEED:           DS    8     ;buňka pro načtení reálné hodnoty FEED
STATE:            DS    4     ;buňka pro zápis DWORD hodnoty STATE

PROC_BEGIN Inicializace
      CONST_GET   OFFS_R_FEED,‘OFFS_R_FEED‘     ;načte offset pro FEED
      LA          FL_OK
      WR          FL_OK
      CONST_GET   OFFS_I_STATE,‘OFFS_I_STATE‘   ;načte offset pro STATE
      LA          FL_OK
      WR          FL_OK

      LDR         -FL_OK
```

```
      ESET1       2           ; PLC0002: Načítání PLC konstant se nepodařilo
PROC_END Inicializace
```

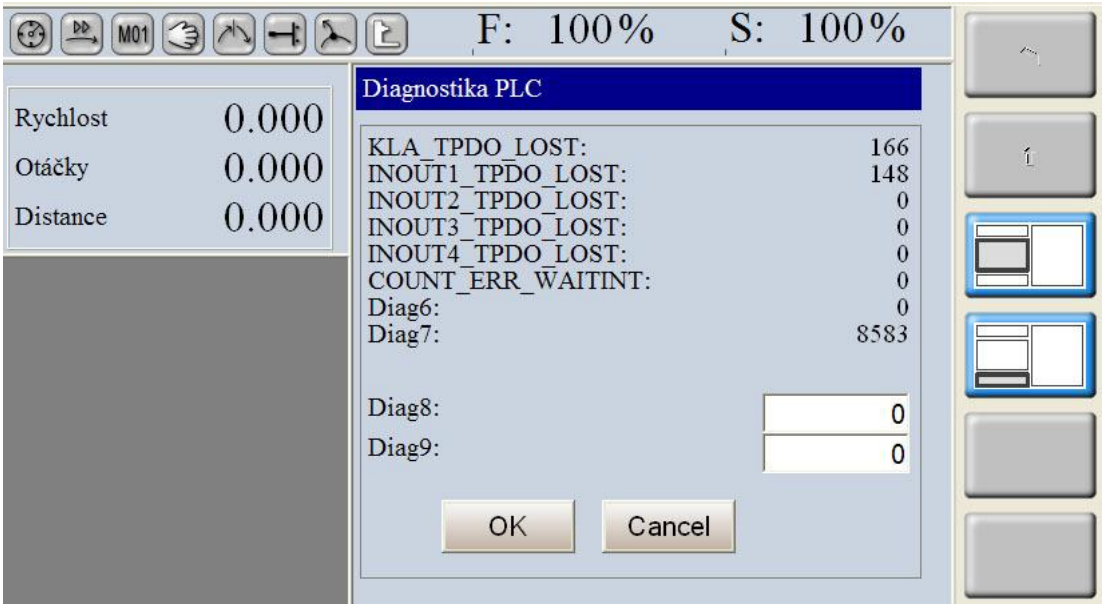
Obsluha čtení a zápisu se sdílené oblasti pomocí instrukci SA_READ a SA_WRITE.

