

# ***PSP3***

## ***Knihovny funkčních modulů***

### **Programátorská příručka**

*Verze 3.36*



2006

---

# **Knihovny funkčních modulů**

# Knihovny funkčních modulů

Tato část manuálu popisuje standardně dodávané moduly k programu PSE a moduly specializovaných knihoven **KOTELNA** a **PRINT**. Pro každý funkční modul je uveden jeho význam, parametry a jeho použití je dokumentováno vzorovým příkladem.

V následující tabulce je uveden seznam všech modulů knihoven **PSE**, **ACTIVE**, **CAN**, **LCD**, **KOTELNA**, **PRINT** a **MATRIX**, které byly standardně dodávány v době vydání tohoto manuálu.

Modul	Popis	Knihovna
AnIn	Čtení analog. hodnoty s převodem na fyzikální veličinu	PSE
AnOut	Zápis hodnoty do AO kanálu převodem z fyzikální veličiny	PSE
AvgVar	Klouzavý průměr	PSE
BinDiff	Detekce náběžné a sestupné hrany	PSE
BinIn	Čtení binárního signálu	PSE
BinMAOut	Binární výstup s řízením ručně/automat	PSE
BinOut	Zápis jednoho binárního signálu	PSE
BKO	Bistabilní klopný obvod	PSE
Break	Přerušení cyklu For, While, Repeat, Switch	PSE
Call	Volání podprogramu	PSE
Case	Větev přepínače pro konkrétní hodnotu	PSE
ChanMode	Nastavení speciálního režimu logického kanálu	PSE
ChkCheck	Kontrola zabezpečení rámce (uživatelská komunikace)	PSE
ChkCreate	Vytvoření zabezpečení rámce (uživatelská komunikace)	PSE
CntDn	Čítač dolů	PSE
CntDnLv	Čítač dolů - hladinový	PSE
CntUp	Čítač nahoru	PSE
CntUpDn	Čítač nahoru i dolů	PSE
CntUpDnLv	Čítač nahoru i dolů - hladinový	PSE
CntUpLv	Čítač nahoru - hladinový	PSE
ColorLED	Ovládání vícebarevných LED	PSE
ComGetLSR	Načtení line status registru, zjištění chyb linky (uživatelská komunikace)	PSE
ComGetMSR	Načtení modem status registru (uživatelská komunikace)	PSE
ComInit	Hlavní modul uživatelské komunikace (uživatelská komunikace)	PSE
ComParams	Přenastavení parametrů uživatelské komunikace za běhu	PSE
ComRead	Čtení z přijímacího buferu (uživatelská komunikace)	PSE
ComREmpty	Dotaz na "prázdnost" přijímacího buferu (uživatelská komunikace)	PSE
ComSetLCR	Nastavení line control registru (uživatelská komunikace)	PSE
ComSetMCR	Nastavení modem control registru (uživatelská komunikace)	PSE
ComSetXfc	Nastavení rozhraní komunikačního kanálu	PSE
ComWEEmpty	Dotaz na "prázdnost" vysílacího buferu (uživatelská komunikace)	PSE
ComWrite	Zápis do vysílacího buferu (uživatelská komunikace)	PSE
DeltaAvg	Výpočet neklouzavého aritmetického průměru	PSE

Modul	Popis	Knihovna
DigIn	Čtení šestnáctice digitálních vstupních signálů	PSE
DigOut	Zápis šestnáctice digitálních vstupních signálů	PSE
DImpt	Čtení a přepočet impulzního vstupu	PSE
DM_MUX	Obsluha analogového multiplexeru DM-MUX3/1	PSE
EEFinish	Dokončení zápisu do EEPROM v bezpečném režimu	PSE
EEMode	Nastavení režimu práce s EEPROM	PSE
EERead	Čtení dat z EEPROM	PSE
EESize	Zjištění velikosti uživatelsky přístupné EEPROM	PSE
EEWrite	Zápis dat do EEPROM	PSE
Else	Pokračovací část podmíněného příkazu If	PSE
EndCase	Ukončení větve přepínače Case	PSE
EndFor	Ukončení cyklu For	PSE
EndIf	Ukončení podmíněného příkazu If	PSE
EndSwitch	Ukončení přepínače Switch	PSE
EndWhile	Ukončení cyklu While	PSE
EqLine	Výpočet doporučené teploty topné vody	PSE
ErrSig	Obsluha chyby v systému	PSE
Exit	Ukončení běhu podprogramu	PSE
FIFO	Fronta typu "první dovnitř - první ven" (roura)	PSE
Filtr1R	Filtr 1.řádu	PSE
FindDay	Nalezení rozsahu indexů v časové matici	PSE
For	Iterační cyklus	PSE
FreqOut	Frekvenční výstup na digitálním výstupu	PSE
GetTime	Čtení aktuálního času a vyhodnocení změn	PSE
Hyst	Test analogového údaje na mez	PSE
If	Podmíněný příkaz	PSE
ImpIn	Ošetření 16 impulzních vstupů	PSE
IncDec	Inkrementace/dekrementace proměnné	PSE
Interpol	Obecný funkční měnič	PSE
InterXY	Lineární interpolace $Z=F(X, Y)$	PSE
IRCIIn	Čtení hodnoty z hardwareového čítačového vstupu / vstupu pro inkrementální čidla polohy	PSE
IRCMMode	Nastavení režimu a konstant hardwareového čítačového vstupu / vstupu pro inkrementální čidla polohy	PSE
IRCPreset	Nastavení přednastavené hodnoty IRC-čítače	PSE
IRCSet	Nastavení hodnoty hardwareového čítačového vstupu / vstupu pro inkrementální čidla polohy	PSE
Let	Aritmetický, relační nebo logický výraz	PSE
Limiter	Omezení proměnné na zadaných mezích	PSE
Limits	Testování, zda je proměnná v daných mezích	PSE
MemCheck	Výpočet kontrolního součtu proměnné nebo paměti FLASH	PSE
MKO	Monostabilní klopný obvod	PSE
MuxAnIn	Čtení hodnoty z AI kanálu multiplexeru s převodem na fyzikální veličinu	PSE

Modul	Popis	Knihovna
MuxNi1000	Čtení teploty z odporového snímače Ni1000 připojeného přes multiplexer	PSE
Ni1000	Čtení teploty z odporového snímače teploty Ni1000.	PSE
Ni1000U2T	Převod napětí na snímači Ni1000 na teplotu	PSE
Nop	Prázdná operace	PSE
ParseTime	Převod času z DB-Net formátu do položek.	PSE
PID	Regulátor PID	PSE
Pt100R2T	Převod odpor na teplotu pro čidlo Pt100	PSE
PulseOut	Impulzní výstup na digitálním výstupu	PSE
PWM	Pulzně šířková modulace	PSE
Relay	Releový regulátor	PSE
REM	Komentář	PSE
Repeat	Cyklus s podmínkou na konci	PSE
Report	Zápis alarmu/hlášení/informace do provozního deníku	PSE
RPM	Měření otáček pomocí modulu AD_FDI8	PSE
RS	Klopný obvod RS	PSE
RS485Rx	Přepnutí RS485 na příjem (uživatelská komunikace)	PSE
RS485Tx	Přepnutí RS485 na vysílání (uživatelská komunikace)	PSE
SeqStep	Řízení sekvence	PSE
SetPwd	Nastavení přístupového hesla LcdShellu	PSE
SetTime	Nastavení času procesní stanice	PSE
StartType	Údaje o startu systému	PSE
Station	Zjistí číslo stanice	PSE
StopWatch	“Stopky” pro měření času procesoru	PSE
StrFormat	Formátování hodnoty (uživatelská komunikace).	PSE
StrParse	Převod formátovaných dat (řetězce) na hodnotu (uživatelská komunikace).	PSE
StrSubst	Náhrada znaků v řetězci (uživatelská komunikace)	PSE
SubInst	Instance podprogramu	PSE
Switch	Přepínač; rozskok podle hodnoty	PSE
SyncArch	Archivace údajů	PSE
SyncMark	Generátor časových značek pro modul SyncArch	PSE
SyncSum	Údržba sum a maxim v maticích se synchronizací.	PSE
Timeout	Údržba Timeoutů pro komunikace a displeje	PSE
Timer	Časování a zpoždování digitálních signálů	PSE
TimerOff	Zpoždění sestupné hrany	PSE
TimerOn	Zpoždění náběžné hrany	PSE
TimerPuls	Generování pulzu od náběžné hrany	PSE
Tmo	Hlavní modul hlídání timeoutu	PSE
TmoCheck	Kontrola stavu timeoutu	PSE
TmoStart	Spuštění timeoutu	PSE
TmoStop	Zrušení timeoutu	PSE
Until	Ukončení cyklu s podmínkou na konci	PSE

Modul	Popis	Knihovna
ValAct1	Buzení ventilu s koncovými stavovými spínači	PSE
ValAct2	Buzení ventilu s impulzními koncovými spínači	PSE
ValAct3	Buzení ventilu bez koncových spínačů	PSE
ValCtrl	Řízení obecného ventilu	PSE
Valve	Ovládání jednoduchého ventilu bez koncových spínačů	PSE
VarWStat	Test příznaku zápisu databázové proměnné.	PSE
Watchdog	Obsluha logického hlídacího časovače	PSE
While	Cyklus s podmínkou na začátku	PSE
EthNetSeg	Definice vzdálené stanice na Ethernetu	ACTIVE
EthReqDb	Žádost o přenos databázové proměnné po síti Ethernet	ACTIVE
EthRqDbDr	Žádost o přímý přenos databázové proměnné po síti Ethernet	ACTIVE
EthRoute	Definice statického směrování	ACTIVE
EthSState	Vrátí stav vzdálené stanice	ACTIVE
ReqDb	Žádost o přenos proměnné po síti DB-NET	ACTIVE
ReqDbDir	Žádost o přímý přenos databázové proměnné po síti.	ACTIVE
ReqTime	Přenos času po síti	ACTIVE
NETStat	Informace o stavu komunikační sítě DB-Net	ACTIVE
ARION	Základní modul obsluhy sítě ARION	ARION
ARN_AI	Čtení analogového vstupu v síti ARION	ARION
ARN_AO	Zápis analogového výstupu v síti ARION	ARION
ARN_DI	Čtení digitálních vstupů v síti ARION	ARION
ARN_DO	Zápis digitálních výstupů v síti ARION	ARION
ARN_NODE	Definice uzlu na sběrnici ARION	ARION
ARN_SfAO	Definice bezpečného stavu AO v síti ARION	ARION
ARN_SfDO	Definice bezpečného stavu DO v síti ARION	ARION
CAN_AI	Modul analogových vstupů na sběrnici CAN	CAN
CAN_AO	Modul analogových výstupů na sběrnici CAN	CAN
CAN_DI	Modul digitálních vstupů na sběrnici CAN	CAN
CAN_DO	Modul digitálních výstupů na sběrnici CAN	CAN
CAN_NMT_M	NMT-Master sběrnice CAN	CAN
CAN_NMT_N	Modul nulové obsluhy NMT-vrstvy sběrnice CAN	CAN
CAN_NMT_S	NMT-Slave sběrnice CAN	CAN
CAN_Node	Definice uzlu pro CAN_NMT_M/CAN_NMT_S	CAN
CAN_PDO	Obecný procesní datový objekt sběrnice CAN	CAN
CAN_S_AI	Slave-modul an. vstupů na sběrnici CAN	CAN
CAN_S_AO	Slave-modul an. výstupů na sběrnici CAN	CAN
CAN_S_DI	Slave-modul dig. vstupů na sběrnici CAN	CAN
CAN_S_DO	Slave-modul dig. výstupů na sběrnici CAN	CAN
CNC_ADCAN	Připojení ke CANu přes modul AD-CAN	CAN
CNC_ADOS	Připojení ke CANu pro ADOS100/200	CAN
CNC_AMP99	Připojení ke CANu pro AMAP99	CAN
CNC_AMR99	Připojení ke CANu pro AMiRiS99	CAN
CNC_APT2k	Připojení ke CANu pro APT2100G	CAN

Modul	Popis	Knihovna
CNC_ART4k	Připojení ke CANu pro ART4000	CAN
CNC_C167	Připojení ke CANu pro systémy s C167	CAN
EscGLCD	Vyslání Escape-sekvence na grafický terminál	LCD
GGraf	Vyslání hodnoty do grafu grafického terminálu	LCD
GGrafClr	Vyprázdnění tabulky hodnot grafu	LCD
LCD200	Interpretr dat z LCDSHELLu pro grafický terminál APT200 připojený po lince RS232	LCD
LCD200_4	Interpretr dat z LCDSHELLu pro grafický terminál APT200 připojený po lince RS485	LCD
LCD2100	Interpretr dat z LCDSHELLu pro grafický terminál APT2100	LCD
LCD485	Interpretr dat z LCDSHELLu pro terminály APT připojené po RS485	LCD
LCDADIR	Interpretr dat z LCDSHELLu pro paralelní terminál 2x8 znaků s miniklávesnicí (ADIR)	LCD
LCDG485	Interpretr dat z LCDSHELLu pro terminál APT2000 připojený po lince RS485	LCD
LCDGTTY	Interpretr dat z LCDSHELL pro grafický terminál APT2000 připojený po lince RS232	LCD
LCDK12	Interpretr dat z LCDSHELLu pro paralelní terminál LCDK12	LCD
LCDK14	Interpretr dat z LCDSHELLu pro paralelní terminál LCDK14	LCD
LCDMest	Interpretr dat z LCDSHELLu pro paralelní terminál (8x21 znaků max.) stanice MESTERM	LCD
LCDMini	Interpretr dat z LCDSHELLu pro paralelní terminál 2x16 znaků s miniklávesnicí (ART267)	LCD
LCD4x20	Interpretr dat z LCDSHELLu pro paralelní terminál 4x20 znaků	LCD
LCDRfsh	Občerstvování LCD displeje	LCD
LCDTTY	Připojení terminálu APT po seriové lince RS232	LCD
Boilers	Výběr z 32 kotlů podle provozních hodin	KOTELNA
DayPlan	Denní časový plán	KOTELNA
Holiday	Definice prázdnin pro modul DayPlan	KOTELNA
HourRun	Sledování provozních hodin zařízení	KOTELNA
PPlan	Periodické plánování hodnot po etapách	KOTELNA
ShortCut	Regulace teploty vratu pomocí simulace zkratu	KOTELNA
Stand_in	Výběr zařízení podle provozních hodin	KOTELNA
ArcPrint	Tisk archivů na tiskárně	PRINT
LogPrint	Tisk provozního deníku na tiskárně	PRINT
PrintBar	Tisk sloupce na základě databázové proměnné	PRINT
PrintBin	Tisk binárních dat na sériové tiskárně	PRINT
PrintMgr	Modul komunikace se sériovou tiskárnou	PRINT
PrintStr	Tisk řetězce na sériové tiskárně	PRINT
PrintTM	Tisk data a času na sériové tiskárně	PRINT
PrintVar	Tisk číselné hodnoty na sériové tiskárně	PRINT
PrtdbStr	Tisk databázového řetězce na sériové tiskárně	PRINT
MDImp	Čtení a přepočítání šestnáctice impulzních vstupů	MATRIX
MImpIn	Ošetření šestnácti impulzních vstupů	MATRIX
MtxAdd	Součet dvou matic podle pravidel maticového počtu	MATRIX

Modul	Popis	Knihovna
MtxCopy	Kopie matice	MATRIX
MtxMul	Násobení matic mezi sebou nebo matice číslem podle pravidel maticového počtu	MATRIX
MtxSub	Rozdíl dvou matic podle pravidel maticového počtu	MATRIX
MtxVMul	Násobení matice vektorem po řádcích nebo po sloupcích	MATRIX



# Způsob popisu modulů, význam symbolů

V této části vysvětlíme jakým způsobem budeme popisovat jednotlivé moduly a jejich parametry. Popíšeme také význam symbolů, které se používají v dalším textu.

## Modul

Popis každého modulu vypadá následovně:

<Jméno modulu>	<Krátký popis funkce>
----------------	-----------------------

### Popis

<Podrobný popis funkce>

### Parametry

<Popis významu jednotlivých parametrů>

### Příklad

<Krátký příklad použití modulu v aplikaci>

## Parametry

Každý parametr je popsán tabulkou:

<jméno>	<IN/OUT>	<typ>	<popis parametru>
---------	----------	-------	-------------------

Význam jednotlivých polí tabulky:

- ♦ <jméno>

Jméno parametru

- ♦ <IN/OUT>

Směr signálového toku:

- 1) **IN**: modul parametr načítá
- 2) **OUT**: modul parametr zapisuje
- 3) **IN/OUT**: modul parametr načítá i zapisuje
- 4) **PAR**: neutrální parametr, modul jej nečte ani nezapisuje, jedná se zpravidla o konstantní parametr (číslo)
- 5) **STI**: modul parametr načítá z logického zásobníku LA-procesu / vstupní kontakt modulu v reléovém schématu
- 6) **STO**: modul parametr ukládá na logický zásobník LA-procesu / výstupní kontakt modulu v reléovém schématu

Parametry s označením STI a STO mohou být pouze v LA-modulech, v normálních modulech nejsou.

- ♦ <typ>

Typ parametru:

Typ	Význam
Konst	Číselná konstanta.
Výběr	Konstanta, která se zadává výběrem z více hodnot. Hodnoty mohou být buď numerické nebo logické (výběr ANO/NE).
I	Jméno databázové proměnné typu I
L	Jméno databázové proměnné typu L
F	Jméno databázové proměnné typu F
MI	Jméno maticové databázové proměnné typu MI. Běžně budeme tímto symbolem rozumět <u>prvek matice</u> . To znamená, že kromě jména se zadává ještě číslo řádku a číslo sloupce matice. V ostatních případech bude výslovně uvedeno, že se jedná buď o <u>celou matici</u> (zadává se pouze jméno) nebo o <u>řádek matice</u> (zadává se jméno a číslo řádku) nebo o <u>sloupec matice</u> (zadává se jméno a číslo sloupce).

Typ	Význam
ML	Jméno maticové databázové proměnné typu <u>ML</u> . Běžně budeme tímto symbolem rozumět <u>prvek matice</u> . To znamená, že kromě jména se zadává ještě číslo řádku a číslo sloupce matice. V ostatních případech bude výslovně uvedeno, že se jedná buď o <u>celou matici</u> (zadává se pouze jméno) nebo o <u>řádek matice</u> (zadává se jméno a číslo řádku) nebo o <u>sloupec matice</u> (zadává se jméno a číslo sloupce).
MF	Jméno maticové databázové proměnné typu <u>F</u> . Běžně budeme tímto symbolem rozumět <u>prvek matice</u> . To znamená, že kromě jména se zadává ještě číslo řádku a číslo sloupce matice. V ostatních případech bude výslovně uvedeno, že se jedná buď o <u>celou matici</u> (zadává se pouze jméno) nebo o <u>řádek matice</u> (zadává se jméno a číslo řádku) nebo o <u>sloupec matice</u> (zadává se jméno a číslo sloupce).
Bit	Bit databázové proměnné. Může být zadán dvěma způsoby: - jméno proměnné typu <u>I</u> a číslo bitu - jméno alias proměnné (přezdívky bitu) Kromě bitu jednoduché databázové proměnné může tento symbol označovat také bit maticové databázové proměnné. V tomto případě se zadává jméno proměnné, číslo řádku, číslo sloupce a číslo bitu..
NONE	Parametr očekává jméno databázové proměnné, které se však nemusí zadávat. Modul pak pracuje s implicitní hodnotou. Symbol NONE bývá uveden vždy v kombinaci s jiným symbolem pro databázovou proměnnou.
DI16	Číslo logického kanálu DI (16-tice signálů)
DI	Signál logického kanálu DI. Lze zadat dvěma způsoby: - číslo kanálu a číslo signálu - jméno signálu
DO16	Číslo logického kanálu DO (16-tice signálů)
DO	Signál logického kanálu DO. Lze zadat dvěma způsoby: - číslo kanálu a číslo signálu - jméno signálu
AI	Logický kanál AI. Lze zadat dvěma způsoby: - číslo kanálu - jméno signálu
AO	Logický kanál AO. Lze zadat dvěma způsoby: - číslo kanálu - jméno signálu
Návěští	Návěští spolupracujícího modulu
Řetězec	Řetězec znaků
bit	Parametr na logickém zásobníku LA-procesu / kontakt modulu v reléovém schématu. Datový typ parametru je <u>bit</u> . Tento typ parametru může být pouze v LA-modulech, v normálních modulech není.
int	Parametr na logickém zásobníku LA-procesu / kontakt modulu v reléovém schématu. Datový typ parametru je <u>int</u> . Tento typ parametru může být pouze v LA-modulech, v normálních modulech není.
word	Parametr na logickém zásobníku LA-procesu / kontakt modulu v reléovém schématu. Datový typ parametru je <u>word</u> . Tento typ parametru může být pouze v LA-modulech, v normálních modulech není.
long	Parametr na logickém zásobníku LA-procesu / kontakt modulu v reléovém schématu. Datový typ parametru je <u>long</u> . Tento typ parametru může být pouze v LA-modulech, v normálních modulech není.
dword	Parametr na logickém zásobníku LA-procesu / kontakt modulu v reléovém schématu. Datový typ parametru je <u>dword</u> . Tento typ parametru může být pouze v LA-modulech, v normálních modulech není.
float	Parametr na logickém zásobníku LA-procesu / kontakt modulu v reléovém schématu. Datový typ parametru je <u>float</u> . Tento typ parametru může být pouze v LA-modulech, v normálních modulech není.
Klávesa	Jméno klávesy LCD displeje. Tento typ parametru může být pouze v zobrazovacích prvcích LCDSHELL.

## Strukturované parametry

Některé konstantní parametry mohou být strukturované. Strukturovaný parametr je složen z několika dílčích parametrů. Editace takového parametru představuje editaci seznamu dílčích parametrů. Popis strukturovaného parametru se skládá z jedné tabulky strukturovaného parametru a několika tabulek s popisem dílčích parametrů. Poznamenejme, že všechny dílčí parametry jsou vždy konstantní a zadávají se jako číslo nebo výběr z hodnot anebo logický výběr ANO/NE. Příklad strukturovaného parametru:

*hlavní parametr:*

<jméno>	<IN/OUT>	<typ>	<popis parametru>
---------	----------	-------	-------------------

*dílčí parametr č. 1:*

<jméno>	<typ>	<popis parametru>
---------	-------	-------------------

*dílčí parametr č. 2:*

<jméno>	<typ>	<popis parametru>
---------	-------	-------------------

atd.

## Variabilní parametry

Některé parametry mohou mít více různých typů. Vhodný typ parametru volí programátor při editaci modulu. Takové parametry se nazývají variabilní. Popis variabilních parametrů je podobný jako popis normálních parametrů s tím rozdílem, že tabulka je rozšířena o několik dalších polí ve sloupečku *Typ*. V těchto polích jsou vypsány všechny typy, které lze pro daný parametr zvolit. Na prvním místě tohoto sloupečku je typ, který je zvolen implicitně, pokud programátor nezvolí jiný.

<jméno>	<IN/OUT>	<typ1>	<popis parametru>
		<typ2>	
		<typ3>	
		...	
		...	

**AnIn**

Čtení hodnoty z AI kanálu s převodem na fyzikální veličinu

**Popis**

Modul čte analogový údaj z logického kanálu AI a přepočítává jej na fyzikální rozměr.

**Parametry**

<b>ChanAI</b>	IN	AI	Číslo vstupního logického kanálu AI.
<b>Hodnota</b>	OUT	F	Přepočítaná hodnota ve fyzikálních jednotkách.
		MF	
<b>Rozsah</b>	IN	Konst	Horní hranice měřicího rozsahu HW modulu v elektrických jednotkách.
		F	
		MF	
<b>EIMin</b>	IN	Konst	Dolní mez signálu v elektrických jednotkách.
		F	
		MF	
<b>EIMax</b>	IN	Konst	Horní mez signálu v elektrických jednotkách.
		F	
		MF	
<b>FyzMin</b>	IN	Konst	Dolní mez signálu ve fyzikálních jednotkách.
		F	
		MF	
<b>FyzMax</b>	IN	Konst	Horní mez signálu ve fyzikálních jednotkách.
		F	
		MF	

Hodnoty parametrů pro jednotlivé měřicí rozsahy. Fyzikální rozsah měřené veličiny necht' je např. -20..50°C.

Elektrický rozsah	Rozsah	EIMin	EIMax	FyzMin	FyzMax
0..5V -5..5V	5	0 -5	5	-20	50
0..10V -10..10V	10	0 -10	10	-20	50
0..20mA 4..20mA -20..20mA	20	0 4 -20	20	-20	50
0..40mA -40..40mA	40	0 -40	40	-20	50

Přepočet na fyzikální veličinu

Pro přepočet na fyzikální veličinu jsou zapotřebí dva body, které udávají hodnotu fyzikální veličiny odpovídající dané hodnotě elektrické veličiny. Tyto dva body jsou zadány parametry [EIMin,FyzMin] a [EIMax,FyzMax]. Obvykle se tyto body zadávají jako krajní hodnoty rozsahu. Není to ovšem nezbytně nutné. Nemusejí to být krajní body, ale mohou ležet i uvnitř rozsahu.

**Příklad**

AnIn #0.9, Hodnota, 20, 4, 20, 0, 25000

Měření tlaku na rozsahu 0 až 25000 Pa. Čidlo s rozsahem 4..20mA je připojeno na logický vstupní kanál bit č. 0.9. Výsledek se zapisuje do proměnné Hodnota.

<b>AnOut</b>	<b>Zápis hodnoty do AO kanálu převodem z fyzikální veličiny</b>
--------------	---

**Popis**

Modul zapisuje hodnotu proměnné, jejíž fyzikální rozměr přepočítává na rozsah převodníku, do logického kanálu.

**Parametry**

<b>ChanAO</b>	OUT	AO	Číslo výstupního logického kanálu AO.
<b>Hodnota</b>	IN	F	Hodnota ve fyzikálních jednotkách.
		MF	
<b>Rozsah</b>	IN	Konst	Horní hranice výstupního rozsahu HW modulu v elektrických jednotkách.
		F	
		MF	
<b>EIMin</b>	IN	Konst	Dolní mez signálu v elektrických jednotkách.
		F	
		MF	
<b>EIMax</b>	IN	Konst	Horní mez signálu v elektrických jednotkách.
		F	
		MF	
<b>FyzMin</b>	IN	Konst	Dolní mez signálu ve fyzikálních jednotkách.
		F	
		MF	
<b>FyzMax</b>	IN	Konst	Horní mez signálu ve fyzikálních jednotkách.
		F	
		MF	

Hodnoty parametrů pro jednotlivé měřicí rozsahy (fyzikální rozsah měřené veličiny necht' je např. -20..50°C):

Elektrický rozsah	Rozsah	EIMin	EIMax	FyzMin	FyzMax
0..5V -5..5V	5	0 -5	5	-20	50
0..10V -10..10V	10	0 -10	10	-20	50
0..20mA 4..20mA -20..20mA	20	0 4 -20	20	-20	50
0..40mA -40..40mA	40	0 -40	40	-20	50

Přepočet z fyzikální veličiny

Pro přepočet z fyzikální veličiny jsou zapotřebí dva body, které udávají hodnotu fyzikální veličiny odpovídající dané hodnotě elektrické veličiny. Tyto dva body jsou zadány parametry [EIMin,FyzMin] a [EIMax,FyzMax]. Obvykle se tyto body zadávají jako krajní hodnoty rozsahu. Není to ovšem nezbytně nutné. Nemusejí to být krajní body, ale mohou ležet i uvnitř rozsahu.

**Příklad**

AnOut #0.1, SERVO, 10.0, 0.0, 10.0, 0.0, 100.0

Proměnná SERVO o rozsahu 0..100% vystupuje na AO kanál 1 s rozsahem převodníku 0..10V.

<b>ArcPrint</b>	Tisk archivů na tiskárně
-----------------	--------------------------

**Popis**

Modul vytiskne až 8 průběhů archivních dat. Průběhy se tisknou vedle sebe na řádek. Řádek začíná časem vzorku a dále pokračuje maximálně 8-mi hodnotami. Na začátek tiskové sestavy se vytiskne název tisku a hlavička s popisy jednotlivých průběhů. Tisk se startuje bitem řídicí proměnné.

**Parametry**

<b>PrintMgr</b>	PAR	Návěští	Návěští modulu <b>PrintMgr</b> .
-----------------	-----	---------	----------------------------------

<b>Řízení</b>	IN/OUT	Bit	Bit řídicí proměnné. Tisk se spustí hodnotou "1" v tomto bitu.
---------------	--------	-----	--

<b>Režim</b>	PAR	Konst	Příznak nulování bitu řídicí proměnné.
--------------	-----	-------	--

Je-li Režim=1, tak se po zahájení tisku nuluje řídicí bit Řízení (aby se netisklo stále dokola). Tento příznak se typicky nastavuje posledním ze série modulů **ArcPrint**. Tím se dosáhne vytištění všech archivů na povel společným řídicím bitem.

<b>Index</b>	IN/OUT	I	Archivní index.
--------------	--------	---	-----------------

<b>Mat.časů</b>	IN/OUT	ML	Matice časů jednotlivých vzorků archivu rozměru [1,n].
-----------------	--------	----	--

<b>Název</b>	PAR	Řetězec	Název tisku. Tiskne se jako první.
--------------	-----	---------	------------------------------------

<b>Počet</b>	PAR	Konst	Skutečný počet průběhů (tisknutých sloupců).
--------------	-----	-------	--

Modul umožňuje maximálně 8 průběhů.

Dále následují parametry jednotlivých průběhů. Popíšeme pouze jeden.

<b>x-Popis</b>	PAR	Řetězec	Hlavička sloupce, např "tlak".
----------------	-----	---------	--------------------------------

(x=<1 až 8>)

<b>x-Popis2</b>	PAR	Řetězec	Podhlavička sloupce - tiskne se pod hlavičku x-Popis, např. "kPa".
-----------------	-----	---------	--

<b>x-Archiv</b>	IN	MI	Řádek matice libovolného typu - řádek archivu s daným průběhem.
		ML	
		MF	

<b>x-Před</b>	PAR	Konst	Počet míst před desetinnou tečkou výpisu hodnoty.
---------------	-----	-------	---

<b>x-Za</b>	PAR	Konst	Počet míst za desetinnou tečkou výpisu hodnoty.
-------------	-----	-------	---

**Příklad**

```
:01000 PrintMgr 0x0900, 2, 9600, 8, 0, 1
```

```
ArcPrint      :01000, PrnOn.0, 1, NdxArcP, ArcTime, "TLAKY", 2,
               "P1", "kPa", ArcP[0,*], 3, 1, "P2", "kPa",
               ArcP[1,*], 3, 1, "", "", NONE[0,*], 2, 1, ... atd.
```

Modul tiskne 2 průběhy archivu ArcP, tisk se řídí bitem PrnOn.0. Výsledný tisk vypadá zhruba takto:

## TLAKY

Datum a čas	P1	P2
	kPa	kPa
6.3.1997,17:30:00	121.3	168.5
6.3.1997,17:45:00	121.1	171.5
6.3.1997,18:00:00	118.9	163.0
...		

<b>ARION</b>	Základní modul obsluhy sítě ARION
--------------	-----------------------------------

**Popis**

Modul zajišťuje obsluhu komunikace na síti ARION, připojené k sériovému komunikačnímu kanálu řídicího systému. Viz též dodatek "Obsluha sítě vzdálených vstup/výstupních zařízení ARION".

**Umístění modulu**

Modul musí být umístěn v procesu ProcINIT, a to kdekoli před (nad) všemi moduly **ARN\_NODE**, které se na něj odkazují pomocí návěští. Modulu **ARION** je proto nutno přidělit návěští.

**Parametry**

<b>Port</b>	PAR	Konst	Číslo použitého komunikačního kanálu.
-------------	-----	-------	---------------------------------------

<b>Rychlost</b>	PAR	Konst	Udává přenosovou rychlost sítě ARION.
-----------------	-----	-------	---------------------------------------

Režim	PAR	Výběr	Udává režim, ve kterém má síť ARION pracovat. Bližší popis jednotlivých režimů viz dodatek "Obsluha sítě vzdálených vstup/výstupních zařízení ARION".	
			Hodnota	Význam
			0	Simplex2 - Simplexní režim na rozhraní RS232/RS422
			1	Simplex4 - Simplexní režim na rozhraní RS485
			2	HalfDupl2 - Poloduplexní režim na RS232/RS422
			3	HalfDupl4 - Poloduplexní režim na rozhraní RS485
			4	Duplex - Plně duplexní režim na RS232/RS422
			6	Autonom2 - Autonomní režim na rozhraní RS232/RS422
			7	Autonom4 - Autonomní režim na rozhraní RS485

**Příklad**

```
ProcInit:
17001 ARION 1, 19200, HalfDupl4
```

Zajišťuje obsluhu sítě ARION na komunikačním kanále číslo 1 (systémové rozhraní RS485) na komunikační rychlosti 19200 Bd, v poloduplexním režimu na rozhraní RS485.



<b>ARN_AI</b>	Čtení analogového vstupu v síti ARION
---------------	---------------------------------------

**Popis**

Modul definuje příjem dat jednoho analogového vstupního signálu ze vstupního zařízení připojeného na síť ARION.

Data se ukládají do proměnné určené parametrem *Hodnota* po přepočtu na fyzikální rozměr. Přepočet probíhá na základě parametrů *Rozsah*, *ElMin*, *ElMax*, *FyzMin* a *FyzMax* analogicky k modulu **AnIn**.

**Řízení přenosu**

Uskutečnění fyzického přenosu dat po síti dosáhneme dosazením jedničky za parametr *Přenést*.

Pokud z jednoho zařízení čteme více vstupních signálů, a tedy se přijímá prostřednictvím více modulů **ARN\_AI**, bylo by zbytečným přetěžováním sítě žádat o přenos v každém z těchto modulů. Je tedy vhodné dosazovat jedničku za *Přenést* jen v jednom z nich (zpravidla v prvním).

**Ukládání dat do výstupní proměnné**

Data se do proměnné uvedené v parametru *Hodnota* neukládají v okamžiku vyvolání modulu **ARN\_AI**, ale asynchronně, při příjmu odpovídajícího rámce po síti. Z toho vyplývá následující:

- V autonomním režimu, kdy jsou data zasílána z iniciativy samotného podřízeného uzlu, není nutno modul **ARN\_AI** vůbec umísťovat do periodického procesu, je možno ho vložit do ProcINIT.
- Není možné používat dočasnou proměnnou, do které se data načtou a potom někde zkopírují. Proměnná uvedená v parametru *Hodnota* musí být trvale vyhrazena jen pro tento účel.
- Chceme-li po vyvolání modulu s parametrem *Přenést* rovným jedné čekat na naplnění výstupní proměnné daty, je třeba vhodně testovat výstupní bit *Stav*.

**Parametry**

Uzel	PAR	Návěští	Návěští modulu <b>ARN_NODE</b> , který definuje příslušný uzel sítě ARION.
------	-----	---------	--

Toto návěští je globální (modul **ARN\_NODE** je v ProcINIT zatímco moduly **ARN\_xy** zpravidla v periodických procesech) a je ho tedy třeba zadat ručně.

Přenést	IN	Konst	Udává, zda se má při vyvolání tohoto modulu iniciovat přenos dat po síti ARION.
		Bit	

Stav	OUT	Bit	Výstupní bit, indikující, že byla přijata data a proměnná <i>Hodnota</i> byla naplněna. Nuluje se při vyvolání přenosu dat, nastavuje se na jedničku při příjmu odpovídajícího rámce (ať už na vyžádání parametrem <i>Přenést</i> , nebo z iniciativy podřízeného uzlu). Nechceme-li stav přenosu testovat, lze za tento parametr dosadit NONE.
------	-----	-----	--

Signál	PAR	Konst	Číslo signálu v rámci uzlu.
--------	-----	-------	-----------------------------

Hodnota	OUT	I	Jméno databázové proměnné (nebo prvku matice), do které se uloží hodnota ve fyzikálních jednotkách při příjmu odpovídajícího rámce.
		L	
		F	
		MI	
		ML	
		MF	

<b>Rozsah</b>	IN	Konst	Horní hranice měřicího rozsahu HW modulu v elektrických jednotkách.
		F	
		MF	
<b>EIMin</b>	IN	Konst	Dolní mez signálu v elektrických jednotkách.
		F	
		MF	
<b>EIMax</b>	IN	Konst	Horní mez signálu v elektrických jednotkách.
		F	
		MF	
<b>FyzMin</b>	IN	Konst	Dolní mez signálu ve fyzikálních jednotkách.
		F	
		MF	
<b>FyzMax</b>	IN	Konst	Horní mez signálu ve fyzikálních jednotkách.
		F	
		MF	

**Příklad**

```

ProcInit:
17001 ARION      1, 19200, HalfDupl4
17002 ARN_NODE   :17001, 1, 100, NONE.0, AI, 24, 0x0010

Proc00:
    ARN_AI        :17002, 1, @StavAI1, 0, Val1[0,0], 20, 4, 20, 0, 100
    ARN_AI        :17002, 0, NONE.0, 2, Val2[0,0], 20, 4, 20, 0, 100

```

Do proměnných Val1 a Val2 jsou načteny hodnoty analogových signálů číslo 0 a 2 ve stupních Celsia (0÷100) z uzlu číslo 1 po přepočtu z proudového rozsahu 4÷20mA. Stav přenosu je ukládán do bitu @StavAI1.

<b>ARN_AO</b>	Zápis analogového výstupu v síti ARION
---------------	--

**Popis**

Modul definuje vysílání dat jednoho analogového výstupního signálu do výstupního zařízení připojeného na síť ARION.

Data se čtou z proměnné určené parametrem *Hodnota* po přepočtu z fyzikálního rozměru na rozsah převodníku. Přepočet probíhá na základě parametrů *Rozsah*, *ElMin*, *ElMax*, *FyzMin* a *FyzMax* analogicky k modulu **AnOut**.

**Řízení přenosu**

Uspokojení fyzického přenosu dat po síti dosáhneme dosazením jedničky za parametr *Přenést*.

Pokud do jednoho zařízení zapisujeme více výstupních signálů, a tedy se vysílá prostřednictvím více modulů **ARN\_AO**, bylo by zbytečným přetěžováním sítě žádat o přenos v každém z těchto modulů. Je tedy vhodné dosazovat jedničku za *Přenést* jen v jednom z nich (zpravidla v posledním).

**Parametry**

<b>Uzel</b>	PAR	Návěští	Návěští modulu <b>ARN_NODE</b> , který definuje příslušný uzel sítě ARION.
-------------	-----	---------	--

Toto návěští je globální (modul **ARN\_NODE** je v ProcINIT zatímco moduly **ARN\_xy** zpravidla v periodických procesech) a je ho tedy třeba zadat ručně.

<b>Přenést</b>	IN	Konst Bit	Udává, zda se má při vyvolání tohoto modulu iniciovat přenos dat po síti ARION.
----------------	----	--------------	---

<b>Stav</b>	OUT	Bit	Výstupní bit, indikující, že data byla úspěšně vyslána na síť. Nuluje se při vyvolání přenosu dat, nastavuje se na jedničku při dokončení vysílání odpovídajícího rámce. Nechceme-li stav přenosu testovat, lze za tento parametr dosadit NONE.
-------------	-----	-----	---

<b>Signál</b>	PAR	Konst	Číslo signálu v rámci uzlu.
---------------	-----	-------	-----------------------------

<b>Hodnota</b>	OUT	I L F MI ML MF	Jméno databázové proměnné (nebo prvku matice), ze které se načte hodnota ve fyzikálních jednotkách, a po patřičném přepočtu se vloží do odpovídajícího rámce.
----------------	-----	-------------------------------	---

<b>Rozsah</b>	IN	Konst F MF	Horní hranice měřicího rozsahu HW modulu v elektrických jednotkách.
---------------	----	------------------	---

<b>ElMin</b>	IN	Konst F MF	Dolní mez signálu v elektrických jednotkách.
--------------	----	------------------	--

<b>ElMax</b>	IN	Konst F MF	Horní mez signálu v elektrických jednotkách.
--------------	----	------------------	--

<b>FyzMin</b>	IN	Konst F MF	Dolní mez signálu ve fyzikálních jednotkách.
---------------	----	------------------	--

FyzMax	IN	Konst	Horní mez signálu ve fyzikálních jednotkách.
		F	
		MF	

**Příklad**

```
ProcInit:
```

```
17001 ARION      1, 19200, HalfDupl4
```

```
17002 ARN_NODE   :17001, 1, 100, NONE.0, AO, 24, 0x0010
```

```
Proc00:
```

```
    ARN_AO        :17002, 0, @StavAO1, 0, Val1[0,0], 20, 4, 20, 0, 100
```

```
    ARN_AO        :17002, 1, NONE.0, 2, Val2[0,0], 20, 4, 20, 0, 100
```

Proměnné Val1 a Val2 jsou zapsány na analogové signály 0 a 2 uzlu číslo 1 po přepočtu z fyzikálního rozsahu 0÷100% na proudový rozsah 4÷20mA. Stav přenosu je ukládán do bitu @StavAO1.

<b>ARN_DI</b>	Čtení digitálních vstupů v síti ARION
---------------	---------------------------------------

**Popis**

Modul definuje příjem dat zvoleného počtu digitálních vstupních signálů ze vstupního zařízení připojeného na síť ARION.

Data se ukládají do proměnné určené parametrem *Proměnná*. Je-li za tento parametr dosazen prvek matice a parametrem *Signálů* zvoleno více signálů, než je počet bitů jedné položky této matice, ukládá se i do následujících prvků téhož řádku matice.

**Řízení přenosu**

Uskutečnění fyzického přenosu dat po síti dosáhneme dosazením jedničky za parametr *Přenést*.

Pokud z jednoho zařízení čteme data do více proměnných, a tedy se přijímá prostřednictvím více modulů **ARN\_DI**, bylo by zbytečným přetěžováním síť žádat o přenos v každém z těchto modulů. Je tedy vhodné dosazovat jedničku za *Přenést* jen v jednom z nich (zpravidla v prvním).

**Ukládání dat do výstupní proměnné**

Data se do proměnné uvedené v parametru *Hodnota* neukládají v okamžiku vyvolání modulu **ARN\_DI**, ale asynchronně, při příjmu odpovídajícího rámce po síti. Z toho vyplývá následující:

- V autonomním režimu, kdy jsou data zasílána z iniciativy samotného podřízeného uzlu, není nutno modul **ARN\_DI** vůbec umísťovat do periodického procesu, je možno ho vložit do ProcINIT.
- Není možné používat dočasnou proměnnou, do které se data načtou a potom někde zkopírují. Proměnná uvedená v parametru *Proměnná* musí být trvale vyhrazena jen pro tento účel.
- Chceme-li po vyvolání modulu s parametrem *Přenést* rovným jedné čekat na naplnění výstupní proměnné daty, je třeba vhodně testovat výstupní bit *Stav*.

**Parametry**

<b>Uzel</b>	PAR	Návěští	Návěští modulu <b>ARN_NODE</b> , který definuje příslušný uzel sítě ARION.
-------------	-----	---------	--

Toto návěští je globální (modul **ARN\_NODE** je v ProcINIT zatímco moduly **ARN\_xy** zpravidla v periodických procesech) a je ho tedy třeba zadat ručně.

<b>Přenést</b>	IN	Konst	Udává, zda se má při vyvolání tohoto modulu iniciovat přenos dat po síti ARION.
		Bit	

<b>Stav</b>	OUT	Bit	Výstupní bit, indikující, že byla přijata data a proměnná <i>Proměnná</i> byla naplněna. Nuluje se při vyvolání přenosu dat, nastavuje se na jedničku při příjmu odpovídajícího rámce (ať už na vyžádání parametrem <i>Přenést</i> , nebo z iniciativy podřízeného uzlu). Nechceme-li stav přenosu testovat, lze za tento parametr dosadit NONE.
-------------	-----	-----	---

<b>Signálů</b>	PAR	Konst	Počet bitů dat, které se mají uložit do proměnné předané za parametr <i>Proměnná</i> .
----------------	-----	-------	--

<b>Počátek</b>	PAR	Konst	Číslo prvního signálu v rámci daného uzlu, který se má uložit do proměnné předané za parametr <i>Proměnná</i> .
----------------	-----	-------	---

Proměnná	OUT	I	Jméno databázové proměnné (nebo prvku matice), do které se uloží hodnoty signálů při příjmu odpovídajícího rámce .
		L	
		F	
		MI	
		ML	
		MF	

**Příklad**

```
ProcInit:
17001 ARION      1, 19200, HalfDupl4
17002 ARN_NODE   :17001, 1, 100, NONE.0, DI, 24, 0x0000
```

```
Proc00:
    ARN_DI        :17002, 1, @StavDI1, 16, 0, Val1[0,0]
    ARN_DI        :17002, 0, NONE.0, 8, 16, Val2[0,0]
```

Do proměnné Val1 (typu I) jsou načítány hodnoty prvních šestnácti vstupních signálů uzlu číslo 1. Data zbylých osmi vstupních signálů jsou načítána do proměnné Val2 (typu I). Stav přenosu je ukládán do bitu @StavDI1.

*Poznámka: je samozřejmě možné použít proměnnou typu L, což umožní načíst všech 24 signálů jediným vyvoláním modulu **ARN\_DI**.*

<b>ARN_DO</b>	Zápis digitálních výstupů v síti ARION
---------------	--

**Popis**

Modul definuje vysílání dat zvoleného počtu digitálních výstupních signálů na výstupní zařízení připojené na síť ARION.

Data se načítají z proměnné určené parametrem *Proměnná*. Je-li za tento parametr dosazen prvek matice a parametrem *Signálů* zvoleno více signálů, než je počet bitů jedné položky této matice, načítá se i z následujících prvků téhož řádku matice.

**Řízení přenosu**

Uskutečnění fyzického přenosu dat po síti dosáhneme dosazením jedničky za parametr *Přenést*.

Pokud do jednoho zařízení zapisujeme data z více proměnných, a tedy se vysílá prostřednictvím více modulů **ARN\_DO**, bylo by zbytečným přetěžováním sítě žádat o přenos v každém z těchto modulů. Je tedy vhodné dosazovat jedničku za *Přenést* jen v jednom z nich (zpravidla v posledním).

**Parametry**

<b>Uzel</b>	PAR	Návěští	Návěští modulu <b>ARN_NODE</b> , který definuje příslušný uzel sítě ARION.
-------------	-----	---------	--

Toto návěští je globální (modul **ARN\_NODE** je v ProcINIT zatímco moduly **ARN\_xy** zpravidla v periodických procesech) a je ho tedy třeba zadat ručně.

<b>Přenést</b>	IN	Konst Bit	Udává, zda se má při vyvolání tohoto modulu iniciovat přenos dat po síti ARION.
----------------	----	--------------	---

<b>Stav</b>	OUT	Bit	Výstupní bit, indikující, že data byla úspěšně vyslána na síť. Nuluje se při vyvolání přenosu dat, nastavuje se na jedničku při dokončení vysílání odpovídajícího rámce. Nechceme-li stav přenosu testovat, lze za tento parametr dosadit NONE.
-------------	-----	-----	---

<b>Signálů</b>	PAR	Konst	Počet bitů dat, které se mají načíst z proměnné předané za parametr <i>Proměnná</i> .
----------------	-----	-------	---

<b>Počátek</b>	PAR	Konst	Číslo prvního signálu v rámci daného uzlu, do kterého se mají zapsat data z proměnné předané za parametr <i>Proměnná</i> .
----------------	-----	-------	--

<b>Proměnná</b>	OUT	I	Jméno databázové proměnné (nebo prvku matice), jejíž příslušný počet bitů se uloží do odpovídajícího rámce.
		L	
		F	
		MI	
		ML	
		MF	

**Příklad**

```

ProcInit:
17001 ARION      1, 19200, HalfDupl4
17002 ARN_NODE   :17001, 1, 100, NONE.0, DO, 24, 0x0000

Proc00:
    ARN_DO        :17002, 0, @StavDO1, 16, 0, Val1[0,0]
    ARN_DO        :17002, 1, NONE.0, 8, 16, Val2[0,0]

```

Proměnná *Val1* (typu *I*) je vyslána do prvních šestnácti výstupních signálů uzlu číslo 1. Data zbylých osmi výstupních signálů jsou naplněna hodnotou proměnné *Val2* (typu *I*). Stav přenosu je ukládán do bitu *@StavDO1*.

*Poznámka: je samozřejmě možné použít proměnnou typu L, což umožní zapsat všech 24 signálů jediným vyvoláním modulu **ARN\_DO**.*

<b>ARN_NODE</b>	Definice uzlu na sběrnici ARION
-----------------	---------------------------------

**Popis**

Modul slouží k přidání uzlu do seznamu uzlů, které má obsluhovat modul **ARION**, na který se modul **ARN\_NODE** odkazuje návěští.  
Seznam uzlů slouží k řízení jejich inicializace a detekce stavu.

**Umístění modulu**

Modul musí být umístěn tak, aby byl poprvé spuštěn později než modul **ARION**, na který se odkazuje pomocí návěští, a zároveň dříve než všechny moduly z knihovny **ARION** (moduly **ARN\_...**), které se pomocí návěští odkazují na něj. Modulu **ARN\_NODE** je proto nutno přidělit návěští.

Typicky se tyto požadavky splní tak, že se modul umístí v procesu ProcINIT, a to kdekoliv za (pod) modulem **ARION**, na který se odkazuje. V případě překročení dovolené spotřeby paměti procesu ProcINIT lze ovšem modul **ARN\_NODE** umístit i do vhodně zvoleného periodického procesu.

**Parametry**

<b>Sít'</b>	PAR	Návěští	Návěští modulu <b>ARION</b> , do jehož seznamu uzlů se má nový uzel přidat.
-------------	-----	---------	---

Toto návěští je nutno zadat ručně, protože řídicí systém může obsluhovat více nezávislých sítí ARION, a tedy obsahovat více modulů **ARION**. Vazbu na správný modul tedy nelze automaticky určit.

<b>Adresa</b>	PAR	Konst	Adresa uzlu na sběrnici.
---------------	-----	-------	--------------------------

<b>Kontrola</b>	PAR	Konst	Čas pro detekci ztráty spojení v milisekundách. Blíže viz dodatek "Obsluha sítě vzdálených vstup/výstupních zařízení ARION". Nechceme-li, aby uzel prováděl detekci ztráty spojení, dosadíme za tento parametr nulu.
-----------------	-----	-------	---

<b>Stav</b>	OUT	Bit	Výstupní bit, indikující, že spojení s uzlem bylo úspěšně navázáno, inicializace ukončena a nebyla detekována ztráta spojení. Nechceme-li stav spojení s uzlem testovat, lze za tento parametr dosadit NONE.
-------------	-----	-----	---

Typ	PAR	Výběr	Udává typ uzlu:	
			Hodnota	Význam
			0	AI - Modul analogových vstupů
			1	AO - Modul analogových výstupů
			2	DI - Modul digitálních vstupů
			3	DO - Modul digitálních výstupů

<b>Signálů</b>	PAR	Konst	Počet vstupních nebo výstupních signálů na daném uzlu.
----------------	-----	-------	--

<b>AnParam</b>	PAR	Výběr	Parametry A/D nebo D/A převodníku. Bere se v úvahu jen pro uzly typu AI nebo AO - viz parametr <b>Typ</b> .
----------------	-----	-------	---

<b>Rozlišení</b>	Konst	Počet datových bitů, na kterých je v komunikačním rámci uložena analogová hodnota (0÷31)
------------------	-------	--

<b>Bipolární</b>	ANO	Hodnota v komunikačním rámci se chápe jako číslo se znaménkem ve dvojkovém doplňku. Tedy např. pro Rozlišení=10 v rozsahu -512÷511.
	NE	Hodnota v komunikačním rámci se chápe jako kladné číslo bez znaménkového bitu. Tedy např. pro Rozlišení=10 v rozsahu 0÷1023.



**Příklad**

```
ProcInit:
17001 ARION      1, 19200, HalfDupl4
17002 ARN_NODE   :17001, 1, 100, @StavAO, AO, 24, 0x0010
17003 ARN_NODE   :17001, 2, 100, @StavDO, DO, 24, 0x0000
```

Síť ARION je naparametrizována pro obsluhu dvou zařízení. Jednoho typu DINA024x s adresou 1, druhého typu DINDO24 s adresou 2. Je zaručeno, že obě zařízení detekují ztrátu spojení s NMT-Masterem do 100 milisekund. Aktuální stav spojení je ukládán do proměnné StavAO, resp StavDO.

**ARN\_SfAO**

Definice bezpečného stavu AO v síti ARION

**Popis**

Modul slouží pro definici bezpečného stavu jednoho analogového výstupního signálu výstupního zařízení připojeného na síť ARION. Do bezpečného stavu jsou výstupní signály uvedeny v případě detekované ztráty spojení s řídicím systémem. Viz též parametr *Kontrola* modulu **ARN\_NODE**.

Vyvolání tohoto modulu má efekt pouze u těch výstupních zařízení, která podporují uživatelskou definici bezpečného stavu. Ostatní zařízení při ztrátě spojení nastavují výstupy do bezpečného stavu, který je pro dané zařízení neměnně určen (viz dokumentaci konkrétního výstupního zařízení). V druhém případě vyvolání modulu **ARN\_SfAO** neznamena chybu, ale jeho skutečný vliv na činnost zařízení se nijak nebude lišit od vyvolání modulu **ARN\_AO**.

Data se čtou z proměnné určené parametrem *Hodnota* po přepočtu z fyzikálního rozměru na rozsah převodníku. Přepočet probíhá na základě parametrů *Rozsah*, *ElMin*, *ElMax*, *FyzMin* a *FyzMax* analogicky k modulu **AnOut**.

**Řízení přenosu**

Data se přenesou do příslušného uzlu ve fázi jeho inicializace. Při prvotní inicializaci se uplatní moduly **ARN\_SfAO** umístěné v procesu *Proclnit*, jakož i moduly umístěné v periodických procesech, pokud v okamžiku prvotní inicializace již došlo k jejich vyvolání. Při prvotní inicializaci (nebo při reinicializaci po ztrátě a obnovení spojení) nerozhoduje hodnota parametru *Přenést*.

Případný nový přenos dat po síti v případě změny definice bezpečného stavu vyžaduje novou inicializaci uzlu. Té dosáhneme dosazením jedničky za parametr *Přenést* modulu **ARN\_SfAO** vyvolávaného z periodického procesu.

Pokud do jednoho zařízení zapisujeme více výstupních signálů, a tedy se vysílá prostřednictvím více modulů **ARN\_SfAO**, bylo by zbytečným přetěžováním sítě žádat o přenos v každém z těchto modulů. Je tedy vhodné dosazovat jedničku za *Přenést* jen v jednom z nich (zpravidla v posledním).

**Parametry**

<b>Uzel</b>	PAR	Návěští	Návěští modulu <b>ARN_NODE</b> , který definuje příslušný uzel sítě ARION.
-------------	-----	---------	--

Toto návěští je globální a je ho tedy třeba zadat ručně.

<b>Přenést</b>	IN	Konst	Udává, zda se má při vyvolání tohoto modulu iniciovat přenos dat po síti ARION, tedy vyvolat reinicializaci uzlu.
		Bit	

<b>Stav</b>	OUT	Bit	Výstupní bit, indikující, že data byla úspěšně vyslána na síť. Nuluje se při vyvolání přenosu dat, nastavuje se na jedničku při dokončení vysílání odpovídajícího rámce. Nechceme-li stav přenosu testovat, lze za tento parametr dosadit NONE.
-------------	-----	-----	---

<b>Signál</b>	PAR	Konst	Číslo signálu v rámci uzlu.
---------------	-----	-------	-----------------------------

<b>Hodnota</b>	OUT	I	Jméno databázové proměnné (nebo prvku matice), ze které se načte hodnota ve fyzikálních jednotkách, a po patřičném přepočtu se vloží do odpovídajícího rámce.
		L	
		F	
		MI	
		ML	
		MF	

<b>Rozsah</b>	IN	Konst	Horní hranice měřicího rozsahu HW modulu v elektrických jednotkách.
		F	
		MF	

<b>EIMin</b>	IN	Konst	Dolní mez signálu v elektrických jednotkách.
		F	
		MF	

<b>EIMax</b>	IN	Konst	Horní mez signálu v elektrických jednotkách.
		F	
		MF	

<b>FyzMin</b>	IN	Konst	Dolní mez signálu ve fyzikálních jednotkách.
		F	
		MF	

<b>FyzMax</b>	IN	Konst	Horní mez signálu ve fyzikálních jednotkách.
		F	
		MF	

**Příklad**

```

ProcInit:
17001 ARION      1, 19200, HalfDupl4
17002 ARN_NODE   :17001, 1, 100, NONE.0, AO, 24, 0x0010
                  ARN_SfAO   :17002, 0, @InitAO1, 0, Val1[0,0], 20, 4, 20, 0, 100
                  ARN_SfAO   :17002, 1, NONE.0, 2, Val2[0,0], 20, 4, 20, 0, 100

```

Proměnné Val1 a Val2 jsou zapsány jako bezpečné hodnoty pro analogové signály 0 a 2 uzlu číslo 1 po přepočtu z fyzikálního rozsahu 0÷100% na proudový rozsah 4÷20mA. Stav přenosu je ukládán do bitu @InitAO1.

**ARN\_SfDO**

Definice bezpečného stavu DO v síti ARION

**Popis**

Modul slouží pro definici bezpečného stavu zvoleného počtu digitálních výstupních signálů výstupního zařízení připojeného na síť ARION. Do bezpečného stavu jsou výstupní signály uvedeny v případě detekované ztráty spojení s řídicím systémem. Viz též parametr *Kontrola* modulu **ARN\_NODE**.

Vyvolání tohoto modulu má efekt pouze u těch výstupních zařízení, která podporují uživatelskou definici bezpečného stavu. Ostatní zařízení při ztrátě spojení nastavují výstupy do bezpečného stavu, který je pro dané zařízení neměnně určen (viz dokumentaci konkrétního výstupního zařízení). V druhém případě vyvolání modulu **ARN\_SfDO** neznamená chybu, ale jeho skutečný vliv na činnost zařízení se nijak nebude lišit od vyvolání modulu **ARN\_DO**.

Data se načítají z proměnné určené parametrem *Proměnná*. Je-li za tento parametr dosazen prvek matice a parametrem *Signálů* zvoleno více signálů, než je počet bitů jedné položky této matice, načítá se i z následujících prvků téhož řádku matice.

**Řízení přenosu**

Data se přenesou do příslušného uzlu ve fázi jeho inicializace. Při prvotní inicializaci se uplatní moduly **ARN\_SfDO** umístěné v procesu *Proclnit*, jakož i moduly umístěné v periodických procesech, pokud v okamžiku prvotní inicializace již došlo k jejich vyvolání. Při prvotní inicializaci (nebo při reinicializaci po ztrátě a obnovení spojení) nerozhoduje hodnota parametru *Přenést*.

Případný nový přenos dat po síti v případě změny definice bezpečného stavu vyžaduje novou inicializaci uzlu. Té dosáhneme dosazením jedničky za parametr *Přenést* modulu **ARN\_SfDO** vyvolávaného z periodického procesu.

Pokud do jednoho zařízení zapisujeme více výstupních signálů, a tedy se vysílá prostřednictvím více modulů **ARN\_SfDO**, bylo by zbytečným přetěžováním sítě žádat o přenos v každém z těchto modulů. Je tedy vhodné dosazovat jedničku za *Přenést* jen v jednom z nich (zpravidla v posledním).

**Parametry**

<b>Uzel</b>	PAR	Návěští	Návěští modulu <b>ARN_NODE</b> , který definuje příslušný uzel sítě ARION.
-------------	-----	---------	--

Toto návěští je globální a je ho tedy třeba zadat ručně.

<b>Přenést</b>	IN	Konst	Udává, zda se má při vyvolání tohoto modulu iniciovat přenos dat po síti ARION, tedy vyvolat reinicializaci uzlu.
		Bit	

<b>Stav</b>	OUT	Bit	Výstupní bit, indikující, že data byla úspěšně vyslána na síť. Nuluje se při vyvolání přenosu dat, nastavuje se na jedničku při dokončení vysílání odpovídajícího rámce. Nechceme-li stav přenosu testovat, lze za tento parametr dosadit <i>NONE</i> .
-------------	-----	-----	---

<b>Signálů</b>	PAR	Konst	Počet bitů dat, které se mají načíst z proměnné předané za parametr <i>Proměnná</i> .
----------------	-----	-------	---

<b>Počátek</b>	PAR	Konst	Číslo prvního signálu v rámci daného uzlu, do kterého se mají zapsat data z proměnné předané za parametr <i>Proměnná</i> .
----------------	-----	-------	--

<b>Proměnná</b>	OUT	I	Jméno databázové proměnné (nebo prvku matice), jejíž příslušný počet bitů se uloží do odpovídajícího rámce.
		L	
		F	
		MI	
		ML	
		MF	

**Příklad**

```
ProcInit:
17001 ARION      1, 19200, HalfDupl4
17002 ARN_NODE   :17001, 1, 100, NONE.0, DO, 24, 0x0000
                ARN_SfDO :17002, 0, @InitDO1, 16, 0, Val1[0,0]
                ARN_SfDO :17002, 1, NONE.0, 8, 16, Val2[0,0]
```

Proměnná `Val1` (typu `I`) je zapsána jako definice bezpečné hodnoty prvních šestnácti výstupních signálů uzlu číslo 1. Definice bezpečných hodnot zbylých osmi výstupních signálů jsou naplněna hodnotou proměnné `Val2` (typu `I`). Stav přenosu je ukládán do bitu `@InitDO1`.

*Poznámka: je samozřejmě možné použít proměnnou typu `L`, což umožní zapsat všech 24 signálů jediným vyvoláním modulu **ARN\_SfDO**.*

<b>AvgVar</b>	Klouzavý průměr
---------------	-----------------

**Popis**

Modul počítá klouzavý průměr z maximálně 12-ti vzorků. Aktuální počet vzorků lze měnit i za chodu. V každém běhu modulu se zpracovává nový vzorek.

**Parametry**

Hodnota	IN	I	Vstupní proměnná.
		L	
		F	
		MI	
		ML	
		MF	

Průměr	OUT	I	Výstupní proměnná - klouzavý průměr.
		L	
		F	
		MI	
		ML	
		MF	

Počet	IN	Konst	Počet vzorků, ze kterých se počítá klouzavý průměr. Lze zadat číslo 1..12.
		I	
		MI	

**Příklad**

AvgVar      P, Pf, 10

Filtrace proměnné P do proměnné Pf klouzavým průměrem z 10-ti vzorků.

**BinDiff**

Modul slouží k detekci náběžné a sestupné hrany

**Popis**

Modul sleduje najednou 16 signálů představovaných 16-ti bity proměnné *Signál*. Pro každý signál lze nastavit, zda se má detekovat náběžná nebo sestupná hrana, případně obě hrany. Pokud dojde ke sledované změně signálu, je tento stav detekován v proměnné *Detekce* nastavením příslušného bitu do log. "1". V případě, že ke sledované změně nedojde, je v příslušném bitu log. "0". Parametr *Náběžná* obsahuje masku pro sledování náběžných hran. Každému bitu parametru odpovídá jeden signál. Je-li bit v log."1", je detekována náběžná hrana tohoto signálu. Obdobně v parametru *Sestupná* je maska pro detekci sestupné hrany. V parametru *Init* je pro každý signál zakódováno chování po výpadku napájení. Parametr udává, s jakou hodnotou signálu se bude porovnávat sledovaný signál při prvním běhu modulu. Jsou možné 3 hodnoty tohoto parametru:

0	Signál se porovnává s hodnotou před výpadkem napájení.
1	Signál se porovnává s log. "1"
2	Signál se porovnává s log. "0".

**Parametry**

<b>Signál</b>	IN	I	Databázová proměnná, která obsahuje vstupní sledované signály.
---------------	----	---	--

<b>Detekce</b>	OUT	I	Výstupní databázová proměnná, která zobrazuje detekci hran.
----------------	-----	---	---

<b>Náběžná</b>	PAR	Výběr	Parametr obsahuje masku na náběžné hrany. Hodnota "1" (ANO) na příslušném bitu znamená, že modul sleduje vzestupnou hranu signálu.
----------------	-----	-------	--

<b>Sestupná</b>	PAR	Výběr	Parametr obsahuje masku na sestupné hrany. Hodnota "1" (ANO) na příslušném bitu znamená, že modul sleduje sestupnou hranu signálu.
-----------------	-----	-------	--

<b>Init</b>	PAR	Výběr	Nastavení porovnávací hodnoty signálu po výpadku napájení.
-------------	-----	-------	--

Je-li pro příslušný bit hodnota 0, je ponechána porovnávací hodnota signálu taková, jaká byla těsně před výpadkem napájení. Je-li hodnota pro bit 1, je nastavena hodnota porovnávacího signálu na "1", tzn. jako by během výpadku napájení přešel signál do "1". Je-li hodnota pro bit 2, je nastavena hodnota porovnávacího signálu na "0", tzn. jako by během výpadku napájení přešel signál do "0" (viz tabulka výše v popisu modulu).

**Příklad**

```
BinDiff          Signaly, Detekce, 0x0003, 0x000C, 0x00000000
```

Jsou detekovány náběžné hrany 0. a 1. signálu a sestupné hrany signálů 2 a 3. Po výpadku napájení se porovnávají všechny signály s hodnotou před výpadkem.

<b>BinIn</b>	Čtení jednoho binárního signálu
--------------	---------------------------------

**Popis**

Modul načte hodnotu jednoho digitálního vstupního signálu z logického kanálu DI do zvoleného bitu databázové proměnné typu `INT`. Čtený signál lze negovat.

**Parametry**

<b>Signál</b>	IN	DI	Signál logického kanálu DI z něžž bude modul číst.
---------------	----	----	--

<b>Negace</b>	PAR	Konst	Příznak negace vstupního signálu.
---------------	-----	-------	-----------------------------------

<b>Výstup</b>	OUT	Bit	Bit proměnné, do něžž bude hodnota načteného signálu uložena.
---------------	-----	-----	---

**Příklad**

```
BinIn #0.1,0,VstupX.1
```

Čte logický kanál DI číslo 0 bit č. 1 do bitu č. 1 proměnné `VstupX`. Signál nebude negován.



<b>BinMAOut</b>	Binární výstup s přepínáním Ručně / Automat
-----------------	---

**Popis**

Pomocí proměnné Přepínání lze přepínat digitální výstup z proměnné Ručně nebo Automat. Je-li nastaven příznak Negace, výstup se neguje. Modul umožňuje odepnout automatické ovládání a přejít na ruční.

**Parametry**

<b>Přepínání</b>	IN	Bit	Bit, který přepíná výstup.
------------------	----	-----	----------------------------

Hodnoty:

"0" - výstup je brán z proměnné Automat

"1" - výstup je brán z proměnné Ručně

<b>Ručně</b>	IN	Bit	Výstup ručního ovládání.
--------------	----	-----	--------------------------

<b>Automat</b>	IN	Bit	Výstup z regulátoru, nebo automatického řízení.
----------------	----	-----	---

<b>Negace</b>	PAR	Výběr	ANO = výstup je negován.
---------------	-----	-------	--------------------------

<b>Výstup</b>	OUT	DO	Výstupní signál DO.
---------------	-----	----	---------------------

**Příklad**

```
BinMAOut      Prepinani.0, Rucne.0, Auto.0, 0x0000, #2.5
```

Pomocí proměnné Prepinani se přepíná mezi proměnnou Rucne.0 a Auto.0 na výstup - 5. bit logického kanálu č. 2.

<b>BinOut</b>	Zápis jednoho binárního signálu
---------------	---------------------------------

**Popis**

Modul zapíše hodnotu zvoleného bitu databázové proměnné typu  $\mathbb{I}$  na výstup logického kanálu DO. Zapisovaný kanál lze negovat.

**Parametry**

<b>Proměnná</b>	IN	Bit	Bit proměnné, který se zapíše do výstupního kanálu.
-----------------	----	-----	---

<b>Negace</b>	PAR	Výběr	ANO=Negovat výstupní signál.
---------------	-----	-------	------------------------------

<b>Kanál</b>	OUT	DO	Číslo zapisovaného signálu DO.
--------------	-----	----	--------------------------------

**Příklad**

```
BinOut Vystup.0, 0, #1.2
```

Zapíše hodnotu 0.bitu proměnné Vystup do 2.bitu logického kanálu DO č. 1. Signál nebude negován.

<b>BKO</b>	Bistabilní klopný obvod
------------	-------------------------

**Popis**

Zvolenou hranou vstupu se překlápí výstup klopného obvodu.

**Parametry**

<b>Režim</b>	PAR	Výběr	Nastavení režimu
	Sestupná	ANO	Překlápění sestupnou hranou
		NE	Překlápění náběžnou hranou
<b>Vstup</b>	IN	Bit	Vstup klopného obvodu
<b>Výstup</b>	IN	Bit	Výstup klopného obvodu

**Příklad**

BKO                      0x0000, @BKO\_Vstup, @BKO\_Vystup

Náběžnou hranou bitu @BKO\_Vstup se překlápí výstupní bit @BKO\_Vystup.

**Boilers**

Rozdělení provozních hodin mezi několik kotlů

**Popis**

Modul ovládá automatické střídání a připínání kotlů (maximálně 32 jednotek). Typicky se používá, když se nějakým algoritmem nebo "natvrdo" určí počet kotlů, které mají v danou chvíli být zapnuty a modul poté sám vybírá vhodné kotle podle jejich stavu (porucha / v pořádku) a provozních hodin tak, aby byly provozní hodiny rovnoměrně rozděleny mezi jednotlivé kotle.

Podle zadaného počtu kotlů a provozních hodin modul vybírá kotel, který bude zapnut. Dojde-li na kotli k poruše, automaticky zapíná další kotel. Zároveň modul zajišťuje pravidelné střídání kotlů maximálně jednou denně (minimálně jednou týdně). Dojde-li v systému k chybě, jsou kotle uvedeny do poruchového stavu.

**Parametry**

<b>Počet</b>	IN	I	Požadovaný počet kotlů, které musí být v chodu najednou.
--------------	----	---	--

<b>Chyba</b>	IN	I	Příznak globální chyby kotelny.
--------------	----	---	---------------------------------

Chyby se dělí do dvou typů. Je-li alespoň jedna z nich nastavena, jsou všechny kotle nastaveny na příslušnou chybu (viz. 2. sloupec proměnné Řízení).

bit 0. - Chyba I.

bit 1. - Chyba II.

Význam chyb a reakce na ně je v rukou aplikátora.

Doporučený význam  
chyb

Chyba I. - vážná porucha, nutno okamžitě všechno vypnout (např. zaplavení kotelny)

Chyba II. - méně závažná porucha, stačí např. vypnout kotle a nemusí se vypínat oběhová čerpadla

<b>Provoz</b>	IN/OUT	ML	Vnitřní proměnná - matice rozměru počet kotlů x 2.
---------------	--------	----	--

Matice obsahuje pro každý kotel v prvním sloupci počet provozních hodin, ve druhém sloupci počet provozních sekund.

<b>Řízení</b>	IN/OUT	MI	Proměnná - matice rozměru počet kotlů x 3.
---------------	--------	----	--

1. sloupec - vstupní, z něj modul čte informaci, zda je kotel v poruše (hodnota 1) nebo v pořádku (hodnota 0).

2. sloupec - výstupní, příkazy pro činnost kotle. Význam jednotlivých bitů:

Bit	Stav "1"	Stav "0"
0	Kotel vypnut	Kotel vypnut
1	Kotel chod	Kotel vypnut
2	Kotel provozní záloha	Kotel není v provozní záloze
4	Kotel je v poruše I.	Kotel není v poruše
5	Kotel je v poruše II.	Kotel není v poruše

3. sloupec - informace o pořadí kotlů. Uživatel nesmí tento sloupec měnit.

Sloupec obsahuje indexy kotlů tak, jak se budou kotle postupně spouštět. Jako první se spustí kotel jehož index je v buňce Řízení[0, 2]. Jako druhý se spustí kotel jehož index je v buňce Řízení[1, 2], atd.

<b>Den</b>	PAR	Výběr	Nastavení jednoho nebo více dnů v týdnu (např. pondělí), kdy se musí střídát hlavní kotel.
------------	-----	-------	--

Hlavní kotel je ten, který se vybírá jako první při zapínání a je tedy nejvíce používán (běží vždy, pokud je požadovaný počet kotlů větší než nula). Současně se při střídání hlavního kotle také přepočítá pořadí, v jakém se budou připínat další kotle. Toto pořadí je opačné vzhledem k počtu provozních hodin. Dřív se tedy vybere pro připnutí kotel s menším počtem provozních hodin.

V nastavený den se tedy hlavním kotlem stane jiný kotel, vybraný na základě provozních hodin.

Čas	PAR	Konst	Počet sekund od začátku dne, kdy se vymění role hlavního kotle. Stane se tak v den zvolený parametrem Den.
-----	-----	-------	--

**Příklad**

Boilers                      PocKotlu, Chyba, Info, Rizeni, 0x0002, 21600, 0, 0

Modul střídá každé pondělí hlavní kotel (parametr 0x0002). Do proměnné PocKotlu se zadává požadovaný počet zapnutých kotlů. Pokud dojde k poruše některého kotle, je nutné nastavit hodnotu 1 do proměnné Rizeni[i,0], kde *i* je číslo kotle. Pokud dojde ke globální poruše (např. únik plynu), je nutné nastavit proměnnou Chyba na hodnotu 1 - chyba I. (vážná porucha), nebo na hodnotu 2 - chyba II. (méně závažná chyba).

<b>Break</b>	Přerušení cyklu <b>For</b> , <b>While</b> , <b>Repeat</b> , <b>Switch</b>
--------------	---

**Popis**

Pomocí modulu lze předčasně přerušit vykonávání cyklů **For**, **While**, **Repeat** a **Switch**. Modul se obvykle volá pouze na základě určité podmínky. To se zajistí umístěním modulu do podmíněné větve programu (konstrukce **If-EndIf** nebo **If-Else-EndIf**).

**Parametry**

Modul nemá žádné parametry.

**Příklad**

```

Let          Suma = 0
For          i, 0.0, 9.0, 1.0, :NONE
| Let       @Zaporne = Pole[0,i] < 0
| If        @Zaporne, :NONE
| | Break
| EndIf
| Let       Suma = Suma + Pole[0,i]
EndFor

```

Do proměnné *Suma* se sčítají buňky maticové proměnné *Pole* rozměru [1,10]. Výpočet se provádí v cyklu **For**. Narazí-li se při výpočtu na buňku se zápornou hodnotou, výpočet se přeruší modulem **Break**. Nechť např. proměnná *Pole* má tyto hodnoty:

Sloupec	0	1	2	3	4	5	6	7	8	9
Hodnota	1	2	3	-1	5	1	2	4	0	-2

Výsledný součet v proměnné *Suma* bude  $1+2+3 = 6$ .

<b>Call</b>	Volání podprogramu (Lib100-999)
-------------	---------------------------------

**Popis**

Modul volá podprogram Lib100 až Lib999 definovaný v tabulce procesů. Provádění dalších modulů je pozdrženo do okamžiku dokončení činnosti podprogramu.

Návrat z podprogramu nastane automaticky po vykonání posledního funkčního modulu v podprogramu nebo po vykonání pseudopříkazu **Exit**.

Podprogramu nelze přímo předat žádné parametry. Lze však (např. pomocí příkazu **Let** uložit potřebné hodnoty do databázových proměnných s dohodnutými jmény). Obdobným způsobem je možné řešit návrat výsledků z podprogramu.

**Parametry**

Číslo	PAR	Konst	Číslo volaného podprogramu (100 až 999).
-------	-----	-------	--

Podprogram musí existovat, jinak se neprovede žádná činnost.

**Příklad**

```
Call 101
```

Volá podprogram Lib101 a čeká na jeho dokončení. Pak pokračuje v činnosti.

<b>CAN_AI</b>	Modul analogových vstupů na sběrnici CAN
---------------	--

**Popis**

Modul (PDO-Master vstupního PDO, viz oddíl “Vrstva objektů procesních dat PDO” dodatku “Obsluha sběrnice CAN”) definuje příjem dat jednoho analogového vstupního signálu ze vstupního zařízení připojeného na sběrnici CAN.

Data se ukládají do proměnné určené parametrem *Hodnota* po přepočtu na fyzikální rozměr. Přepočet probíhá na základě parametrů *Rozsah*, *ElMin*, *ElMax*, *FyzMin* a *FyzMax* analogicky k modulu **AnIn**.

**Řízení přenosu**

U vstupních PDO je obvyklé, že přenos dat vyvolává PDO-Slave. U analogových signálů zpravidla v pravidelných intervalech. Přesto je možné vyžádat si přenos ze strany PDO-Mastera dosazením jedničky za parametr *Přenést*. V tom případě se vyšle remote-frame, na který PDO-Slave reaguje vysláním dat na sběrnici.

Pokud se jeden PDO skládá z více vstupních signálů, a tedy se přijímá prostřednictvím více modulů **CAN\_AI**, bylo by zbytečným přetěžováním sběrnice žádat o přenos v každém z těchto modulů. Je tedy vhodné dosazovat jedničku za *Přenést* jen v jednom z nich (zpravidla v prvním).

**Ukládání dat do výstupní proměnné**

Data se do proměnné uvedené v parametru *Hodnota* neukládají v okamžiku vyvolání modulu **CAN\_AI**, ale asynchronně, při příjmu odpovídajícího PDO. Z toho vyplývá následující:

- Spoléháme-li na zasílání dat z iniciativy PDO-Slavea (parametr *Přenést* má nulovou hodnotu), není nutno modul **CAN\_AI** vůbec umísťovat do periodického procesu, je možno ho vložit do ProclNIT.
- Není možné používat dočasnou proměnnou, do které se data načtou a potom někam zkopírují. Proměnná uvedená v parametru *Hodnota* musí být trvale vyhrazena jen pro tento účel.
- Chceme-li po vyvolání modulu s parametrem *Přenést* rovným jedné čekat na naplnění výstupní proměnné daty, je třeba vhodně testovat výstupní bit *Stav*.

**Identifikátor (COB-Id) použitého PDO**

Modul používá implicitní mapování PDO dle specifikace CANopen Minimum Capability Device, tedy pracuje s PDO s identifikátorem COB-Id = 641 + Node-Id - 1.

**Parametry**

<b>Připojení</b>	PAR	Návěští	Návěští připojovacího modulu <b>CNC_...</b> , který zajišťuje fyzické připojení ke sběrnici CAN.
------------------	-----	---------	--

Toto návěští je globální (modul **CNC\_...** je v ProclNIT zatímco moduly vrstvy PDO zpravidla v periodických procesech) a je ho tedy třeba zadat ručně.

<b>Node-Id</b>	PAR	Konst	Identifikátor uzlu (běžného nebo virtuálního) sběrnice CAN, který je PDO-Slavem příslušného PDO.
----------------	-----	-------	--

<b>Přenést</b>	IN	Konst Bit	Udává, zda se má při vyvolání tohoto modulu iniciovat přenos dat po sběrnici CAN.
----------------	----	--------------	---

<b>Stav</b>	OUT	Bit	Výstupní bit, indikující, že byla přijata data a proměnná <i>Hodnota</i> byla naplněna. Nuluje se při vyvolání přenosu dat, nastavuje se na jedničku při příjmu odpovídajícího PDO (ať už na vyžádání parametrem <i>Přenést</i> , nebo z iniciativy PDO-Slavea). Nechceme-li stav přenosu testovat, lze za tento parametr dosadit NONE.
-------------	-----	-----	--



<b>Parametry</b>	PAR	Výběr	Parametry převodníku.
	Rozlišení	Konst	Počet datových bitů, na kterých je v PDO uložena analogová hodnota (0÷31)
	Bipolární	ANO	Hodnota v PDO se chápe jako číslo se znaménkem ve dvojkovém doplňku. Tedy např. pro Rozlišení=10 v rozsahu -512÷511.
		NE	Hodnota v PDO se chápe jako kladné číslo bez znaménkového bitu. Tedy např. pro Rozlišení=10 v rozsahu 0÷1023.
<b>Počátek</b>	PAR	Konst	Pořadí (0÷7) byte v datové části PDO, od kterého je uložena hodnota příslušného signálu.
<b>Hodnota</b>	OUT	I	Jméno databázové proměnné (nebo prvku matice), do které se uloží hodnota ve fyzikálních jednotkách při příjmu odpovídajícího PDO.
		L	
		F	
		MI	
		ML	
		MF	
<b>Rozsah</b>	IN	Konst	Horní hranice měřicího rozsahu HW modulu v elektrických jednotkách.
		F	
		MF	
<b>EIMin</b>	IN	Konst	Dolní mez signálu v elektrických jednotkách.
		F	
		MF	
<b>EIMax</b>	IN	Konst	Horní mez signálu v elektrických jednotkách.
		F	
		MF	
<b>FyzMin</b>	IN	Konst	Dolní mez signálu ve fyzikálních jednotkách.
		F	
		MF	
<b>FyzMax</b>	IN	Konst	Horní mez signálu ve fyzikálních jednotkách.
		F	
		MF	

### Příklad

```

ProcINIT:
17001 CNC_C167      125kbit/s
17002 CAN_NMT_M     :17001
      CAN_Node      :17002, 1, 100, 5, NONE.0, NeníVirt, NMT_Full

Proc00:
      CAN_AI        :17001, 1, 1, @StavAI1, 0x0010, 0, Val1[0,0], 20, 4, 20, 0, 100
      CAN_AI        :17001, 1, 0, NONE.0, 0x0010, 2, Val2[0,0], 20, 4, 20, 0, 100

```

Do proměnných Val1 a Val2 jsou načteny hodnoty analogových signálů ve stupních Celsia (0÷100) z uzlu číslo 1 po přepočtu z proudového rozsahu 4÷20mA. Stav přenosu je ukládán do bitu @StavAI1, přenos je vyvoláván PDO-Masterem.

<b>CAN_AO</b>	Modul analogových výstupů na sběrnici CAN
---------------	---

**Popis**

Modul (PDO-Master výstupního PDO, viz oddíl “Vrstva objektů procesních dat PDO” dodatku “Obsluha sběrnice CAN”) definuje vysílání dat jednoho analogového výstupního signálu na výstupní zařízení připojené na sběrnici CAN.

Data se čtou z proměnné určené parametrem *Hodnota* po přepočtu z fyzikálního rozměru na rozsah převodníku. Přepočet probíhá na základě parametrů *Rozsah*, *ElMin*, *ElMax*, *FyzMin* a *FyzMax* analogicky k modulu **AnOut**.

**Řízení přenosu**

U výstupních PDO musí přenos dat vyvolávat PDO-Master. Toho dosáhneme dosazením jedničky za parametr *Přenést*.

Pokud se jeden PDO skládá z více výstupních signálů, a tedy se PDO skládá postupně prostřednictvím více modulů **CAN\_AO**, bylo by zbytečným přetěžováním sběrnice žádat o přenos v každém z těchto modulů. Je tedy vhodné dosazovat jedničku za *Přenést* jen v jednom z nich (zpravidla v posledním).

**Čtení dat ze vstupní proměnné**

Data se z proměnné uvedené v parametru *Hodnota* načtou v okamžiku vyvolání každé instance modulu **CAN\_AO**, ale až v okamžiku vyvolání s hodnotou 1 v parametru *Přenést*. Z toho vyplývá, že není možné používat dočasnou proměnnou, do které se data kopírují krátkodobě před předáním do modulu **CAN\_AO**. Proměnná uvedená v parametru *Hodnota* musí být trvale vyhrazena jen pro tento účel.

**Identifikátor (COB-Id) použitého PDO**

Modul používá implicitní mapování PDO dle specifikace CANopen Minimum Capability Device, tedy pracuje s PDO s identifikátorem COB-Id = 769 + Node-Id - 1.

**Parametry**

<b>Připojení</b>	PAR	Návěští	Návěští připojovacího modulu <b>CNC_...</b> , který zajišťuje fyzické připojení ke sběrnici CAN.
------------------	-----	---------	--

Toto návěští je globální (modul **CNC\_...** je v ProclNIT zatímco moduly vrstvy PDO zpravidla v periodických procesech) a je ho tedy třeba zadat ručně.

<b>Node-Id</b>	PAR	Konst	Identifikátor uzlu (běžného nebo virtuálního) sběrnice CAN, který je PDO-Slavem příslušného PDO.
----------------	-----	-------	--

<b>Přenést</b>	IN	Konst	Udává, zda se má při vyvolání tohoto modulu iniciovat přenos dat po sběrnici CAN.
		Bit	

<b>Stav</b>	OUT	Bit	Výstupní bit, indikující, že data byla úspěšně vyslána na sběrnici. Nuluje se při vyvolání přenosu dat, nastavuje se na jedničku při dokončení vysílání odpovídajícího PDO. Nechceme-li stav přenosu testovat, lze za tento parametr dosadit NONE.
-------------	-----	-----	--

<b>Parametry</b>	PAR	Výběr	Parametry převodníku.
------------------	-----	-------	-----------------------

Rozlišení	Konst	Počet datových bitů, na kterých je v PDO uložena analogová hodnota (0÷31)
-----------	-------	---

Bipolární	ANO	Hodnota v PDO se chápe jako číslo se znaménkem ve dvojkovém doplňku. Tedy např. pro Rozlišení=10 v rozsahu -512÷511.
	NE	Hodnota v PDO se chápe jako kladné číslo bez znaménkového bitu. Tedy např. pro Rozlišení=10 v rozsahu 0÷1023.

Počátek	PAR	Konst	Pořadí (0÷7) byte v datové části PDO, od kterého je uložena hodnota příslušného signálu.
---------	-----	-------	--

Hodnota	IN	I	Jméno databázové proměnné (nebo prvku matice), ze které se načte hodnota ve fyzikálních jednotkách před vysláním odpovídajícího PDO.
		L	
		F	
		MI	
		ML	
		MF	

Rozsah	IN	Konst	Horní hranice výstupního rozsahu HW modulu v elektrických jednotkách.
		F	
		MF	

ElMin	IN	Konst	Dolní mez signálu v elektrických jednotkách.
		F	
		MF	

ElMax	IN	Konst	Horní mez signálu v elektrických jednotkách.
		F	
		MF	

FyzMin	IN	Konst	Dolní mez signálu ve fyzikálních jednotkách.
		F	
		MF	

FyzMax	IN	Konst	Horní mez signálu ve fyzikálních jednotkách.
		F	
		MF	

### Příklad

```
ProcINIT:
17001 CNC_C167      125kbit/s
17002 CAN_NMT_M     :17001
      CAN_Node       :17002, 1, 100, 5, NONE.0, NeníVirt, NMT_Full
```

```
Proc00:
      CAN_AO         :17001, 1, 1, @StavAO1, 0x0010, 0, Val1[0,0], 20, 4, 20, 0, 100
      CAN_AO         :17001, 1, 0, NONE.0, 0x0010, 2, Val2[0,0], 20, 4, 20, 0, 100
```

Proměnné Val1 a Val2 jsou zapsány na analogové signály uzlu číslo 1 po přepočtu z fyzikálního rozsahu 0÷100% na proudový rozsah 4÷20mA. Stav přenosu je ukládán do bitu @StavAO1, přenos je vyvoláván PDO-Masterem.

<b>CAN_DI</b>	Modul digitálních vstupů na sběrnici CAN
---------------	--

**Popis**

Modul (PDO-Master vstupního PDO, viz oddíl “Vrstva objektů procesních dat PDO” dodatku “Obsluha sběrnice CAN”) definuje příjem dat zvoleného počtu digitálních vstupních signálů ze vstupního zařízení připojeného na sběrnici CAN.

Data se ukládají do proměnné určené parametrem *Proměnná*. Je-li za tento parametr dosazen prvek matice a parametrem *Bajtů* zvoleno více bajtů, než je velikost jedné položky této matice, ukládá se i do následujících prvků téhož řádku matice.