```
MECH_BEGIN  M_SA_RW
CYK:  EX
      SA_READ     0,OFFS_R_FEED,8,R_FEED  ;načte reální hodnotu FEED
      EX0
      SA_WRITE    0,OFFS_I_STATE,4,STATE  ;zapíše DWORD hodnotu STATE
      EX0
      JUM         CYK
MECH_END    M_SA_RW
```

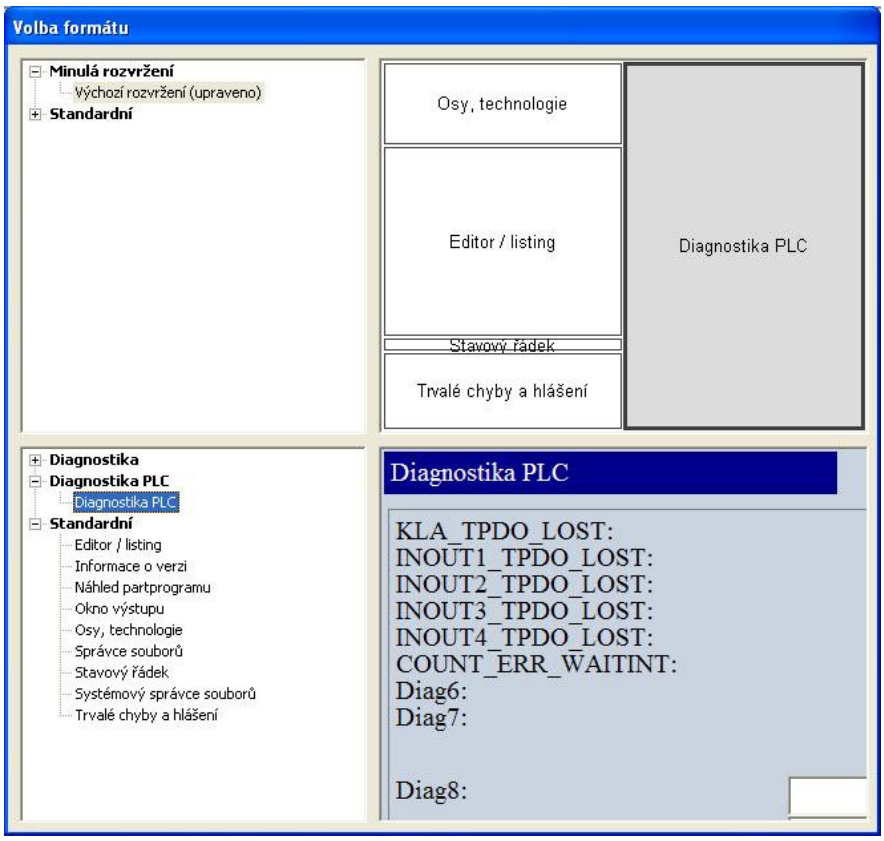
18.3.3 Využití sdílené paměti pro tvorbu okna

Uvedeme příklad použití sdílené paměti pro tvorbu diagnostického okna PLC. Okno má obsahovat osm výstupních a dvě vstupní pole. V systému bude začleněno jako normální okno, které se vyvolá prostřednictvím volby indikace.

Upozornění: Příklad slouží jen jako ukázka práce se sdílenou pamětí. V praxi se sdílená paměť na tyto účely nepoužívá, protože mnohem snadněji dosáhneme stejného výsledku využitím přímého sdílení (instrukce SHARE_VAR,…) .



Volba diagnostického okna PLC prostřednictvím „volby indikace“:



Definice dat v PLC programu. Pro zjednodušení budeme uvažovat 2 vstupní a 2 výstupní pole:

```
;Offsety pro diagnostiku
OFFS_DIAG0:             DS    2           ;PLC diagnostika
OFFS_DIAG1:             DS    2
OFFS_DIAG2:             DS    2
OFFS_DIAG3:             DS    2

;Buňky pro PLC diagnostiku
Diag0:                  DS    4           ;výstup do okna
Diag1:                  DS    4
Diag2:                  DS    4           ;vstup z okna
Diag3:                  DS    4
```

Načtení offsetů pro sdílenou paměť v inicializaci:

```
PROC_BEGIN INICIALIZACE_CNF
            FL          1,CNFOK

            CONST_GET   OFFS_DIAG0,'OFFS_DIAG0'
            LA          CNFOK
            WR          CNFOK
            CONST_GET   OFFS_DIAG1,'OFFS_DIAG1'
            LA          CNFOK
```

```
            WR          CNFOK
            CONST_GET   OFFS_DIAG2,'OFFS_DIAG2'
            LA          CNFOK
            WR          CNFOK
            CONST_GET   OFFS_DIAG3,'OFFS_DIAG3'
            LA          CNFOK
            WR          CNFOK