**Řízení přenosu**

U vstupních PDO je obvyklé, že přenos dat vyvolává PDO-Slave. U digitálních signálů zpravidla při změně hodnoty některého signálu. Přesto je možné vyžádat si “pro jistotu” jednou za čas (řádově sekundy až desítky sekund) přenos ze strany PDO-Mastera dosazením jedničky za parametr *Přenést*. V tom případě se vyšle remote-frame, na který PDO-Slave reaguje vysláním dat na sběrnici.

Pokud se data jednoho PDO ukládají do více proměnných, a tedy se přijímá prostřednictvím více modulů **CAN\_DI**, bylo by zbytečným přetěžováním sběrnice žádat o přenos v každém z těchto modulů. Je tedy vhodné dosazovat jedničku za *Přenést* jen v jednom z nich (zpravidla v prvním).

**Ukládání dat do výstupní proměnné**

Data se do proměnné uvedené v parametru *Proměnná* neukládají v okamžiku vyvolání modulu **CAN\_DI**, ale asynchronně, při příjmu odpovídajícího PDO. Z toho vyplývá následující:

- Spoléháme-li na zasílání dat z iniciativy PDO-Slavea (parametr *Přenést* má nulovou hodnotu), není nutno modul **CAN\_DI** vůbec umísťovat do periodického procesu, je možno ho vložit do ProcINIT.
- Není možné používat dočasnou proměnnou, do které se data načtou a potom někam zkopírují. Proměnná uvedená v parametru *Proměnná* musí být trvale vyhrazena jen pro tento účel.
- Chceme-li po vyvolání modulu s parametrem *Přenést* rovným jedné čekat na naplnění výstupní proměnné daty, je třeba vhodně testovat výstupní bit *Stav*.

**Identifikátor (COB-Id) použitého PDO**

Modul používá implicitní mapování PDO dle specifikace CANopen Minimum Capability Device, tedy pracuje s PDO s identifikátorem COB-Id = 385 + Node-Id - 1.

**Parametry**

<b>Připojení</b>	PAR	Návěští	Návěští připojovacího modulu <b>CNC_...</b> , který zajišťuje fyzické připojení ke sběrnici CAN.
------------------	-----	---------	--

Toto návěští je globální (modul **CNC\_...** je v ProcINIT zatímco moduly vrstvy PDO zpravidla v periodických procesech) a je ho tedy třeba zadat ručně.

<b>Node-Id</b>	PAR	Konst	Identifikátor uzlu (běžného nebo virtuálního) sběrnice CAN, který je PDO-Slavem příslušného PDO.
----------------	-----	-------	--

<b>Přenést</b>	IN	Konst Bit	Udává, zda se má při vyvolání tohoto modulu iniciovat přenos dat po sběrnici CAN.
----------------	----	--------------	---

<b>Stav</b>	OUT	Bit	Výstupní bit, indikující, že byla přijata data a proměnná <i>Hodnota</i> byla naplněna. Nuluje se při vyvolání přenosu dat, nastavuje se na jedničku při příjmu odpovídajícího PDO (ať už na vyžádání parametrem <i>Přenést</i> , nebo z iniciativy PDO-Slavea). Nechceme-li stav přenosu testovat, lze za tento parametr dosadit NONE.
-------------	-----	-----	--

<b>Bajtů</b>	PAR	Konst	Počet byte dat, které se mají uložit do proměnné předané za parametr <i>Proměnná</i> .
--------------	-----	-------	--

<b>Počátek</b>	PAR	Konst	Pořadí (0÷7) byte v datové části PDO, od kterého jsou uloženy hodnoty načítaných signálů.
----------------	-----	-------	---

<b>Proměnná</b>	OUT	I	Jméno databázové proměnné (nebo prvku matice), do které se uloží hodnoty signálů při příjmu odpovídajícího PDO.
		L	
		F	
		MI	
		ML	
		MF	

### Příklad

```

ProcINIT:
17001 CNC_C167      125kbit/s
17002 CAN_NMT_M     :17001
      CAN_Node      :17002, 1, 100, 5, NONE.0, NeníVirt, NMT_Full
      CAN_Node      :17002, 2, 100, 5, NONE.0, 1, NMT_Full
      CAN_DI         :17001, 1, 0, @StavDI1, 8, 0, Val1[0,0]
      CAN_DI         :17001, 2, 0, NONE.0, 2, 0, Val2[0,0]

```

Do 4 prvků matice *Val1* (typu MI) jsou načítány hodnoty z prvních osmi digitálních vstupních modulů na zařízení ADC\_CAN s Node-Idem 1 (prvních 5 DIP přepínačů je OFF). Data devátého a desátého digitálního vstupního modulu jsou načítána do proměnné *Val2* (typu I). Stav přenosu je ukládán do bitu *@StavDI1*, přenos je vyvoláván PDO-Slavem.

<b>CAN_DO</b>	Modul digitálních výstupů na sběrnici CAN
---------------	---

**Popis**

Modul (PDO-Master výstupního PDO, viz oddíl “Vrstva objektů procesních dat PDO” dodatku “Obsluha sběrnice CAN”) definuje vysílání dat zvoleného počtu digitálních výstupních signálů na výstupní zařízení připojené na sběrnici CAN.

Data se načítají z proměnné určené parametrem *Proměnná*. Je-li za tento parametr dosazen prvek matice a parametrem *Bajtů* zvoleno více bajtů, než je velikost jedné položky této matice, načítá se i z následujících prvků téhož řádku matice.

**Řízení přenosu**

U výstupních PDO musí přenos dat vyvolávat PDO-Master. Toho dosáhneme dosazením jedničky za parametr *Přenést*.

Pokud se jeden PDO skládá z více proměnných, a tedy se PDO skládá postupně prostřednictvím více modulů **CAN\_DO**, bylo by zbytečným přetěžováním sběrnice žádat o přenos v každém z těchto modulů. Je tedy vhodné dosazovat jedničku za *Přenést* jen v jednom z nich (zpravidla v posledním).

**Čtení dat ze vstupní proměnné**

Data se z proměnné uvedené v parametru *Proměnná* načtou v okamžiku vyvolání každé instance modulu **CAN\_DO**, ale až v okamžiku vyvolání s hodnotou 1 v parametru *Přenést*. Z toho vyplývá, že není možné používat dočasnou proměnnou, do které se data kopírují krátkodobě před předáním do modulu **CAN\_DO**. Proměnná uvedená v parametru *Proměnná* musí být trvale vyhrazena jen pro tento účel.

**Identifikátor (COB-Id) použitého PDO**

Modul používá implicitní mapování PDO dle specifikace CANopen Minimum Capability Device, tedy pracuje s PDO s identifikátorem COB-Id = 513 + Node-Id - 1.

**Parametry**

<b>Připojení</b>	PAR	Návěští	Návěští připojovacího modulu <b>CNC_...</b> , který zajišťuje fyzické připojení ke sběrnici CAN.
Toto návěští je globální (modul <b>CNC_...</b> je v ProclNIT zatímco moduly vrstvy PDO zpravidla v periodických procesech) a je ho tedy třeba zadat ručně.			
<b>Node-Id</b>	PAR	Konst	Identifikátor uzlu (běžného nebo virtuálního) sběrnice CAN, který je PDO-Slavem příslušného PDO.
<b>Přenést</b>	IN	Konst Bit	Udává, zda se má při vyvolání tohoto modulu iniciovat přenos dat po sběrnici CAN.
<b>Stav</b>	OUT	Bit	Výstupní bit, indikující, že data byla úspěšně vyslána na sběrnici. Nuluje se při vyvolání přenosu dat, nastavuje se na jedničku při dokončení vysílání odpovídajícího PDO. Nechceme-li stav přenosu testovat, lze za tento parametr dosadit NONE.
<b>Bajtů</b>	PAR	Konst	Počet byte dat, které se mají uložit do PDO.
<b>Počátek</b>	PAR	Konst	Pořadí (0÷7) byte v datové části PDO, od kterého se mají uložit hodnoty zapisovaných signálů.

Proměnná	IN	I	Jméno databázové proměnné (nebo prvku matice), ze které se načtou hodnoty signálů před vysláním odpovídajícího PDO.
		L	
		F	
		MI	
		ML	
		MF	

### Příklad

```

ProcINIT:
17001 CNC_C167      125kbit/s
17002 CAN_NMT_M     :17001
      CAN_Node      :17002, 1, 100, 5, NONE.0, NeníVirt, NMT_Full
      CAN_Node      :17002, 2, 100, 5, NONE.0, 1, NMT_Full

Proc00:
      CAN_DO         :17001, 1, 0, @StavDO1, 8, 0, Val1[0,0]
      CAN_DO         :17001, 2, 0, NONE.0, 2, 0, Val2[0,0]

```

Data ze 4 prvků matice Val1 (typu MI) jsou zapsána na prvních osm digitálních výstupních modulů na zařízení ADC\_CAN s Node-Idem 1 (prvních 5 DIP přepínačů je OFF). Na devátý a desátý digitálního výstupního modul jsou zapsána data z proměnné Val2 (typu I). Stav přenosu je ukládán do bitu @StavDO1, přenos je vyvoláván PDO-Masterem.

<b>CAN_NMT_M</b>	NMT-Master sběrnice CAN
------------------	-------------------------

**Popis**

Modul zajišťuje funkci NMT-Mastera sběrnice CAN dle standardu CANopen. Blíže k funkci vrstvy NMT viz oddíl "Vrstva správy sítě NMT" dodatku "Obsluha sběrnice CAN".

**Umístění modulu**

Modul musí být umístěn v procesu ProcINIT, a to kdekoli za (pod) modulem zajišťujícím fyzické připojení ke sběrnici CAN (modul **CNC\_...**), na který se odkazuje pomocí návěští.

**Definice seznamu uzlů**

Modul musí být následován potřebným počtem modulů **CAN\_Node**, které se na něj odkazují pomocí návěští, a které postupně definují všechny uzly sběrnice, které má NMT-Master inicializovat (jsou-li nějaké).

**Parametry**

Připojení	PAR	Návěští	Návěští připojovacího modulu <b>CNC_...</b> , který zajišťuje fyzické připojení ke sběrnici CAN.
-----------	-----	---------	--

Toto návěští je nutno zadat ručně, protože v řídicím systému schopném vložení více řadičů sběrnice CAN může být více modulů **CNC\_...** a **CAN\_NMT...**, vazby mezi nimi nelze automaticky určit.

**Příklad**

```
ProcInit:
17001 CNC_C167      125kbit/s
17002 CAN_NMT_M    :17001
```

Na stanici vybavené integrovaným řadičem procesoru C167 je naparametrizován NMT-Master sběrnice CAN, nejsou naparametrizovány žádné uzly (Parametrizace uzlů viz modul **CAN\_Node**).



<b>CAN_NMT_N</b>	Modul nulové obsluhy NMT-vrstvy sběrnice CAN
------------------	--

**Popis**

Modul je určen k použití na místě modulů **CAN\_NMT\_M** nebo **CAN\_NMT\_S** v případě systémů, které nemají vzhledem k vrstvě NMT vykonávat ani funkci NMT-Mastera, ani funkci NMT-Slavea sběrnice CAN dle standardu CANopen.

Modul je určen:

- 1) pro připojení ke sběrnicím, kde je z důvodu připojení zařízení třetích stran (a správné detekce třídy sítě těmito zařízeními) důležité, aby řídicí systém nevysílal na sběrnici žádné komunikační objekty, užívané vrstvou NMT,
- 2) zejména pak pro propojení se zařízeními neodpovídajícími standardu CANopen, která využívají k přenosu dat identifikátory objektů rezervované pro vrstvu NMT.

Modul je komunikačně zcela pasivní, nevysílá žádné komunikační objekty. Na rozdíl od modulů **CAN\_NMT\_M** a **CAN\_NMT\_S** umožňuje přijímat pomocí modulů **CAN\_PDO** i komunikační objekty s identifikátory vyhrazenými pro vrstvu NMT (0, 129÷255, 2025, 2026, 1793÷1919).

Modul slouží pouze k tomu, aby poskytl ostatním modulům knihovny CAN některé nezbytné funkce, kvůli kterým je přítomnost některého z modulů **CAN\_NMT\_x** v každé aplikaci připojené ke sběrnici CAN vyžadována.

Blíže k funkci vrstvy NMT viz oddíl "Vrstva správy sítě NMT" dodatku "Obsluha sběrnice CAN".

**Umístění modulu**

Modul musí být umístěn v procesu ProcINIT, a to kdekoli za (pod) modulem zajišťujícím fyzické připojení ke sběrnici CAN (modul **CNC\_...**), na který se odkazuje pomocí návěští.

**Definice seznamu uzlů**

Případné moduly **CAN\_Node**, které by se na tento modul odkazovaly pomocí návěští, jsou tiše (tj. bez hlášení jakékoliv chyby) ignorovány.

**Parametry**

Připojení	PAR	Návěští	Návěští připojovacího modulu <b>CNC_...</b> , který zajišťuje fyzické připojení ke sběrnici CAN.
-----------	-----	---------	--

Toto návěští je nutno zadat ručně, protože v řídicím systému schopném vložení více řadičů sběrnice CAN může být více modulů **CNC\_...** a **CAN\_NMT...**, vazby mezi nimi nelze automaticky určit.

**Příklad**

```
ProcInit:
17001 CNC_C167      125kbit/s
17002 CAN_NMT_N    :17001
```

Na stanici vybavené integrovaným řadičem procesoru C167 je naparametrizována nulová obsluha NMT-vrstvy sběrnice CAN.

<b>CAN_NMT_S</b>	NMT-Slave sběrnice CAN
------------------	------------------------

**Popis**

Modul zajišťuje funkci NMT-Slavea sběrnice CAN dle standardu CANopen. Blíže k funkci vrstvy NMT viz oddíl "Vrstva správy sítě NMT" dodatku "Obsluha sběrnice CAN".

**Umístění modulu**

Modul musí být umístěn v procesu ProcINIT, a to kdekoli za (pod) modulem zajišťujícím fyzické připojení ke sběrnici CAN (modul **CNC\_...**), na který se odkazuje pomocí návěští.

**Definice seznamu uzlů**

Modul musí být následován potřebným počtem modulů **CAN\_Node**, které se na něj odkazují pomocí návěští, a které postupně definují všechny CANopen kompatibilní uzly sběrnice obsahující PDO-slavey, se kterými má tento uzel komunikovat (jsou-li nějaké).

**Parametry**

<b>Připojení</b>	PAR	Návěští	Návěští připojovacího modulu <b>CNC_...</b> , který zajišťuje fyzické připojení ke sběrnici CAN.
------------------	-----	---------	--

Toto návěští je nutno zadat ručně, protože v řídicím systému schopném vložení více řadičů sběrnice CAN může být více modulů **CNC\_...** a **CAN\_NMT...**, vazby mezi nimi nelze automaticky určit.

<b>Node-Id</b>	PAR	Konst	Identifikátor tohoto uzlu sběrnice CAN
----------------	-----	-------	--

<b>Guard-Time</b>	PAR	Konst	Navrhovaná perioda Node-Guardingu v milisekundách. Blíže viz oddíl "Vrstva správy sítě NMT" dodatku "Obsluha sběrnice CAN". NMT-Master nemusí tuto navrhovanou dobu respektovat a může uzlu přidělit periodu podle své vlastní parametrizace. NMT-Master realizovaný modulem <b>CAN_NMT_M</b> nebere navrhovaný Guard-Time v úvahu.
-------------------	-----	-------	--

<b>LTFactor</b>	PAR	Konst	Navrhovaný LifeTime-Factor (násobek periody Node-Guardingu). Blíže viz oddíl "Vrstva správy sítě NMT" dodatku "Obsluha sběrnice CAN". NMT-Master nemusí tuto navrhovanou hodnotu respektovat a může uzlu přidělit násobek podle své vlastní parametrizace. NMT-Master realizovaný modulem <b>CAN_NMT_M</b> nebere navrhovaný LifeTime-Factor v úvahu.
-----------------	-----	-------	--

<b>Chyba</b>	OUT	Bit	Výstupní bit, indikující, že došlo ke ztrátě spojení s NMT-Masterem (případně že dosud nebylo navázáno). Aplikace by měla na jedničkovou hodnotu tohoto bitu reagovat přechodem do tzv. bezpečného stavu. Způsob realizace tohoto bezpečného stavu je na autorovi aplikace. Nechceme-li stav spojení s NMT-Masterem testovat, lze za tento parametr dosadit NONE.
--------------	-----	-----	---

MCD	PAR	Výběr	Udává úroveň implementace vrstvy NMT.	
			Hodnota	Význam
			0	NMT_Full - úplná implementace vrstvy NMT
			1	NMT_MCD - zjednodušená implementace na úrovni specifikace CANopen Minimum Capability Device. Tato úroveň implementace je ochuzena o některé důležité rysy.
Použitá úroveň se musí přizpůsobit možností NMT-Mastera sběrnice. Je-li NMT-Master realizován modulem <b>CAN_NMT_M</b> , měla by se vždy volit úroveň NMT_Full.				

**Příklad**

```
ProcInit:  
17001 CNC_C167      125kbit/s  
17002 CAN_NMT_S    :17001, 1, 500, 4, @Lost, NMT_Full
```

Na stanici vybavené integrovaným řadičem procesoru C167 je naparametrizován NMT-Slave sběrnice CAN jako uzel s Node-Idem 1, navrhovaný Guard-Time je 500ms, navrhovaný LifeTime-Factor je 4, při ztrátě spojení s NMT-Masterem se nastavuje bit @Lost, je zvolena úplná implementace vrstvy NMT. Nejsou naparametrizovány žádné uzly (Parametrizace uzlů viz modul **CAN\_Node**).

<b>CAN_Node</b>	Definice uzlu pro CAN_NMT_M/CAN_NMT_S
-----------------	---------------------------------------

**Popis**

Modul slouží k přidání uzlu do seznamu uzlů, které má obsluhovat modul vrstvy NMT, na který se modul **CAN\_Node** odkazuje návěští.

V případě NMT-Mastera seznam uzlů slouží k řízení jejich inicializace a detekce stavu, v případě NMT-Slavea pouze k řízení detekce stavu.

**Umístění modulu**

Modul musí být umístěn v procesu ProcINIT, a to kdekoli za (pod) modulem vrstvy NMT (**CAN\_NMT...**), na který se odkazuje pomocí návěští, a před (nad) případnými moduly vrstvy PDO, pracujícími s daným uzlem.

**Parametry**

<b>NMT</b>	PAR	Návěští	Návěští modulu vrstvy NMT ( <b>CAN_NMT...</b> ), do jehož seznamu uzlů se má nový uzel přidat.
------------	-----	---------	--

Toto návěští je nutno zadat ručně, protože v řídicím systému schopném vložení více řadičů sběrnice CAN může být více modulů **CAN\_NMT...**, vazbu na správný modul vrstvy NMT tedy nelze automaticky určit.

<b>Node-Id</b>	PAR	Konst	Identifikátor definovaného uzlu sběrnice CAN
----------------	-----	-------	--

<b>Guard-Time</b>	PAR	Konst	Určená perioda Node-Guardingu v milisekundách. Blíže viz oddíl "Vrstva správy sítě NMT" dodatku "Obsluha sběrnice CAN".
-------------------	-----	-------	---

<b>LTFactor</b>	PAR	Konst	Určený LifeTime-Factor (násobek periody Node-Guardingu). Blíže viz oddíl "Vrstva správy sítě NMT" dodatku "Obsluha sběrnice CAN".
-----------------	-----	-------	---

<b>Chyba</b>	OUT	Bit	Výstupní bit, indikující, že došlo ke ztrátě spojení s uzlem (případně že dosud nebylo navázáno). Nechceme-li stav spojení s uzlem testovat, lze za tento parametr dosadit <b>NONE</b> .
--------------	-----	-----	---

<b>Virtuální</b>	PAR	Konst	Pro běžné uzly sběrnice CAN ponechte implicitní nulovou hodnotu (zobrazí se jako <b>NeníVirt</b> ). Pro virtuální uzly zadejte <b>Node-Id</b> běžného uzlu, se kterým je tento virtuální uzel svázán. Blíže viz oddíl "Virtuální uzly" dodatku "Obsluha sběrnice CAN".
------------------	-----	-------	---

MCD	PAR	Výběr	Udává úroveň implementace vrstvy NMT definovaným uzlem.	
			Hodnota	Význam
			0	NMT_Full - úplná implementace vrstvy NMT
			1	NMT_MCD - zjednodušená implementace na úrovni specifikace CANopen Minimum Capability Device. Tato úroveň implementace je ochuzena o některé důležité rysy.

Použitá úroveň musí odpovídat skutečným schopnostem NMT-Slavea. Je-li NMT-Slavem zařízení ADC\_CAN, zvolte NMT\_Full. Je-li NMT-Slavem řídicí systém s modulem **CAN\_NMT\_S**, zvolte na obou stranách NMT\_Full. U zařízení jiných výrobců je třeba úroveň implementace vrstvy NMT najít v jejich dokumentaci. V době tvorby tohoto textu např. zařízení firmy WAGO implementují NMT na úrovni NMT\_MCD, zařízení firmy Weidmüller na úrovni NMT\_Full.

### Příklad

```

ProcInit:
17001 CNC_C167      125kbit/s
17002 CAN_NMT_M      :17001
      CAN_Node       :17002, 1, 100, 5, NONE.0, NeníVirt, NMT_Full
      CAN_Node       :17002, 3, 100, 5, NONE.0, NeníVirt, NMT_Full
      CAN_Node       :17002, 4, 100, 5, NONE.0, 3, NMT_Full

```

Na stanici vybavené integrovaným řadičem procesoru C167 je naparametrizován NMT-Master sběrnice CAN. Ten má naparametrizovánu obsluhu dvou zařízení typu ADC\_CAN, první s Node-Idem 1 (nemá více než 8 modulů stejného typu), druhý s Node-Idem 3 (má více než 8 modulů stejného typu, proto se na něm vytváří ještě virtuální uzel s Node-Idem 4). Je zaručeno, že obě zařízení detekují ztrátu spojení s NMT-Masterem do 0.5 sekundy.

**CAN\_PDO**

Obecný procesní datový objekt sběrnice CAN

**Popis**

Obecný modul vrstvy PDO sběrnice CAN (viz oddíl "Vrstva objektů procesních dat PDO" dodatku "Obsluha sběrnice CAN"). Používá se zejména pro komunikaci se zařízeními, která nepoužívají implicitní mapování PDO dle specifikace CANopen Minimum Capability Device.

Parametr *Směr* a způsob používání parametru *Přenést* určuje, zda se jedná o PDO-Mastera či PDO-Slavea vstupního či výstupního PDO. COB-Id použitého PDO lze libovolně zvolit parametrem modulu:

- PDO-Master vstupního PDO: *Směr* je *Příjem*, parametr *Přenést* se může nebo nemusí používat.
- PDO-Master výstupního PDO: *Směr* je *Vysílání*, alespoň jedna instance modulu pro každý COB-Id musí mít nastaven parametr *Přenést*.
- PDO-Slave vstupního PDO: *Směr* je *Odpovídač*, parametr *Přenést* se může nebo nemusí používat.
- PDO-Slave výstupního PDO: *Směr* je *Příjem*, parametr *Přenést* je zpravidla nulový.

Data se při režimech *Vysílání* nebo *Odpovídač* načítají z proměnné určené parametrem *Proměnná*. Je-li za tento parametr dosazen prvek matice a parametrem *Bajtů* zvoleno více bajtů, než je velikost jedné položky této matice, načítá se i z následujících prvků téhož řádku matice.

Při režimu *Příjem* se data do proměnné určené parametrem *Proměnná* obdobným způsobem naopak ukládají.

**Řízení přenosu**

U výstupních PDO musí přenos dat vyvolávat PDO-Master. Toho dosáhneme dosazením jedničky za parametr *Přenést* v příslušné instanci modulu **CAN\_PDO**. PDO-Slave výstupního PDO sice může nastavit parametr *Přenést*, čímž se vyšle remote-frame, žádající o data, ale v praxi je velmi nepravděpodobné setkat se s PDO-Masterem výstupního PDO, který by byl připraven na tento remote-frame reagovat.

U vstupních PDO může přenos dat vyvolávat jak PDO-Slave (například při změně dat digitálního signálu, periodicky u analogových signálů), tak PDO-Master. V obou případech toho dosáhneme dosazením jedničky za parametr *Přenést* v příslušné instanci modulu **CAN\_PDO**.

Pokud se jeden PDO skládá z více proměnných, a tedy se PDO skládá postupně prostřednictvím více modulů **CAN\_PDO**, bylo by zbytečným přetěžováním sběrnice žádat o přenos v každém z těchto modulů. Je tedy vhodné dosazovat jedničku za *Přenést* jen v jednom z nich (zpravidla v posledním při režimu *Vysílání* nebo *Odpovídač*, zpravidla v prvním při režimu *Příjem*).

**Čtení/Ukládání dat z/do proměnné**

Data se z proměnné uvedené v parametru *Proměnná* načtou vždy v okamžiku vyvolání každé instance modulu **CAN\_PDO**, ale až v okamžiku:

- V režimu *Vysílání* při vyvolání s hodnotou 1 v parametru *Přenést*.
- V režimu *Příjem* při příjmu odpovídajícího PDO
- V režimu *Odpovídač* při každém vyvolání modulu **CAN\_PDO**.

Z toho vyplývá následující:

- Společně-li u PDO-Mastera vstupního PDO na zasílání dat z iniciativy PDO-Slavea (parametr *Přenést* má nulovou hodnotu), není nutno modul **CAN\_PDO** vůbec umísťovat do periodického procesu, je možno ho vložit do ProcINIT.
- Kromě režimu *Odpovídač* není možné používat dočasnou proměnnou pro hodnoty čtené z / zapisované do PDO. Proměnná uvedená v parametru *Hodnota* musí být trvale vyhrazena jen pro tento účel.
- Chceme-li při režimu *Příjem* po vyvolání modulu s parametrem *Přenést* rovným jedné čekat na naplnění výstupní proměnné daty, je třeba vhodně testovat výstupní bit *Stav*.

Identifikátor (COB-Id)  
použitého PDO

Modul nepoužívá žádné implicitní mapování PDO, COB-Id lze volit parametrem modulu nezávisle na Node-Id.

### Parametry

<b>Připojení</b>	PAR	Návěští	Návěští připojovacího modulu <b>CNC_...</b> , který zajišťuje fyzické připojení ke sběrnici CAN.
------------------	-----	---------	--

Toto návěští je globální (modul **CNC\_...** je v ProcINIT zatímco moduly vrstvy PDO zpravidla v periodických procesech) a je ho tedy třeba zadat ručně.

<b>Node-Id</b>	PAR	Konst	Identifikátor uzlu (běžného nebo virtuálního) sběrnice CAN, který je PDO-Slavem příslušného PDO.
----------------	-----	-------	--

<b>COB-Id</b>	PAR	Konst	Identifikátor komunikačního objektu (PDO) sběrnice CAN.
---------------	-----	-------	---

<b>Směr</b>	PAR	Výběr	Udává režim přenosu.	
			Hodnota	Význam
			0	Příjem - používá se pro PDO-Mastera vstupního objektu nebo PDO-Slavea výstupního objektu.
			1	Vysílání - používá se pro PDO-Mastera výstupního objektu.
			2	Odpovídač - používá se pro PDO-Slavea vstupního objektu.

<b>Přenést</b>	IN	Konst	Udává, zda se má při vyvolání tohoto modulu iniciovat přenos dat po sběrnici CAN.
		Bit	

<b>Stav</b>	OUT	Bit	Výstupní bit, indikující, že data byla úspěšně vyslána na sběrnici. Nuluje se při vyvolání přenosu dat, nastavuje se na jedničku při dokončení vysílání/příjmu odpovídajícího PDO. Nechceme-li stav přenosu testovat, lze za tento parametr dosadit NONE.
-------------	-----	-----	---

<b>Bajtů</b>	PAR	Konst	Počet byte dat v PDO, se kterými modul pracuje.
--------------	-----	-------	---

<b>Počátek</b>	PAR	Konst	Pořadí (0÷7) byte v datové části PDO, od kterého začíná oblast dat v PDO, se kterou modul pracuje.
----------------	-----	-------	--

<b>Proměnná</b>	IN/OUT	I	Jméno databázové proměnné (nebo prvku matice), ze které se načtou data před vysláním odpovídajícího PDO (Směr=Vysílání nebo Odpovídač), nebo do které se data uloží (Směr=Příjem)
		L	
		F	
		MI	
		ML	
		MF	

### Příklad

PDO-Master jednoho vstupního a jednoho výstupního PDO:

```
ProcINIT:
17001 CNC_C167      125kbit/s
17002 CAN_NMT_M     :17001
      CAN_Node      :17002, 1, 100, 5, NONE.0, NeníVirt, NMT_Full

Proc00:
```

```
CAN_PDO      :17001, 1, 1000, Příjem, 0, @StavIn, 2, 0, Val1[0,0]  
CAN_PDO      :17001, 1, 1001, Vysílání, 1, @StavOut, 2, 0, Val2[0,0]
```

**Odpovídající PDO-Slave:**

ProcINIT:

```
17001 CNC_C167      125kbit/s  
17002 CAN_NMT_S     :17001
```

Proc00:

```
CAN_PDO      :17001, 1, 1000, Odpovídač, @Vyslat, @StavIn, 2, 0, Val1[0,0]  
CAN_PDO      :17001, 1, 1001, Příjem, 0, @StavOut, 2, 0, Val2[0,0]
```

PDO-slave s Node-Idem 1 vlastní jeden vstupní a jeden výstupní PDO s COB-Idy 1000 a 1001. Při změně hodnoty Val1 naství bit @Vyslat, čímž vyvolá přenos PDO 1000 PDO-Masterovi, kde se tato hodnota uloží do stejnojmenné proměnné. PDO-Master periodicky zasílá hodnotu Val2 do výstupního PDO 1001 PDO-Slaveovi, kde se tato hodnota uloží do stejnojmenné proměnné. Stavby přenosu obou PDO se na obou uzlech ukládají do bitů @StavIn a @StavOut.



<b>CAN_S_AI</b>	Slave-modul an. vstupů na sběrnici CAN
-----------------	--

**Popis**

Modul (PDO-Slave vstupního PDO, viz oddíl “Vrstva objektů procesních dat PDO” dodatku “Obsluha sběrnice CAN”) definuje vysílání dat jednoho analogového vstupního signálu ze vstupního zařízení připojeného na sběrnici CAN.

Data se čtou z proměnné určené parametrem *Hodnota* po přepočtu z fyzikálního rozměru na rozsah převodníku. Přepočet probíhá na základě parametrů *Rozsah*, *ElMin*, *ElMax*, *FyzMin* a *FyzMax* analogicky k modulu **AnOut**.

**Řízení přenosu**

U vstupních PDO je obvyklé, že přenos dat vyvolává PDO-Slave. U analogových signálů zpravidla v pravidelných intervalech. Toho dosáhneme dosazením jedničky za parametr *Přenést*.

Pokud se jeden PDO skládá z více vstupních signálů, a tedy se přijímá prostřednictvím více modulů **CAN\_S\_AI**, bylo by zbytečným přetěžováním sběrnice žádat o přenos v každém z těchto modulů. Je tedy vhodné dosazovat jedničku za *Přenést* jen v jednom z nich (zpravidla v posledním).

**Čtení dat ze vstupní proměnné**

Data se z proměnné uvedené v parametru *Hodnota* čtou v okamžiku vyvolání každé instance modulu **CAN\_S\_AI**.

**Identifikátor (COB-Id) použitého PDO**

Modul používá implicitní mapování PDO dle specifikace CANopen Minimum Capability Device, tedy pracuje s PDO s identifikátorem COB-Id = 641 + Node-Id - 1.

**Parametry**

<b>Připojení</b>	PAR	Návěští	Návěští připojovacího modulu <b>CNC_...</b> , který zajišťuje fyzické připojení ke sběrnici CAN.
------------------	-----	---------	--

Toto návěští je globální (modul **CNC\_...** je v ProclNIT zatímco moduly vrstvy PDO zpravidla v periodických procesech) a je ho tedy třeba zadat ručně.

<b>Node-Id</b>	PAR	Konst	Identifikátor uzlu (běžného nebo virtuálního) sběrnice CAN, který je PDO-Slavem příslušného PDO. Zpravidla je roven Node-Idu uzlu, na němž je modul použit, pokud nepoužíváme virtuální či nenakonfigurované uzly (viz dodatek “Obsluha sběrnice CAN”)
----------------	-----	-------	--

<b>Přenést</b>	IN	Konst	Udává, zda se má při vyvolání tohoto modulu iniciovat přenos dat po sběrnici CAN.
		Bit	

<b>Stav</b>	OUT	Bit	Výstupní bit, indikující, že data byla úspěšně vyslána na sběrnici. Nuluje se při vyvolání přenosu dat, nastavuje se na jedničku při dokončení vysílání odpovídajícího PDO, ať už v důsledku nastavení bitu <i>Přenést</i> , nebo na žádost PDO-Mastera. Nechceme-li stav přenosu testovat, lze za tento parametr dosadit <i>NONE</i> .
-------------	-----	-----	---

<b>Parametry</b>	PAR	Výběr	Parametry převodníku.
------------------	-----	-------	-----------------------

<b>Rozlišení</b>	Konst	Počet datových bitů, na kterých je v PDO uložena analogová hodnota (0÷31)
------------------	-------	---

Bipolární	ANO	Hodnota v PDO se chápe jako číslo se znaménkem ve dvojkovém doplňku. Tedy např. pro Rozlišení=10 v rozsahu -512÷511.
	NE	Hodnota v PDO se chápe jako kladné číslo bez znaménkového bitu. Tedy např. pro Rozlišení=10 v rozsahu 0÷1023.

Počátek	PAR	Konst	Pořadí (0÷7) byte v datové části PDO, od kterého je uložena hodnota příslušného signálu.
---------	-----	-------	--

Hodnota	IN	I	Jméno databázové proměnné (nebo prvku matice), ze které se načte hodnota ve fyzikálních jednotkách před vysláním odpovídajícího PDO.
		L	
		F	
		MI	
		ML	
		MF	

Rozsah	IN	Konst	Horní hranice výstupního rozsahu HW modulu v elektrických jednotkách.
		F	
		MF	

ElMin	IN	Konst	Dolní mez signálu v elektrických jednotkách.
		F	
		MF	

ElMax	IN	Konst	Horní mez signálu v elektrických jednotkách.
		F	
		MF	

FyzMin	IN	Konst	Dolní mez signálu ve fyzikálních jednotkách.
		F	
		MF	

FyzMax	IN	Konst	Horní mez signálu ve fyzikálních jednotkách.
		F	
		MF	

### Příklad

```
ProcINIT:
17001 CNC_C167      125kbit/s
17002 CAN_NMT_S     :17001, 1, 500, 4, @Lost, NMT_Full
```

```
Proc00:
CAN_S_AI      :17001, 1, 0, NONE.0, 0x0010, 0, Val1[0,0], 20, 4, 20, 0, 100
CAN_S_AI      :17001, 1, 1, @StavAI1, 0x0010, 2, Val2[0,0], 20, 4, 20, 0, 100
```

Proměnné Val1 a Val2 jsou periodicky vysílány PDO-Masterovi jako analogové signály uzlu číslo 1 po přepočtu z fyzikálního rozsahu 0÷100% na proudový rozsah 4÷20mA. Stav přenosu je ukládán do bitu @StavAI1, přenos je vyvoláván PDO-Slavem.

<b>CAN_S_AO</b>	Slave-modul an. výstupů na sběrnici CAN
-----------------	---

**Popis**

Modul (PDO-Slave výstupního PDO, viz oddíl “Vrstva objektů procesních dat PDO” dodatku “Obsluha sběrnice CAN”) definuje příjem dat jednoho analogového výstupního signálu výstupním zařízením připojeným na sběrnici CAN.

Data se ukládají do proměnné určené parametrem *Hodnota* po přepočtu na fyzikální rozměr. Přepočet probíhá na základě parametrů *Rozsah*, *ElMin*, *ElMax*, *FyzMin* a *FyzMax* analogicky k modulu **AnIn**.

**Řízení přenosu**

U výstupních PDO musí přenos dat vyvolávat PDO-Master. PDO-Slave výstupního PDO sice může nastavit parametr *Přenést*, čímž se vyšle remote-frame, žádající o data, ale v praxi je velmi nepravděpodobné setkat se s PDO-Masterem výstupního PDO, který by byl připraven na tento remote-frame reagovat.

**Ukládání dat do výstupní proměnné**

Data se do proměnné uvedené v parametru *Hodnota* neukládají v okamžiku vyvolání modulu **CAN\_S\_AO**, ale asynchronně, při příjmu odpovídajícího PDO. Z toho vyplývá následující:

- Modul **CAN\_S\_AO** není nutné vůbec umísťovat do periodického procesu, je možno ho vložit do ProclNIT.
- Není možné používat dočasnou proměnnou, do které se data načtou a potom někde zkopírují. Proměnná uvedená v parametru *Hodnota* musí být trvale vyhrazena jen pro tento účel.
- Chceme-li čekat na naplnění výstupní proměnné daty, je třeba vhodně testovat výstupní bit *Stav*. Je ovšem rovněž nutné ho ve vhodném okamžiku nulovat, protože parametr *Přenést* se z výše uvedených důvodů obvykle nepoužívá, tudíž modul **CAN\_S\_AO** tento bit nikdy nenuluje.

**Identifikátor (COB-Id) použitého PDO**

Modul používá implicitní mapování PDO dle specifikace CANopen Minimum Capability Device, tedy pracuje s PDO s identifikátorem COB-Id = 769 + Node-Id - 1.

**Parametry**

<b>Připojení</b>	PAR	Návěští	Návěští připojovacího modulu <b>CNC_...</b> , který zajišťuje fyzické připojení ke sběrnici CAN.
------------------	-----	---------	--

Toto návěští je globální (modul **CNC\_...** je v ProclNIT zatímco moduly vrstvy PDO zpravidla v periodických procesech) a je ho tedy třeba zadat ručně.

<b>Node-Id</b>	PAR	Konst	Identifikátor uzlu (běžného nebo virtuálního) sběrnice CAN, který je PDO-Slavem příslušného PDO. Zpravidla je roven Node-Idu uzlu, na němž je modul použit, pokud nepoužíváme virtuální či nenakonfigurované uzly (viz dodatek “Obsluha sběrnice CAN”).
----------------	-----	-------	---

<b>Přenést</b>	IN	Konst Bit	Udává, zda se má při vyvolání tohoto modulu iniciovat přenos dat po sběrnici CAN. Viz upozornění v oddílu “Řízení přenosu” výše.
----------------	----	--------------	--

<b>Stav</b>	OUT	Bit	Výstupní bit, indikující, že byla přijata data a proměnná <i>Hodnota</i> byla naplněna. Nuluje se při vyvolání přenosu dat (ale viz upozornění v oddílu “Řízení přenosu” výše), nastavuje se na jedničku při příjmu odpovídajícího PDO (ať už na vyžádání parametrem <i>Přenést</i> , nebo z iniciativy PDO-Mastera). Nechceme-li stav přenosu testovat, lze za tento parametr dosadit NONE.
-------------	-----	-----	---

<b>Parametry</b>	PAR	Výběr	Parametry převodníku.
	Rozlišení	Konst	Počet datových bitů, na kterých je v PDO uložena analogová hodnota (0÷31)
	Bipolární	ANO	Hodnota v PDO se chápe jako číslo se znaménkem ve dvojkovém doplňku. Tedy např. pro Rozlišení=10 v rozsahu -512÷511.
		NE	Hodnota v PDO se chápe jako kladné číslo bez znaménkového bitu. Tedy např. pro Rozlišení=10 v rozsahu 0÷1023.
<b>Počátek</b>	PAR	Konst	Pořadí (0÷7) byte v datové části PDO, od kterého je uložena hodnota příslušného signálu.
<b>Hodnota</b>	OUT	I	Jméno databázové proměnné (nebo prvku matice), do které se uloží hodnota ve fyzikálních jednotkách při příjmu odpovídajícího PDO.
		L	
		F	
		MI	
		ML	
		MF	
<b>Rozsah</b>	IN	Konst	Horní hranice měřicího rozsahu HW modulu v elektrických jednotkách.
		F	
		MF	
<b>EIMin</b>	IN	Konst	Dolní mez signálu v elektrických jednotkách.
		F	
		MF	
<b>EIMax</b>	IN	Konst	Horní mez signálu v elektrických jednotkách.
		F	
		MF	
<b>FyzMin</b>	IN	Konst	Dolní mez signálu ve fyzikálních jednotkách.
		F	
		MF	
<b>FyzMax</b>	IN	Konst	Horní mez signálu ve fyzikálních jednotkách.
		F	
		MF	

### Příklad

```

ProcINIT:
17001 CNC_C167      125kbit/s
17002 CAN_NMT_S     :17001, 1, 500, 4, @Lost, NMT_Full
      CAN_S_AO       :17001, 1, 0, @StavAO1, 0x0010, 0, Val1[0,0], 20, 4, 20, 0, 100
      CAN_S_AO       :17001, 1, 0, NONE.0, 0x0010, 2, Val2[0,0], 20, 4, 20, 0, 100

```

Do proměnných Val1 a Val2 jsou ukládány hodnoty analogových signálů zaslaných PDO-Masterem na uzel s Node-Idem 1 ve stupních Celsia (0÷100) z uzlu číslo 1 po přepočtu z proudového rozsahu 4÷20mA. Stav přenosu je ukládán do bitu @StavAO1, přenos je vyvoláván PDO-Masterem.

<b>CAN_S_DI</b>	Slave-modul dig. vstupů na sběrnici CAN
-----------------	---

**Popis**

Modul (PDO-Slave vstupního PDO, viz oddíl “Vrstva objektů procesních dat PDO” dodatku “Obsluha sběrnice CAN”) definuje vysílání dat zvoleného počtu digitálních vstupních signálů ze vstupního zařízení připojeného na sběrnici CAN.

Data se načítají z proměnné určené parametrem *Proměnná*. Je-li za tento parametr dosazen prvek matice a parametrem *Bajtů* zvoleno více bajtů, než je velikost jedné položky této matice, načítá se i z následujících prvků téhož řádku matice.

**Řízení přenosu**

U vstupních PDO je obvyklé, že přenos dat vyvolává PDO-Slave. U digitálních signálů zpravidla při změně hodnoty některého signálu. Toho dosáhneme dosazením jedničky za parametr *Přenést*.

Pokud se jeden PDO skládá z více proměnných, a tedy se PDO skládá postupně prostřednictvím více modulů **CAN\_DO**, bylo by zbytečným přetěžováním sběrnice žádat o přenos v každém z těchto modulů. Je tedy vhodné dosazovat jedničku za *Přenést* jen v jednom z nich (zpravidla v posledním).

**Čtení dat ze vstupní proměnné**

Data se z proměnné uvedené v parametru *Hodnota* čtou v okamžiku vyvolání každé instance modulu **CAN\_S\_AI**.

**Identifikátor (COB-Id) použitého PDO**

Modul používá implicitní mapování PDO dle specifikace CANopen Minimum Capability Device, tedy pracuje s PDO s identifikátorem COB-Id = 385 + Node-Id - 1.

**Parametry**

<b>Připojení</b>	PAR	Návěští	Návěští připojovacího modulu <b>CNC_...</b> , který zajišťuje fyzické připojení ke sběrnici CAN.
------------------	-----	---------	--

Toto návěští je globální (modul **CNC\_...** je v ProclNIT zatímco moduly vrstvy PDO zpravidla v periodických procesech) a je ho tedy třeba zadat ručně.

<b>Node-Id</b>	PAR	Konst	Identifikátor uzlu (běžného nebo virtuálního) sběrnice CAN, který je PDO-Slavem příslušného PDO. Zpravidla je roven Node-Idu uzlu, na němž je modul použit, pokud nepoužíváme virtuální či nenakonfigurované uzly (viz dodatek “Obsluha sběrnice CAN”)
----------------	-----	-------	--

<b>Přenést</b>	IN	Konst	Udává, zda se má při vyvolání tohoto modulu iniciovat přenos dat po sběrnici CAN.
		Bit	

<b>Stav</b>	OUT	Bit	Výstupní bit, indikující, že data byla úspěšně vyslána na sběrnici. Nuluje se při vyvolání přenosu dat, nastavuje se na jedničku při dokončení vysílání odpovídajícího PDO, ať už v důsledku nastavení bitu <i>Přenést</i> , nebo na žádost PDO-Mastera. Nechceme-li stav přenosu testovat, lze za tento parametr dosadit <i>NONE</i> .
-------------	-----	-----	---

<b>Bajtů</b>	PAR	Konst	Počet byte dat, které se mají uložit do PDO.
--------------	-----	-------	--

<b>Počátek</b>	PAR	Konst	Pořadí (0÷7) byte v datové části PDO, od kterého se mají uložit hodnoty zapisovaných signálů.
----------------	-----	-------	---

Proměnná	IN	I	Jméno databázové proměnné (nebo prvku matice), ze které se načtou hodnoty signálů před vysláním odpovídajícího PDO.
		L	
		F	
		MI	
		ML	
		MF	

**Příklad**

```

ProcINIT:
17001 CNC_C167      125kbit/s
17002 CAN_NMT_S     :17001, 1, 500, 4, @Lost, NMT_Full

Proc00:
    CAN_S_DI        :17001, 1, 0, NONE.0, 2, 0, Val1[0,0]
    CAN_S_DI        :17001, 1, @Zmena, @StavDI1, 2, 2, Val2[0,0]
    LET              @Zmena = 0

```

Data proměnných Val1 a Val2 jsou zaslána NMT-Masterovi jako 32 digitálních signálů uzlu s Node-Idem1. Stav přenosu je ukládán do bitu @StavDI1, přenos je vyvoláván PDO-Slavem při změně některého signálu, která způsobí nastavení bitu @Zmena (způsob detekce změny zde není uveden).

<b>CAN_S_DO</b>	Slave-modul dig. výstupů na sběrnici CAN
-----------------	--

**Popis**

Modul (PDO-Slave vstupního PDO, viz oddíl “Vrstva objektů procesních dat PDO” dodatku “Obsluha sběrnice CAN”) definuje příjem dat zvoleného počtu digitálních výstupních signálů výstupním zařízením připojeným na sběrnici CAN.

Data se ukládají do proměnné určené parametrem *Proměnná*. Je-li za tento parametr dosazen prvek matice a parametrem *Bajtů* zvoleno více bajtů, než je velikost jedné položky této matice, ukládá se i do následujících prvků téhož řádku matice.

**Řízení přenosu**

U výstupních PDO musí přenos dat vyvolávat PDO-Master. PDO-Slave výstupního PDO sice může nastavit parametr *Přenést*, čímž se vyšle remote-frame, žádající o data, ale v praxi je velmi nepravděpodobné setkat se s PDO-Masterem výstupního PDO, který by byl připraven na tento remote-frame reagovat.

**Ukládání dat do výstupní proměnné**

Data se do proměnné uvedené v parametru *Hodnota* neukládají v okamžiku vyvolání modulu **CAN\_S\_DO**, ale asynchronně, při příjmu odpovídajícího PDO. Z toho vyplývá následující:

- Modul **CAN\_S\_DO** není nutné vůbec umísťovat do periodického procesu, je možno ho vložit do ProclNIT.
- Není možné používat dočasnou proměnnou, do které se data načtou a potom někam zkopírují. Proměnná uvedená v parametru *Proměnná* musí být trvale vyhrazena jen pro tento účel.
- Chceme-li čekat na naplnění výstupní proměnné daty, je třeba vhodně testovat výstupní bit *Stav*. Je ovšem rovněž nutné ho ve vhodném okamžiku nulovat, protože parametr *Přenést* se z výše uvedených důvodů obvykle nepoužívá, tudíž modul **CAN\_S\_DO** tento bit nikdy nenuluje.

**Identifikátor (COB-Id) použitého PDO**

Modul používá implicitní mapování PDO dle specifikace CANopen Minimum Capability Device, tedy pracuje s PDO s identifikátorem COB-Id = 513 + Node-Id - 1.

**Parametry**

<b>Připojení</b>	PAR	Návěští	Návěští připojovacího modulu <b>CNC_...</b> , který zajišťuje fyzické připojení ke sběrnici CAN.
------------------	-----	---------	--

Toto návěští je globální (modul **CNC\_...** je v ProclNIT zatímco moduly vrstvy PDO zpravidla v periodických procesech) a je ho tedy třeba zadat ručně.

<b>Node-Id</b>	PAR	Konst	Identifikátor uzlu (běžného nebo virtuálního) sběrnice CAN, který je PDO-Slavem příslušného PDO. Zpravidla je roven Node-Idu uzlu, na němž je modul použit, pokud nepoužíváme virtuální či nenakonfigurované uzly (viz dodatek “Obsluha sběrnice CAN”).
----------------	-----	-------	---

<b>Přenést</b>	IN	Konst	Udává, zda se má při vyvolání tohoto modulu iniciovat přenos dat po sběrnici CAN. Viz upozornění v oddílu “Řízení přenosu” výše.
		Bit	

<b>Stav</b>	OUT	Bit	Výstupní bit, indikující, že byla přijata data a proměnná <i>Hodnota</i> byla naplněna. Nuluje se při vyvolání přenosu dat (ale viz upozornění v oddílu “Řízení přenosu” výše), nastavuje se na jedničku při příjmu odpovídajícího PDO (ať už na vyžádání parametrem <i>Přenést</i> , nebo z iniciativy PDO-Mastera). Nechceme-li stav přenosu testovat, lze za tento parametr dosadit NONE.
-------------	-----	-----	---

<b>Bajtů</b>	PAR	Konst	Počet byte dat, které se mají uložit do proměnné předané za parametr <i>Proměnná</i> .
--------------	-----	-------	--

<b>Počátek</b>	PAR	Konst	Pořadí (0÷7) byte v datové části PDO, od kterého jsou uloženy hodnoty načítaných signálů.
----------------	-----	-------	---

<b>Proměnná</b>	OUT	I	Jméno databázové proměnné (nebo prvku matice), do které se uloží hodnoty signálů při příjmu odpovídajícího PDO.
		L	
		F	
		MI	
		ML	
		MF	

**Příklad**

```

ProcINIT:
17001 CNC_C167      125kbit/s
17002 CAN_NMT_S     :17001, 1, 500, 4, @Lost, NMT_Full
                  CAN_S_DO :17001, 1, 0, @StavDO1, 2, 0, Val1[0,0]
                  CAN_S_DO :17001, 1, 0, NONE.0, 2, 2, Val2[0,0]

```

Do proměnných Val1 a Val2 jsou ukládány hodnoty 32 digitálních výstupních signálů zasílaných PDO-Masterem na uzel s Node-Idem 1. Stav přenosu je ukládán do bitu @StavDO1, přenos je vyvoláván PDO-Masterem.



**Popis**

Modul řídí kaskádu kotlů (maximálně 32 kotlů). Modul zajišťuje:

- ♦ *rovnoměrné opotřebení kotlů*
- ♦ *určení počtu jednotek, které mají být v chodu (tzv. sekundární regulace)*
- ♦ *regulaci vlastního kotle (tzv. primární regulace)*
- ♦ *speciální funkce a ochrany*

Dále rozebereme jednotlivé činnosti:

- ♦ ***rovnoměrné opotřebení kotlů***

Modul určuje počet kotlů, které mají být v chodu a pořadí v jakém se připojují. Každému kotli se měří provozní čas, což je celková doba, po kterou byl kotel v chodu. Pořadí kotlů je pak určeno obráceným pořadím provozních časů. Tedy kotel s menším provozním časem se připojuje dříve.

Výpočtem pořadí dle provozních časů se dosahuje rovnoměrného opotřebení kotlů. Navíc modul ještě zajišťuje pravidelné střídání kotlů, a to tak, že jako první v pořadí je tzv. *hlavní kotel* nezávisle na svém provozním čase. Hlavní kotel je tedy nejvíce používán (běží vždy, pokud je požadovaný počet kotlů větší než nula).

Přepočet pořadí kotlů a zároveň vystřídání hlavního kotle se provádí bitem `Pořadí`. Pro periodické generování bitu se typicky používá modul `SyncMark` nastavený např. tak, aby jedenkrát za týden nastavil bit `Pořadí` do "1". Modul poté, co provede patřičné akce, bit vynuluje.

- ♦ ***určení počtu jednotek, které mají být v chodu (tzv. sekundární regulace)***

Nejprve vysvětlíme pojem *Jednotka*.

**Jednotka**

Jednotkou rozumíme buď jeden kotel, pokud je daný kotel jednostupňový, nebo jeden stupeň kotle, pokud je daný kotel vícestupňový. Máme-li např. dva jednostupňové kotle a dva dvoustupňové, je celkový počet jednotek 6.

Pro připojování dalšího kotle se používá integrální kritérium, jehož okamžitá hodnota je dána:

$$I(t) = \int_0^t x(t) \cdot dt,$$

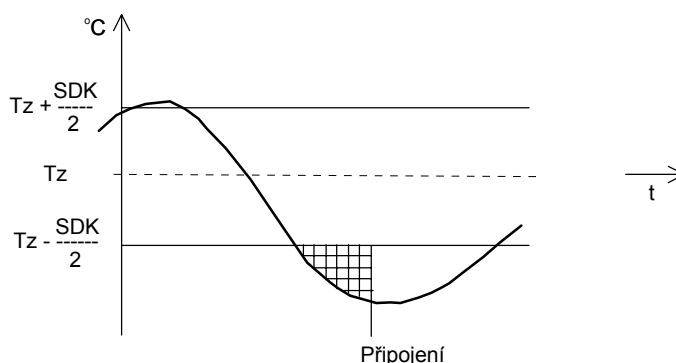
kde  $I$  je hodnota integrálu a  $x$  je modifikovaná regulační odchylka, která je určena:

$$x(t) = (T_z(t) - \frac{SDK}{2}) - T_m(t),$$

kde  $T_m$  je měřená teplota,  $T_z$  je žádaná teplota a  $SDK$  je tzv. *spínací difference kotle*, která představuje "necitlivost" sekundární regulace.

Integrál se počítá pouze ve stavu, když  $T_m < T_z - \frac{SDK}{2}$ , jinak je vynulován. Další kotel se připojí, jestliže integrál překročí hodnotu `PřipojInt`.

Na obrázku je velikost integrálu znázorněna šrafovanou plochou.

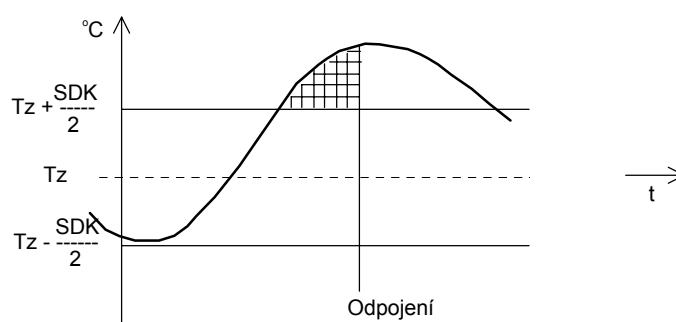


Pro odpojování dalšího kotle se používá stejné integrální kritérium jako pro připojování, liší se však modifikovaná regulační odchylka:

$$x(t) = T_m(t) - \left(T_z(t) + \frac{SDK}{2}\right).$$

Integrál se počítá pouze ve stavu, když  $T_m > T_z + \frac{SDK}{2}$ , jinak je integrál vynulován. Další kotel se odpojí, jestliže integrál překročí hodnotu  $OdpojInt$ .

Na obrázku je velikost integrálu znázorněna šrafovanou plochou.



#### Postup připojování jednotek:

U dvoustupňových kotlů typicky bývá výkon 1. stupně větší než výkon 2. stupně, což se projevuje poněkud složitějším připojováním dvoustupňových kotlů (obdobné je to i s více-stupňovými kotli). Postup ukážeme na příkladě. Nejprve však musíme vysvětlit pojmy *cyklování* a *regulační hladina*.

#### Cyklování

Regulace probíhá tak, že jedna jednotka z určeného počtu tzv. *cykluje* a ostatní jednotky z určeného počtu jedou naplno. Cyklování je v podstatě pulsně-šířková modulace, kdy se výstup regulátoru 0 až 100% převádí na velikost střídavé modulace (viz dále popis regulace vlastního kotle). Cyklování se používá při regulaci kotle typu VYP/ZAP, pro proporcionálně řízené kotle se používá přímo procentuální hodnota výkonu.

#### Regulační hladina

Regulační hladina je číslo, které charakterizuje celkový sepnutý výkon. Celkový výkon je dán součtem výkonů všech sepnutých jednotek. Při určování počtu kotlů se zvyšuje nebo snižuje regulační hladina. Dle této hladiny se potom sepnou určité jednotky. Hladina 0 je stav, kdy nejede žádná jednotka. Pro každou konfiguraci kaskády existuje maximální možná hladina, při které jsou všechny jednotky v chodu.

Příklad 1)

Připínání tří dvoustupňových kotlů (stupně jsou označeny I a II, cyklující stupeň je označen Ic nebo Ilc):

Hladina	Kotel A	Kotel B	Kotel C
1	Ic		
2	I+Ilc		
3	I	Ic	
4	I+Ilc	I	
5	I+II	I+Ilc	
6	I+II	I	Ic
7	I+II	I+Ilc	I
8	I+II	I+II	I+Ilc

Příklad 2)

A dvoustupňový, B jednostupňový, C dvoustupňový:

Hladina	Kotel A	Kotel B	Kotel C
1	Ic		
2	I+Ilc		
3	I	Ic	
4	I+Ilc	I	
5	I+II	I	Ic
6	I+II	I	I+Ilc

Příklad 3)

A dvoustupňový, B dvoustupňový, C jednostupňový:

Hladina	Kotel A	Kotel B	Kotel C
1	Ic		
2	I+Ilc		
3	I	Ic	
4	I+Ilc	I	
5	I+II	I+Ilc	
6	I+II	I	Ic
7	I+II	I+Ilc	I

U kombinace jedno- a vícestupňových kotlů může být celkový počet hladin o jedna menší, je-li jako poslední dvoustupňový kotel.

Určení maximální hladiny:

Chceme-li zjistit jaká může být maximální hladina pro danou konfiguraci kaskády, postupujeme takto:

- začneme s nulovou hodnotou
- za každý jednostupňový kotel připočteme hodnotu 1
- za každý dvoustupňový kotel připočteme hodnotu 3
- za každý třístupňový kotel připočteme hodnotu 4
- je-li při připojování jako poslední vícestupňový kotel, odečteme od výsledku hodnotu 1

Příklady:

Konfigurace	Maximální hladina
4 x jednostupňový kotel	4
4 x dvoustupňový kotel	11
4 x třístupňový kotel	15

♦ **regulace regulované jednotky (tzv. primární regulace)**

Primární regulace je PI regulátor, který reguluje jednu jednotku. Tato jednotka se nazývá regulovaná a je vždy pouze jediná v celé kaskádě.

## Regulovaná jednotka

Regulační hladina, která je výsledkem sekundární regulace, určuje rozdělení jednotek kaskády do tří skupin:

- jednotky, které jsou vypnuty
- jednotky, které jsou v chodu (trvale na 100%)
- jedna jednotka, která je regulovaná

Primární regulace se stará výhradně o tuto regulovanou jednotku.

Výsledkem primární regulace je procentuální hodnota výkonu regulované jednotky. Dle této procentuální hodnoty se jednotka řídí. Řízení může probíhat dvojím způsobem, dle typu kotle:

- cyklování - klasický kotel, který má spínání ZAP/VYP
  - proporcionální - kotel, který má plynule měnitelný výkon
- (oba dva typy kotlů mohou být jedno- i víceúpržné)

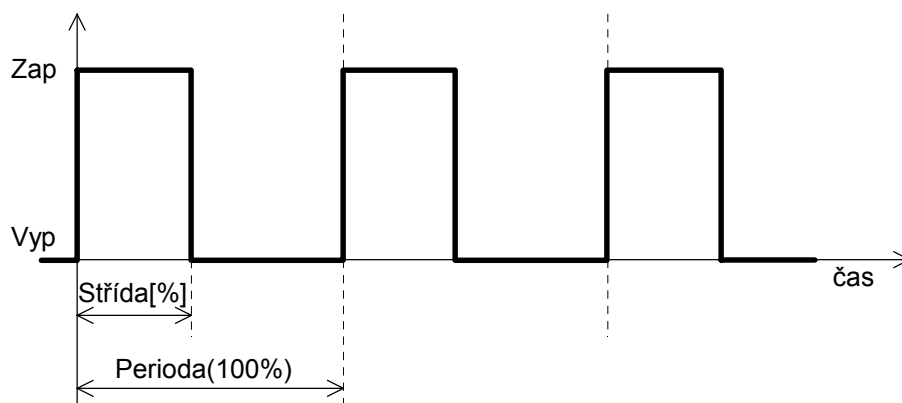
Z regulačního hlediska je jedno, o jaký typ kotle jde, protože výstupem PI regulátoru je v obou případech procentuální hodnota výkonu kotle. Ta se pak u cyklování převádí na pulsně-šířkovou modulaci a u proporcionálního kotle se přímo používá tato procentuální hodnota.

## Pulsně-šířková modulace

Procentuální hodnota výkonu 0% až 100% odpovídá střídě modulace. Aby se kotel nenamáhal zbytečně krátkými pulzy, je velikost střídě omezena:

Výkon P [%]	Omezená střída [%]
$P < 10$	0 (trvale vypnuto)
$10 \leq P < 20$	20
$20 \leq P < 80$	P
$80 \leq P < 90$	80
$90 \leq P$	100 (trvale zapnuto)

Velikost periody modulace se zadává v rozsahu (5 až 20) min., interně je hodnota omezena v rozsahu (1 až 20) min.



## Regulátor PI

Používá se klasický PI regulátor se vzorcem pro výpočet akčního zásahu:

$$y(t) = K \cdot \left( e(t) + \frac{1}{T_i} \int_0^t e(t) \cdot dt \right), \text{ kde}$$

$y$  .... akční veličina (výkon [%])

$e$  .... regulační odchylka (rozdíl mezi žádanou a měřenou teplotou)

$K$  ... zesílení (parametr Zesílení)

$T_i$  ... integrační časová konstanta (parametr ČasKonst)

## ♦ speciální funkce a ochrany

## Postupné spínání jednotek:

Modul zajišťuje, aby se nesešlo více jednotek současně. Při současném zapnutí více jednotek najednou může např. dojít ke krátkodobému poklesu tlaku a ochrany kotlů to mohou vyhodnotit jako poruchu. Nastane-li situace, že by se mělo připojit více jednotek, tak modul zajistí jejich postupné připojování s časovým odstupem. Délka časového odstupu je dána velikostí periody procesu, ve kterém je modul umístěn.

## Omezení žádané hodnoty:

Modulu se dá zadat omezení žádané hodnoty. Funguje to tak, že vstupní žádanou hodnotu si modul interně omezí na zadaný rozsah. **Pozor, není to ochrana omezení výstupní teploty vody z kaskády.** Ochrana omezení výstupní teploty je buď řešena přímo v kotli, nebo se musí naprogramovat mimo modul CasCon.

## Princip regulace:

Rozlišují se dva druhy regulace: regulace regulované jednotky a určování počtu jednotek, které mají být v chodu. Rozebereme je podrobněji:

## 1. Regulace regulované jednotky (primární regulace)

Jedná se o přímou regulaci teploty výstupní vody. Výstupem regulátoru je požadovaný výkon jedné jednotky (tj. jednoho kotle nebo jednoho stupně vícestupňového kotle). Pro regulaci se používá regulátor PI. Pokud se ukáže, že ani plný výkon jednotky nestačí k ohřátí vody na požadovanou teplotu nastupuje sekundární regulace, která připojí další jednotku. Dosavadní regulovaná jednotka pak jede naplno a primární regulací se dále reguluje výkon nové jednotky. Opačný postup nastává v případě soustavného přetápění, kdy ani nulový výkon regulované jednotky nedokáže přetápění odstranit a je nutno odepnout jednotku.

## 2. Určování počtu jednotek (sekundární regulace)

Jedná se o nepřímou (sekundární) regulaci, která se uplatňuje tehdy, když primární regulace nedostačuje. Pro sekundární regulaci se používá časový integrál, který se počítá, když regulační odchylka přesáhne mez danou SDK/2. Hodnota SDK představuje jakousi "necitlivost" sekundární regulace. Je nežádoucí, aby se počet jednotek změnil vlivem přechodných regulačních dějů. Sekundární regulace musí být proto navržena jako "pomalejší" vůči regulaci primární. Dosahuje se toho pomocí dostatečně velké hodnoty integrálů pro připojování a odpojování. Tyto parametry se volí tak, aby se sekundární regulace projevila teprve až po bezpečném odeznění přechodných regulačních dějů. Doporučuje se volit tyto hodnoty raději větší, aby nedocházelo k nežádoucímu připínání / odepínání jednotek.

Sekundární regulace se uplatňuje spíše výjimečně např. při najíždění kotelný nebo při změně počasí. Za běžné situace se uplatňuje pouze primární regulace.

## Parametry

<b>Žádaná</b>	IN	F	Žádaná hodnota teploty výstupní vody kaskády [°C].
		MF	
<b>Měřená</b>	IN	F	Měřená hodnota teploty výstupní vody kaskády [°C].
		MF	
<b>Vyp</b>	IN	Bit	Vypnutí modulu. Na náběžnou hranu bitu modul vypne všechny kotle a dále již do žádných proměnných nezapisuje.
<b>Pořadí</b>	INOUT	Bit	Příkaz pro přepočítání pořadí spínání jednotek a pro změnu hlavního kotle. Modul bit po zpracování vynuluje. Je-li bit trvale v "0", lze pořadí jednotek nastavovat zvnějšku modulu v proměnné <i>Chod</i> (viz dále).

<b>Perioda</b>	IN	Konst	Perioda cyklování (5 .. 20) [min].
		F	
		MF	

<b>Tmin</b>	IN	Konst	Omezení minimální žádané hodnoty teploty výstupní vody. Hodnota 0 znamená bez omezení.
		F	
		MF	

<b>Tmax</b>	IN	Konst	Omezení maximální žádané hodnoty teploty výstupní vody. Hodnota 0 znamená bez omezení.
		F	
		MF	

Typ	IN	MI	Matice rozměru $[n, 1]$ , kde $n$ je počet kotlů. Typy jednotlivých kotlů.	
			Hodnota	Význam
			1	Jednostupňový
			2	Dvoustupňový
			3	Třístupňový
		Kotle mohou být řízeny ZAP/VYP nebo proporcionálně.		

<b>Poruchy</b>	IN	MI	Matice rozměru $[n, 1]$ , kde $n$ je počet kotlů. Poruchy jednotlivých kotlů. Je-li kotel v poruše a nelze jej spouštět, zapíše se zvnějšku modulu do odpovídajícího řádku nenulová hodnota. Modul pak daný kotel nespo uští (vyřadí jej ze seznamu provozuschopných kotlů). Nulová hodnota znamená, že kotel je provozuschopný.

Chod	INOUT	MI	Matice rozměru $[n, 3]$ , kde $n$ je počet kotlů. Význam sloupců:		
			0	1	2
			Řízení ZAP/VYP	Pořadí sepnutí	Aktivita jednotek

### Řízení ZAP/VYP:

Význam bitů buňky Řízení ZAP/VYP:

Bit	Význam
0	1. stupeň zapnut
1	2. stupeň zapnut
2	3. stupeň zapnut

Sloupec Řízení ZAP/VYP se používá pro řízení kotlů ZAP/VYP. Hodnoty bitů jsou nastavovány takto:

Stav jednotky	Hodnota bitu
Vypnuta (0%)	"0"
Trvale v chodu (100%)	"1"
Regulovaná (výkon se mění)	Hodnota bitu se mění v rámci periody cyklování

Modul zajišťuje, že se nikdy do proměnné nezapíše kombinace "1. stupeň vypnut, 2. stupeň zapnut". Je-li zapnut 2. stupeň, musí být vždy zapnut i 1. stupeň.

### Pořadí sepnutí:

Hodnoty 0 až  $n-1$ , kde  $n$  je počet kotlů. Nejdříve se spíná kotel s hodnotou 0. Modul přepočítává pořadí na žádost pomocí bitu Pořadí. Způsob, jak modul určuje pořadí byl popsán výše v popisu "rovnoměrné opotřebení kotlů".

Určování pořadí zvnějšku modulu:

Chceme-li pořadí určovat zvnějšku modulu, lze jednoduše zapisovat do 1. sloupce této proměnné a bit Pořadí nechat trvale v "0".

**Aktivita jednotek:**

Význam jednotlivých bitů:

Bit	Význam
0	1. stupeň je aktivní
1	2. stupeň je aktivní
2	3. stupeň je aktivní

Bit má hodnotu "1" je-li daná jednotka buď trvale v chodu (výkon je 100%), nebo je to regulovaná jednotka (výkon se mění):

Stav jednotky	Hodnota bitu
Vypnuta (0%)	"0"
Trvale v chodu (100%)	"1"
Regulovaná (výkon se mění)	"1"

<b>Výkon</b>	OUT	MF	Regulovaný výkon jednotlivých stupňů. Matice rozměru $[n, m]$ , kde $n$ je počet kotlů a $m$ je maximální počet stupňů. Ve sloupcích jsou výkony jednotlivých stupňů počínaje prvním. Pro kotle typu ZAP/VYP mají hodnoty výkonu pouze informativní hodnotu, u proporcionálního kotle se tyto hodnoty převádějí na akční veličiny jednotlivých stupňů kotle.
--------------	-----	----	--

<b>Provoz</b>	INOUT	ML	Doby provozu [min]. Matice rozměru $[n, 1]$ , kde $n$ je počet kotlů. V proměnné se udržují doby provozu jednotlivých kotlů. Hodnoty lze zvětškovat modulu přepisovat, např. vynulovat. Podle těchto hodnot modul určuje pořadí spínání jednotlivých kotlů. Pořadí modul přepočítává pouze na žádost pomocí bitu Pořadí.
---------------	-------	----	---

<b>Hladina</b>	INOUT	I	Regulační hladina. Číslo, které charakterizuje velikost sepnutého výkonu. Popis viz výše. Je-li parametr $SDK=0$ (popsán dále), modul hodnotu hladiny neurčuje, ale pouze ji načítá. Takovým způsobem lze hladinu zadávat zvětškovat modulu např. pomocí tabulky dle hodnoty venkovní teploty. Nezávisle na hodnotě parametru $SDK$ lze hodnotu hladiny zvětškovat změnit, např. vynulovat. Je-li hladina určována modulem ( $SDK < 0$ ), je minimální hodnota omezena na 1, tj. vždy je alespoň jedna jednotka aktivní. V případě, že hladina je zadávána zvětškovat ( $SDK=0$ ), hodnota omezena není.
----------------	-------	---	---

<b>Zesílení</b>	IN	Konst	Zesílení regulátoru (primární regulace). Jedná se o regulátor PI, který určuje střihu cyklování regulované jednotky, případně výkon proporcionálně řízené jednotky.
		F	
		MF	

<b>ČasKonst</b>	IN	Konst	Integrační časová konstanta regulátoru [min] (primární regulace). Jedná se o regulátor PI, který určuje střihu cyklování regulované jednotky, případně výkon proporcionálně řízené jednotky.
		F	
		MF	

<b>SDK</b>	IN	Konst	Spínací diference kotle (sekundární regulace). Udává meze regulační odchylky, po jejichž překročení se začíná počítat integrál pro připojení nebo odpojení další jednotky. Bližší popis viz výše.
		F	
		MF	

<b>PřipojInt</b>	IN	Konst	Hodnota integrálu pro připojení další jednotky $^{\circ}\text{C} \cdot \text{min}$ (sekundární regulace).
		F	
		MF	

<b>OdpojInt</b>	IN	Konst	Hodnota integrálu pro odpojení další jednotky [°C.min] (sekundární regulace).
		F	
		MF	

<b>Integral</b>	OUT	F	Aktuální hodnota integrálu pro připojení / odpojení jednotky [°C.min] (sekundární regulace).
		NONE	

**Příklad**

Modul lze použít ve dvou základních režimech:

- ♦ *Počet jednotek určuje modul (sekundární regulaci provádí modul)*

Podmínkou tohoto režimu je  $SDK < > 0$ . Modul určuje a zapisuje hodnotu regulační hladiny. Nechť je dána konfigurace: 2 dvoustupňové kotle a 2 jednostupňové kotle. Změna pořadí kotlů a změna hlavního kotle se provádí každé pondělí v 6:00. Perioda cyklování je 10 min. Omezení žádané hodnoty není. Zesílení regulátoru je 8, integrační časová konstanta je 20 min. Spínací difference kotle je 4 °C. Integrál pro připojení další jednotky je 200 °C.min a integrál pro odpojení je 50 °C.min.

Uvedenému zadání odpovídají následující hodnoty proměnných:

Perioda: 10

Typ:

2
2
1
1

Zesílení: 8

CasKonst: 20

SDK: 4

PripojInt: 200

OdpojInt: 50

Pro generování bitu Pořadí se použije modul SyncMark.

SyncMark Týden, 1, 6, 0, 0, @Poradi, NONE

CasCon Zadana, Merena, @Vyp, @Poradi, Perioda, 0, 0, Typ, Poruchy, Chod,

Vykon, Provoz, Hladina, Zesileni, CasKonst, SDK, PripojInt, OdpojInt

- ♦ *Počet jednotek se určuje zvnějšku modulu (sekundární regulaci neprovádí modul, nýbrž aplikace)*

Určující podmínkou je  $SDK = 0$ . Počet jednotek se zadává pomocí hodnoty regulační hladiny, kterou modul pouze načítá. Dle této hodnoty a pořadí připojování v proměnné Chod modul odvozuje, které jednotky budou zapnuty, a které ne.

Nechť parametry jsou stejné, jako v předchozím příkladu. Pouze parametr SDK je nulový.

Hodnoty regulační hladiny jsou zadány formou tabulky dle venkovní teploty:

Venkovní teplota [°C]	Hladina
pod -11	8
-11 až -8	7
-8 až -5	6
-5 až -2	5
-2 až 1	4
1 až 5	3
5 až 12	2
12 až 20	1
nad 20	0

Tabulka je realizována pomocí modulu Interpol. Postupuje se tak, že pomocí interpolace se nejprve vytvoří spojitá hodnota hladiny HladinaF (typ F) a následujícím převodem na typ I (proměnná Hladina) se vytvoří celočíselná skoková hodnota. Vstupní hodnota venkovní teploty Tvenk do modulu Interpol je upravena pomocí nelinearity tzv. *plovoucí vůle v převodech*, aby se odstranilo případné kmitání výsledné



celočíselné hodnoty hladiny. K tomu může dojít, pokud se venkovní teplota pohybuje kolem některého ze zlomových bodů.

Hodnoty matice  $F_{ceHladina}$  představující interpolační tabulku:

-100	8
-11	8
-8	7
-5	6
-2	5
1	4
5	3
12	2
20	1
100	0

Hodnotu plovoucí vůle stanovíme:  $Hyst: 0.1$

```
Let      Tx = IF(Tvenk > Tx + Hyst, Tvenk, IF(Tvenk < Tx - Hyst, Tvenk, Tx))
Interpol Tx, HladinaF, FceHladina
Let      Hladina = HladinaF
SyncMark Týden, 1, 6, 0, 0, @Poradi, NONE
CasCon   Zadana, Merena, @Vyp, @Poradi, Perioda, 0, 0, Typ, Poruchy, Chod,
        Vykon, Provoz, Hladina, Zesileni, CasKonst, 0, PripojInt, OdpojInt
```

## Doporučení

Volba periody procesu

Co perioda procesu určuje:

- **časový odstup připojování jednotlivých jednotek**  
Jedná se o ochranu, aby se nikdy nepřipnulo více jednotek najednou. Nastane-li taková potřeba, spínají se jednotky postupně s periodou danou periodou procesu. Tato perioda musí být tedy alespoň tak dlouhá, jaká je nejkratší dovolená doba připínání jednotek po sobě.
- **rozlišení střídý cyklování**  
Střída může nabývat hodnot v celých násobcích periody procesu. S rostoucí periodou se snižuje rozlišení a regulace je tak méně "přesná".

V praxi se volí velikost periody procesu alespoň 10 krát menší než je perioda cyklování. Optimální hodnota je cca 100 krát menší než perioda cyklování.

## Nastavení regulačních parametrů

Nejprve je třeba nastavit parametry primární regulace, tj. hodnoty **Zesílení** a **CasKonst**. Při nastavování primární regulace je vhodné dočasně odpojit sekundární regulaci. Proveďte se to tak, že se nastaví  $SDK = 0$ . To způsobí, že modul nebude měnit regulační hladinu **Hladina**, která rozhoduje o počtu připojených jednotek. Regulační hladinu v tomto případě nastavíme na nějakou konstantní hodnotu. Tím dosáhneme toho, že část jednotek pojede naplno a jedna jednotka bude regulovaná (primární regulací). Pro nalezení vhodných parametrů primární regulace uvádíme dále jednoduchou praktickou metodu. Až se podaří nastavit uspokojivě parametry primární regulace, lze přistoupit k nastavení sekundární regulace.

U sekundární regulace se nastavují parametry **SDK**, **PřipojInt** a **OdpojInt**. Hodnotu **SDK** nastavíme např. na 4 °C. Zvolíme výchozí hodnoty integrálů **PřipojInt** a **OdpojInt**. Pro počáteční nastavení doporučujeme vyjít z hodnot parametrů primární regulace:

$$PřipojInt = 50 \cdot \frac{CasKonst}{Zesileni}$$

Hodnota **OdpojInt** se obvykle nastavuje stejná nebo menší (až poloviční) než je hodnota **PřipojInt**. To způsobí, že odpojování je rychlejší než připojování, což je doporučováno.

Hodnota integrálu určuje rychlost sekundární regulace. Zvyšováním hodnoty integrálu se sekundární regulace zpomaluje, jednotky se připínají pomaleji a nedochází k překmitu sekundární regulace. Překmitem sekundární regulace se myslí stav, kdy při větších

skocích žádané hodnoty přípne sekundární regulace postupně více jednotek, než je potřeba. Teprve následně začne počet jednotek snižovat na správnou hodnotu. Tento překmit sekundární regulace se pochopitelně projeví i překmitem regulované teploty. Nicméně jistá velikost překmitu regulované teploty nemusí být na závadu. Snižováním hodnoty integrálu se sekundární regulace zrychluje, což se ovšem také může projevit zvyšováním překmitu.

V běžném provozu se velké skoky žádané hodnoty provádějí jen zřídka. Z toho důvodu lze nastavit hodnoty integrálů nižší, je-li potřeba zajistit vyšší rychlost připojování a odpojování jednotek. Při menších skocích nebo při postupné změně žádané hodnoty totiž k nežádoucím překmitům sekundární regulace obvykle vůbec nedochází.

Ověření nastavených hodnot lze provést sledováním odezev při větších skocích žádané hodnoty, kdy je modul nucen připnout alespoň jednu jednotku.

Nastavení správné hodnoty integrálu pro odpojení `OdpojInt` lze ověřit podobným způsobem jako nastavení `PřipojInt` - skoky žádané z vyšší hodnoty na nižší.

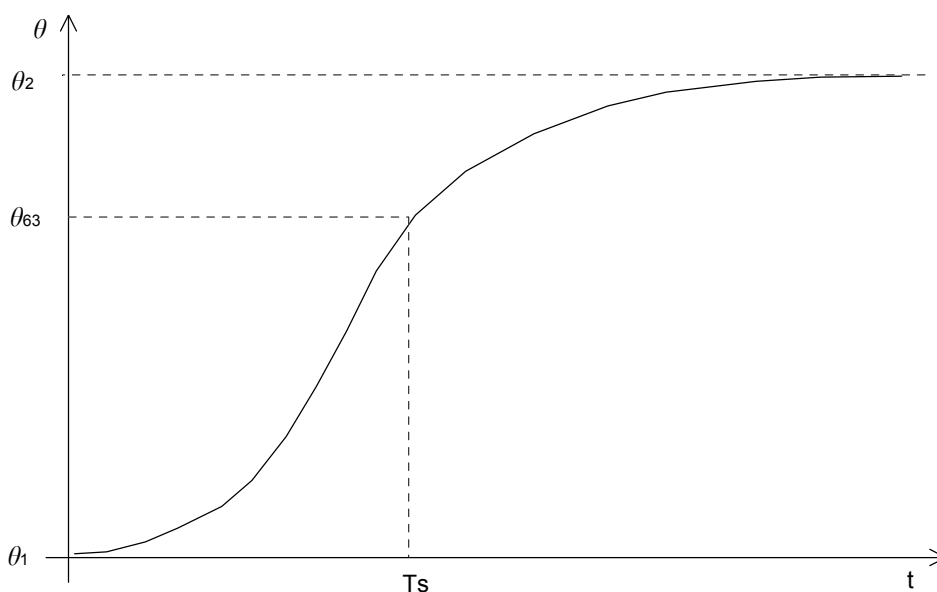
### Jednoduchá metoda pro nastavení parametrů primární regulace

Primární regulace je regulátor PI, regulující jednu jednotku. Pro nastavení regulátoru tj. vhodnou volbu parametrů `Zesílení` a `ČasKonst` popíšeme jednoduchý praktický postup, kterým lze dojít k solidním výchozím hodnotám. Tyto lze později "doladit".

Metoda spočívá ve změření přechodové charakteristiky soustavy tj. odezvy na skokovou změnu vstupu soustavy.

#### Postup:

- a) Zrušíme omezení žádané hodnoty, tj. nastavíme  $T_{\min} = 0, T_{\max} = 0$ .
- b) Vyřadíme sekundární regulaci  
Provedeme to nastavením  $SDK = 0$ .
- c) Primární regulaci nastavíme na 0%  
Žádanou teplotu `Žádaná` nastavíme na nějakou nízkou hodnotu např. -1000. Nízká žádaná hodnota způsobí, že výstup PI regulátoru bude trvale 0%.
- d) Soustavu necháme ustálit v "běžném" stavu  
To znamená ve stavu, ve kterém se často nachází. Např. jedna jednotka běží, ostatní stojí. Tj. hodnotu `Hladina` nastavíme na odpovídající hodnotu (např. 1) a počkáme, až se měřená teplota ustálí. Výsledkem je tedy stav, kdy část jednotek běží naplno vlivem hodnoty `Hladina` a regulovaná jednotka neběží resp. běží na 0% vlivem nízké žádané hodnoty.
- e) Provedeme skokovou změnu primární regulace na 100%  
Žádanou teplotu `Žádaná` nastavíme na nějakou vysokou hodnotu např. 1000. Vysoká žádaná hodnota způsobí, že výstup PI regulátoru bude trvale 100%. Sledujeme měřenou výstupní teplotu kaskády. Průběh teploty bude podobný následující křivce.



$\theta_1$  ... počáteční ustálená teplota

$\theta_2$  ... koncová ustálená teplota

$\theta_{63}$  ... teplota 63 % celkové změny teploty ( $\theta_1 + 0.63 * (\theta_2 - \theta_1)$ )

$T_s$  ... čas, za který teplota dosáhne 63 % celkové změny

- f) Zesílení regulátoru stanovíme:

$$\text{Zesílení} = 0.7 * \frac{100}{\theta_2 - \theta_1}$$

- g) Integrační časovou konstantu regulátoru stanovíme:

$$\text{CasKonst} = T_s$$

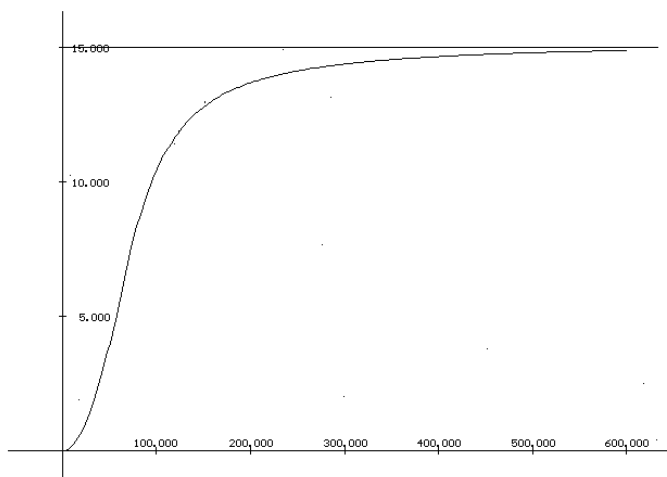
- h) Doladění konstant

Takto stanovené parametry regulátoru by měly zajistit, že výsledný regulovaný systém bude mít mírně přetlumený charakter. To znamená, že výsledná odezva na skok žádané hodnoty bude bez překmitu. Zmenšením zesílení lze případně systém ještě více ztlumit. Zvýšením zesílení se zrychlí regulační děj, ale sníží se tlumení, takže výsledná odezva může být mírně kmitavá. Výrazným zvyšováním zesílení mohou kmity narůstat až k nestabilitě systému.

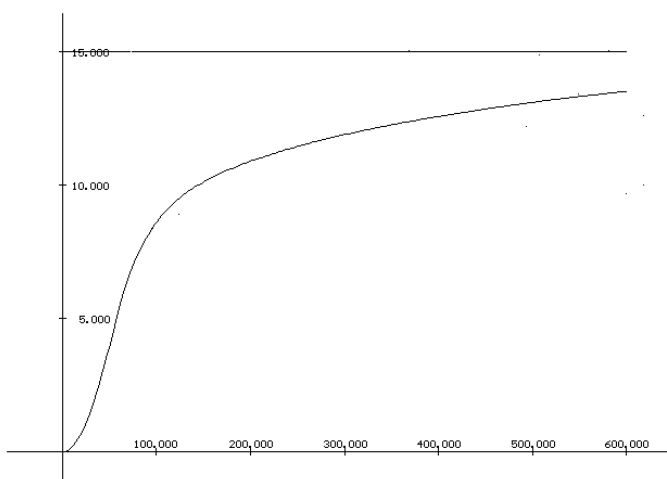
Skok žádané hodnoty můžeme provést např. tak, že nastavíme žádanou teplotu na teplotu, na které se systém ustálil na začátku předchozího měření, tj.  $Zadana = \theta_1$ . Počkáme, až se měřená teplota ustálí na této hodnotě a pak provedeme skok o polovinu rozsahu teplot z předchozího měření, tj.  $Zadana = \frac{\theta_2 - \theta_1}{2}$ . Vše provádíme s vyřazenou sekundární regulací, tj.  $SDK = 0$ .

Integrační časovou konstantu obvykle není potřeba doladovat. Uvádíme proto jen příklady odezev výsledného regulovaného systému v závislosti na velikosti zvolené integrační časové konstanty regulátoru:

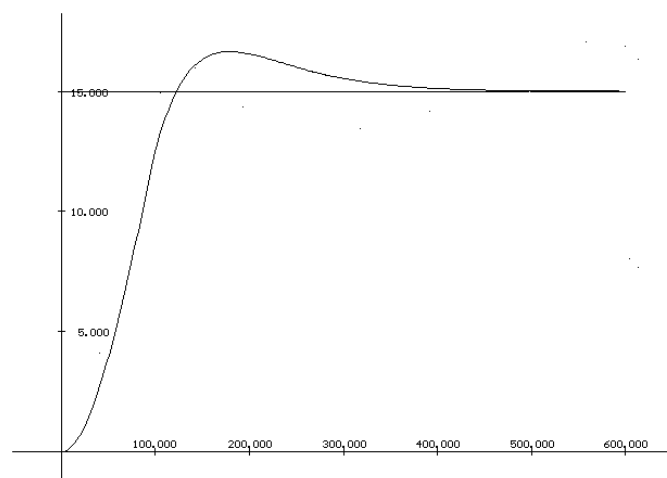
1) Správně zvolená integrační časová konstanta:



2) Regulátor s dvojnásobnou integrační časovou konstantou (dlouho “dotahuje”):



3) Regulátor s poloviční integrační časovou konstantou (nežádoucí překmit):



<b>Case</b>	Větev přepínače pro jednu konkrétní hodnotu
-------------	---

**Popis**

Příkaz **Case** definuje spolu s příkazem **EndCase** větev přepínače **Switch-EndSwitch** pro jednu konkrétní hodnotu přepínací proměnné. Detailní popis je uveden v popisu k modulu **Switch**.