            LDR         -CNFOK
            ESET1       ERR_KONSTANTY     ;Chyba načtení PLC konstant
PROC_END INICIALIZACE_CNF
```

Mechanizmus pro kontinuální obsluhu PLC diagnostiky. Zapisují a čtou se data z okna diagnostiky:

```
            FL          1, M_PLCDGN       ;mechanizmus trvale běží

MECH_BEGIN  M_PLCDGN
M_CYKL:     LOD         BYTE.KLA_TPDO_LOST      ;plnění buněk výstupu
            STO         Diag0
            LOD         BYTE.INOUT_TPDO_LOST
            STO         Diag1

            SA_WRITE    0,OFFS_DIAG0,4,Diag0    ;Diagnostika - výstup
            EX0
            EX
            SA_WRITE    0,OFFS_DIAG1,4,Diag1
            EX0
            EX
            SA_READ     0,OFFS_DIAG2,4,Diag2    ;Diagnostika - vstup
            EX0
            EX
            SA_READ     0,OFFS_DIAG3,4,Diag3
            EX0
            EX
            JUM         M_SCYKL
MECH_END    M_PLCDGN
```

PLC konstanty pro definici offsetů ve sdílené paměti jsou zadány v souboru „SAVars.PlcConstans“.

```
<PlcConstants>
  <!-- Diagnostics -->
  <Constant ID="OFFS_DIAG0" Value="SA0INT"></Constant> <!-- Diagnostika -->
  <Constant ID="OFFS_DIAG1" Value="SA0INT"></Constant>
  <Constant ID="OFFS_DIAG2" Value="SA0INT"></Constant>
  <Constant ID="OFFS_DIAG3" Value="SA0INT"></Constant>
</PlcConstants>
```

Části HTML souboru pro PLC diagnostiku „PlcDiagnostics.html“. Jsou použity kaskádové styly. Na tomto místě nebudeme detailně rozepisovat umístění jednotlivých prvků.

Zobrazení hodnot se provede například v elementu DIV s atributem CNCType=“AsyncIndicatorPainter“ a vstup hodnot se provede pomocí elementu INPUT s atributem name=“PlcSAValueEdit“ (viz dále).

Část HTML kódu BODY:

```
<BODY scroll="no">
  <DIV ID="Background">

    <DIV class="WindowTitle"> <!-- nadpis -->
      Diagnostika PLC
    </DIV>

    <DIV id="Values">
      <DIV id="Diag0_Lbl" class="LabelMedium">KLA_TPDO_LOST:</DIV>
      <DIV id="Diag0_Val" class="ValueMedium"
       CNCType="AsyncIndicatorPainter" AIPLibrary="StdPlugins"
       AIPType="PlcSAValue" AIPOptions="Channel: 0; Area: 0;
       Offset: OFFS_DIAG0; DataType: UByte; NumberPrecision:0;">+000</DIV>

      <DIV id="Diag1_Lbl" class="LabelMedium">INOUT1_TPDO_LOST:</DIV>
      <DIV id="Diag1_Val" class="ValueMedium"
       CNCType="AsyncIndicatorPainter" AIPLibrary="StdPlugins"
       AIPType="PlcSAValue" AIPOptions="Channel: 0; Area: 0;
       Offset: OFFS_DIAG1; DataType: UByte; NumberPrecision:0;">+000</DIV>

      <DIV id="Diag8_Lbl" class="LabelMedium">Diag8:</DIV>
      <INPUT id="Diag8_Val" type="text" class="EditMedium"
       name="PlcSAValueEdit" Area="0" Offset="OFFS_DIAG2"DataType="UDWord">

      <DIV id="Diag9_Lbl" class="LabelMedium">Diag9:</DIV>
      <INPUT id="Diag9_Val" type="text" class="EditMedium"
       name="PlcSAValueEdit" Area="0" Offset="OFFS_DIAG3"DataType="UDWord">


      <BUTTON id="OK" class="Button">OK</BUTTON>
      <BUTTON id="Cancel" class="Button">Cancel</BUTTON>
    </DIV>
  </DIV>
</BODY>
```

Kopírování a registrace pro tvorbu SETUPu PLC v souboru „PLC.NSI“:

Část pro instalaci souborů:

```
Function InstallPlcFiles

File "/oname=Html\PlcDiagnostics.html" "..\Html\PlcDiagnostics.html"
```

Část pro odinstalování:

```
Function un.InstallPlcFiles

Delete "$INSTDIR\Html\PlcDiagnostics.html"
```

Část pro registraci dialogů:

```
Function InstallPlcConfig

WriteRegStr HKLM ~
"Software\MEFI\WinCNC\Machine\UserInterface\Dialogs\PlcDiagnostics" ~
"Library" "StdPlugins"
WriteRegStr HKLM ~
"Software\MEFI\WinCNC\Machine\UserInterface\Dialogs\PlcDiagnostics" ~
"Type" "PlcSAEditor"
WriteRegStr HKLM ~
"Software\MEFI\WinCNC\Machine\UserInterface\Dialogs\PlcDiagnostics" ~
"HtmlFile" "PlcDiagnostics.html"
```

Část pro registraci oken:

```
WriteRegStr HKLM ~
"Software\MEFI\WinCNC\Machine\UserInterface\Windows\PlcDiagnostics" ~
"Library" "StdPlugins"
WriteRegStr HKLM ~
"Software\MEFI\WinCNC\Machine\UserInterface\Windows\PlcDiagnostics" ~
"Type" "PlcSAEditor"
WriteRegStrHKLM ~
"Software\MEFI\WinCNC\Machine\UserInterface\Windows\PlcDiagnostics" ~
"HtmlFile" "PlcDiagnostics.html"
WriteRegStr HKLM ~
"Software\MEFI\WinCNC\Machine\UserInterface\Windows\PlcDiagnostics" ~
"Category" "Diagnostika PLC"
WriteRegStr HKLM ~
"Software\MEFI\WinCNC\Machine\UserInterface\Windows\PlcDiagnostics" ~
"Name" "Diagnostika PLC"
```

(znak „~“ znamená pokračování řádku – ve skutečnosti řádek nesmí být rozdělen)

18.3.4 Zobrazování dat ze sdílené paměti

Zobrazení hodnot ze sdílené paměti se provede v HTML kódu pomocí atributu `CNCType="AsyncIndicatorPainter"`. Dále uvedeme možnosti použití tohoto atributu, které souvisí se sdílenou pamětí.

Některé vybrané Prvky `"AsyncIndicatorPainter"` nabízené knihovnou `StdPlugins`.

AIPLibrary="StdPlugins" **CNCType="AsyncIndicatorPainter"**				
AIPType="PlcSAValue"			**Zobrazení hodnot různou formou**	
	AIPOptions		Vlastnosti prvku	
		Channel	0,1,..	číslo suportu (0,1,2..)
		Area	0,1,..	číslo sdílené oblasti (0,1)
		Offset	xxx	offset prvku ve sdílené paměti (v bajtech od začátku)
		DataMask	xxx	maska
		DataType	BIT	bitová proměnná
			UBYTE	celočíselná hodnota BYTE bez znaménka
			SBYTE	celočíselná hodnota BYTE se znaménkem
			UWord	celočíselná hodnota WORD bez znaménka
			SWord	celočíselná hodnota WORD se znaménkem
			UDWord	celočíselná hodnota DWORD bez znaménka
			SDWord	celočíselná hodnota DWORD se znaménkem
			Real	reálná hodnota
		GraphType	Number	číselná hodnota
			Time	formát času
			Image	obrázek
			Text	text
			HBar	horizontální indikátor
			VBar	vertikální indikátor
			Meter120,..	potenciometr 120,...
		NumberWidth	xx	
		NumberPrecision	0,1,2,..	
		ImgN	N=0,1,2,..	obrázek řazený podle hodnoty proměnné (N=0..31)
		TextN	N=0,1,2,..	text řazený podle hodnoty proměnné (N=0..31)
		Výčet dalších vlastností: `MinVal, MaxVal, NumberSign, PointerImg, FaceImg, ImgDefault, ImgWidth, ImgHeight, TextDefault`		
AIPType="PlcSAData"			**Výpis paměti (hexadecimálně)**	
	AIPOptions		Vlastnosti prvku	
		Channel	0,1,..	číslo suportu (0,1,2..)
		Area	0,1,..	číslo sdílené oblasti (0,1)
		Offset	xxx	offset prvku ve sdílené paměti (v bajtech od začátku oblasti)
		DataSize	xxx	počet
		GroupBy	BIT	bitová proměnná
			BYTE	proměnná BYTE
			WORD	proměnná WORD
			DWORD	proměnná DWORD

18.3.5 Využití sdílené paměti pro zobrazování obrázků

Uvedeme příklad použití sdílené paměti pro vykreslení obrázků na základě hodnoty v datové proměnné. Budeme řídit zobrazování vlastní signálky (LED diody) v horní části obrazovky (oblast TOP). Signálka má mít 3 stavy a bude se řídit z PLC pomocí buňky „MACHINE_STATE“:

MACHINE_STATE	stav	obrázky signálek	názvy souborů
0	not ready	 šedá	Machine0.png
1	ready	 zelená	Machine1.png
2	error	 červená	Machine2.png

Definice dat v PLC programu.