**Parametry**

Hodnota	PAR	Konst	Hodnota řídící proměnné příkazu <b>Switch</b> , pro niž je tato větev příkazu <b>Case</b> aktivní.
---------	-----	-------	--

Návěští	PAR	Návěští	Návěští následujícího příkazu <b>EndCase</b> (automatické, lokální - lze generovat automaticky)
---------	-----	---------	---

**Příklad**

```

Switch Test, :00010      Přepínač podle hodnoty
                          proměnné Test
      Case 0, :00000      Větev pro hodnotu proměnné
                          Test=0
      . . .              Příkazy/moduly větve
:00000 EndCase           Konec větve
      Case 1, :00001      Větev pro hodnotu proměnné=1
      . . .
:00001 EndCase
      Case 2, :00002      Větev pro hodnotu proměnné=2
      . . .
:00002 EndCase
:00010 EndSwitch         Konec přepínače

```

<b>ChanMode</b>	Nastavení speciálního režimu logického kanálu
-----------------	---

**Popis**

Příkaz **ChanMode** slouží k nastavení speciálního režimu logického kanálu.

Většina logických kanálů nevyžaduje nastavení režimu. Rozsahy analogových vstupních signálů, negace digitálních vstupních signálů apod. se obvykle určují pouze pomocí parametrů modulů jako např. **AnIn**, **DigIn**. U takovýchto obvyčejných vstupně/výstupních signálů nemá použití modulu **ChanMode** žádný efekt.

Některé vstupně/výstupní signály ovšem vyžadují speciální parametrizaci na úrovni operačního systému. Jedná se pouze o signály, vybavené speciálními obvody, o jejichž nastavení nemá operační systém NOS přímou informaci, například analogové vstupny vybavené speciálním obvodem pro kompenzaci napětí analogové země. Právě k této parametrizaci režimu signálu na úrovni operačního systému slouží modul **ChanMode**.

**Parametry**

Typ	PAR	Výběr	Udává typ logického kanálu, se kterým má modul ChanMode pracovat.	
			Hodnota	Význam
			0	DI - digitální vstupní kanál
			1	DO - digitální výstupní kanál
			2	AI - analogový vstupní kanál
			3	AO - analogový výstupní kanál

Kanál	PAR	Konst	Číslo logického kanálu.
-------	-----	-------	-------------------------

Signál	PAR	Konst	Číslo signálu v rámci logického kanálu.
--------	-----	-------	---

Režim	IN	Výběr	Režim pro zvolený kanál/signál:	
			Hodnota	Význam
		I		
		MI	0	<b>COMPAI5V</b> - Analogový vstupní signál s rozsahem 5V a s kompenzací napětí analogové země.
			1	<b>COMPAI10V</b> - Analogový vstupní signál s rozsahem 10V a s kompenzací napětí analogové země.
			2	<b>COMPAI-I</b> - Analogový proudový vstupní signál s kompenzací napětí analogové země.
			3	<b>COMPAI-R</b> - Analogový odporový vstupní signál s kompenzací napětí analogové země.
			4	<b>DIFFAI</b> - Analogový vstupní signál, který spolu s následujícím signálem v témže kanálu tvoří diferenciální vstupní dvojici.
			5	<b>SNGLAI</b> - Single-ended analogový vstupní signál. Používá se pro zrušení předchozí volby DIFFAI (viz výše) nebo pro přepnutí vstupů, jejichž režim je implicitně diferenciální, do režimu jednoduchého vstupu.
			6	<b>MODEREG</b> - Následující parametr Data udává hodnotu, která bude zapsána do registru ModeReg A/D převodníku při přepnutí na signál určený parametrem Signál. Hodnota se uplatní až po vyvolání modulu <b>ChanMode</b> s parametrem Režim rovným 10 ( <b>REINIT</b> ).

	Má význam pouze u tenzometrických vstupů.
7	<b>FILTREG-L</b> - Následující parametr Data udává hodnotu, která bude zapsána do nižších 16ti bitů 24bitového registru FilterReg A/D převodníku. Hodnota se uplatní až po vyvolání modulu <b>ChanMode</b> s parametrem Režim rovným 10 ( <b>REINIT</b> ). Má význam pouze u tenzometrických vstupů. Hodnota parametru Signál se nebere v úvahu, jedná se o globální atribut celého kanálu.
8	<b>FILTREG-H</b> - Následující parametr Data udává hodnotu, která bude zapsána do vyšších 8mi bitů 24bitového registru FilterReg A/D převodníku. Hodnota se uplatní až po vyvolání modulu <b>ChanMode</b> s parametrem Režim rovným 10 ( <b>REINIT</b> ). Má význam pouze u tenzometrických vstupů. Hodnota parametru Signál se nebere v úvahu, jedná se o globální atribut celého kanálu.
9	<b>INTERVAL</b> - Následující parametr Data udává prodlevu v milisekundách mezi okamžikem, kdy je A/D převodník přepnut na některý signál, a okamžikem, kdy je hodnota tohoto signálu načtena z převodníku do logického kanálu. Hodnota se uplatní až po vyvolání modulu <b>ChanMode</b> s parametrem Režim rovným 10 ( <b>REINIT</b> ). Má význam pouze u tenzometrických vstupů. Hodnota parametru Signál se nebere v úvahu, jedná se o globální atribut celého kanálu.
10	<b>REINIT</b> - Provede reinicializaci obsluhy kanálu s promítnutím změn, u kterých je to v této tabulce uvedeno. Hodnota parametru Signál se nebere v úvahu, jedná se o globální ovládání celého kanálu.
11	<b>NI1000</b> - Analogový vstupní signál přepnutý na měření teploty odporovým snímačem Ni1000, resp. měření odporu. Má význam pouze u analogových vstupů se softwareovým přepínáním napětového nebo odporového měření, tedy ne u vstupů, kde se tyto režimy rozlišují hardwareovými prostředky - propojkou.

		Tuto volbu lze zrušit (vrátit signál do výchozího nastavení) volbou režimu číslo 15 (DEFAULT).
12		<b>ACDI</b> - Digitální vstupní signál přepnutý na buzení střídavým napětím 50 Hz. Má význam pouze u systémů, kde se nepoužívají separátní kanály pro stejnosměrné a střídavé vyhodnocování vstupního signálu. Tuto volbu lze zrušit (vrátit signál do výchozího nastavení) volbou režimu číslo 15 (DEFAULT).
13		<b>CONTACTDI</b> - Digitální vstupní signál přepnutý na vyhodnocování stavu pasivního kontaktu, zapojeného vůči zemi (zavřeno=1, otevřeno=0). Má význam pouze u digitálních vstupů se softwareovým přepínáním napětového buzení nebo vyhodnocování kontaktu. Tuto volbu lze zrušit (vrátit signál do výchozího nastavení) volbou režimu číslo 15 (DEFAULT).
14		<b>ACTIVE</b> - Aktivace výstupního signálu (digitálního nebo analogového). Má význam pouze u kombinovaných I/O obvodů, u kterých tatáž svorka může být softwareově konfigurována jako vstupní či výstupní, či umožňuje softwareově zvolit analogový nebo digitální výstupní režim. Tuto volbu lze zrušit (vrátit signál do výchozího nastavení) volbou režimu číslo 15 (DEFAULT).
15		<b>DEFAULT</b> - Návrat k výchozímu nastavení po volbě některého z těchto režimů, u nichž je možnost takového návratu uvedena.

Data	IN	Konst	Doplňující data pro nastavení režimu. Má význam jen u režimů, u kterých je to uvedeno v popisu parametru Režim viz výše. U ostatních režimů se nebere v úvahu.
		I	
		MI	

## Příklady

ChanMode AI,1,0,COMPAI-R,0

Nastaví logický kanál #AI1.0 do režimu kompenzace napětí analogové země při použití jako odporový vstup.

ChanMode AI,0,0,NI1000,0

ChanMode DI,0,1,ACDI,0

ChanMode DI,0,2,CONTACTDI,0

ChanMode DO,0,3,ACTIVE,0

Nastaví jednotlivé signály řídicího systému ADiR následovně:

IO0	analogový vstup: Měření teploty odporovým snímačem Ni1000 zapoužití funkčního modulu <b>Ni1000</b> nad logickým signálem AI0.0
IO1	digitální vstup: vyhodnocování přítomnosti střídavého napětí za použití např.



	funkčního modulu <b>BinIn</b> nad logickým signálem DI0.1
IO2	digitální vstup: vyhodnocování stavu pasivního kontaktu zapojeného mezi svorky IO2 a GND za použití např. funkčního modulu <b>BinIn</b> nad logickým signálem DI0.2
IO3	digitální výstup: ovládaný např. za použití funkčního modulu <b>BinOut</b> nad logickým signálem DO0.3
IO4	Výchozí nastavení - analogový nebo digitální vstup. Je možno: a) vyhodnocovat přítomnost stejnosměrného napětí za použití např. funkčního modulu <b>BinIn</b> nad logickými signály DI0.4, DI0.5, nebo b) měřit analogovou veličinu s elektrickým rozsahem 0 až 5 V za použití funkčního modulu <b>AnIn</b> nad logickými signály AI0.4, AI0.5.
IO5	

**ChkCheck** Kontrola zabezpečení rámce (uživatelská komunikace)

**Popis**

Modul se používá v uživatelské komunikaci ke kontrole, zda je přijatý rámec platný, zda nedošlo při přenosu ke zkreslení dat v rámci.

**Parametry**

<b>Data</b>	IN	MI	Kontrolovaný rámec - řádková matice. Jedna buňka matice představuje jeden znak rámce.
-------------	----	----	---

<b>IndexOd</b>	IN	Konst	Index v matici <i>Data</i> , od kterého se začíná počítat zabezpečení.
		I	
		MI	

<b>Délka</b>	IN	Konst	Počet znaků rámce (buňek matice <i>Data</i> ), ze kterých se počítá zabezpečení.
		I	
		MI	

<b>IndexChk</b>	IN	Konst	Od kterého indexu matice <i>Data</i> začíná vlastní zabezpečení.
		I	
		MI	

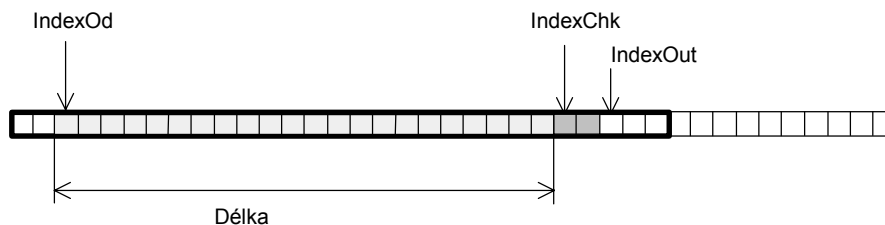
Typ	IN	Konst	Typ zabezpečení		
			Hodnota	Typ	Popis
		MI	0	CRC-16	16-ti bitové CRC s polynomem $x^{16} + x^{15} + x^2 + 1$ . Byty jdou v pořadí LE (B0, B1).
			1	CRC16 BE	16-ti bitové CRC s polynomem $x^{16} + x^{15} + x^2 + 1$ . Byty jdou v pořadí BE (B1, B0).
			2	CCIT	16-ti bitové CRC s polynomem $x^{16} + x^{12} + x^5 + 1$ . Byty jdou v pořadí LE (B0, B1).
			3	CCIT BE	16-ti bitové CRC s polynomem $x^{16} + x^{12} + x^5 + 1$ . Byty jdou v pořadí BE (B1, B0).
			4	SUM-16	16-ti bitová suma. Byty jdou v pořadí LE (B0, B1).
			5	SUM-16 BE	16-ti bitová suma. Byty jdou v pořadí BE (B1, B0).
			6	SUM-8	8-mi bitová suma.
			7	SUM-8+	8-mi bitová suma s přičítáním při přetečení. Při přičítání jednotlivých bytů se kontroluje, zda suma nepřesáhla hodnotu 255. Pokud k tomu dojde, ořízne se suma na byte a připočte se k ní jednička. Tento typ zabezpečení používá DB-Net.
			8	XOR-8	8-mi bitový XOR.

<b>IndexOut</b>	OUT	I	Index v matici <i>Data</i> posunutý o počet znaků vlastního zabezpečení.
		NONE	

<b>OK</b>	OUT	Bit	Příznak úspěchu (1 = úspěch , 0 = chyba).
		NONE	

Grafické znázornění významu jednotlivých parametrů:

Na obrázku je znázorněna matice `Data`. Silně orámovaná část představuje celý přijatý rámec. Tmavší buňky tvoří tu část rámce, ze které se počítá zabezpečení. Nejtmaší buňky tvoří vlastní zabezpečení - v tomto případě je zabezpečení na dvou bytech.



### Příklad

Přijímáme rámce, které mají tuto strukturu:

- 0x02 ... 1 byte začínající rámec
- LEN ... 1 byte určující délku dat rámce
- DATA ... data rámce o délce LEN
- CHK ... 2 byty zabezpečení typu CRC-16
- 0x03 ... 1 byte ukončující rámec

Přijatý rámec je v matici `Tlg`. Budeme kontrolovat, zda nedošlo při přenosu ke zkreslení rámce.

```
LET      Delka = Tlg[0, 1]
LET      NdxChk = 2 + Delka
ChkCheck Tlg, 2, Delka, NdxChk, CRC-16, NONE, @TlgOK
```

Výsledek kontroly je v bitu `@TlgOK`.

**ChkCreate** Vytvoření zabezpečení rámce (uživatelská komunikace)

**Popis**

Modul se používá v uživatelské komunikaci k výpočtu kontrolního zabezpečení vysílaného rámce. Vypočtenou hodnotu zabezpečení modul zapíše na zadanou pozici rámce. Na přijímací straně se opětovným výpočtem zabezpečení může následně kontrolovat, zda nedošlo při přenosu ke zkreslení dat v rámci.

**Parametry**

<b>Data</b>	IN/OUT	MI	Kontrolovaný rámec - řádková matice. Jedna buňka matice představuje jeden znak rámce.
-------------	--------	----	---

<b>IndexOd</b>	IN	Konst	Index v matici <i>Data</i> , od kterého se začíná počítat zabezpečení.
		I	
		MI	

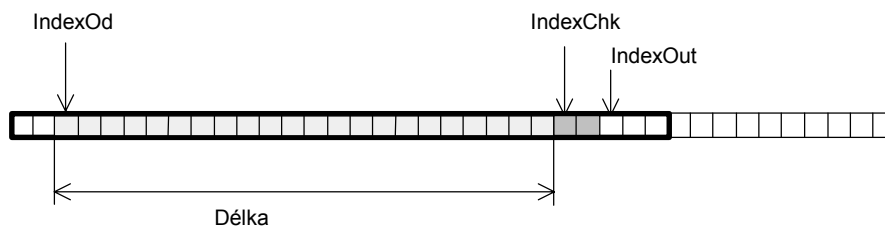
<b>Délka</b>	IN	Konst	Počet znaků rámce (buňek matice <i>Data</i> ), ze kterých se počítá zabezpečení.
		I	
		MI	

<b>IndexChk</b>	IN	Konst	Od kterého indexu matice <i>Data</i> se zapíše vlastní hodnota zabezpečení.
		I	
		MI	

Typ	IN	Konst	Typ zabezpečení		
			Hodnota	Typ	Popis
		I	0	CRC-16	16-ti bitové CRC s polynomem $x^{16} + x^{15} + x^2 + 1$ . Byty jdou v pořadí LE (B0, B1).
		MI	1	CRC16 BE	16-ti bitové CRC s polynomem $x^{16} + x^{15} + x^2 + 1$ . Byty jdou v pořadí BE (B1, B0).
			2	CCIT	16-ti bitové CRC s polynomem $x^{16} + x^{12} + x^5 + 1$ . Byty jdou v pořadí LE (B0, B1).
			3	CCIT BE	16-ti bitové CRC s polynomem $x^{16} + x^{12} + x^5 + 1$ . Byty jdou v pořadí BE (B1, B0).
			4	SUM-16	16-ti bitová suma. Byty jdou v pořadí LE (B0, B1).
			5	SUM-16 BE	16-ti bitová suma. Byty jdou v pořadí BE (B1, B0).
			6	SUM-8	8-mi bitová suma.
			7	SUM-8+	8-mi bitová suma s přičítáním při přetečení. Při přičítání jednotlivých bytů se kontroluje, zda suma nepřesáhla hodnotu 255. Pokud k tomu dojde, ořízne se suma na byte a připočte se k ní jednička. Tento typ zabezpečení používá DB-Net.
			8	XOR-8	8-mi bitový XOR.

Grafické znázornění významu jednotlivých parametrů:

Na obrázku je znázorněna matice *Data*. Silně orámovaná část představuje celý vytvářený rámec. Tmavší buňky tvoří tu část rámce, ze které se počítá zabezpečení. Nejtmavší buňky tvoří vlastní zapisované zabezpečení - v tomto případě je zabezpečení na dvou bytech.

**Příklad**

Vysíláme rámce, které mají tuto strukturu:

- 0x02 ... 1 byte začínající rámec
- LEN ... 1 byte určující délku dat rámce
- DATA ... data rámce o délce LEN
- CHK ... 2 byty zabezpečení typu CRC-16
- 0x03 ... 1 byte ukončující rámec

Vysílaný rámec je připraven v matici `Tlg`, v proměnné `Delka` je délka dat rámce. Do rámce budeme doplňovat zabezpečení.

```
LET      NdxChk = 2 + Delka
ChkCreate Tlg, 2, Delka, NdxChk, CRC-16, NONE
```

<b>CNC_ADCAN</b>	Připojení ke CANu přes modul AD-CAN
------------------	-------------------------------------

**Popis**

Modul obsluhuje komunikaci na sběrnici CAN prostřednictvím řadiče v rozšiřujícím modulu AD-CAN modulárního řídicího systému řady ADiS. Poskytuje jednotné rozhraní modulům vyšších komunikačních vrstev NMT a PDO.

*Pozn.: Rozšiřující modul AD-CAN se neuvádí v Konfiguraci V/V v prostředí PSE3.*

**Indikace stavu sběrnice****CAN**

Modul zároveň zajišťuje indikaci stavu sběrnice CAN pomocí LED označené "RDY" na rozšiřujícím modulu AD-CAN. Způsob indikace je závislý na tom, zda je řídicí systém naparametrizován jako NMT-Master (pomocí funkčního modulu **CAN\_NMT\_M**) nebo NMT-Slave (pomocí funkčního modulu **CAN\_NMT\_S**).

Stav LED RDY	NMT-Master	NMT-Slave
Trvale svítí	Komunikace na sběrnici v pořádku, všechny uzly (typu NMT-Slave) uvedené v konfiguraci vrstvy NMT jsou úspěšně zinicializovány.	
Bliká (50% svítí, 50% nesvítí)	Ztráta spojení s některým NMT-Slavem, uvedeným v konfiguraci vrstvy NMT	Ztráta spojení s NMT-Masterem
Zablikává (80% svítí, 20% nesvítí)	Tento stav se u NMT-Mastera nevyskytuje	Spojení s NMT-Masterem v pořádku, uzel zinicializován, ale spojení s některým z ostatních NMT-Slaveů, uvedených v konfiguraci vrstvy NMT, se ztratilo.
Trvale zhasnutá	Řadič sběrnice detekoval kritickou chybu na sběrnici CAN (např. zkrat), vůbec se nedaří vysílat rámce na sběrnici. <i>Pozn.: Ne každý zkrat na sběrnici je takto řadičem detekován.</i>	

**Umístění modulu**

Modul musí být umístěn v procesu ProcINIT, a to kdekoli před (nad) všemi moduly z knihovny **CAN**, které se na něj odkazují.

**Parametry**

Pozice	PAR	Konst	Pozice (0÷15) rozšiřujícího modulu AD-CAN v sestavě vstup/výstupních modulů modulárního řídicího systému řady ADiS.
--------	-----	-------	---

Rychlost	PAR	Výběr	Udává přenosovou rychlost sběrnice CAN.	
			Hodnota	Význam
			0	10kbit/s
			1	20kbit/s
			2	50kbit/s
			3	100kbit/s
			4	125kbit/s
			5	250kbit/s
			6	500kbit/s
			7	800kbit/s
			8	1Mbit/s

**Příklad**

```
ProcInit:
17001 CNC_ADCAN 0, 125kbit/s
```

Zajišťuje obsluhu sběrnice CAN prostřednictvím modulu AD-CAN, umístěného v sestavě na pozici 0 (bezprostředně sousedící s CPU), komunikační rychlost je 125 kbit/s.

**CNC\_ADOS**

## Připojení ke CANu pro ADOS100/200

Zajišťuje připojení ke sběrnici CAN na řídicích systémech typu ADOS100/200.  
Modul je funkčně plně ekvivalentní a stoprocentně zaměnitelný s modulem **CNC\_C167**,  
jakož i s ostatními moduly, které jsou s **CNC\_C167** plně ekvivalentní.  
Viz popis funkčního modulu **CNC\_C167**.

<b>CNC_AMP99</b>	Připojení ke CANu pro AMAP99
------------------	------------------------------

Zajišťuje připojení ke sběrnici CAN na řídicích systémech typu AMAP99.  
Modul je funkčně plně ekvivalentní a stoprocentně zaměnitelný s modulem **CNC\_C167**,  
jakož i s ostatními moduly, které jsou s **CNC\_C167** plně ekvivalentní.  
Viz popis funkčního modulu **CNC\_C167**.



**CNC\_AMR99**

## Připojení ke CANu pro AMiRiS99

Zajišťuje připojení ke sběrnici CAN na řídicích systémech typu AMiRiS99.  
Modul je funkčně plně ekvivalentní a stoprocentně zaměnitelný s modulem **CNC\_C167**,  
jakož i s ostatními moduly, které jsou s **CNC\_C167** plně ekvivalentní.  
Viz popis funkčního modulu **CNC\_C167**.

<b>CNC_APT2k</b>	Připojení ke CANu pro APT2100G
------------------	--------------------------------

Zajišťuje připojení ke sběrnici CAN na řídicích systémech typu APT2100G.  
Modul je funkčně plně ekvivalentní a stoprocentně zaměnitelný s modulem **CNC\_C167**,  
jakož i s ostatními moduly, které jsou s **CNC\_C167** plně ekvivalentní.  
Viz popis funkčního modulu **CNC\_C167**.

<b>CNC_ART4k</b>	Připojení ke CANu pro ART4000
------------------	-------------------------------

Zajišťuje připojení ke sběrnici CAN na řídicích systémech typu ART4000.  
Modul je funkčně plně ekvivalentní a stoprocentně zaměnitelný s modulem **CNC\_C167**,  
jakož i s ostatními moduly, které jsou s **CNC\_C167** plně ekvivalentní.  
Viz popis funkčního modulu **CNC\_C167**.

<b>CNC_C167</b>	Připojení ke CANu pro systémy s C167
-----------------	--------------------------------------

**Popis**

Modul obsluhuje komunikaci na sběrnici CAN prostřednictvím interního řadiče sběrnice CAN na procesoru C167 u řídicích systémů, které jsou vybaveny tímto procesorem a zároveň potřebnými obvody pro fyzické připojení ke sběrnici. Poskytuje jednotné rozhraní modulům vyšších komunikačních vrstev NMT a PDO.

Indikace stavu sběrnice  
CAN

Modul zároveň zajišťuje indikaci stavu sběrnice CAN pomocí LED označené "ERR" na standardním systémovém indikátoru (jen u systémů, které jsou tímto indikátorem vybaveny, tedy ne např. u systémů řady ART). Způsob indikace je závislý na tom, zda je řídicí systém naparametrizován jako NMT-Master (pomocí funkčního modulu **CAN\_NMT\_M**) nebo NMT-Slave (pomocí funkčního modulu **CAN\_NMT\_S**).

Stav LED ERR	NMT-Master	NMT-Slave
Trvale zhasnutá	Komunikace na sběrnici v pořádku, všechny uzly (typu NMT-Slave) uvedené v konfiguraci vrstvy NMT jsou úspěšně zinicizovány.	
Bliká (50% svítí, 50% nesvítí)	Ztráta spojení s některým NMT-Slavem, uvedeným v konfiguraci vrstvy NMT	Ztráta spojení s NMT-Masterem
Zablikává (20% svítí, 80% nesvítí)	Tento stav se u NMT-Mastera nevyskytuje	Spojení s NMT-Masterem v pořádku, uzel zinicizován, ale spojení s některým z ostatních NMT-Slaveů, uvedených v konfiguraci vrstvy NMT, se ztratilo.
Trvale svítí	Řadič sběrnice detekoval kritickou chybu na sběrnici CAN (např. zkrat), vůbec se nedaří vysílat rámce na sběrnici. Pozn.: Ne každý zkrat na sběrnici je takto řadičem detekován.	

**Umístění modulu**

Modul musí být umístěn v procesu ProcINIT, a to kdekoli před (nad) všemi moduly z knihovny **CAN**, které se na něj odkazují.

**Parametry**

Rychlost	PAR	Výběr	Udává přenosovou rychlost sběrnice CAN.	
			Hodnota	Význam
			0	10kbit/s
			1	20kbit/s
			2	50kbit/s
			3	100kbit/s
			4	125kbit/s
			5	250kbit/s
			6	500kbit/s
			7	1Mbit/s

**Příklad**

```
ProcInit:
17001 CNC_C167    125kbit/s
```

Zajišťuje obsluhu sběrnice CAN prostřednictvím interního řadiče sběrnice CAN na procesoru C167, komunikační rychlost je 125 kbit/s.

<b>CntDn</b>	Čítač dolů
--------------	------------

**Popis**

Modul čítače, který čítá směrem dolů.

**Parametry**

<b>Čítej</b>	IN	Bit	Na náběžnou hranu bitu čítá dolů.
<b>Nastav</b>	IN	Bit	Nastavením bitu do "1" se hodnota čítače nastaví na hodnotu <i>Hodnota</i> . Čítač setrvá v tomto stavu, dokud není bit zvenku vynulován.
<b>Hodnota</b>	IN	Konst	Nastavovaná hodnota.
		I	
		MI	
<b>Výstup</b>	OUT	Bit	Výstup se nastaví do "1", je-li hodnota čítače <i>Čítač</i> ≤ 0.
<b>Čítač</b>	OUT	I	Hodnota čítače. Minimální hodnota, na které se čítač zastaví je -32768.
		MI	
		NONE	

**Příklad**

CntDn                      Beh.0, Tik.0, Perioda, Tik.0, Citac

Dosazením stejného bitu Tik.0 do parametrů *Nastav* a *Výstup* realizujeme cyklický čítač. V proměnné *Citac* je hodnota čítače, která se z hodnoty *Perioda* postupně snižuje na 0, aby pak v dalším kroku opět nabyla hodnoty *Perioda*, a tak stále dokola. Náběžnou hranou bitu Beh.0 se čítač sníží o jedničku.

<b>CnDnLv</b>	Čítač dolů - hladinový
---------------	------------------------

**Popis**

Modul čítače, který čítá směrem dolů.

**Parametry**

<b>Čítej</b>	IN	Bit	1=čítej, 0=stůj
<b>Nastav</b>	IN	Bit	Nastavením bitu do "1" se hodnota čítače nastaví na hodnotu <i>Hodnota</i> . Modul poté bit vynuluje.
<b>Hodnota</b>	IN	Konst	Nastavovaná hodnota.
		I	
		MI	
<b>Výstup</b>	OUT	Bit	Výstup se nastaví do "1", je-li hodnota čítače <i>Čítač</i> ≤ 0.
<b>Čítač</b>	OUT	I	Hodnota čítače.
		MI	
		NONE	

**Příklad**

CnDnLv                      Beh.0, Tik.0, Perioda, Tik.0, Citac

Dosazením stejného bitu Tik.0 do parametrů *Natav* a *Výstup* realizujeme cyklický čítač. V proměnné *Citac* je hodnota čítače, která se z hodnoty *Perioda* postupně snižuje na 0, aby pak v dalším kroku opět nabyla hodnoty *Perioda*, a tak stále dokola. Pomocí bitu *Beh.0* se zastavuje a spouští čítání.

<b>CntUp</b>	Čítač nahoru
--------------	--------------

**Popis**

Modul čítače, který čítá směrem nahoru.

**Parametry**

<b>Čítej</b>	IN	Bit	Na náběžnou hranu bitu čítá nahoru
<b>Reset</b>	IN	Bit	Nastavením bitu do "1" se hodnota čítače nastaví na hodnotu 0. Čítač setrvá v tomto stavu, dokud není bit zvenku vynulován.
<b>Hodnota</b>	IN	Konst	Porovnávaná hodnota.
		I	
		MI	
<b>Výstup</b>	OUT	Bit	Výstup se nastaví do "1", je-li hodnota čítače Čítač >= Hodnota.
<b>Čítač</b>	OUT	I	Hodnota čítače. Maximální hodnota, na které se čítač zastaví je 32767.
		MI	
		NONE	

**Příklad**

CntUp                      Beh.0, Reset.0, 100, Max.0, Citac

Čítač čítá v intervalu od 0 do 100. Hodnota čítače je v proměnné Citac. Jakmile čítač dosáhne hodnoty 100. Nastaví se 0. bit proměnné Max do "1". Na náběžnou hranu bitu Beh.0 se čítač zvýší o jedničku.

<b>CnUpDn</b>	Čítač nahoru i dolů
---------------	---------------------

**Popis**

Modul čítače, který podle potřeby čítá směrem nahoru i dolů.

**Parametry**

<b>Nahoru</b>	IN	Bit	Na náběžnou hranu bitu čítá nahoru.
<b>Dolů</b>	IN	Bit	Na náběžnou hranu bitu čítá dolů.
<b>Reset</b>	IN	Bit	Nastavením bitu do "1" se hodnota čítače nastaví na hodnotu 0. Čítač setrvá v tomto stavu, dokud není bit zvenku vynulován.
<b>Nastav</b>	IN	Bit	Nastavením bitu do "1" se hodnota čítače nastaví na hodnotu <i>Hodnota</i> . Čítač setrvá v tomto stavu, dokud není bit zvenku vynulován.
<b>Hodnota</b>	IN	Konst	Porovnávaná nebo nastavovaná hodnota.
		I	
		MI	
<b>Nahoře</b>	OUT	Bit	Výstup se nastaví do "1", je-li hodnota čítače $\text{Čítač} \geq \text{Hodnota}$ .
<b>Dole</b>	OUT	Bit	Výstup se nastaví do "1", je-li hodnota čítače $\text{Čítač} \leq 0$ .
<b>Čítač</b>	OUT	I	Hodnota čítače. Maximální hodnota, na které se čítač zastaví je 32767. Minimální hodnota, na které se čítač zastaví je -32768.
		MI	
		NONE	

**Příklad**

CnUpDn      Citej.0, Citej.1, Nastav.0, Nastav.1, 20, Konec.0, Konec.1, Citac

Čítač čítá v intervalu 0 až 20. Hodnota čítače je v proměnné *Citac*. Dosažení maximální hodnoty čítače (20) je signalizováno 0. bitem proměnné *Konec*. Podobně, dosažení minimální hodnoty čítače (0) je signalizováno 1. bitem proměnné *Konec*. Na náběžnou hranu bitu č. 0 proměnné *Citej* se čítá nahoru, na náběžnou hranu bitu č. 1 proměnné *Citej* se čítá dolů. Bitem č. 0 proměnné *Nastav* se čítač resetuje na hodnotu 0, bitem č.1 proměnné *Nastav* se čítač nastaví na hodnotu načtenou z proměnné *Hodnota*.



<b>CnUpDnLv</b>	Čítač nahoru i dolů - hladinový
-----------------	---------------------------------

**Popis**

Modul čítače, který podle potřeby čítá směrem nahoru i dolů.

**Parametry**

<b>Nahoru</b>	IN	Bit	Bit povolující čítání nahoru.
<b>Dolů</b>	IN	Bit	Bit povolující čítání dolů.
<b>Reset</b>	IN	Bit	Nastavením bitu do "1" se hodnota čítače nastaví na hodnotu 0. Modul poté bit vynuluje.
<b>Nastav</b>	IN	Bit	Nastavením bitu do "1" se hodnota čítače nastaví na hodnotu <i>Hodnota</i> . Modul poté bit vynuluje.
<b>Hodnota</b>	IN	Konst	Porovnávaná nebo nastavovaná hodnota.
		I	
		MI	
<b>Nahoře</b>	OUT	Bit	Výstup se nastaví do "1", je-li hodnota čítače <i>Čítač</i> >= <i>Hodnota</i> .
<b>Dole</b>	OUT	Bit	Výstup se nastaví do "1", je-li hodnota čítače <i>Čítač</i> <= 0.
<b>Čítač</b>	OUT	I	Hodnota čítače.
		MI	
		NONE	

**Příklad**

CnUpDnLv    Citej.0, Citej.1, Nastav.0, Nastav.1, 20, Konec.0, Konec.1, Citac

Čítač čítá v intervalu 0 až 20. Hodnota čítače je v proměnné *Citac*. Dosažení maximální hodnoty čítače (20) je signalizováno 0. bitem proměnné *Konec*. Podobně, dosažení minimální hodnoty čítače (0) je signalizováno 1. bitem proměnné *Konec*. Bitem č. 0 proměnné *Citej* se spouští čítání nahoru, bitem č. 1 proměnné *Citej* se spouští čítání dolů. Bitem č. 0 proměnné *Nastav* se čítač resetuje na hodnotu 0, bitem č.1 proměnné *Nastav* se čítač nastaví na hodnotu načtenou z proměnné *Hodnota*.

<b>CntUpLv</b>	Čítač nahoru - hladinový
----------------	--------------------------

**Popis**

Modul čítače, který čítá směrem nahoru.

**Parametry**

<b>Čítej</b>	IN	Bit	Bit povolující čítání: 1=čítej, 0=stůj
<b>Reset</b>	IN	Bit	Nastavením bitu do "1" se hodnota čítače nastaví na hodnotu 0. Modul poté bit vynuluje.
<b>Hodnota</b>	IN	Konst	Porovnávaná hodnota.
		I	
		MI	
<b>Výstup</b>	OUT	Bit	Výstup se nastaví do "1", je-li hodnota čítače Čítač >= Hodnota.
<b>Čítač</b>	OUT	I	Hodnota čítače.
		MI	
		NONE	

**Příklad**

CntUpLv                      Beh.0, Reset.0, 100, Max.0, Citac

Čítač čítá v intervalu od 0 do 100. Hodnota čítače je v proměnné Citac. Jakmile čítač dosáhne hodnoty 100. Nastaví se 0. bit proměnné Max do "1". Čítání se spouští a zastavuje bitem Beh.0.

**ColorLED**

Ovládání vícebarevných LED na zařízeních, která jsou jimi vybavena.

**Popis**

Modul rozsvítí na žádanou barvu LED, identifikovanou číslem, popřípadě ji zhasne. Je též možno zvolit blikání příslušnou barvou včetně určení rychlosti blikání.

Modul ovládá LED na připojeném terminálu. Pokud se tedy k zařízení připojuje externí terminál vybavený LED (např. APT200, APT1100), ovládá modul jednotlivé LED na připojeném terminálu. Pokud je však dané zařízení zároveň i terminálem (např. Mesterm), ovládá modul jednotlivé LED na tomto zařízení.

Modul vyžaduje ke své činnosti, aby byl do aplikace vložen modul pro obsluhu LcdShellu.

**Parametry**

LED	IN	Konst	Číslo ovládané LED. LED jsou číslovány $0+n-1$ , kde $n$ je počet dostupných diod. V případě, že je možno ovládat jas diod, používá se modul ColorLED s parametrem LED rovným $n$ pro nastavování jasu. Požadovaný jas se v takovém případě předává v parametru <code>Perioda</code> . Počet dostupných stupňů jasu, a tím i maximální hodnota parametru <code>Perioda</code> , je závislý na konkrétním typu zařízení.
		I	
		MI	

Barva	IN	Konst	Číselný kód barvy, kterou má LED svítit nebo blikat. Hodnota 0 (= "zhasnuto") se používá pro zhasnutí LED. Dostupné barvy závisí na konkrétním zařízení. Barvy jsou kódovány takto:
		I	
		MI	

Kód	Význam
0	nesvítí
1	červená
2	zelená
3	žlutá

Perioda	IN	Konst	Parametr určuje půlperiodu blikání v násobcích základního kroku 10ms. Např. hodnota 50 určuje sekundovou periodu (500ms svítí, 500 ms nesvítí). Pro setrvalý stav (trvale svítí nebo trvale zhasnuto) se zadává nulová hodnota parametru <code>Perioda</code> . Zvláštní použití parametru <code>Perioda</code> pro ovládání jasu LED viz popis parametru <code>LED</code> výše.
		I	
		MI	

**Příklad**

ColorLED            0, žlutá, neblinká

Způsobí trvalé rozsvícení první LED žlutou barvou.

ColorLED            1, červená, 50

Způsobí blikání druhé LED červenou, LED bude 500ms rozsvícena, 500ms zhasnuta.

<b>ComGetLSR</b>	Načtení line status registru, zjištění chyb linky (uživatelská komunikace)
------------------	--

**Popis**

Modul načte line status registr obvodu seriové linky. V tomto registru jsou mimo jiné i příznaky chyb linky. Využíváme-li v aplikaci přerušení od chyby linky, umísťuje se modul do podprogramu přerušení.

**Parametry**

<b>ComInit</b>	PAR	Návěští	Návěští modulu ComInit.
----------------	-----	---------	-------------------------

LSR	IN	I MI	Hodnota registru. Význam jednotlivých bitů:	
			Bit	Význam
			0	Data Ready (DR) - přišel znak. Bit se resetuje čtením znaku. Čtení provádí automaticky modul ComInit do svého interního buferu, takže pro aplikaci nemá tento bit význam.
			1	Overrun Error (OE) - přišel nový znak, ještě než se stačil vybrat předchozí. Čtení přijatých znaků provádí automaticky modul ComInit do svého interního buferu, takže by k této chybě nemělo nikdy dojít. Dojde-li k přeplnění přijímacího buferu, nastaví se bit 15.
			2	Parity Error (PE) - chyba parity. Bit se nuluje čtením registru LSR.
			3	Framing Error (FE) - ztráta stopbitu. Bit se nuluje čtením registru LSR.
			4	Break Indicator (BI) - nastavuje se, pokud je signál RxD v "1" déle než jeden znak. Bit se nuluje čtením registru LSR.
			5	Transmitter Holding Register Empty (THRE) - indikuje, že lze zapsat další znak do registru THR (vysílací registr). Bit se nuluje zápisem do registru THR.
			6	Transmitter Empty (TEMT) - indikuje, že se zrovna nic nevysílá. Bit se nuluje čtením registru LSR.
			7	(pouze obvody 16550+) - bit je nastaven, pokud je alespoň jeden znak v přijímacím FIFO obvodu přijat s chybou. Bit se nuluje čtením registru LSR, pokud již žádný znak ve FIFO není přijat s chybou.
			15	Přeplnění přijímacího buferu - bit se nastaví, dojde-li při příjmu modulem ComInit k přeplnění přijímacího buferu. Bit se nuluje čtením registru LSR.

**Příklad**

Budeme načítat stav line status registru do proměnné LSR. Komunikační kanál je definován v procesu INIT pomocí modulu ComInit, který má návěští :01000.

```
ComGetLSR      :01000, LSR
```

<b>ComGetMSR</b>	Načtení modem status registru (uživatelská komunikace)
------------------	--

**Popis**

Modul načte modem status registr obvodu seriové linky. V tomto registru jsou stavy modemových signálů. Využíváme-li v aplikaci přerušení od změny modemových signálů, umísťuje se modul do podprogramu přerušení.

**Parametry**

<b>ComInit</b>	PAR	Návěští	Návěští modulu ComInit.
----------------	-----	---------	-------------------------

MSR	OUT	I MI	Hodnota registru. Význam jednotlivých bitů:	
			Bit	Význam
			0	Delta CTS - změna CTS. Bit je nastaven, pokud se od minulého čtení registru změnil stav signálu CTS. Bit je nulován čtením registru MSR.
			1	Delta DSR - změna DSR. Bit je nastaven, pokud se od minulého čtení registru změnil stav signálu DSR. Bit je nulován čtením registru MSR.
			2	TERI - náběh RI. Bit je nastaven, pokud se od minulého čtení registru změnil stav signálu RI z "0" do "1". Bit je nulován čtením registru MSR.
			3	Delta DCD - změna DCD. Bit je nastaven, pokud se od minulého čtení registru změnil stav signálu DCD. Bit je nulován čtením registru MSR.
			4	CTS - stav signálu CTS.
			5	DSR - stav signálu DSR.
			6	RI - stav signálu RI.
			7	DCD - stav signálu DCD.

**Příklad**

Budeme načítat stav modemového registru do proměnné MSR. Komunikační kanál je definován v procesu INIT pomocí modulu ComInit, který má návěští :01000.

```
ComGetMSR      :01000, MSR
```

<b>Comlnit</b>	Hlavní modul uživatelské komunikace (uživatelská komunikace)
----------------	--

**Popis**

Jádro uživatelské komunikace.

**Parametry**

<b>Režim</b>	PAR	Výběr	Režim činnosti.
PoslPřer	ANO		Generovat přerušení od vyslaného znaku pouze po posledním znaku (tj. po vyprázdnění vysílacího buferu).
		NE	Generovat přerušení od vyslaného znaku po každém znaku.
Řídit směr	ANO		Řídit směr buzením linky. Používá se při komunikaci přes kanál RS485. Na kanálu RS232 se to projeví nastavením signálu RTS do "1" před vysláním a nastavením signálu RTS zpět do "0" po odvyslání posledního znaku z vysílacího buferu.
	NE		Řízení směru se neprovádí. Používá se při komunikaci přes kanál RS232.
<b>Kanál</b>	PAR	Konst	Číslo komunikačního kanálu (0 .. 32).
<b>Rychlost</b>	PAR	Konst	Komunikační rychlost [Bd].
<b>Délka</b>	PAR	Konst	Délka znaku [počet bitů].
<b>Parita</b>	PAR	Konst	0 = žádná, 1 = sudá, 2 = lichá.
<b>Stopbity</b>	PAR	Konst	Počet stopbitů (1, 2).
<b>Přijltr</b>	PAR	Návěští	Návěští modulu SubInst s instancí podprogramu na obsluhu přerušení od přijatého znaku.
<b>Vyslitr</b>	PAR	Návěští	Návěští modulu SubInst s instancí podprogramu na obsluhu přerušení od vyslaného znaku.
<b>Errltr</b>	PAR	Návěští	Návěští modulu SubInst s instancí podprogramu na obsluhu přerušení od chyby linky (špatná parita, ztracený stopbit atd.).
<b>Modemltr</b>	PAR	Návěští	Návěští modulu SubInst s instancí podprogramu na obsluhu přerušení od změny modemových signálů.
<b>PřijBuf</b>	OUT	MI	Databázová matice MI[1, n] sloužící jako interní přijímací bufer. Jedna buňka matice odpovídá dvěma znakům pro případ délky znaku 8 bitů a méně. V případě vícebitových znaků odpovídá jedna buňka matice jednomu znaku.
<b>VyslBuf</b>	IN	MI	Databázová matice MI[1, n] sloužící jako interní vysílací bufer. Jedna buňka matice odpovídá dvěma znakům pro případ délky znaku 8 bitů a méně. V případě vícebitových znaků odpovídá jedna buňka matice jednomu znaku.

Modul slouží jako základní jádro komunikace. Nastavují se pomocí něj parametry komunikace. Umísťuje se do procesu INIT.

**Příjem znaků**

Z hlediska zpracování přijatých znaků umožňuje modul dva přístupy:

- a) Dotazovací (pooling)  
Modul ukládá přijaté znaky do interního buferu (parametr `PřijBuf`). Aplikace se periodicky dotazuje, zda něco přišlo. Provádí to modulem `ComRead`, který znaky z interního buferu přesouvá do svého vlastního buferu. Tento vlastní bufer je potom přístupný pro další zpracování v aplikaci.  
Při dotazovacím přístupu se parametr `PřijItr` nezadává (ponechává se hodnota `NONE`).
- b) Přerušovací (interrupt)  
Modul vyvolá okamžitě po přijetí znaku uživatelský podprogram na zpracování přijatého znaku. V uživatelském podprogramu se znak vybere opět pomocí modulu `ComRead`. Jelikož doba provádění tohoto podprogramu může být v některých případech relativně dlouhá, může dojít k tomu, že další znak přijde dříve než se stačí podprogram dokončit. Aby tato situace nezpůsobila ztrátu přijatého znaku, ukládá modul `ComInit` přijatý znak do interního buferu (parametr `PřijBuf`) a speciálním mechanismem vyvolává uživatelský podprogram tak, aby jej vyvolal vždy až po dokončení předchozího vyvolání. Tak je zajištěno, že se přijaté znaky neztrácí. **Vedlejším účinkem tohoto mechanismu je to, že na jedno vyvolání podprogramu může modul `ComRead` načíst i více znaků než jeden.** Počet vyvolání podprogramu tak může být menší než počet přijatých znaků.  
Při přerušovacím přístupu se parametr `PřijItr` musí zadat.

Doporučujeme používat raději přerušovací přístup, protože méně zatěžuje systém. Přijaté znaky může aplikace zpracovávat přímo v podprogramu přerušení. Reakce na přijatý rámec tak může být rychlejší než v případě dotazovacího přístupu.

### Vysílání znaků

Vysílání znaků se provádí pomocí modulu `ComWrite`. Tomuto modulu se zadá odkaz na bufer (databázovou proměnnou), ve kterém je připraven vysílaný rámec. Modul provede to, že znaky ze svého buferu přesune do interního vysílacího buferu modulu `ComInit` (parametr `VyslBuf`). Modul `ComInit` začne znaky okamžitě vysílat. O vysílání jednotlivých znaků se aplikace dále nemusí starat. Je-li to z nějakého důvodu potřeba, lze vyplnit parametr `VyslItr` a dostávat tak přerušení od vyslaného znaku i do aplikace. Modul `ComInit` pak vyvolává příslušný podprogram bezprostředně po odeslání každého znaku. Parametrem `PoslPřer` lze určit, zda se má vyvolávat podprogram přerušení po každém odvysílaném znaku nebo pouze po posledním. Poslední znakem se myslí situace, kdy modul `ComInit` při vysílání vybere celý bufer a nemá již co vysílat. Prakticky to znamená, že když modulem `ComWrite` vložíme celý rámec, dostaneme přerušení po odvysílání posledního znaku tohoto rámce. Pokud bychom ovšem ještě předtím, než se stačí rámec odvysílat, vložili další vysílané znaky opětovným voláním modulu `ComWrite`, dostaneme přerušení až po odvysílání všech těchto vložených znaků.

### Chyby linky

Je-li třeba kontrolovat chyby linky (špatná parita, ztracený přijímaný znak, ztracený stopbit, break), lze to udělat modulem `GetLSR`, který vrací hodnotu line status registru. Tento modul se vkládá do podprogramu na zpracování přerušení od chyby (je třeba zadat parametr `ErrItr`) příp. jinak, pokud přerušení od chyby nevyužíváme.

### Změna modemových signálů

Přerušení lze vyvolávat i při změně modemových signálů. V tom případě je nutné zadat parametr `ModemItr`. Do podprogramu na zpracování přerušení se umísťuje modul `GetMSR`, který načítá modem status registr.

### Volba velikosti přijímacího buferu

U dotazovacího přístupu musí být přijímací bufer minimálně tak velký, aby mohl pojmout všechny znaky přijaté mezi jednotlivými "dotazy" - vyprázdněními přijímacího buferu

pomocí modulu `ComRead`. Délka tedy závisí na komunikační rychlosti, periodě volání modulu `ComRead` a způsobu komunikace (např. u potvrzovacího protokolu se dá předpokládat, že se přijímací bufer napřed vyprázdní než se zašle odpověď, a teprve následně může přijít další rámec).

U přerušovacího přístupu musí být bufer schopen pojmout znaky přijaté za dobu nejdelšího možného trvání běhu podprogramu přerušení od přijatého znaku. Typicky se volí délka buferu stejná jako délka nejdelšího přijímaného rámce, což dává dostatečnou rezervu.

### **Celková koncepce uživatelské komunikace**

Popíšeme jednotlivé části, ze kterých se komunikace obecně skládá. Dále pak popíšeme, jak jednotlivé části realizovat - které moduly se mají použít a jakým způsobem.

- ♦ *Definice parametrů linky*  
Parametry se definují v hlavním modulu `ComInit`.
- ♦ *Příjem znaků přijímaného rámce*  
Na příjem znaků se používá modul `ComRead`. Vkládá se buď do podprogramu přerušení od přijatého znaku (přerušovací přístup) nebo do periodického procesu (dotazovací přístup).
- ♦ *Kontrola maximální časové prodlevy (timeoutu) mezi přijímanými znaky rámce*  
Na kontrolu časových prodlev se používají moduly `TmoInit`, `TmoStart`, `TmoStop` a `TmoCheck`. Modul `TmoInit` definuje jeden kanál timeoutu. Modulem `TmoStart` se startuje timeout, modulem `TmoStop` se timeout zastavuje. Modul `TmoCheck` slouží ke kontrole stavu timeoutu (zda již nastal nebo ne). Typicky se moduly používají v přerušovacím režimu. Modulu `TmoInit` se zadá odkaz na podprogram, který bude modulem spouštěn jako přerušení bezprostředně po uplynutí timeoutu. Do tohoto podprogramu se umísťuje kód na reakci na timeout.  
Modul `TmoCheck` slouží naopak pro dotazovací režim. Lze jej volat periodicky a zjišťovat, zda již timeout nastal. V přerušovacím režimu se modul nepoužívá, protože vypršení timeoutu je jednoznačně identifikováno vyvoláním podprogramu přerušení.
- ♦ *Rozpoznání začátku a konce rámce*  
Jsou v zásadě tři možnosti:  
- očekávání rámce  
Jedná se o nejjednodušší (nezabezpečený) způsob rozpoznávání rámců. Lze jej provádět u protokolů typu bod-bod master-slave, kdy se komunikuje způsobem dotaz-odpověď. Master zasílá dotazy nebo příkazy a na každý takový rámec následuje odpověď od slavea. Tehdy se dá předpokládat, že první znak, který přijde po odeslání rámce stanicí, je začátek rámce protistanici. Rámce mohou mít pevnou strukturu, ze které jednoznačně vyplývá délka rámce. Při příjmu (obvykle v podprogramu přerušení od přijatého znaku) se kontroluje délka rámce a dle ní se rozpoznává konec rámce.  
  
- speciální znaky začátku a konce rámce  
Začátek a konec rámce je označen speciálními znaky začátku a konce, které se na jiném místě v rámci již nesmí vyskytnout. Pokud se při sestavování uvnitř rámce objeví znak stejný jako znak začátku nebo konce rámce, musí být tento znak nějak nahrazen. Obvykle se provádí náhrada dvěma znaky, přičemž první znak má význam přesmykače, a druhý znak je definovaným způsobem překódovaný původní znak. Když se potom rámec na protistanici rozebírá, tak přesmykač oznamuje, že následný znak je překódován a je jej tedy nutno překódovat zpět. Na překódování speciálních znaků uvnitř rámce tam i zpět se používá modul `StrSubst`.  
Vlastní rozpoznávání začátku a konce rámce se provádí obvykle v podprogramu přerušení od přijatého znaku. Je ovšem nutno pamatovat na to, že z výše uvedených důvodů může modul `ComRead` na jedno vyvolání podprogramu načíst i více než jeden znak. Z toho důvodu nelze jednoduše kontrolovat poslední přijatý znak a porovnávat jej se znakem začátku rámce. V tomto případě se musí projít v cyklu všechny znaky, které modul v každém běhu načte.



Kontrola znaku konce rámce je jednodušší, protože za posledním znakem rámce bývá obvykle delší časová mezera, kdy protistrana čeká na odpověď. V takovém případě se lze spolehnout na kontrolu pouze posledního přijatého znaku.

*Pozn.: Problémům s načtením více znaků na jedno vyvolání podprogramu se lze vyhnout tím, že modulu ComRead se zadá za parametr Bufer místo matice jednoduchá proměnná. Modul pak na jedno vyvolání načte pouze jeden znak. Aplikace pak ovšem musí zajistit postupné ukládání celého přijímaného rámce do nějaké matice ve své režii.*

- časová mezera mezi rámci

Konec rámce je identifikován časovou mezerou mezi znaky. Je-li tedy mezera mezi přijímanými znaky větší než stanovená ukončovací mezera, jedná se o konec rámce. Takto zadané rámce se rozpoznávají tak, že po příjmu každého znaku se nastartuje timeout s hodnotou stanovené ukončovací mezery. Nastane-li timeout, vyvolá se podprogram přerušení od timeoutu a v tomto podprogramu se zpracuje přijatý rámeček.

- ♦ *Kontrola zabezpečení přijatého rámce*

Aby byl rámeček zabezpečen proti zkreslení při přenosu, provede se před vysláním nějaký kontrolní součet jednotlivých znaků dat rámce a takto vzniklé zabezpečení se přidá do vysílaného rámce. Protistrana po přijetí rámce provede znovu kontrolní součet znaků dat a porovná jej s přijatým zabezpečením. Pokud se zabezpečení liší, vyhodnotí se to jako chyba při přenosu rámce. Pojem "kontrolní součet" zde uvedený znamená obecně nějaký typ zabezpečení.

Pro vytváření zabezpečení vysílaného rámce se používá modul ChkCreate. Pro kontrolu zabezpečení přijatého rámce se používá modul ChkCheck. Těmto modulům se zadává konkrétní typ zabezpečení (CRC, suma, XOR apod.).

- ♦ *Rozbor rámce*

Rozborem rámce se myslí vypreparování dat z přijatého rámce do databázových proměnných. Slouží k tomu modul StrParse. Modul umí vypreparovat jednu hodnotu. Skládá-li se rámeček z více hodnot, řeší se to vícenásobným vložením modulu.

- ♦ *Reakce na přijatý rámeček*

Po rozboru - zpracování rámce následuje obvykle odpověď. To je typický případ reakce slavea na rámeček od mastera. Sestavení odpovědi a její vyslání lze povést přímo v podprogramu přerušení od přijatého znaku příp. v podprogramu přerušení od timeoutu mezi znaky.

V případě, kdy je stanice sama masterem, obvykle na přijatý rámeček neodpovídá. Master většinou zasílá žádosti a příkazy cyklicky, tj. sestavuje je a vysílá v nějakém periodickém procesu.

- ♦ *Sestavení vysílaného rámce*

Sestavením rámce se myslí zformátování hodnot nějakých databázových proměnných do formátu vysílaného rámce. Slouží k tomu modul StrFormat. Modul umí zformátovat jednu hodnotu. Skládá-li se rámeček z více hodnot, řeší se to vícenásobným vložením modulu.

- ♦ *Zabezpečení vysílaného rámce*

Pro tvorbu zabezpečení vysílaného rámce se používá modul ChkCreate viz "Kontrola zabezpečení přijatého rámce" výše.

- ♦ *Vyslání celého rámce*

Vysílaný rámeček se nejprve zformátuje (modulem StrFormat) do řetězce uloženého v databázové proměnné typu MI, kde jednotlivý znak odpovídá jedné buňce matice. Tento řetězec se výše najednou pomocí modulu ComWrite.

- ♦ *Vysílání jednotlivých znaků vysílaného rámce*

O vysílání jednotlivých znaků se aplikace nestará, je zabezpečeno interně modulem ComInit.

- ♦ *Kontrola maximálního času čekání na odpověď*

Na kontrolu časových prodlev se používají moduly TmoInit, TmoStart, TmoStop a TmoCheck. Viz "Kontrola maximální časové prodlevy (timeoutu) mezi přijímanými znaky rámce" výše.

### Příklad

Chceme komunikovat na kanálu RS232 (kanál č. 0) s těmito parametry: 9600 Bd, délka znaku 8 bitů, parita sudá, 1 stopbit. Maximální délka rámce je 256 znaků. Znaky přijímaných rámců budeme načítat v přerušovacím režimu v podprogramu Lib100.

```
ProcINIT:
:01000 SubInst 100 Instance podprog. přeruš. od přij. znaku
:02000 ComInit 0x0000, 0, 9600, 8, Sudá, 1, :01000, :NONE, :NONE, :NONE,
PrijsBuf, VyslBuf
```

```
Lib100: Podprogram přer. od přijatého znaku
... načítání přijímaných znaků pomocí modulu ComRead ...
```

Proměnné PrijsBuf a VyslBuf jsou matice MI[1,256] představující interní přijímací a vysílací bufer. Jelikož se jedná o 8-mi bitový přenos, připadají na jednu buňku matice dva znaky. Do každého buferu se tak vejdou až dva telegramy. Jelikož se při příjmu přijaté znaky bezprostředně z buferu vybírají pomocí modulu ComRead, je velikost přijímacího buferu víc než dostatečná. Při vysílání se vysílá vždy pouze jeden telegram, takže velikost vysílacího buferu je také dostatečná.

*Pozn.: Jelikož nelze na jednoduchém příkladu v manuálu osvětlit celou problematiku uživatelské komunikace, jsou v distribuční sadě PSP3 dodávány příklady aplikací uživatelské komunikace. Lze je nalézt v adresáři \EXAMPLES\USRCOM instalované sady PSP3. Přímou v jednotlivých aplikacích lze v editoru PSE zobrazit podrobnější informace příkazem <Popis>.*

<b>ComParams</b>	Přenastavení parametrů uživatelské komunikace za běhu
------------------	---

**Popis**

Modul umožňuje změnit za běhu parametry jako komunikační rychlost, délku znaku, paritu a počet stopbitů.

**Parametry**

<b>ComInit</b>	PAR	Návěští	Návěští modulu ComInit.
<b>Set</b>	PAR	Bit	Přenastavení parametrů komunikačního kanálu. Nastavením hodnoty parametru na "1" se přenastaví komunikační kanál dle zadaných parametrů. Modul po té bit vynuluje. Uvede-li se za parametr <code>None</code> , tak modul provádí přenastavení v každém běhu. V tomto případě je nutno modul umístit do podmíněné větve programu.
		None	
<b>Rychlost</b>	PAR	Konst	Komunikační rychlost [Bd].
		I	
		L	
		MI	
		ML	
<b>Délka</b>	PAR	Konst	Délka znaku [počet bitů].
		I	
		MI	
<b>Parita</b>	PAR	Konst	0 = žádná, 1 = sudá, 2 = lichá.
		I	
		MI	
<b>Stopbity</b>	PAR	Konst	Počet stopbitů (1, 2).
		I	
		MI	

**Příklad**

Nastavením bitu `High.0` na "1" se komunikační kanál přenastaví na parametry 19200 Bd, 8 bitů, sudá parita, 1 stopbit. Bit `High.0` je po té modulem vynulován.

```
ComParams :01000, High.0, 19200, 8, 1, 1
```

<b>ComRead</b>	Čtení z přijímacího buferu (uživatelská komunikace)
----------------	---

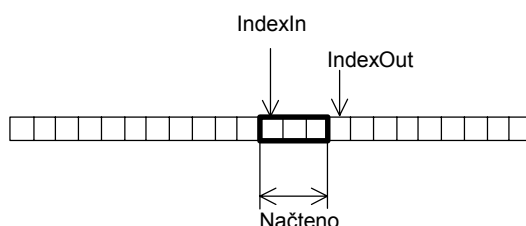
**Popis**

Modul slouží pro načtení přijatých znaků z přijímacího buferu.

**Parametry**

<b>ComInit</b>	PAR	Návěští	Návěští modulu ComInit.
<b>Bufer</b>	OUT	I	Bufer, do kterého se načtou znaky z přijímacího buferu modulu ComInit. Zadá-li se jednoduchá proměnná, načte modul jeden nebo žádný znak. Zadá-li se matice, pokusí se modul načíst tolik znaků, kolik se jich do matice vejde.
		MI	
<b>IndexIn</b>	IN	Konst	Index v matici Bufer, od kterého se začínají do matice zapisovat načtené znaky. Je-li Bufer jednoduchá proměnná, nemá parametr význam.
		I	
		MI	
<b>IndexOut</b>	OUT	I	Index v matici Bufer posunutý o počet načtených znaků.
		NONE	
<b>Načteno</b>	OUT	I	Počet načtených znaků.
		NONE	

Obrázek objasňující význam některých parametrů. Je na něm znázorněna matice Bufer. V silně orámované části jsou právě načtené znaky (v jednom konkrétním běhu modulu):



Modul se typicky umísťuje do podprogramu přerušení od přijatého znaku (viz popis modulu ComInit). Má-li modul za parametr Bufer zadánu matici, může se stát, že na jedno vyvolání podprogramu modul načte více jak jeden znak. Na to je potřeba pamatovat při vytváření aplikace. Zadá-li se za parametr Bufer jednoduchá proměnná, načte se na jedno vyvolání podprogramu vždy jeden znak. Je-li třeba průběžně kontrolovat jednotlivé znaky přijímaného rámce, je tedy jednodušší kontrolovat jednoduchou proměnnou než procházet v cyklu jednotlivé načtené znaky maticové proměnné. Maticová proměnná má zase tu výhodu, že může představovat přijímaný rámec, který modul postupně automaticky plní (zadá-li se za parametry IndexIn a IndexOut stejná proměnná).

**Příklad**

Budeme přijímat znaky v přerušovacím režimu. Přijímaný rámec budeme postupně plnit do matice TlgIn. Jakmile přijde poslední znak rámce, rámec zpracujeme a příjem nastavíme tak, aby se matice TlgIn mohla plnit zase od začátku dalším přijímaným rámcem. V procesu INIT definujeme komunikaci pomocí modulu ComInit s parametry: 9600 Bd, délka znaku 8 bitů, parita sudá, 1 stopbit.

ProcINIT:

```
:01000 SubInst 100 Instance podprog. přeruš. od přij. znaku
:02000 ComInit 0x0000, 0, 9600, 8, Sudá, 1, :01000, :NONE, :NONE, :NONE,
PrijsBuf, VyslBuf
```

```
Lib100:                                Podprogram přer. od přijatého znaku
                                         Postupné plnění TlgIn. TlgNdx se posouvá.
ComRead :02000, TlgIn, TlgNdx, TlgNdx, NONE
... Kontrola, zda je konec rámce do bitu @KonecRamce ...
If      @KonecRamce
|      ... Zpracování rámce. Délka rámce je v TlgNdx. ...
|      Let  TlgNdx = 0          Příště začneme zase od začátku
EndIf
```

<b>ComREmpty</b>	Dotaz na "prázdnost" přijímacího buferu (uživatelská komunikace)
------------------	--

**Popis**

Modul slouží ke kontrole přijímacího buferu. Vrací příznak, zda je bufer prázdný, navíc vrací ještě počet vložených znaků v buferu.

**Parametry**

<b>ComInit</b>	PAR	Návěští	Návěští modulu ComInit.
----------------	-----	---------	-------------------------

<b>Prázdný</b>	OUT	Bit	Příznak, zda je bufer prázdný.
		NONE	

<b>Vloženo</b>	OUT	I	Počet vložených znaků v buferu.
		NONE	

**Příklad**

Chceme zjišťovat počet znaků v přijímacím buferu. Výsledek budeme zapisovat do proměnné `ZnakuIn`. Hlavní modul komunikace `ComInit` je vložen do procesu `INIT` a má návěští :01000.

```
ComREmpty :01000, NONE.0, ZnakuIn
```

<b>ComSetLCR</b>	Nastavení line control registru (uživatelská komunikace)
------------------	--

**Popis**

Modul nastavuje line control registr obvodu sériové linky.

**Parametry**

<b>ComInit</b>	PAR	Návěští	Návěští modulu ComInit.
----------------	-----	---------	-------------------------

LCR	IN	Konst	Nastavovaná hodnota registru.	
		I	Význam jednotlivých bitů:	
		MI	Bity	Význam
			1, 0	Délka slova: 00 = 5 bitů 01 = 6 bitů 10 = 7 bitů 11 = 8 bitů
			2	Počet stopbitů: 0 = 1 stopbit 1 = 2/1.5 stopbitů. (hodnota 1.5 platí pro nastavení délky slova 5 bitů) Počet stopbitů 1.5 nepodporují všechny typy komunikačních obvodů.
			5, 4, 3	Typ parity: xx0 = bez parity 001 = lichá 011 = sudá 101 = paritní bit je vždy "1" 111 = paritní bit je vždy "0"
			6	SOUT režim: 0 = normální 1 = nastaví TxD do "1", což způsobí na protějšku detekci "break"
			7	Funkce DLAB (nastavuje přístup k alternativním registrům na adresách +0, +1): 0 = normální registry (+0=THR, +1=IER) 1 = nastavení předděličky DL (+0=DL-lo, +1=DL-hi)

**Příklad**

Komunikujeme 9-bitovým protokolem. Začátek rámce se rozlišuje pomocí nejvyššího bitu. Znak označující začátek rámce má tento bit nastaven do "1", ostatní znaky mají tento bit nastaven do "0". Délka slova je v tomto případě nastavena na 8, devátým bitem znaku je paritní bit. Pomocí modulu jej budeme nastavovat na požadovanou hodnotu.

```

If          @StartChar, :NONE
| ComSetLCR :01010, 0x002B
Else       :NONE
| ComSetLCR :01010, 0x003B
EndIf

```

Je-li bit @StartChar v "1", znamená to, že se bude vysílat první znak rámce, proto modulem ComSetLCR nastavíme paritní bit na "1". V opačném případě nastavíme paritní bit na "0".

<b>ComSetMCR</b>	Nastavení modem control registru (uživatelská komunikace)
------------------	---

**Popis**

Modul nastavuje modem control registr obvodu seriové linky. Tímto registrem se řídí stav modemových signálů.

**Parametry**

ComInit	PAR	Návěští	Návěští modulu ComInit.
---------	-----	---------	-------------------------

MCR	IN	Konst	Nastavovaná hodnota registru.	
		I	Význam jednotlivých bitů:	
		MI	Bit	Význam
			0	DTR signál
			1	RTS signál
			2	OUT1, nepoužívá se. Nejlépe ponechat hodnotu "0".
			3	OUT2, "1" povoluje generování přerušení komunikačního obvodu. Ponechat hodnotu "1", jinak nebudou fungovat přerušení.
			4	LOOP - lokální smyčka, všechny odeslané znaky se rovnou zpět přijímají, ven nejde nic. Ponechat hodnotu "0".

**Příklad**

Komunikujeme na kanálu RS232, na který máme připojen převodník na RS485. Převodník se řídí signálem DTR. Před vysláním se musí nastavit DTR do "1", aby se vybudila linka, po ukončení vysílání se musí DTR nastavit do "0", aby se linka odbudila. Vysílání rámce budeme pro jednoduchost provádět periodicky s periodou 5 s. Budeme jej provádět pomocí modulu ComWrite. Před vyvoláním modulu nejprve nastavíme signál DTR do "1", pomocí modulu ComSetMCR. Dokončení vysílání budeme detekovat přerušením od posledního vyslaného znaku. V podprogramu, který se tímto přerušením vyvolá, modulem ComSetMCR signál zase DTR nastavíme zpět do "0". Modulu ComInit musíme v parametru Režim nastavit, aby přerušení od vyslaného znaku vyvolával pouze po posledním vyslaném znaku.

ProcINIT:

```
:01000 SubInst 100 Instance podprog. přeruš. od příj. znaku
:01001 SubInst 101 Instance podprog. přeruš. od vysl. znaku
:02000 ComInit 0x0001, 0, 9600, 8, Sudá, 1, :01000, :01001, :NONE, :NONE,
PrijsBuf, VyslBuf
```

Proc00:

Periodický proces 5s

```
ComSetMCR :02000, 0x0009 "Nahození" DTR před vysláním
... vyslání rámce pomocí modulu ComWrite ...
```

Lib100:

Podprogram přer. od přijatého znaku

```
... načítání přijímaných znaků pomocí modulu ComRead ...
```

Lib101:

Podprogram přer. od vyslaného znaku

```
ComSetMCR :02000, 0x0008 "Shození" DTR po posl. vyslaném znaku
```



**ComSetXfc****Nastavení rozhraní komunikačního kanálu****Popis**

Modul nastavuje rozhraní u těch komunikačních kanálů, u nichž je možno fyzické rozhraní programově zvolit z více možností. Je nutno vybrat jednu z možností, které daný komunikační kanál podporuje - viz popis I/O konfigurace konkrétního systému.

**Parametry**

Kanál	PAR	Konst	Číslo komunikačního kanálu (0 .. 32).
-------	-----	-------	---------------------------------------

Rozhraní	IN	Konst	Volba rozhraní:	
		I	Hodnota	Význam
		MI	0 (RS232)	Aktivovat rozhraní RS232
			1 (RS485)	Aktivovat rozhraní RS485
			2 (IrDA)	Aktivovat infračervené rozhraní IrDA

**Příklad**

Chceme aktivovat infračervené rozhraní IrDA (namísto implicitního RS232) na komunikačním kanále 0 systému ADOS100 (který takovou volbu podporuje):

```
ComSetXfc 0, IrDA
```

<b>ComWEmpty</b>	Dotaz na "prázdnost" vysílacího buferu (uživatelská komunikace)
------------------	---

**Popis**

Modul slouží ke kontrole vysílacího buferu. Vrací příznak, zda je bufer prázdný, navíc vrací ještě počet vložených znaků v buferu.

**Parametry**

<b>ComInit</b>	PAR	Návěští	Návěští modulu ComInit.
----------------	-----	---------	-------------------------

<b>Prázdný</b>	OUT	Bit	Příznak, zda je bufer prázdný.
		NONE	

<b>Vloženo</b>	OUT	I	Počet vložených znaků v buferu.
		NONE	

**Příklad**

Chceme zjišťovat počet znaků ve vysílacím buferu. Výsledek budeme zapisovat do proměnné `ZnakuOut`. Hlavní modul komunikace `ComInit` je vložen do procesu `INIT` a má návěští :01000.

```
ComWEmpty :01000, NONE.0, ZnakuOut
```

<b>ComWrite</b>	Zápis do vysílacího buferu (uživatelská komunikace)
-----------------	---

**Popis**

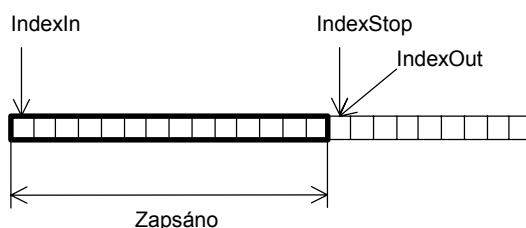
Modul slouží pro zápis vysílaných znaků do vysílacího buferu.

**Parametry**

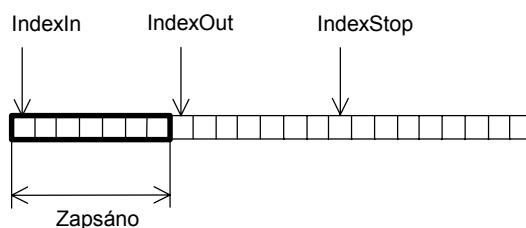
<b>ComInit</b>	PAR	Návěští	Návěští modulu ComInit.
<b>Bufer</b>	IN	I	Bufer, ze kterého se zapisují znaky do interního vysílacího buferu modulu ComInit. Zadává se jednoduchá proměnná I nebo řádková matice MI[1, n].
		MI	
<b>IndexIn</b>	IN	Konst	Index v matici Bufer, od kterého začínají vysílaná data. Je-li Bufer jednoduchá proměnná, nemá parametr význam.
		I	
		MI	
<b>IndexStop</b>	IN	Konst	Index v matici Bufer, před kterým končí vysílaná data. Je-li Bufer jednoduchá proměnná, nemá parametr význam.
		I	
		MI	
<b>IndexOut</b>	OUT	I	Index v matici Bufer posunutý o počet zapsaných znaků.
		NONE	
<b>Zapsáno</b>	OUT	I	Počet zapsaných znaků.
		NONE	

Počet zapisovaných znaků:

- ♦ *Bufer je jednoduchá proměnná typu I*  
Zapisuje se jeden znak.
- ♦ *Bufer je řádková matice MI[1, n]*  
Na obrázcích je znázorněna matice Bufer. V silně orámované části jsou znaky zapsané do vysílacího buferu (v jednom konkrétním běhu modulu).
  - a) Povedlo se zapsat všechny znaky (normální stav):



- b) Povedlo se zapsat pouze část znaků:  
Situace běžně nenastává. Může nastat pouze v případě, kdy se interní výstupní bufer modulu ComInit přeplní (znaky se zapisují rychleji než se stačí vysílat a kapacita buferu už nestačuje).



Dojde-li při zápisu do vysílacího buferu k jeho přeplnění, modul zapíše pouze tolik znaků, kolik se do vysílacího buferu vejde. Opakovaným voláním modulu `ComWrite`, lze zapsat i zbývající znaky. V tomto případě lze s výhodou využít toho, že se za parametry `IndexIn` a `IndexOut` zadá stejná proměnná. Potom stačí pouze opakovaně volat modul `ComWrite`, dokud se nevyšle vše (dokud `IndexOut` nedosáhne hodnoty `IndexStop`). V praxi je ovšem jednodušší navrhnout velikost vysílacího buferu modulu `ComInit` s dostatečnou rezervou (větší než je nejdelší vysílaný rámec) a vysílat rámce s dostatečnou časovou prodlevou, aby se vždy stihnul odovysílat předchozí rámec ještě před zahájením vysílání dalšího. U potvrzovaných protokolů je tato časová mezera dána již přicipem komunikace, kdy po odovysílání rámce master čeká na odpověď.

### Příklad

Budeme vysílat rámce po seriové lince. Vysílaný rámec bude připraven v matici `TlgOut`, jeho délka bude v proměnné `LenOut`. Rámce budeme vysílat periodicky s periodou 2 s. Komunikační kanál je definován v procesu `INIT` pomocí modulu `ComInit`, který má návěští :01000.

```
Proc00:                                     Perioda 2 s
... vytvoření rámce (v proměnné TlgOut) ...
ComWrite :01000, TlgOut, 0, LenOut, NONE, NONE
```

<b>Cond</b>	Podmíněný příkaz - trvalý
-------------	---------------------------

**Popis**

Modul **Cond** je obdobou příkazu **If**. Realizuje podmíněný příkaz typu **Cond-EndCond** nebo **Cond-ElseCond-EndCond**. Rozdíl oproti **If** příkazu je v tom, že příkaz **Cond** testuje podmínku pouze jednou po restartu. Pozdější změna podmínky již nemá vliv. Druhým rozdílem oproti příkazu **If** je to, že moduly, které jsou umístěny v neaktivní větvi příkazu **Cond**, nejen, že nejsou vykonávány, ale nejsou na začátku ani inicializovány. Jsou tak trvale vyřazeny z činnosti a systém se chová, jako by tyto moduly nebyly vůbec vloženy. Tato vlastnost je důležitá pro to, aby se daly podmíněně vyřazovat i ty moduly, které by jinak ve své inicializaci provedly činnosti ovlivňující následné chování systému. Typickým příkladem jsou komunikační moduly (**LCDTTY**, **LCD485**, **LCDG485**, **LCDGTTY**, **ComInit**, **Modem**, apod.), které si v inicializaci pro sebe vyhradí některý komunikační kanál. Chceme-li mít možnost dočasně používat tento komunikační kanál pro něco jiného, např. pro komunikaci se servisním počítačem s PSP3, můžeme takovéto moduly umístit do větve příkazu **Cond**. Použití příkazu **If** pro tento případ nepomáhá, protože příkaz **If** neblokuje inicializaci modulů.

Za běhu stanice se příkaz **Cond** "přepíná" tak, že se nastaví podmínka na příslušnou hodnotu a pak se provede restart stanice, aby se podmínka v modulu **Cond** projevila. Podmínku můžeme určovat buď hodnotou bitu databázové proměnné nebo stavem DIP přepínače, zadáme-li do bitového parametru **NONE**.

*Pozn.: Vyřazujeme-li konstrukci Cond-ElseCond-EndCond modul, na který se odkazují jiné moduly v aplikaci (např. ComInit), musíme současně vyřadit konstrukci Cond-ElseCond-EndCond i všechny tyto odkazující se moduly (např. ComRead, ComWrite, ...). Jinak se systém může chovat nedefinovaně.*

**Parametry**

Podmínka	IN	Bit	Podmínka provádění příkazů/modulů mezi <b>Cond</b> a následujícím <b>ElseCond</b> nebo <b>EndCond</b> . Hodnota "1" způsobí provádění těchto příkazů /modulů.
		NONE	

DIPKanál	PAR	Konst	Číslo kanálu DIP přepínače. Lze volit tyto kanály:					
			Číslo	Kanál	Popis			
			0	SW	SW DIP přepínače. Stavů přepínačů se zapisují do jednotlivých bitů proměnné DIP. Bit 0 odpovídá přepínači č. 1, bit 1 přepínači č. 2, atd.			
			1	HW	HW DIP přepínače. Všechny HW přepínače nejsou softwareově přístupné. Dle typu procesní stanice lze načítat buď pouze 1. přepínač (do bitu 0) nebo žádný.			
			1 000	USER	Uživatelské přepínače. Do tohoto kanálu se mapují pouze ty přepínače, které jsou uživatelsky využitelné. Tj. jsou přístupné a systém je nijak nevyužívá. Mapování přepínačů do proměnné DIP: <table><tr><td>Přepínač</td><td>Bit prom. DIP</td></tr><tr><td>SW-10.</td><td>0</td></tr><tr><td>HW-1.</td><td>1</td></tr></table>	Přepínač	Bit prom. DIP	SW-10.
Přepínač	Bit prom. DIP							
SW-10.	0							
HW-1.	1							

DIPBit	PAR	Konst	Číslo bitu DIP přepínače - číslováno od nuly. Bit 0 odpovídá přepínači č. 1, bit 1 přepínači č. 2, atd. DIP přepínač se používá jako alternativní podmínka provádění, je-li parametr <b>Podmínka</b> zadán <b>NONE</b> . Stav ON přepínače způsobí provádění příkazů/modulů mezi <b>Cond</b> a následujícím <b>ElseCond</b> nebo <b>EndCond</b> .
--------	-----	-------	---

<b>Návěští</b>	PAR	Návěští	Návěští následujícího příkazu <b>ElseCond</b> nebo <b>EndCond</b> - automatické, lokální, je tedy generované automaticky.
----------------	-----	---------	---

**Příklad**

Na stanici máme komunikační kanál RS485 (číslo 1) obsazen komunikací s nějakým zařízením. Na komunikační kanál RS232 (číslo 0) máme připojen terminál APT110. Potřebujeme, mít možnost komunikaci s terminálem dočasně zrušit a místo terminálu na tento kanál připojit servisní počítač s PSP3. Modul pro komunikaci s terminálem tedy dáme do podmíněné větve **Cond-EndCond**. Pro přepínání mezi komunikací s terminálem a komunikací s PSP3 použijeme DIP přepínač SW č. 10. Zvolíme tedy DIPKanál=SW a DIPBit=9 (9. bit odpovídá přepínači č. 10):

```
Cond          NONE.0, SW, 9, :NONE
LCDTTY       0, 1, 9600, 1
EndCond
```

Je-li DIP přepínač v poloze ON, tak komunikujeme s terminálem, jinak je komunikační kanál č. 0 uvolněný pro komunikaci s PSP3.

<b>DayPlan</b>	Denní časový plán
----------------	-------------------

**Popis**

Modul provádí plánování hodnoty do databázové proměnné **Výstup** z matice hodnot **Hodnoty** a časového plánu **Časy**. Pomocí tohoto modulu lze vytvořit zvláštní průběhy plánu pro každý den v týdnu (příp. skupinu dnů). Pomocí jediného modulu **DayPlan** tak lze realizovat  $n$  různých plánů..

Modul se aktivuje jen ve dnech nastavených v parametrech 0.KódDne až 7.KódDne (pondělí, úterý, ...). Vyjímečné dny (svátky atd.) jsou uloženy v matici **Svátky**. Každý parametr  $x$ .KódDne odpovídá jednomu řádku v maticích **Časy** a **Hodnoty**.

Je-li nastaven příznak **Plánování**, je plánování lineární. To znamená, že mezi dvěma časovými zlomy je hodnota **Výstup** lineárně interpolována tak, aby výchozí bod odpovídal plánu prvního časového zlomu a koncový bod odpovídal plánu druhého časového zlomu. Při úrovním plánování je při přechodu přes časový bod, hodnota **Výstup** změněna na odpovídající hodnotu v matici **Hodnoty**. Mezi časovými body pak zůstává hodnota konstantní.

**Parametry**

<b>Režim</b>	PAR	Konst	Způsob jakým modul pracuje
--------------	-----	-------	----------------------------

<b>Plánování</b>	Konst	Je-li tento příznak nastaven, je plánování lineární. Jinak se používá úrovním plánování.
------------------	-------	--

<b>Odřádku</b>	Konst	První řádkový index matic <b>Časy</b> a <b>Hodnoty</b> . Daného modulu se pak týkají řádky <b>Odřádku</b> až ( <b>Odřádku</b> + <b>Počet</b> -1)
----------------	-------	--

<b>Počet</b>	PAR	Konst	Počet jednotlivých plánů <1..8>. Tomuto počtu musí odpovídat počet řádků matic <b>Časy</b> a <b>Hodnoty</b> . Každý řádek plánů se zpracovává ve dni (dnech) zadaných odpovídajícím parametrem $x$ .KódDne ( $x < 0$ , <b>Počet</b> -1).
--------------	-----	-------	--

<b>x.KódDne</b>	PAR	Výběr	( $x < 0$ , 7)
-----------------	-----	-------	----------------

Maska nastavení dnů v týdnu, pro které modul pracuje. V závorce je kód dne, výsledná maska je dána součtem zvolených dnů.

název	den	maska (dec)	maska (hex)
Pondělí	pondělí	1	0x0001
Úterý	úterý	2	0x0002
Středa	středa	4	0x0004
Čtvrtek	čtvrtek	8	0x0008
Pátek	pátek	16	0x0010
Sobota	sobota	32	0x0020
Neděle	neděle	64	0x0040
Svátek	svátek	128	0x0080
User0	spec.den 0	256	0x0100
User1	spec.den 1	512	0x0200
User2	spec.den 2	1024	0x0400
User3	spec.den 3	2048	0x0800
User4	spec.den 4	4096	0x1000
User5	spec.den 5	8192	0x2000
User6	spec.den 6	16384	0x4000
User7	spec.den 7	32768	0x8000

<b>Svátky</b>	IN	MI	Matice má rozměr $[3 \times n]$ kde $n$ je počet svátků, které chceme zadat.
---------------	----	----	--

Každý svátek je dán dnem, měsícem a typem dne (viz. KódDne). Typ dne udává, jak se má tento svátek chápat. Např. jako by to byla neděle. Může to však být i zcela zvláštní typ dne. K tomu slouží uživatelsky zadaný typ dne Svátek, User0 až User7.

Význam řádků matice je následující:

0 - den

1 - měsíc

2 - maska (součet jednotlivých masek z KódDne). Lze nastavit např. všechny všední dny.

Aktuální typ dne modul zjistí nejprve z datumu (např. úterý), poté se kontroluje v matici Svátky, zda nesouhlasí některý sloupec s aktuálním datem. Pokud ano (tzn. "je svátek"), nebere se v úvahu aktuální typ dne, nýbrž maska typů dne zadaná v tomto sloupci matice Svátky. Plán se potom určuje z toho řádku matice, který má v parametru x.KódDne nastaven příslušný typ dne. Jinými slovy, v normální dny modul plánuje dle řádku, pro který má např. nastaveno pondělí a zrovna je pondělí, a ve dnech svátků modul plánuje z řádku, pro který má nastaven alespoň jeden typ dne shodující se s některým v masce typů dne daného svátku (viz výše).

Je-li maska svátku nastavena na nulu, tak modul daný svátek ignoruje.

Pro zvláštní období v roce (např. prázdniny) můžeme definovat speciální chování plánovače. K tomu je zapotřebí modul **Holiday**, který určuje pro každý den tohoto období (prázdnin) typ dne, jak má být tento den chápán (např. jako sobota nebo některý uživatelský typ). V proměnné Svátky je potom třeba vyhradit jeden sloupec navíc, do kterého zapisuje hodnotu modul **Holiday** v každý den během období prázdnin. Tento sloupec inicializujeme nulami, což má význam "nepoužitý".

Pokud nepožadujeme zvláštní plánování během svátků nebo prázdnin, musíme nechat matici nevyplněnou (tj. samé nuly) a matice může mít pouze jeden sloupec.

<b>Časy</b>	IN	ML	Matice časových zlomů o velikosti $[N \times M]$ , kde $N$ je větší nebo rovno Počet a $M$ je počet zlomů.
-------------	----	----	--

Každý prvek obsahuje čas od začátku dne v sekundách. Každému prvku v matici odpovídá příslušný prvek v matici Hodnoty. V příslušný časový okamžik se plánuje hodnota z Hodnoty do Výstup.

<b>Hodnoty</b>	IN	MI	Matice hodnot o velikosti $[N \times M]$ , kde $N$ je větší nebo rovno Počet.
		ML	
		MF	

<b>Výstup</b>	OUT	I	Databázová proměnná, do které se uloží plánovaná hodnota.
		L	
		F	

### Příklad

```
DayPlan      0x0000, 2, 0x001F, 0x00E0, 0x0000, 0x0000, 0x0000,
              0x0000, 0x0000, 0x0000, Holy, TPlan, Plan, T1
```

Jsou dva různé plány: pro všední dny a pro víkendy. Všední dny se plánují podle 1. řádku matic TPlan a Plan, víkendy se plánují podle 2. řádku. Svátky se plánují dle proměnné Holy. Aktuální plán je v proměnné T1.



<b>DeltaAvg</b>	Výpočet neklouzavého aritmetického průměru
-----------------	--

**Popis**

Modul je určen k výpočtu aritmetického průměru z  $N$  vzorků rozdílu  $x_1 - x_2$ . Výsledek je uložen do  $Y$ . Neklouzavý aritmetický průměr znamená, že modul nemění výstupní proměnnou v každém kroku, ale vždy na  $N$  kroků (běhů modulu).

**Parametry**

<b>X1</b>	IN	F	První sledovaná proměnná.
<b>X2</b>	IN	F	Druhá sledovaná proměnná.
<b>N</b>	PAR	Konst	Počet vzorků. Může být libovolně velký.
<b>Y</b>	OUT	F	Výsledný průměr.

**Příklad**

```
AnaIn  #0.0, F1, 0.004, 0.0, 0.020, 100.0, 1.0
```

```
AnaIn  #0.1, F2, 0.004, 0.0, 0.020, 100.0, 1.0
```

```
DeltaAvg  F1, F2, N, F
```

Nechť  $N$  je inicializovaná proměnná s hodnotou 10. Pak v proměnné  $F$  je aritmetický průměr z deseti vzorků rozdílu dvou analogových vstupů log. kanálů 0 a 1.

<b>DigIn</b>	Čtení šestnáctice digitálních vstupních signálů
--------------	---

**Popis**

Modul čte šestnáct digitálních vstupních signálů z logického kanálu DI do databázové proměnné typu `I`. Každý ze čtených signálů lze jednotlivě negovat.

**Parametry**

<b>Kanál</b>	IN	DI16	Číslo logického kanálu DI, z něž bude modul číst.
<b>Proměnná</b>	OUT	I	Jméno proměnné, do níž budou načtené (příp. ještě negované) signály uloženy.
<b>Negace</b>	PAR	Výběr	Lze nastavit jednotlivé příznaky negace pro každý ze vstupních bitů (signálů) č. 0 až 15.

**Příklad**

```
DigIn #2,DigVstup,0x0031
```

Čte logický kanál DI číslo 2 do proměnné `DigVstup`. Signály č. 0, 4 a 5 budou negovány, ostatní budou přeneseny přímo.

<b>DigOut</b>	Zápis šestnáctice digitálních výstupních signálů
---------------	--

**Popis**

Modul zapíše šestnáct digitálních výstupních signálů z databázové proměnné typu I na výstupní logický kanál DO. Každý ze zapisovaných kanálů lze jednotlivě negovat.

**Parametry**

<b>Proměnná</b>	IN	I	Jméno proměnné, která obsahuje signály, které modul zapíše do výstupního kanálu (příp. negaci).
<b>Kanál</b>	OUT	DO16	Číslo logického kanálu DO, na nějž bude modul psát.
<b>Negace</b>	PAR	Výběr	Lze nastavit jednotlivé příznaky negace pro každý z výstupních bitů (signálů) č. 0 až 15.