```
OFFS_MACHINE_STATE:     DS    2     ;offset pro stavovou buňku

MACHINE_STATE:          DS    4     ;stavová buňka (0, 1, 2)
```

Načtení offsetu pro sdílenou paměť v inicializaci:

```
PROC_BEGIN INICIALIZACE_CNF
            FL          1,CNFOK

            CONST_GET   OFFS_MACHINE_STATE,'OFFS_MACHINE_STATE'
            LA          CNFOK
            WR          CNFOK

            LDR         -CNFOK
            ESET1       ERR_KONSTANTY     ;Chyba načtení PLC konstant
PROC_END INICIALIZACE_CNF
```

Obsluha zápisu stavu do sdílené oblasti

```
            FL          1, M_SA_STATE     ;mechanizmus trvale běží

MECH_BEGIN  M_SA_STATE
CYK:        LOD         cnst.0            ;nastavení MACHINE_STATE
            STO         MACHINE_STATE
            LDR         MReady            ;Stroj ready ?
            LOD         cnst.1
            STO1        MACHINE_STATE
            LDR         MError            ;Error stroje ?
            LOD         cnst.2
            STO1        MACHINE_STATE
            EX
            SA_WRITE    0,OFFS_MACHINE_STATE,4,MACHINE_STATE
            EX0
            JUM         CYK
MECH_END    M_SA_STATE
```

Úprava souboru „TOP.HTML“ pro zavedení nové signálky. Soubor TOP.HTML překopírujeme z instalační části z adresáře HTML do projektu PLC. V části pro signálky přidáme řádek:

```
<TD width="36" height="32" CNCType="AsyncIndicatorPainter"
 AIPLibrary="StdPlugins" AIPType="PlcSAValue" AIPOptions="Channel: 0;
 Area: 0; Offset: OFFS_MACHINE_STATE; GraphType: Image; DataType: UDWord;
 Img0: Machine0.png; Img1: Machine1.png; Img2: Machine2.png; ">LED</TD>
```

Zavedení PLC konstanty pro offset ve sdílené paměti v souboru „SAVars.PlcConstans“.

```
<Constant ID="OFFS_MACHINE_STATE" Value="SA0INT"></Constant> <!-- stav -->
```

Upravíme SETUP pro PLC aby se soubor TOP.HTML kopíroval do adresáře „CNC Machine Files“

Část pro instalaci souborů:

```
Function InstallPlcFiles

File "/oname=Html\Top.html" "..\Html\Top.html"
```

Část pro odinstalování:

```
Function un.InstallPlcFiles

Delete "$INSTDIR\Html\Top.html"
```