**Příklad**

```
DigOut DigVystup, #5, 0x0002
```

Zapíše obsah proměnné DigVystup na logický kanál č. 5. Bit č. 1 bude negován, ostatní budou zapsány přímo.

**DImp**

## Čtení a přepoččet impulzního vstupu

**Popis**

Modul **DImp** čte údaj z impulzního vstupu a přepočte ho na fyzikální rozměr. Modul spolupracuje s modulem pro obsluhu 16 impulzních vstupů **Impln**. Modul **Impln** musí být umístěn v rychlém procesu (ProcQUICK) a s ním spolupracující moduly **DImp** (1 až 16 exemplářů) v kterémkoli řádném procesu (Proc00 až Proc15).

Modul **DImp** zpracovává údaje z čidel s impulzním výstupem typu "1 impuls = N fyzikálních jednotek". Na základě této hodnoty *N* (konstanty čidla) odvozuje stav průběžného počítadla (např. počítadlo vodoměru, plynoměru, elektroměru atd.) a odhaduje okamžitou hodnotu.

Zatímco stav počítadla je velmi přesný (chyba měření způsobená procesní stanicí je menší než 0.00001%), je odhad okamžité hodnoty zatížen poměrně velkou chybou, kterou lze navíc poměrně těžko odhadnout - závisí totiž zejména na četnosti a pravidelnosti impulsů. Obvykle se tato chyba pohybuje v rozsahu 1 - 10% i přesto, že modul **DImp** disponuje relativně dokonalým korekčním mechanismem. Odhad okamžité hodnoty je proto třeba brát především jako informativní údaj a pracovat především se stavem počítadla.

Aby se zamezilo vzniku zaokrouhlovacích chyb, které nastávají při vysokých hodnotách počítadla a nízkých hodnotách přírůstku, používá se pro sumaci speciální algoritmus. Ten zajišťuje, že celková suma počítadla nemá zaokrouhlovací chybu, ale jako vedlejší efekt se mohou projevit "nepravidelné" přírůstky počítadla, které neodpovídají přesným přírůstkům v proměnné *Delta*.

**Parametry**

<b>Impulz</b>	PAR	Návěští	Návěští modulu <b>Impln</b> , který zajišťuje prvotní zpracování šestnáctice impulzních vstupů.
---------------	-----	---------	---

Toto návěští je globální (modul **Impln** je v ProcQUICK zatímco modul **DImp** v Proc00 až Proc15) a je ho tedy třeba zadat ručně.

<b>Index</b>	PAR	Konst	Číslo signálu z šestnáctice obsluhovaného modulem <b>Impln</b> (0 až 15).
--------------	-----	-------	---

<b>Delta</b>	OUT	F	Jméno proměnné, která bude obsahovat přírůstek průběžného počítadla od minulého běhu. Nechceme-li počítat přírůstek, parametr lze zadat <b>NONE</b> .
		NONE	

<b>Suma</b>	OUT	F	Jméno proměnné, která bude obsahovat průběžné počítadlo. Nechceme-li udržovat počítadlo, parametr lze zadat <b>NONE</b> .
		NONE	

<b>Okamžitá</b>	OUT	F	Jméno proměnné, která bude obsahovat odhad okamžité hodnoty v jednotkách za hodinu. Nechceme-li počítat okamžitou hodnotu, parametr lze zadat <b>NONE</b> .
		NONE	

<b>Konstanta</b>	IN	F	Konstanta měřidla, tedy <i>N</i> jednotek na 1 impuls. Zadáme-li parametr <b>NONE</b> , modul dosadí za konstantu hodnotu 1.
		NONE	

<b>SyncIn</b>	IN	Bit	Bit databázové proměnné, který se použije jako synchronizační, to znamená, že je-li v něm při vstupu do modulu jednička, změní modul jeho hodnotu na nulu a vynulují se vnitřní čítače. Proměnná <i>Suma</i> obsahuje hodnotu načítanou od předchozího volání modulu, ale od příštího volání se začne přičítat znovu od nuly.
---------------	----	-----	---

<b>SyncOut</b>	OUT	Bit	Bit databázové proměnné, který se nastaví do jedničky při příchodu synchronizačního pulzu v parametru <i>SyncIn</i> .
		NONE	

Je-li tedy po provedení modulu **DImp** v tomto bitu jednička, obsahuje proměnná *Suma* definitivní načítané hodnoty za poslední periodu synchronizace.

Parametr lze zadat **NONE**.

**Příklad**

```
ProcQUICK:
:01000  ImpIn 3, 0, 30, 0xFFFF, 0x0000
Proc00:
      DImp      :01000,2,Delta,Pocitadlo,Hodnota,Konst, Sync.0,Sync.0
      DImp      :01000,3,Delta2,Pocitadlo2,Hodnota2,Konst2,Sync.0, NONE
```

Modul ImpIn v rychlém procesu zpracovává 16 impulzních vstupů z logického kanálu DI č. 3. Maximální vzdálenost impulzů v čase je 30 sekund, aktivní je pouze náběžná hrana každého impulzu.

Modul **DImp** v procesu č. 0 zpracovává signál č. 2 modulu **ImpIn**. V proměnné **Konst** je konstanta měřidla (např. 0.125 m<sup>3</sup>/impulz). Proměnná **Pocitadlo** je průběžným počítadlem, proměnná **Delta** obsahuje přírůstek počítadla od minula, proměnná **Hodnota** obsahuje odhad okamžité hodnoty (dle příkladu v rozměru [m<sup>3</sup>/h]). Nastavením bitu proměnné **Sync.0** na "1" se vynuluje počítadlo v dalším běhu procesu. Synchronizační bit **Sync.0** je zřetězen (je zadán na místě parametru **SyncIn** i **SyncOut**) a využívá se v dalším modulu **DImp**, viz druhý řádek procesu č. 0 .

<b>DIP</b>	Načtení stavu DIP přepínače
------------	-----------------------------

**Popis**

Modul načítá zadaný DIP přepínač procesní stanice.

**Parametry**

Kanál	IN	Konst	Číslo kanálu. Lze volit tyto kanály:								
		I	Číslo	Kanál	Popis						
		MI	0	SW	SW DIP přepínače. Stavy přepínačů se zapisují do jednotlivých bitů proměnné DIP. Bit 0 odpovídá přepínači č. 1, bit 1 přepínači č. 2, atd.						
			1	HW	HW DIP přepínače. Všechny HW přepínače nejsou softwareově přístupné. Dle typu procesní stanice lze načítat buď pouze 1. přepínač (do bitu 0) nebo žádný.						
			1 000	USER	Uživatelské přepínače. Do tohoto kanálu se mapují pouze ty přepínače, které jsou uživatelsky využitelné. Tj. jsou přístupné a systém je nijak nevyužívá. Mapování přepínačů do proměnné DIP: <table><tr><th>Přepínač</th><th>Bit prom. DIP</th></tr><tr><td>SW-10.</td><td>0</td></tr><tr><td>HW-1.</td><td>1</td></tr></table>	Přepínač	Bit prom. DIP	SW-10.	0	HW-1.	1
Přepínač	Bit prom. DIP										
SW-10.	0										
HW-1.	1										

DIP	OUT	I	Výstupní proměnná - zapisované stavy přepínačů.
		MI	

**Příklad**

DIP SW, Prepinace

Stavy SW přepínačů se zapíší do proměnné Prepinace.

<b>DM_MUX3</b>	Obsluha analogového multiplexeru DM-MUX3/1
----------------	--

**Popis**

Multiplexer DM-MUX3/1 se společnými svorkami A/B připojuje na jeden analogový vstup řídicího systému, jeho řízení se provádí buďto dvěma digitálními výstupy, nebo jedním analogovým výstupem (funkční modul **DM\_MUX3** ošetřuje obě tyto varianty). Na svorkách multiplexeru A0/B0, A1/B1 a A2/B2 vzniká trojice samostatných analogových vstupů.

Příslušný analogový vstup řídicího systému může být hardwareově nastaven (pro všechny tři vstupy multiplexeru společně) buďto na měření teploty snímačem Ni1000, nebo na měření napětí či proudu ve zvoleném (pro všechny tři vstupy stejném) rozsahu. V prvním případě se pro načtení hodnot jednotlivých vstupů používá funkční modul **MuxNi1000** (pozor, modul **DM\_MUX3** musí být v tom případě navázán na logický vstupní kanál určený pro měření teploty). Ve všech ostatních případech se pro načtení hodnot jednotlivých vstupů používá funkční modul **MuxAnIn**.

**Parametry**

<b>Vstup</b>	IN	AI	Číslo vstupního logického kanálu AI, na který jsou připojeny společné svorky A/B multiplexeru.
--------------	----	----	--

<b>ŘízeníAO</b>	OUT	F	Hodnota řídicího napětí (ve Voltech - 0 až 10 V), které je třeba v daném kroku přivést na řídicí analogový výstup, zpravidla pomocí funkčního modulu <b>AnOut</b> - viz příklad. Při použití digitálního řízení multiplexeru lze za tento parametr dosadit <b>NONE</b> .
		MF	

<b>ŘízeníDO0</b>	OUT	Bit	Bit proměnné, do něž bude uložena logická hodnota, kterou je třeba v daném kroku přivést na řídicí digitální výstup, připojený ke svorce D0 multiplexeru. Zpravidla se k tomu používají funkční moduly <b>DigOut</b> nebo <b>BinOut</b> - viz příklad. Při použití analogového řízení multiplexeru lze za tento parametr dosadit <b>NONE</b> .
------------------	-----	-----	--

<b>ŘízeníDO1</b>	OUT	Bit	Bit proměnné, do něž bude uložena logická hodnota, kterou je třeba v daném kroku přivést na řídicí digitální výstup, připojený ke svorce D1 multiplexeru. Zpravidla se k tomu používají funkční moduly <b>DigOut</b> nebo <b>BinOut</b> - viz příklad. Při použití analogového řízení multiplexeru lze za tento parametr dosadit <b>NONE</b> .
------------------	-----	-----	--

**Umístění modulu**

Modul je třeba periodicky vyvolávat, a to tak, aby mezi skutečným vystavením hodnot parametrů **Řízení...** na fyzické výstupy (na konci vyvolání procesu) a příštím spuštěním procesu obsahujícího funkční modul **DM\_MUX3** uplynulo nejméně 900 ms. Zpravidla se tato podmínka splní umístěním modulu, jakož i modulu(ů) **AnOut/DigOut/BinOut**, realizujících fyzický výstup řídicích signálů, do procesu s periodou nejméně 1 s (předpokládáme-li skutečnou dobu vykonávání procesu kratší než 100 ms, což lze v případě pochybností u delších procesů ověřit funkčním modulem **StopWatch**).

Moduly **MuxAnIn/MuxNi1000**, sloužící k načtení hodnot jednotlivých multiplexovaných vstupů, lze potom umístit do libovolně rychlého procesu. Musíme si ovšem uvědomit, že jejich umístěním do seberychejšího procesu se nic nemění na tom, že vždy po dobu tří sekund (přesněji tří vyvolání procesu s modulem **DM\_MUX3**) zůstane hodnota získaná pomocí těchto modulů neměnná.

**Příklad**

```
Proc01 (perioda: 1 s)
10001:  DM_MUX3      #0.0, MUX_Ctrl, NONE.0, NONE.0
        AnOut        #0.0, MUX_Ctrl, 10V, 0.0, 10.0, 0.0, 10.0
```

```

Proc02 (perioda: 100 ms)
    MuxAnIn    10001:, 0, InpA0B0, 5V, 0.0, 5.0, 0.0, 5.0
    MuxAnIn    10001:, 1, InpA1B1, 5V, 1.0, 5.0, 30.0, 150.0
    MuxAnIn    10001:, 2, InpA2B2, 5V, 1.0, 5.0, 30.0, 150.0

```

Výše uvedený příklad demonstruje analogové řízení multiplexeru s periodou přepínání 1 sekunda. Multiplexer je připojen na vstup AI.0, jeho řízení je prováděno výstupem AO.0. V jiném procesu jsou načítány hodnoty jednotlivých vstupů multiplexeru do proměnných InpA0B0, InpA1B1 a InpA2B2. První ze vstupů je ukládán přímo ve voltech (0 až 5 V), ostatní jako hodnota fyzikální veličiny před (nějakým) připojeným převodníkem v rozsahu 30 (odpovídá napětí 1 V) až 150 (odpovídá napětí 5 V). Je třeba upozornit, že vzhledem ke skutečnosti diskutované v oddíle "Umístění modulu" se hodnota každé z výše uvedených proměnných bude měnit jen cca při každém třicátém vyvolání procesu Proc02, mezitím zůstane konstantní.

```

Proc01 (perioda: 1 s)
    10001:    DM_MUX3    #1.0, NONE, DOBits.0, DOBits.1
              DigOut     DOBits, #0, 0000

```

```

Proc02 (perioda: 10 s)
    MuxNi1000 10001:, 0, TempA0B0, 6180
    MuxNi1000 10001:, 1, TempA0B0, 5000
    MuxNi1000 10001:, 2, TempA0B0, 5000

```

Výše uvedený příklad demonstruje digitální řízení multiplexeru s periodou přepínání 1 sekunda. Multiplexer je připojen na vstup AI.0, jeho řízení je prováděno výstupy DO.0, DO.1. Pro výstup řídicích signálů byl zvolen funkční modul **DigOut**. Je třeba upozornit, že v takovém případě je třeba v celé aplikaci výstup na všechny ostatní digitální výstupy kanálu 0 realizovat prostřednictvím proměnné DOBits, aby nedocházelo k vzájemnému přepisování hodnot na výstupech.

V jiném procesu jsou načítány teploty z jednotlivých vstupů multiplexeru do proměnných TempA0B0, TempA1B1 a TempA2B2. První ze snímačů má citlivost 6180 ppm, ostatní 5000 ppm.

```

Proc01 (perioda: 1 s)
    10001:    DM_MUX3    #1.0, NONE, DOBits.0, DOBits.1
              BinOut     DOBits.0, NE, #0.0
              BinOut     DOBits.1, NE, #0.1

```

```

Proc02 (perioda: 10 s)
    MuxNi1000 10001:, 0, TempA0B0, 6180
    MuxNi1000 10001:, 1, TempA0B0, 5000
    MuxNi1000 10001:, 2, TempA0B0, 5000

```

Výše uvedený příklad demonstruje digitální řízení multiplexeru s periodou přepínání 1 sekunda. Multiplexer je připojen na vstup AI.0, jeho řízení je prováděno výstupy DO.0, DO.1. Pro výstup řídicích signálů byl zvolen funkční modul **BinOut**. Je třeba upozornit, že v takovém případě je třeba v celé aplikaci výstup na všechny ostatní digitální výstupy kanálu 0 realizovat buďto rovněž modulem BinOut, nebo prostřednictvím proměnné DOBits, aby nedocházelo k vzájemnému přepisování hodnot na výstupech.

V jiném procesu jsou načítány teploty z jednotlivých vstupů multiplexeru do proměnných TempA0B0, TempA1B1 a TempA2B2. První ze snímačů má citlivost 6180 ppm, ostatní 5000 ppm.



**EEFinish**

Dokončení zápisu do EEPROM v bezpečném režimu

**Popis**

Modul zkopíruje data, která byla zapsána do EEPROM pomocí modulů **EEWrite**, do záložní oblasti v EEPROM. Kopírují se všechna data zapsaná od posledního vyvolání modulu **EEFinish**, případně od startu systému. Zároveň se nuluje příznak v EEPROM, který udává, že probíhá aktualizace dat v základní oblasti EEPROM. Blíže viz rozbor režimů práce s EEPROM v popisu modulu **EEMode**.

Není-li modulem **EEMode** vybrán režim *Bezpečný*, nemá vyvolání modulu **EEFinish** žádný efekt.

**Doba provádění modulu**

Dobu provádění modulu **EEFinish** je možno přibližně určit jako dvojnásobek součtu doby provádění všech modulů **EEWrite**, které byly vyvolány od posledního vyvolání modulu **EEFinish**, případně od startu systému.

Není-li modulem **EEMode** vybrán režim *Bezpečný*, je doba provádění modulu **EEFinish** zanedbatelně krátká.

**Parametry**

Žádné.

**Příklad**

```
ProcINIT:
    EEMode    Bezpečný
Proc01:
    EEWwrite  10,Prom_LONG[0,0],1,1
    EEWwrite  14,Prom_INT[0,0],1,1
    EEFinish
```

Tento úsek programu zapíše dvě proměnné na adresy 10 a 14 v EEPROM v bezpečném režimu, modul **EEFinish** zajistí potřebnou kopii do záložní oblasti EEPROM.

<b>EEMode</b>	Nastavení režimu práce s EEPROM
---------------	---------------------------------

**Popis**

Modul nastaví režim, ve kterém se bude pracovat s EEPROM.

Modul **EEMode** musí být umístěn v procesu INIT, a to před prvním použitím kteréhokoliv z modulů **EERead**, **EEWrite** nebo **EEFinish**.

Pokud se modul **EEMode** do aplikace vůbec nevloží, pracují moduly **EERead**, **EEWrite** a **EEFinish** s EEPROM stejně, jako by byl modulem **EEMode** nastaven režim *Standard*.

**Režim Standard**

V tomto režimu se data do EEPROM zapisují přímo, bez jakéhokoliv zabezpečení konzistence. Pokud tedy dojde k výpadku napájení procesní stanice nebo k resetu právě v době, kdy se provádí modul **EEWrite**, může dojít k narušení konzistence dat. Některé bajty, ze kterých se zapisovaná hodnota skládá, mohou mít v EEPROM ještě starou hodnotu, případně mohou být ve vymazaném stavu. Po novém startu procesní stanice tedy modul **EERead** může načíst z EEPROM nesprávnou hodnotu.

Tento režim se může použít tehdy, pokud se do EEPROM zapisuje jen výjimečně (např. po změně konstant uživatelem z terminálu) a riziko výpadku během zápisu je akceptovatelné, nebo pokud se autor aplikace rozhodne v aplikaci naprogramovat vlastní schéma zabezpečení konzistence, které může být pro danou konkrétní aplikaci výhodnější než níže uvedené schéma režimu *Bezpečný*.

Výhodou režimu *Standard* je rychlejší přístup k EEPROM, není třeba používat modul **EEFinish** (pokud se přesto použije, nemá jeho vyvolání žádný efekt). Dalšími výhodami jsou možnost využití celé dostupné kapacity EEPROM a maximální využití životnosti EEPROM (jedno vyvolání modulu **EEWrite** představuje pro každou ze zapisovaných buněk jeden zápisový cyklus).

**Využitelná kapacita EEPROM** v režimu *Standard* je rovna dostupné kapacitě, kterou je možno najít v dokumentaci příslušného typu procesní stanice.

*Pozn.: U procesních stanic typu APT2100 je "dostupná kapacita" o 32 byte nižší než celková kapacita uváděná v dokumentaci APT2100. Těchto 32 byte je využito operačním systémem pro uložení konfigurace terminálu.*

**Režim Bezpečný**

V tomto režimu je použito následující schéma pro zabezpečení konzistence dat, zapisovaných do EEPROM, aby se odstranilo riziko popsané u režimu *Standard*:

Paměť EEPROM je rozdělena na dvě oblasti - základní a záložní. Moduly **EEWrite** provádějí zápis do základní oblasti, před zápisem se v EEPROM nastaví příznak, že probíhá aktualizace základní oblasti. Po ukončení zápisů se vyvolá modul **EEFinish**, který zapsaná data zkopíruje do záložní oblasti a nuluje příznak probíhající aktualizace základní oblasti. Pokud při startu systému modul **EEMode** najde v EEPROM nastavený příznak probíhající aktualizace základní oblasti (tzn. že došlo k ukončení běhu aplikace před vyvoláním **EEFinish** a data v základní oblasti mohou být nekonzistentní), provede kopii záložní oblasti do základní oblasti. Po vyvolání modulu **EEMode** tedy základní oblast v každém případě obsahuje konzistentní data.

Životnost EEPROM (měřená v počtu vyvolání modulu **EEWrite**) je oproti režimu *Standard* snížena o polovinu. To za předpokladu, že se modul **EEFinish** použije v aplikaci jen jednou, vždy po provedení všech modulů **EEWrite**. Častější vyvolávání modulu **EEFinish** životnost dále snižuje (jedno vyvolání modulu **EEFinish** představuje pro každou ze zapisovaných buněk v záložní oblasti jeden zápisový cyklus, dále ovšem představuje dva zápisové cykly pro jednu, stále stejnou, paměťovou buňku s příznakem probíhající aktualizace základní oblasti).

Režim *Bezpečný* je vhodné použít vždy, když se zápis dat do EEPROM provádí periodicky.

**Využitelná kapacita EEPROM** v režimu *Bezpečný* je o jeden byte zmenšená polovina dostupné kapacity. Údaj o dostupné kapacitě je možno najít v dokumentaci příslušného typu procesní stanice.

*Pozn.: "Dostupná kapacita" je oproti fyzické velikosti EEPROM snížena o rozsah adres, které využívá operační systém např. pro uložení konfigurací ethernetových rozhraní, nastavení terminálu APT2100G apod. Tato dostupná velikost se může dynamicky měnit. Chystáme-li se tedy využívat většinu fyzicky dostupné EEPROM, je důrazně doporučeno*

nejprve stanici kompletně nakonfigurovat (zejména co se týče případné komunikace po Ethernetu) a potom zjistit dostupnou kapacitu pomocí funkčního modulu **EESize**.

Doba provádění modulů

Zvolený režim práce s EEPROM výrazně ovlivňuje doby provádění modulů **EEWrite** a **EEFinish**. Přesné doby provádění modulů závisejí na více faktorech - kromě množství zapisovaných dat a typu EEPROM ještě na momentálním stavu řadiče EEPROM, na aktuálním obsahu dotyčné paměťové buňky a úrovni opotřebení EEPROM. Uvedené časy je tedy nutno považovat za orientační.

Modul	Typ dat	Kapacita EEPROM ≤2kB		Kapacita EEPROM >2kB	
		Režim Standard	Režim Bezpečný	Režim Standard	Režim Bezpečný
<b>EERead</b>	I	3.5 ms	3.5 ms	4.5 ms	4.5 ms
	L, F	7 ms	7 ms	9 ms	9 ms
<b>EEWrite</b>	I	3.5 ms	5.5 ms	8.5 ms	17.5 ms
	L, F	7 ms	9 ms	17 ms	26 ms
<b>EEFinish</b>	viz popis modulu <b>EEFinish</b>				

Při práci s maticemi nebo jejich výřezy se uvedené časy násobí součinem počtu řádků a sloupců zvoleného výřezu.

Práce s EEPROM je časově náročná. Musíme brát v úvahu, že po dobu provádění jednoho procesu se nemohou spouštět ostatní procesy s výjimkou procesů QUICK, Hlx a ITRx.

Máme-li tedy v aplikaci obyčejný proces s krátkou periodou (např. 100 ms), jehož provedení trvá cca 20ms, a chceme zaručit, že práce s EEPROM nenaruší spouštění tohoto procesu, musíme dbát na to, abychom jednotlivé operace s EEPROM rozdělili mezi více vyvolání vhodného procesu. Chceme-li například zapisovat v bezpečném režimu dvě proměnné typu I a jednu proměnnou typu L, vložíme do vhodného procesu (do jiného procesu s periodou 100÷500ms, do téhož procesu nebo do procesu IDLE) následující sekvenci:

```

IF @UpdateEE
  Switch StatusEE
    Case 0
      EEWrite 0,IntVar1
      EEWrite 6,IntVar2
    EndCase
    Case 1
      EEFinish
    EndCase
    Case 2
      EEWrite 2,LongVar
    EndCase
    Case 3
      EEFinish
      Let StatusEE = -1
      Let @UpdateEE = 0
    EndCase
  EndSwitch
  LET StatusEE = StatusEE+1
EndIf

```

Zápis do EEPROM se spouští nastavením bitu @UpdateEE z kteréhokoliv jiného procesu. Tímto postupem zaručíme, že doba provádění procesu, pracujícího s EEPROM nepřekročí ani v nejhorším případě (tj. vyvolání **EEFinish** při hodnotě 1 v proměnné StatusEE) 22 ms na procesních stanicích vybavených EEPROM s kapacitou do 2kB, resp. 70 ms v případě EEPROM s kapacitou větší než 2kB.

Daní za toto rozložení zápisů se současným zabezpečením konzistence je snížení životnosti EEPROM na čtvrtinu oproti režimu *Standard* (pro kompletní zápis dat se dvakrát vyvolává modul **EEFinish**).

**Parametry**

Režim	PAR	Výběr	Režim práce s EEPROM: 0 - Standard 1 - Bezpečný
-------	-----	-------	---

**Příklad**

EEMode Bezpečný

Nastaví režim práce s EEPROM na Bezpečný.

<b>EERead</b>	Čtení dat z EEPROM
---------------	--------------------

**Popis**

Modul načte databázovou proměnnou ze zadané adresy v EEPROM.

Modul pracuje s paměťovými buňkami v EEPROM na adresách *Adresa* až *Adresa+Rozměr-1* včetně, kde *Rozměr* je 2 pro proměnné typu *I*, 4 pro proměnné typů *L* a *F*, pro databázové matice se rozměr ještě násobí součinem parametrů *Řádků*×*Sloupců*.

*Adresa* každé z výše uvedených buněk musí být menší než "využitelná kapacita EEPROM" (definováno v popisu modulu **EEMode**), důsledky překročení této meze jsou nepředvídatelné.

Doba provádění modulu

Viz tabulku v popisu modulu **EEMode**.

**Parametry**

<b>Adresa</b>	PAR	Konst	Počáteční adresa v EEPROM
---------------	-----	-------	---------------------------

<b>Data</b>	IN	I	Proměnná, jejíž hodnota má být načtena z EEPROM. V případě databázové matice obsahuje i pozici levého horního rohu výřezu.
		L	
		F	
		MI	
		ML	
		MF	

<b>Řádků</b>	PAR	Konst	Počet řádků výřezu databázové matice. V případě jednoduché proměnné nemá hodnota tohoto parametru na nic vliv.
--------------	-----	-------	---

<b>Sloupců</b>	PAR	Konst	Počet sloupců výřezu databázové matice. V případě jednoduché proměnné nemá hodnota tohoto parametru na nic vliv.
----------------	-----	-------	---

**Příklad**

```
EERead 0,VarInt[0,0],1,1
EERead 2,MtxLong[0,0],2,2
```

Jednoduchá proměnná *VarInt* se načte z adresy 0 v EEPROM, čtyři prvky matice *MtxLong* počínaje prvkem *MtxLong[0,0]* se načtou z adresy 2 v EEPROM.

<b>EESize</b>	Zjištění velikosti uživatelsky přístupné EEPROM
---------------	---

**Popis**

Modul zjistí velikost té části EEPROM, která je použitelná pro moduly **EERead**, **EEWrite**, ... Jedná se o "dostupnou velikost" ve smyslu popisu modul **EEMode**.

Modul bere v úvahu velikost EEPROM využívané operačním systémem a odečítá ji od fyzické velikosti EEPROM.

**Parametry**

Velikost	OUT	I	Výstupní parametr, do kterého se uloží dostupná velikost EEPROM.
		L	
		MI	
		ML	

**Příklad**

```
EESize  VelEE
```

Do proměnné `VelEE` se uloží dostupná velikost EEPROM.

<b>EEWrite</b>	Zápis dat do EEPROM
----------------	---------------------

**Popis**

Modul zapíše databázovou proměnnou na zadanou adresu v EEPROM.

Modul pracuje s paměťovými buňkami v EEPROM na adresách Adresa až Adresa+Rozměr-1 včetně, kde Rozměr je 2 pro proměnné typu I, 4 pro proměnné typů L a F, pro databázové matice se rozměr ještě násobí součinem Řádků×Sloupců.

Adresa každé z výše uvedených buněk musí být menší než "využitelná kapacita EEPROM" (definováno v popisu modulu **EEMode**), důsledky překročení této meze jsou nepředvídatelné.

Rozsahy adres použitých různými moduly **EEWrite** v aplikaci by se neměly překrývat.

Je-li pomocí modulu **EEMode** zvolen režim Bezpečný, je po provedení zápisů nutno vyvolat modul **EEFinish**, jinak budou zapsané hodnoty při nejbližším restartu procesní stanice přepsány!

Doba provádění modulu

Viz tabulku v popisu modulu **EEMode**.

**Parametry**

<b>Adresa</b>	PAR	Konst	Počáteční adresa v EEPROM
---------------	-----	-------	---------------------------

<b>Data</b>	OUT	I	Proměnná, jejíž hodnota má být zapsána do EEPROM. V případě databázové matice obsahuje i pozici levého horního rohu výřezu.
		L	
		F	
		MI	
		ML	
		MF	

<b>Řádků</b>	PAR	Konst	Počet řádků výřezu databázové matice. V případě jednoduché proměnné nemá hodnota tohoto parametru na nic vliv.
--------------	-----	-------	---

<b>Sloupců</b>	PAR	Konst	Počet sloupců výřezu databázové matice. V případě jednoduché proměnné nemá hodnota tohoto parametru na nic vliv.
----------------	-----	-------	---

**Příklad**

```
EEWrite 0,VarInt[0,0],1,1
EEWrite 2,MtxLong[0,0],2,2
EEFinish
```

Jednoduchá proměnná VarInt se zapíše na adresu 0 v EEPROM, čtyři prvky matice MtxLong počínaje prvkem MtxLong[0,0] se zapíše na adresu 2 v EEPROM. První volná adresa v EEPROM je 18.

Není-li modulem **EEMode** zvolen režim Bezpečný, je možno vynechat volání modulu **EEFinish**.

<b>Else</b>	Pokračovací část podmíněného příkazu <b>If</b>
-------------	--

**Popis**

Příkaz **Else** uvozuje pokračovací část příkazu **If**. Tato část se provede, není-li splněna podmínka uvedená v příkazu **If**. Pokračovací část příkazu končí příkazem **EndIf**.

**Parametry**

Návěští	PAR	Návěští	Návěští následujícího příkazu <b>EndIf</b> . Toto návěští je automatické, lokální, generuje se tedy automaticky.
---------	-----	---------	--

**Příklad**

```

                If      Test.0, :00000      Je-li digitální signál
                                                (bit) nastaven,
                . . .                          proved' tuto sekvenci
                                                příkazů/modulů,
:00000      Else  :00001                    a není-li nastaven,
                . . .                          proved' tuto sekvenci
:00001      EndIf                          Následující
                                                příkazy/moduly prováděj
                                                vždy
```



<b>ElseCond</b>	Pokračovací část podmíněného příkazu <b>Cond</b>
-----------------	--

**Popis**

Příkaz **ElseCond** uvozuje pokračovací část příkazu **Cond**. Tato část se provádí, není-li splněna podmínka uvedená v příkazu **Cond**. Pokračovací část příkazu končí příkazem **EndIf**.

Upozorňujeme, že podmínka v příkazu **Cond** se kontroluje pouze pro restartu. Za běhu již nelze chování příkazu **Cond-ElseCond-EndCond** změnit.

**Parametry**

Návěští	PAR	Návěští	Návěští následujícího příkazu <b>EndCond</b> . Toto návěští je automatické, lokální, generuje se tedy automaticky.
---------	-----	---------	--

**Příklad**

```
Cond      RunLcd.0, -1, :NONE
LCDTTY    0, 1, 9600, 1
ElseCond
Modem     0, 9600, 8, 0, 1, 0x0200, 10, "", "", "", 0, 0
EndCond
```

Na komunikační kanál č. 0 připojujeme buď terminál APT110 nebo modem. Má-li bit RunLcd.0 hodnotu "1", komunikujeme s terminálem APT110. Je-li hodnota bitu "0", komunikujeme s modemem. Po každé změně bitu musíme provést reset stanice, aby se změna projevila.

<b>EndCase</b>	Ukončení větve přepínače
----------------	--------------------------

**Popis**

Příkaz ukončuje větev přepínače. Podrobnější popis je uveden v popisu příkazu **Case** a příkazu **Switch**.

**Parametry**

žádné

**Příklad**

	Switch Test, :00010	Přepínač podle hodnoty proměnné Test
	Case 0, :00000	Větev pro hodnotu proměnné=0
	. . .	Příkazy/moduly větve
:00000	EndCase	Konec větve
	Case 1, :00001	Větev pro hodnotu proměnné=1
	. . .	
:00001	EndCase	
	Case 2, :00002	Větev pro hodnotu proměnné=2
	. . .	
:00002	EndCase	
:00010	EndSwitch	Konec přepínače

<b>EndCond</b>	Ukončení podmíněného příkazu <b>Cond</b>
----------------	--

**Popis**

Modul ukončí podmíněný příkaz **Cond-EndCond** nebo **Cond-ElseCond-EndCond**.  
Podrobnější popis viz příkaz **ElseCond** a příkaz **Cond**.

**Parametry**

žádné

<b>EndFor</b>	Ukončení iteračního cyklu <b>For</b>
---------------	--------------------------------------

**Popis**

Příkaz **EndFor** ukončuje iterační příkaz **For**. Podrobnější popis je uveden u příkazu **For**.

**Parametry**

žádné

**Příklad**

```
For    I, 0, 100, 1, :00000  
      . . .  
:00000 EndFor
```

Čítačová proměnná cyklu typu **F** bude postupně nabývat hodnot 0, 1, 2 ... 99, 100. Pro tyto hodnoty se vykonají příkazy/moduly v těle iteračního cyklu.

<b>EndIf</b>	Ukončení podmíněného příkazu <b>If</b>
--------------	--

**Popis**

Modul ukončí podmíněný příkaz **If-EndIf** nebo **If-Else-EndIf**. Podrobnější popis viz příkaz **Else** a příkaz **If**.

**Parametry**

žádné

**Příklad**

If	Test.0, :00000	Je-li digitální signál
		(bit) nastaven,
	. . .	proved' tuto sekvenci
		příkazů/modulů,
:00000	Else 00001	a není-li nastaven,
	. . .	proved' tuto sekvenci
:00001	EndIf	Následující
		příkazy/moduly prováděj
		vždy

<b>EndSwitch</b>	Ukončení přepínače <b>Switch</b>
------------------	----------------------------------

**Popis**

Příkaz **EndSwitch** ukončuje přepínač - příkaz **Switch**. Podrobnější popis je uveden v popisu příkazu **Switch**.

**Parametry**

žádné

**Příklad**

Switch Test, :00010	Přepínač podle hodnoty proměnné Test
Case 0, :00000	Větev pro hodnotu proměnné=0
. . .	Příkazy/moduly větve
:00000 EndCase	Konec větve
Case 1, :00001	Větev pro hodnotu proměnné=1
. . .	
:00001 EndCase	
Case 2, :00002	Větev pro hodnotu proměnné=2
. . .	
:00002 EndCase	
:00010 EndSwitch	Konec přepínače

<b>EndWhile</b>	Ukončení cyklu <b>While</b>
-----------------	-----------------------------

**Popis**

Příkaz **EndWhile** ukončuje cyklus s podmínkou na začátku **While**. Podrobnější popis je uveden v popisu příkazu **While**.

**Parametry**

žádné

**Příklad**

```
While      Test.1,:00000  
    . . .  
:00000EndWhile
```

Pokud je bit 1 proměnné `Test` nastaven, provádí příkaz procesní stanice moduly/příkazy uzavřené mezi příkazy **While** a **EndWhile**.

<b>EqLine</b>	Výpočet doporučené teploty topné vody
---------------	---------------------------------------

**Popis**

Modul počítá doporučenou teplotu topné vody na základě požadované teploty v místnosti, měřené venkovní teploty a ekvitemní konstanty.

**Parametry**

<b>ŽádanáT</b>	IN	F	Požadovaná teplota v místnosti.
		MF	
<b>VenkovníT</b>	IN	F	Měřená venkovní teplota.
		MF	
<b>OTV</b>	OUT	F	Doporučená teplota vody.
		MF	
<b>K</b>	IN	Konst	Ekvitemní konstanta v rozsahu <2.0 - 4.0>.
		F	
		MF	

**Příklad**

AnIn #0.9, Tvenk, 20.0, 4.0, 20.0, 0.0, 100.0

EqLine Tmístnost, Tvenk, Totv, K

Modul počítá doporučenou teplotu topné vody  $T_{otv}$ , na základě měření venkovní teploty  $T_{venk}$  (měřeno např. teploměrem PT100 s proudovým výstupem 4-20mA) a zadané teploty v místnosti  $T_{místnost}$ . Ekvitemní konstanta je většinou empiricky stanovená a je to inicializovaná proměnná K.



**ErrSig**

Modul pro obsluhu a detekci chyb a jejich signalizaci klaksonem

**Popis**

Ve vstupní databázové proměnné *Chyba* je maskou *Chyba\_OR* určen bit (příp. skupina bitů), dle kterého modul detekuje chybu. Pokud došlo k nastavení vybraného bitu na "1":

$$(Chyba \text{ and } Chyba\_OR) \neq 0$$

je detekována chyba, což se projeví nastavením bitu *Klakson* na "1".

Pokud došlo k chybě (*Chyba* v "1") a je nastaveno zpoždění *ČasChyby*, není chyba detekována okamžitě, ale detekce chyby je zpožděna o čas zadaný násobkem *ČasChyby* period modulu. Pokud je do této doby chyba odstraněna (*Chyba* v "0"), není chyba detekována (*Klakson* zůstává v "0").

Chyby lze kvitovat. Ke kvitaci dochází, nastaví-li se bit (příp. skupina bitů) na určený maskou *Kvit\_OR* na "1":

$$(Kvitace \text{ and } Kvit\_OR) \neq 0$$

Kvitování se projeví nastavením bitu *Klakson* zpět na "0".

Je-li parametr *ČasOK* nenulový, tak je po kvitaci potlačena detekce chyby na dobu *ČasOK* period procesu. Zůstává-li tedy po kvitaci chybový stav, ale podaří se jej v době potlačení detekce odstranit, není chyba znovu detekována. Pokud se nepodaří chybu odstranit ani do této doby, je znovu detekována.

Je-li parametr *ČasOK* nulový, je detekce chyby potlačena až do doby, dokud se chyba neodstraní. Takže k opětné detekci chyby může dojít až po té, co se chyba odstranila a znovu nastala.

Obsluhu chyby lze zablokovat signálem *Ignorovat*. V takovém případě modul neprovádí nic.

Při detekci chyby (*Klakson* jde do "1"), je do provozního deníku vloženo hlášení s kódem *Kód* a s parametry *WID/data1*, *Data2*, *Zdroj*. Při ukončení chyby (chybová podmínka jde do "0"), je do provozního deníku vloženo hlášení s kódem *Kód+1* a parametry *WID/data1*, *Data2*, *Zdroj*. Při kvitaci je do provozního deníku vloženo hlášení *Kód+2* s parametry *WID/data1*, *Data2*, *Zdroj*. Pokud se po kvitaci nepodařilo včas odstranit chybu a chyba byla znovu detekována, je do provozního deníku vloženo hlášení *Kód+3* s parametry *WID/data1*, *Data2*, *Zdroj*.

**Parametry**

<b>Chyba</b>	IN	I	Databázová proměnná pro detekci chyby.
<b>Chyba_OR</b>	PAR	Výběr	Maska na detekci chyby. Je-li alespoň jeden bit z masky v proměnné <i>Chyba</i> nastaven, je detekována chyba.
<b>Kvitace</b>	IN	I	Databázová proměnná pro detekci kvitace.
<b>Kvit_OR</b>	PAR	Výběr	Maska na detekci kvitace. Je-li alespoň jeden bit z masky v proměnné <i>Kvitace</i> nastaven je chyba kvitována.
<b>Klakson</b>	OUT	Bit	Výstup - bit databázové proměnné pro signalizaci chyby. Bit je nastaven na "1", pokud je signalizována chyba.
<b>Ignorovat</b>	IN	Bit	Vstup - ignorování chyby. Je-li bit nastaven, je obsluha chyby zablokována a negeneruje se žádné hlášení.
<b>ČasChyby</b>	PAR	Konst	Počet period zpoždění detekce chyby. Pokud chyba nastane na dobu kratší, není signalizována ani hlášena do provozního deníku.

<b>ČasOK</b>	PAR	Konst	Počet period, za které se musí chyba odstranit po kvitaci.
--------------	-----	-------	--

Pokud obsluha nestihne do této doby chybu odstranit, je chyba znovu detekována. Zvláštní chování je pro hodnotu parametru 0. V tomto případě je detekce chyby zablokována do doby, dokud se chyba neodstraní (přejde do stavu OK). Takže k opětné detekci chyby může dojít až po té, co se chyba odstranila a znovu nastala.

<b>Kód</b>	PAR	Konst	Zakladní kód pro provozní deník.
------------	-----	-------	----------------------------------

Aplikátor může volit kódy v rozsahu 100..30000, ostatní kódy jsou rezervovány pro systémové účely. Modul generuje až čtyři hlášení s následujícím významem:

kód	význam
Kód	chyba nastala
Kód + 1	chyba skončila
Kód + 2	chyba kvitována
Kód + 3	kvitace skončila - chybu se nepodařilo odstranit

Další parametry slouží k identifikaci chyby a jsou plně v moci programátora.

<b>WID/ data1</b>	PAR	Konst	Parametr do provozního deníku.
-----------------------	-----	-------	--------------------------------

<b>Data2</b>	PAR	Konst	Parametr do provozního deníku.
--------------	-----	-------	--------------------------------

<b>Zdroj</b>	PAR	Konst	Parametr do provozního deníku.
--------------	-----	-------	--------------------------------

#### Příklad

```
ErrSig          Chyba, 0x0001, Kvitace, 0x0001, Err.0, Err.1, 5,
                600, 1000, 1, 0, 0
```

Předpokládáme, že perioda modulu je 1s. Pokud přejde 0. bit proměnné Chyba do "1" a nevrátí se do 5 vteřin zpět na "0", je detekována chyba: 0. bit proměnné Err se nastaví do "1" a do provozního denníku se generuje hlášení s kódem 1000 a daty (1,0,0). Chyba se dá odkvitovat 0. bitem proměnné Kvitace, což způsobí, že se 0. bit proměnné Err vrátí do "0" a generuje se hlášení (1002,1,0,0). Pokud se do doby 600 vteřin nepodaří chybu odstranit, je znovu detekována chyba - Err.0 do "1" a hlášení (1003,1,0,0). Pokud chyba skončí, je zhozen příznak chyby Err.0 do "0" a generováno hlášení (1001,1,0,0).

<b>EscGLCD</b>	Vyslání Escape-sekvence na grafický terminál
----------------	--

**Popis**

Modul slouží pro vyslání libovolné řídicí sekvence na grafický terminál (např. APT2000). Lze jej použít rovněž k "vyslání" sekvence do části operačního systému, obsluhující grafické rozhraní v rámci řídicího terminálu APT2100.

Sekvence může mít až pět číselných parametrů, které lze do ní dosadit na správná místa v kódování příslušného typu, který vyžaduje terminál.

Dosazování číselných parametrů se řídí formátovacím řetězcem, který se předává jako parametr `ESC` modulu **EscGLCD**.

Požadovaný formát sekvence naleznete v uživatelské příručce použitého terminálu.

Formátovací řetězec se používá následovně. Všechny "obyčejné" znaky formátovacího řetězce se do sekvence ukládají v tom tvaru, v jakém jsou uvedeny ve formátovacím řetězci. Dosazení parametru se dosáhne pomocí znaku "%", čímž se vytvoří formátovací předpis.

Znak, který následuje za znakem "%", určuje datový formát, ve kterém se parametr vloží podle následující tabulky. Další formátovací předpis použije následující parametr ze seznamu `Prm1` až `Prm5`.

Speciálním případem je kombinace "%e", která vyvolá vložení znaku "Esc" (27d, 1Bh) do sekvence. Přitom nepoužívá žádný parametr ani nevyvolává posunutí v seznamu parametrů.

Vyskytne-li se za znakem "%" znak neuvedený v následující tabulce, vloží se do sekvence tento znak, jako by před ním nebyl uveden znak "%". Přitom se nepoužívá žádný parametr ani se nevyvolává posunutí v seznamu parametrů. Chceme-li tedy opravdu vyslat znak "%" vložíme do formátovacího řetězce "%%".

Velikost písmene ve formátovacím předpisu nerozhoduje.

Formátovací předpis	Význam
%e	Vložení znaku "Esc" (bez použití parametru)
%c	Vložení <code>Prm<sub>x</sub></code> jako hodnoty typu char (1 byte se znaménkem, -128 ÷ 127)
%b	Vložení <code>Prm<sub>x</sub></code> jako hodnoty typu byte (1 byte bez znaménka, 0 ÷ 255)
%i	Vložení <code>Prm<sub>x</sub></code> jako hodnoty typu int (2 byte se znaménkem, -32768 ÷ 32767)
%w	Vložení <code>Prm<sub>x</sub></code> jako hodnoty typu word (2 byte bez znaménka, 0 ÷ 65535)
%l	Vložení <code>Prm<sub>x</sub></code> jako hodnoty typu long (4 byte se znaménkem, -2147483648 ÷ 2147483647)
%d	Vložení <code>Prm<sub>x</sub></code> jako hodnoty typu dword (4 byte bez znaménka, 0 ÷ 4294967295)
%f	Vložení <code>Prm<sub>x</sub></code> jako hodnoty typu float (4 byte IEEE32:24 - pohyblivá řádová čárka)

V aplikaci musí být použit modul pro obsluhu grafického terminálu, např. LCDGTTY, LCD2100.

**Parametry**

<b>ESC</b>	IN	Řetězec	Formátovací řetězec - viz výše.
------------	----	---------	---------------------------------

<b>Prm1</b>	IN	Konst	První číselný parametr Escape-sekvence.
		I	
		L	
		F	
		MI	
		ML	
		MF	

Prm2	IN	Konst	Druhý číselný parametr Escape-sekvence.
		I	
		L	
		F	
		MI	
		ML	
		MF	

Prm3	IN	Konst	Třetí číselný parametr Escape-sekvence.
		I	
		L	
		F	
		MI	
		ML	
		MF	

Prm4	IN	Konst	Čtvrtý číselný parametr Escape-sekvence.
		I	
		L	
		F	
		MI	
		ML	
		MF	

Prm5	IN	Konst	Pátý číselný parametr Escape-sekvence.
		I	
		L	
		F	
		MI	
		ML	
		MF	

### Příklad

EscGLCD            "%e9l%w%w%w%w", Bod1X, Bod1Y, Bod2X, Bod2Y, 0.0

Modul vyšle do terminálu sekvenci <Esc>9l<x1><y1><x2><y2>. Tím dosáhne vykreslení přímky určené dvěma krajními body, jejichž souřadnice jsou v příslušných databázových proměnných.

*Pozn.: Zpravidla dosáhneme stejného výsledku jednodušeji použitím vhodného prvku LCDShellu (v tomto případě **GEntity**).*

<b>EthNetSeg</b>	Definice vzdálené stanice na Ethernetu
------------------	--

**Popis**

Modul definuje jinou vzdálenou stanici (resp. skupinu stanic) na Ethernetu se kterou je možno komunikovat pomocí modulů **EthReqDb** nebo **EthRqDbDr**. Stanice připojená na Ethernet se chová jako brána (gateway) a přeposílá (routuje) potřebnou komunikaci z Ethernetu na RS232/485 DB-Net - zpřístupňuje vlastně celý "DB-Net segment". Všechny vzdálené stanice mají shodnou IP adresu (definovanou tímto modulem) a rozlišují se podle čísla stanice na DIP přepínači.

Modul se typicky umísťuje do procesu `Init`.

Vyhodnocení parametrů probíhá pouze jedenkrát při startu, a to při prvním spuštění modulu. Změna hodnot databázových proměnných za dalšího běhu stanice se již nijak neprojeví (aby se změna projevila je nutno provést restart (vypnutí, zapnutí) stanice).

**IP konfigurace**

Aby byla možná komunikace pomocí Ethernetu je nutné, aby příslušné komunikační rozhraní (Ethernetová karta) měla správně nastavenou IP konfiguraci. Ta se ukládá do paměti EEPROM a zůstává zachována jak přes přehrání aplikace tak i přes přehrání NOSu.

IP konfigurace se nastavuje z prostředí PSP3 pomocí příkazu menu `Přenos|Konfigurace IP` (viz blíže manuál PSP3) nebo z některých dalších nástrojů (např. `PseView`).

**Parametry**

<b>IPadresa</b>	IN	Konst	IP adresa vzdálené stanice - brány. Konstantní se zadává jako čtveřice čísel v rozsahu 0÷255. V proměnné se zadává jako 32 bitové číslo, přičemž části IP adresy 1÷4 se umísťují od nejvyššího po nejnižší bajt hodnoty. Například adresa 192.168.0.1 se запиše jako 0xC0A80001.
		L	
		ML	
<b>Port</b>	IN	Konst	Číslo UDP portu vzdálené stanice - brány na kterém probíhá komunikace.
		I	
		MI	
<b>Heslo</b>	IN	Konst	Přihlašovací heslo vzdálené stanice - brány.
		L	
		ML	
<b>Timeout</b>	IN	Konst	Maximální doba čekání (v ms) na odpověď od vzdálené stanice.
		L	
		ML	
<b>Stanice</b>	IN	Konst	Příznaky, zda se má pro vzdálené stanice aktivně udržovat jejich stav. V hodnotě z proměnné určuje bit D0 chování pro stanici číslo 0, bit D1 pro stanici číslo 1 atd. Při volbě <code>ANO</code> nebo je-li odpovídající bit proměnné roven 1, je jednou za 30s vzdálená stanice dotázána na svůj stav. Vlastní funkci zjištění stavu vzdálené proměnné zajišťuje modul <b>EthSState</b> . Nastavení tohoto parametru nemá vliv na přenos dat pomocí modulů <b>EthReqDb</b> či <b>EthRqDbDr</b> .
		L	
		ML	

Stav	OUT	I	Jméno proměnné do které se bude průběžně ukládat stav komunikace se vzdáleným "DB-Net segmentem". Proměnná neobsahuje stav vzdálených stanic (tuto funkci zajišťuje modul <b>EthSState</b> ), ale interní stav komunikačního automatu. Parametr je určen pro bližší diagnostiku komunikace například při ladění a zprovoznování komunikace. Nezajímá-li nás stav lze dosadit <b>NONE</b> .
		NONE	

**Stav** může nabývat hodnot:

0x0000	Vše je OK, neprovádí se žádná činnost.
0x0001	Čeká se na odpověď na dotaz na stav vzdálené stanice.
0x0002	Čeká se na odpověď na dotaz modulu <b>EthReqDb</b> nebo <b>EthRqDbDr</b> .
0x0003	Zahájení zpracování dotazu modulu <b>EthReqDb</b> nebo <b>EthRqDbDr</b> .
0x0104	CHYBA, nelze najít komunikační rozhraní (Ethernetovou kartu) přes kterou by bylo možno komunikovat se zadanou IP adresou.
0x0204	CHYBA, nenakonfigurované komunikační rozhraní (Ethernetová karta).
0x0304	CHYBA, duplicitní IP adresa.
0x0404	CHYBA, neexistuje žádné komunikační rozhraní na Ethernet.

### Příklad

```
EthNetSeg 0x584C6CC2, 59, 123456789, 3500, 0x00000030, STAV
```

Definujeme stanici s adresou 194.108.76.88, port 59, heslo 123456789. Timeout je zvolen na 3,5s a aktivně se udržuje stav stanic 4 a 5. Stav komunikace se vzdálenou stanicí se bude ukládat do proměnné **STAV**.

<b>EthReqDb</b>	Žádost o přenos databázové proměnné po síti Ethernet
-----------------	--

**Popis**

Při každém spuštění modulu se přenáší žádaná proměnná buď směrem do vzdálené stanice (zápis) nebo směrem ven ze vzdálené stanice (čtení). Pokud je typ proměnné matice, lze přenášet jak celou matici, tak i její výřez.

Pokud se přenáší matice, lze přenášet jen část matice. Maximální velikost výřezu matice, kterou je modul schopen v současné verzi přenést je 240B.

Tabulka udává maximální velikost výřezu (počet řádků x počet sloupců) pro jednotlivé databázové typy:

Typ	poč.řádků x poč. sloupců
MI	120
ML	60
MF	60

Zadá-li se výřez větší, modul jej interpretuje jako chybu parametrů. Chybový kód se objeví v proměnné **Vložení**.

Přenos větší matice se musí rozdělit na části a postupně použít více modulů **EthReqDb**.

**Parametry**

<b>EthNetSeg</b>	PAR	Návěští	Návěští modulu <b>EthNetSeg</b> , který definuje vzdálenou stanici, se kterou se má komunikovat.
<b>Zapisovat</b>	PAR	Výběr	Nastavení příznaku znamená zápis proměnné, v opačném případě se jedná o čtení.
<b>Stanice</b>	PAR	Konst	Číslo stanice, se kterou se má komunikovat. Podle tohoto čísla se rozpoznává, zda komunikace probíhá přímo se stanicí připojenou na Ethernetu nebo zda se komunikace přeposílá (routuje) dále na RS232/485 síť.
<b>Proměnná</b>	IN/OUT	I	Proměnná ze které se data přenáší/zapisují. Pokud se přenáší matice, lze přenášet jen část matice. Tento "výřez" pak začíná na pozici dané řádkem a sloupcem a má rozměry zadané parametry <b>Řádků</b> a <b>Sloupců</b> .
		L	
		F	
		MI	
		ML	
		MF	
<b>Řádků</b>	PAR	Konst	Počet řádků přenášené části matice.
<b>Sloupců</b>	PAR	Konst	Počet sloupců přenášené části matice.
<b>WID</b>	PAR	Konst	WID proměnné na vzdálené stanici, která se komunikuje. Pokud je zadáno -1, použije se WID proměnné dosazené za parametr <b>Proměnná</b> .
<b>Vložení</b>	OUT	I	Proměnná - chybový kód vložení požadavku. Lze zadat NONE. Popis kódů viz. dodatek "Stavy sériové komunikace".
		NONE	
<b>Stav</b>	IN/OUT	I	Proměnná - stav přenosového požadavku. Lze zadat NONE. Popis kódů viz. dodatek "Stavy sériové komunikace".
		NONE	

**Příklad**

```
EthReqDb :01111, 0x0001, 4, Stav[2, 3], 1, 2, -1, Succ, Res
```

Jde o zápis výřezu matice **Stav** na stanici číslo 4 na IP adresu definovanou modulem **EthNetSeg** s návěštím 01111. Na vzdálené stanici se hodnoty uloží do proměnné se stejným WIDem jako má proměnná **Stav**.

Výsledek vložení požadavku je v proměnné `Succ`, stav požadavku je v proměnné `Res`.  
V tabulce je znázorněn výřez matice.

řádek\sloupec	0	1	2	3	4	5
0						
1						
2						
3						
4						



<b>EthRqDbDr</b>	Žádost o přímý přenos databázové proměnné po síti Ethernet
------------------	--

**Popis**

Přímý přenos znamená, že v databázi vlastní stanice nemusí být kopie vzdálené proměnné, jak je tomu u modulu **EthReqDb**. Číslo vzdálené stanice se nebere z definice zdroje proměnné v databázi, ale určuje se explicitně parametrem modulu. Lze tedy přenášet vlastní lokální proměnnou do libovolné jiné vzdálené proměnné nebo opačně vzdálenou libovolnou proměnnou do lokální. Navíc lze parametry přenosu zadávat v proměnných a měnit je tak za běhu.

Při každém spuštění modulu se přenáší žádaná proměnná buď směrem do vzdálené stanice (zápis) nebo směrem ze vzdálené stanice (čtení). Pokud je typ proměnné matice, lze přenášet jak celou matici, tak i její výřez.

Typ lokální a vzdálené proměnné musí být buď shodný, nebo jedna proměnná může být matice a druhá jednoduchá přičemž musí být shodný jejich bazový typ. Povolené kombinace jsou tedy např. I-I, I-MI, MI-I apod. Příklad zakázaných kombinací je I-L, MI-F apod.

Pokud se přenáší matice, lze přenášet také jen část matice. Zapisovaný výřez matice se musí celý "vejít" do vzdálené matice, stejně tak čtený výřez matice se musí celý "vejít" do lokální matice. Maximální velikost výřezu matice, kterou je modul schopen v současné verzi přenést je 240B.

Tabulka udává maximální velikost výřezu (počet řádků x počet sloupců) pro jednotlivé databázové typy:

Typ	poč.řádků x poč. sloupců
MI	120
ML	60
MF	60

Zadá-li se výřez větší, modul jej interpreтуje jako chybu parametrů. Chybový kód se objeví v proměnné `vložení`.

Přenos větší matice se musí rozdělit na části a postupně použít více modulů **EthReqDb**.

**Parametry**

Většinu parametrů lze zadat dvojím způsobem: buď jako číslo - odpovídající proměnná pak musí být zadána jako `NONE`, nebo jako proměnnou - v tom případě číselný parametr nemá význam.

<b>EthNetSeg</b>	PAR	Návěští	Návěští modulu <b>EthNetSeg</b> , který definuje vzdálenou stanici, se kterou se má komunikovat.
<b>Zapisovat</b>	PAR	Výběr	Nastavení příznaku znamená zápis proměnné, v opačném případě se jedná o čtení.
<b>Stanice</b>	PAR	Konst	Číslo vzdálené stanice.
<b>hStanice</b>	IN	I NONE	Proměnná - číslo vzdálené stanice. Je-li <code>NONE</code> , bere se hodnota z <code>Stanice</code> .
<b>WID</b>	PAR	Konst	WID vzdálené proměnné. Typ vzdálené proměnné musí být shodný s typem lokální proměnné.
<b>hWID</b>	IN	I NONE	Proměnná - WID vzdálené proměnné. Je-li <code>NONE</code> , bere se hodnota z <code>WID</code> .
<b>Typ</b>	PAR	Konst	Databázový typ vzdálené proměnné.

Typ	Číslo
I	0
L	1

F	2
MI	3
ML	4
MF	5

<b>hTyp</b>	IN	I	Proměnná - databázový typ vzdálené proměnné. Je-li NONE, bere se hodnota z Typ.
		NONE	
<b>Řádek</b>	PAR	Konst	Počáteční řádek výřezu vzdálené matice.
<b>hŘádek</b>	IN	IN	Proměnná - počáteční řádek výřezu vzdálené matice. Je-li NONE, bere se hodnota z Řádek.
		NONE	
<b>Sloupec</b>	PAR	Konst	Počáteční sloupec výřezu vzdálené proměnné.
<b>hSloupec</b>	IN	I	Proměnná - počáteční sloupec výřezu vzdálené matice. Je-li NONE, bere se hodnota z Sloupec.
		NONE	
<b>Řádků</b>	PAR	Konst	Počet řádků výřezu vzdálené matice.
<b>hŘádků</b>	IN	I	Proměnná - počet řádků výřezu vzdálené matice. Je-li NONE, bere se hodnota z Řádků.
		NONE	
<b>Sloupců</b>	PAR	Konst	Počet sloupců výřezu vzdálené matice.
<b>hSloupců</b>	IN	I	Proměnná - počet sloupců výřezu vzdálené matice. Je-li NONE, bere se hodnota z Sloupců.
		NONE	
<b>LokProm</b>	IN/OUT	I	Lokální databázová proměnná libovolného typu shodného s typem vzdálené proměnné.
		L	
		F	
		MI	
		ML	
		MF	
<b>LokŘ</b>	PAR	Konst	Počáteční řádek výřezu lokální matice.
<b>hLokŘ</b>	IN	I	Proměnná - počáteční řádek výřezu lokální matice. Je-li NONE, bere se hodnota z LokŘ.
		NONE	
<b>LokSI</b>	PAR	Konst	Počáteční sloupec výřezu lokální matice.
<b>hLokSI</b>	IN	I	Proměnná - počáteční sloupec výřezu lokální matice. Je-li NONE, bere se hodnota z LokSI.
		NONE	
<b>Vložení</b>	OUT	I	Proměnná - chybový kód vložení požadavku. Lze zadat NONE. Popis kódů viz. dodatek "Stavy sériové komunikace".
		NONE	
<b>Stav</b>	IN/OUT	I	Proměnná - stav přenosového požadavku. Popis kódů viz. dodatek "Stavy sériové komunikace".
		NONE	

**Příklad**

```
EthReqDbDr :01111, 0x0000, 7, NONE, 7002, NONE, 2, NONE, 0, NONE, 0, NONE, 1,  
            NONE, 1, NONE, S7_Tlak, 0, NONE, 0, NONE, NONE, NONE
```

Jde o čtení jednoduché proměnné typu F s WIDem 7002 ze stanice číslo 7 na IP adrese definované modulem **EthNetSeg** s návěštím 01111 do proměnné S7\_Tlak. Výsledek vložení požadavku ani stav požadavku se v aplikaci nekontrolují.

**EthRoute**

## Definice statického směrování

**Popis**

Stanice normálně směruje veškeré nelokální IP pakety (pakety s adresátem mimo síť než ke které je stanice připojena) na výchozí bránu (default gateway) určenou IP konfigurací v EEPROM. Modulem **EthRoute** je možno vytvořit seznam výjimek, takže pro určité IP adresy se místo výchozí brány použije nějaká brána jiná.

Pokud je nalezeno více pravidel, která by vyhovovala, použije se co nejkonkrétnější pravidlo. Například pravidlo pro směrování konkrétní IP adresy má přednost před pravidlem pro směrování podsítě typu C, obě mají přednost před pravidlem pro směrování podsítě typu B atd.

Modul se typicky umísťuje do procesu `Init`.

**IP konfigurace**

Stanice jako celek má svoji IP konfiguraci, která se ukládá do paměti EEPROM. Zůstává tak zachována jak přes přehrání aplikace tak i přes přehrání NOSu.

IP konfigurace se nastavuje z prostředí PSP3 pomocí příkazu menu `Přenos|Konfigurace IP` (viz blíže manuál PSP3) nebo z některých dalších nástrojů (např. `PseView`).

**Parametry**

IPadresa	IN	Konst	IP adresa sítě. Společně s hodnotou <b>IPmaska</b> určují rozsah IP adres, pro které bude platit toto pravidlo. Konstantní se zadává jako čtveřice čísel v rozsahu 0÷255. V proměnné se zadává jako 32 bitové číslo, přičemž části IP adresy 1÷4 se umísťují od nejvyššího po nejnižší bajt. Například adresa 10.0.3.128 se zapíše jako 0x0A000380.
		L	
		ML	

IPmaska	IN	Konst	IP maska sítě. Společně s hodnotou <b>IPadresa</b> určují rozsah IP adres, pro které bude platit toto pravidlo. Konstantní se zadává jako čtveřice čísel v rozsahu 0÷255. V proměnné se zadává jako 32 bitové číslo, přičemž části masky 1÷4 se umísťují od nejvyššího po nejnižší bajt. Například maska 255.255.255.0 se zapíše jako 0xFFFFFFFF00.
		L	
		ML	

IPbrana	IN	Konst	IP adresa brány (gateway), na kterou se směrují pakety určené pro adresáty v síti určené parametry <b>IPadresa</b> a <b>IPmaska</b> . Konstantní se zadává jako čtveřice čísel v rozsahu 0÷255. V proměnné se zadává jako 32 bitové číslo, přičemž části IP adresy 1÷4 se umísťují od nejvyššího po nejnižší bajt. Například adresa 192.168.0.1 se zapíše jako 0xC0A80001.
		L	
		ML	

**Příklad**

```
EthRoute 0xC0A81300, 0xFFFFFFFF00, 0xC0A8A82A
```

Definujeme směrování pro síť 192.168.19.0 / 255.255.255.0 na bránu 192.168.168.42.

<b>EthSState</b>	Vrátí stav vzdálené stanice
------------------	-----------------------------

**Popis**

Modul vrací stav vzdálené stanice na DB-Net segmentu definovaném modulem **EthNetSeg**.

Stav stanic se aktualizuje při zpracování každé žádosti modulem **EthReqDb/EthRqDbDR** a taktéž jedenkrát za 30s pro stanice uvedené v parametru **Stanice** modulu **EthNetSeg**. Modul vrací pouze poslední známý stav stanice, neprovádí žádnou komunikaci.

**Parametry**

<b>EthNetSeg</b>	PAR	Návěští	Návěští modulu <b>EthNetSeg</b> , který definuje požadovaný DB-Net segment.
------------------	-----	---------	---

<b>Stanice</b>	IN	Konst	Číslo stanice jejíž stav se zjišťuje.
		I	
		MI	

<b>Stav</b>	OUT	I	Proměnná do které se uloží zjištěný stav.
-------------	-----	---	---

Možné stavy jsou:

- 1 Stanice není připojena (neodpovídá).
- 0 Stanice je pasivní.
- 1 Stanice je aktivní a je ve stavu prvotního poslechu sítě před připojením se na síť (neřídí provoz sítě)
- 2 Stanice je aktivní a je ve stavu čekání na připojení se na síť (neřídí provoz sítě)
- 3 Stanice je aktivní a je připojena (řídí provoz sítě)

**Příklad**

```
EthSStav :01111, 4, Stav
```

Uloží stav stanice číslo 4 na DB-Net segmentu definovaný modulem **EthNetSeg** do proměnné **Stav**.

<b>Exit</b>	Ukončení běhu podprogramu
-------------	---------------------------

**Popis**

Modul **Exit** ukončuje zpracování programu (viz příkaz **Call**) nebo, je-li to žádoucí, řádného procesu.

**Parametry**

žádné

**Příklad**

```
Call      100
Lib100:
    . . .
    If      Test.0, :00000
Exit
:00000 EndIf
    . . .
```

Příkazem **Call** je volán knihovní podprogram `Lib100`. Ten zpracuje část své činnosti. Je-li však nastaven bit č. 0 proměnné `Test`, dojde k předčasnému ukončení výkonu podprogramu příkazem **Exit**. V opačném případě se pokračuje s výkonem podprogramu až do dokončení posledního příkazu/modulu.

**Popis**

Modul byl ve verzi 3.35 distribuční sady PSP3 zrušen.

Modul sloužil ve starších verzích PSP3 ke zrychlení běhu programu univerzálního (pro systémy v provedení pro "industry" i "standard" rozsah provozních teplot) systému NOS na rychlejší úroveň, odpovídající časování sběrnic pro provedení "standard" (implicitně používal univerzální NOS pomalejší časování pro provedení "industry").

Ve V3.35 byla provedena hloubkové revize časování sběrnic, z níž vyšla možnost použít jednotné časování pro obě provedení řídicích systémů (oba rozsahy provozních teplot). Toto jednotné časování je přitom stejně rychlé (na některých systémech dokonce o několik procent rychlejší) jako dříve používané (rychlejší) časování pro systémy v provedení "standard".

Modul FastRun proto od V3.35 není potřebný, a ani by s novou verzí NOSu nefungoval.

Nové univerzální časování je použitelné pouze na systémech vybavených procesory C167 V3.x a vyšší, popř. procesry ST80C269. Při pokusu spustit NOS V3.35 na systému vybaveném starší verzí procesoru C167 (tj. vyrobeném cca před rokem 2000) bude nahlášena chyba, viz kapitolu "Stavové a chybové informace" první části uživatelské příručky (man\_i.pdf).

<b>FIFO</b>	Fronta typu "první dovnitř - první ven" (roura)
-------------	---

**Popis**

Modul **FIFO** slouží jako buffer pro předávání bloků dat mezi dvěma moduly. Tyto moduly se na něj odkazují přes návěští. Jako vlastní buffer slouží databázová matice DbBuf typu MI. Jednotlivé bloky dat jsou tzv. "položky fronty". Délka fronty (maximální počet vložených položek) je určena velikostí matice a velikostí položky:

$$n = \frac{\text{Radku} * \text{Sloupcu} * 2}{\text{Položka}}, \text{ kde}$$

$n$  ..... délka fronty,

Radku ..... počet řádků matice DbBuf,

Sloupcu .. počet sloupců matice DbBuf,

Položka .. velikost položky v bytech.

**Parametry**

<b>DbBuf</b>	INOUT	MI	Databázová matice tvořící vlastní buffer. Rozměry lze volit libovolně, typicky se však volí řádková matice. Celá matice představuje v paměti souvislý blok, buňky jsou uloženy v pořadí 1. řádek, 2. řádek, atd. Délka fronty je určena vzorcem uvedeným výše. Na každou buňku matice připadají 2 B dat. Je-li velikost položky zvolena např. 8, ukládá se do 4 buňek matice.
--------------	-------	----	---

<b>Ztrácet</b>	PAR	Výběr	Volba, zda se má při přeplnění fronty ztrácet nejstarší položka.
----------------	-----	-------	--

Ztrácet	ANO	Při přeplnění fronty ztrácet nejstarší položku
	NE	Při přeplnění fronty zakázat vložení nové položky

<b>Položka</b>	PAR	Konst	Velikost položky [B].
----------------	-----	-------	-----------------------

<b>Vloženo</b>	OUT	I	Počet vložených položek ve frontě.
		MI	
		NONE	

<b>První</b>	OUT	I	Index matice, kde začíná nejstarší položka. Je-li zvolena řádková matice, tak parametr udává přímo číslo sloupce matice. U víceřádkové matice je řádek určen: řádek = První div poč.sloupců a sloupec je určen: sloupec = První mod poč.sloupců
		MI	
		NONE	

<b>Poslední</b>	OUT	I	Index matice, kde začíná nejnovější položka. Je-li zvolena řádková matice, tak parametr udává přímo číslo sloupce matice. U víceřádkové matice je řádek určen: řádek = První div poč.sloupců a sloupec je určen: sloupec = První mod poč.sloupců
		MI	
		NONE	

<b>Přeplnění</b>	OUT	Bit	Modul bit nastaví do "1", pokud došlo k přeplnění fronty. Bit zůstane nastaven. Musí se vynulovat na jiném místě aplikace.
		NONE	

Parametry Vloženo, První, Poslední a Přeplnění jsou pouze informativní. Není-li potřeba mít tyto údaje k dispozici, lze za parametry zadat NONE a modul umístit do procesu INIT. Pokud však chceme tyto údaje využívat, je třeba modul vložit do některého periodického procesu. Perioda procesu potom určuje periodu oživování těchto údajů.



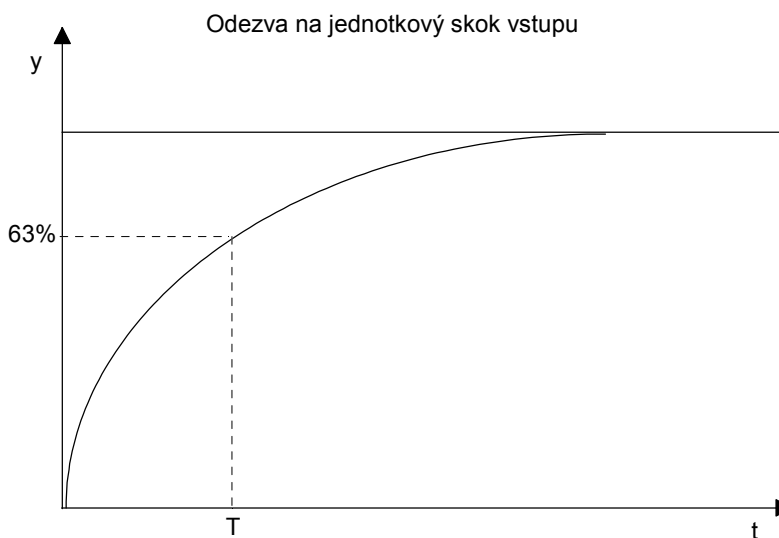
**Příklad**

```
:01000  FIFO  Fifo, 0x0001, 8, Vlozeno, Prvni, Posledni, @Preplnen
```

Modul definuje FIFO frontu. Buffer je v matici `Fifo`, velikost položky je 8 bytů. Modul při přeplnění ztrácí nejstarší položku. Modul je umístěn v periodickém procesu s periodou 1s, aby oživoval informační údaje v proměnných `Vlozeno`, `Prvni`, `Posledni` a `@Preplnen`. V proměnné `Vlozeno` je počet vložených položek, v proměnných `Prvni` a `Posledni` je index nejstarší a nejnovější položky a bit `@Preplnen` signalizuje přeplnění fronty.

**Filtr1R**

Modul realizuje filtr 1. řádu (setrvačný článek)

**Popis****Parametry**

<b>Vstup</b>	IN	F	Vstupní proměnná.
		MF	
<b>Výstup</b>	OUT	F	Výstupní proměnná.
		MF	
<b>T</b>	IN	Konst	Časová konstanta filtru [s]. Je to doba, za kterou výstup dosáhne 63% hodnoty vstupu při odezvě na skok vstupu.
		F	
		MF	

**Příklad**

```
AnaIn          #0.0, AI1, 10.0, 0.0, 10.0, 0.0, 100.0
```

```
Filtr1R        AI1, AI1filtr, 30
```

Filtrace analogového vstupu. Za 30 s od skokové změny vstupu AI1 filtrovaná hodnota dosáhne 63% změněné hodnoty vstupu.

<b>FindDay</b>	Nalezení rozsahu indexů v časové matici
----------------	---

Nalezení rozsahu indexů v časové matici, vytvořené modulem **Archive** nebo **SyncArch** (dále jen archivační modul), ve kterém jsou uložena data ze zadaného dne nebo rozmezí dnů..

**Popis**

Modul nalezne v časové matici data odpovídající požadovanému dni nebo rozsahu dnů. Vráť sloupcový index do matice, na kterém začínají časové údaje příslušného rozmezí a index, na kterém tato data končí. Vzhledem k tomu, že archivované hodnoty jsou v archivní matici uloženy na stejných sloupcích jako příslušné časy vzorků v matici časů, lze získané indexy použít i jako indexy do vlastní archivní matice.

**Určení data**

Datum je určeno vždy čtveřicí parametrů - Den, Měsíc, Rok a DenTýdne. Může být určeno jako absolutní (kalendářní), relativní (vůči běžnému dni - dnešku) nebo jako den v týdnu.

**Absolutní datum**

Datum se považuje za absolutní, pokud den, měsíc i rok jsou větší než nula a den v týdnu je NONE nebo je hodnota příslušné proměnné větší než 6. Za kteroukoliv složku data lze dosadit NONE, v tom případě se za ni dosadí příslušná část aktuálního (dnešního) data.

Je-li datum zadáno neúplně (rok nebo měsíc i rok je NONE), provádí se automatická úprava data tak, aby 1) vyhledávaný interval dní ležel v minulosti (hledat v archivu budoucnost nemá smysl) a 2) počátek intervalu ležel před koncem intervalu.

Např.:

DEN(=1), NONE, NONE, NONE

znamená: prvního tohoto měsíce. Je-li však dnes prvního, znamená to prvního minulého měsíce

DEN(=25), MESIC(=2), NONE, NONE

znamená: 25. února tohoto roku, pokud je již alespoň 26. února, jinak to znamená 25. února loňského roku

DEN(=25), MESIC(=2), ROK(=1996), DVT(=7)

znamená: 25. února 1996

**Relativní datum**

Datum se považuje za relativní, pokud některá jeho složka je menší nebo rovna nule. Přitom den v týdnu musí být NONE nebo hodnota příslušné proměnné musí být větší než 6. Za kteroukoliv složku data lze dosadit NONE, v tom případě se za ni dosadí nula.

Např.:

DEN(=0), NONE, NONE, NONE

znamená: dnes

DEN(=-1), NONE, NONE, NONE

znamená: včera

NONE, MESIC(=-1), NONE, NONE

znamená: před měsícem

Je-li některá složka data větší než nula, chápe se (tato složka) jako absolutní, např.:

DEN(=1), MESIC(=-1), NONE, NONE

znamená: prvního minulého měsíce.

**Den v týdnu**

Datum je určeno dnem v týdnu, pokud je za příslušný parametr dosazena databázová proměnná s hodnotou 0 až 6.

Dny v týdnu jsou číselně kódovány podle následující tabulky:

Číslo	Den
0	neděle
1	pondělí
2	úterý
3	středa
4	čtvrtek
5	pátek
6	sobota

Hledá se vždy poslední den daného typu, který celý leží v minulosti, to jest, je-li dnes středa, pak zadání:

NONE, NONE, NONE, DVT(=4)

znamená: minulý čtvrtek, zatímco:

NONE, NONE, NONE, DVT(=1)

znamená: toto pondělí.

U počátku intervalu se navíc zajišťuje, aby počátek intervalu ležel před jeho koncem, takže při zadání "od úterý do pondělí" se hledá od minulého úterý do tohoto pondělí.

Při zadání dne v týdnu lze zároveň zadat parametr Den, v tom případě se jedná o kombinaci s relativním určením data, zadaná hodnota (kladná i záporná) se chápe jako relativní posun ve dnech. Např.:

DEN(=-7), NONE, NONE, DVT(=1)

znamená: týden před posledním pondělím.

Určení běžného dne  
(dneška)

Dnešek se zapíše:

NONE, NONE, NONE, NONE

případně:

DEN(=0), MESIC(=0), ROK(=0), DVT(=7)

### Parametry

<b>Časy</b>	IN	ML	Matice časů vytvořená archivačním modulem.
<b>Index</b>	IN	I	Archivní index - viz popis příslušného archivačního modulu.
<b>Den1</b>	IN	I	Jméno databázové proměnné, která obsahuje buďto přímo určení kalendářního dne pro začátek hledaného intervalu, nebo relativní posun vůči běžnému dni (dnešku) ve dnech. Viz oddíl "určení data" výše.
		NONE	
<b>Měsíc1</b>	IN	I	Jméno databázové proměnné, která obsahuje buďto přímo určení kalendářního měsíce pro začátek hledaného intervalu, nebo relativní posun vůči běžnému dni (dnešku) v měsících. Viz oddíl "určení data" výše.
		NONE	
<b>Rok1</b>	IN	I	Jméno databázové proměnné, která obsahuje buďto přímo určení kalendářního roku pro začátek hledaného intervalu, nebo relativní posun vůči běžnému dni (dnešku) v letech. Viz oddíl "určení data" výše.
		NONE	

<b>DenTýdne1</b>	IN	I	Jméno proměnné, ve které je číselné označení dne v týdnu pro začátek hledaného intervalu. Viz oddíl "určení data" výše.
		NONE	
<b>hPosun1</b>	IN	I	Jméno databázové proměnné, ve které je uloženo posunutí půlnoci v sekundách pro začátek intervalu. Používá se, pokud za přelom dne není považována půlnoc, ale například 22:00. Dá se také použít k tomu, chceme-li například aby se vzorky z poslední čtvrt hodiny předešlého dne objevily ve výpisu daného dne. Oba přístupy lze kombinovat.
		L	
		F	
		NONE	
<b>Posun1</b>	PAR	Konst	Hodnota posunutí půlnoci, která se použije, dosadí-li se za hPosun1 NONE.
<b>Den2</b>	IN	I	Jméno databázové proměnné, která obsahuje buďto přímo určení kalendářního dne pro konec hledaného intervalu, nebo relativní posun vůči běžnému dni (dnešku) ve dnech. Viz oddíl "určení data" výše.
		NONE	
<b>Měsíc2</b>	IN	I	Jméno databázové proměnné typu I, která obsahuje buďto přímo určení kalendářního měsíce pro konec hledaného intervalu, nebo relativní posun vůči běžnému dni (dnešku) v měsících. Viz oddíl "určení data" výše.
		NONE	
<b>Rok2</b>	IN	I	Jméno databázové proměnné, která obsahuje buďto přímo určení kalendářního roku pro konec hledaného intervalu, nebo relativní posun vůči běžnému dni (dnešku) v letech. Viz oddíl "určení data" výše.
		NONE	
<b>DenTýdne2</b>	IN	I	Jméno proměnné, ve které je číselné označení dne v týdnu pro konec hledaného intervalu. Viz oddíl "určení data" výše.
		NONE	
<b>hPosun2</b>	IN	I	Jméno databázové proměnné, ve které je uloženo posunutí půlnoci v sekundách pro konec intervalu. Používá se, pokud za přelom dne není považována půlnoc, ale například 22:00. Dá se také použít k tomu, chceme-li například, aby se vzorky z první čtvrt hodiny následujícího dne objevily ve výpisu daného dne. Oba přístupy lze kombinovat.
		L	
		F	
		NONE	
<b>Posun2</b>	PAR	Konst	Hodnota posunutí půlnoci, která se použije, dosadí-li se za hPosun2 NONE.
<b>Od</b>	OUT	I	Jméno proměnné, do které se uloží index, na kterém nalezená data začínají.
<b>Do</b>	OUT	I	Jméno proměnné, do které se uloží index, na kterém nalezená data končí. Archiv je organizován jako kruhový buffer, hodnoty se do něj zapisují od začátku do konce a po dosažení konce matice se pokračuje znovu od začátku s přepisováním nejstarších hodnot. Proto se může stát, že hodnota indexu Do je menší, než hodnota indexu Od. Při použití modulu <b>MtxCopy</b> je možné tyto hodnoty spojit do souvislého bloku, protože zmíněný modul je řešen tak, aby se s popsanou situací správně vypořádal.
<b>Aktivace</b>	IN	Bit	Jméno databázové proměnné, jejíž jeden bit (0 až 15) se používá k aktivaci modulu. Je-li při provedení modulu tento bit roven jedné, nuluje se a provede se hledání. Jinak modul nevykonává žádnou činnost. Dosadí-li se za jméno proměnné NONE, provádí se hledání při každém vyvolání modulu.
		NONE	

Ukončeno	OUT	Bit	Jméno databázové proměnné typu <code>I</code> , do jejíhož jednoho bitu (0 až 15) modul zapíše 1, poté co ukončí hledání, nastaví bit <code>Nalezeno</code> a případně naplní proměnné <code>Od</code> a <code>Do</code> . Za jméno proměnné lze dosadit <code>NONE</code> , v tom případě se příznak ukončení hledání nikam nezapisuje.
		NONE	

Nalezeno	OUT	Bit	Jméno proměnné typu <code>I</code> , v níž se jeden bit (0 až 15) nastaví do jedničky, pokud se požadovaný interval podařilo nalézt nebo do nuly, pokud požadovaný interval v archivu není. Pokud se data nepodaří nalézt, modul nemění hodnoty proměnných, dosazených za <code>Od</code> a <code>Do</code> .
----------	-----	-----	---

**Příklad**

```
FindDay      Casy, Index,
             Den, NONE, NONE, NONE, NONE, -8100, Den, NONE,
             NONE, NONE, NONE, -6300, Od, Do, NONE.0, NONE.0,
             Uspech.0
```

Pokud proměnná `Den` obsahuje například číslo 2, naplní se při každém vyvolání modulu proměnné `Od` a `Do` hodnotami sloupců, na kterých začínají a končí údaje, které modul **SyncArch** uložil do archivu předešlé úterý. Pokud v archivu nějaká taková data jsou, nastaví se nultý bit proměnné `Uspech` na jedničku, jinak na nulu. Přelom dne je posunut na 22:00 předešlého dne, poslední čtvrt hodina předešlého a první čtvrt hodina následujícího dne spadají rovněž do vyhledávaného intervalu.

<b>For</b>	Iterační cyklus
------------	-----------------

**Popis**

Modul **For** umožňuje realizaci iteračních cyklů. Provádění cyklu řídí tzv. *řídící proměnná cyklu*, která nabývá postupně hodnot z nějakého intervalu. Pro každou hodnotu řídící proměnné se provede tělo cyklu - vyvolají se vnořené moduly/příkazy.

**Parametry**

<b>Proměnná</b>	IN	I	Jméno řídící proměnné cyklu.
		L	
		F	

<b>Počátek</b>	PAR	Konst	Počáteční hodnota řídící proměnné.
----------------	-----	-------	------------------------------------

<b>Konec</b>	PAR	Konst	Konečná hodnota řídící proměnné.
--------------	-----	-------	----------------------------------

<b>Krok</b>	PAR	Konst	Krok změny řídící proměnné. Implicitní hodnota tohoto parametru je 1.
-------------	-----	-------	---

<b>Návěští</b>	PAR	Návěští	Návěští následujícího příkazu <b>EndFor</b> - automatické, lokální, je tedy generováno automaticky.
----------------	-----	---------	---

**Příklad**

```

For          I, 80.0, 23.0, -0.1, :00000
. . .
:00000 EndFor

```

Tělo cyklu se provede celkem 571-krát pro hodnoty řídící proměnné *I*. Krok iterace je desetinný, tedy postupně 80.0, 79.9, 79.8, ... , 23.1 a 23.0.

Všimněte si prosím, že krok změny řídící proměnné může být i desetinný a záporný. Tuto skutečnost musí ovšem respektovat počáteční a konečná hodnota.

<b>FreqOut</b>	Frekvenční výstup na digitálním výstupu
----------------	---

**Popis**

Modul umožňuje použít některé digitální výstupy jako frekvenční výstupy, případně výstupy pro pulzní šířkovou modulaci (PWM). Viz též dodatek *"Použití digitálních výstupů jako frekvenčních nebo impulzních výstupů"*.

**Parametry**

<b>Signál</b>	OUT	DO	Číslo zapisovaného signálu DO.
---------------	-----	----	--------------------------------

<b>Frekvence</b>	IN	Konst.	Výstupní frekvence v Hz.
		F	
		MF	

<b>Střída</b>	IN	Konst.	Střída udává poměr (v procentech) mezi dobou, kdy je výstup ve stavu ON, a celou periodou výstupního signálu. Implicitní střída 50 % tedy znamená, že signál bude ve stavu ON stejně dlouho jako ve stavu OFF. Pokud se střída přiblíží k nule, resp. k hodnotě 100 %, natolik, že by doba trvání jednoho stavu ON, resp. OFF, byla kratší než 10 µs, výstup se zastaví ve stavu OFF, resp. ON.
		F	
		MF	

<b>MinFrekv</b>	PAR	Konst	Minimální generovaná frekvence v Hz - viz rozbor v dodatku <i>"Použití digitálních výstupů jako frekvenčních nebo impulzních výstupů"</i> . Je-li za parametr Frekvence předána menší hodnota, frekvenční výstup se zastaví - uvede se do polohy OFF.
-----------------	-----	-------	--

Inverze	PAR	Výběr	Udává vztah mezi stavy ON/OFF a fyzickým stavem výstupu. To má vliv jednak na elektrický význam parametru Střída, za druhé na to, v jakém stavu bude výstup v případě jeho zastavení zadáním Frekvence nižší než MinFrekv.	
			Hodnota	Význam
			0	BezInverze - ON odpovídá logické jedničce, OFF odpovídá logické nule. V případě zastavení zůstane výstup ve stavu logické nuly.
			1	Inverzní - ON odpovídá logické nule, OFF odpovídá logické jedničce. V případě zastavení zůstane výstup ve stavu logické jedničky.

*Pozn.: V případě, že je v aplikaci použito více modulů FreqOut nebo PulseOut se stejnou hodnotou parametru Signál, je výsledné chování jejich společného výstupu určeno takto:*

*Má-li v kterémkoliv z dotyčných modulů parametr Inverze hodnotu Inverzní, výstup funguje inverzně.*

*Má-li ve všech dotyčných modulech parametr Inverze hodnotu BezInverze, výstup funguje normálně.*

**Příklad**

FreqOut #0.0, Frekv, 50.0, 1.0, BezInverze

Na digitálním signálu DO0.0 je generována frekvence určená proměnnou Frekv, s pevnou střídou 50% (1:1) nejnižší možná frekvence je 1 Hz. Pro hodnoty proměnné Frekv nižší než 1 je frekvenční výstup zastaven ve stavu logické nuly.

FreqOut #0.1, 100.0, Střida, 1.0, Inverzní

Na digitálním signálu DO0.0 je generována pulzní šířková modulace (PWM) s pevnou frekvencí 100 Hz, střída 0÷100 % je určena proměnnou Střida. Výstup pracuje inverzně, takže střídě 100 % odpovídá trvalá logická nula, střídě 0 % trvalá logická jednička.



<b>GetTime</b>	Načtení okamžitého času do databázové proměnné a vyhodnocení změn
----------------	---

**Popis**

Modul získá okamžitý čas v DB-Net formátu do proměnné `Cas`. Dále rozloží okamžitý čas na položky a uloží je do proměnné `DMYhms`. Do proměnné `DeltaTime` uloží změny, ke kterým došlo od posledního volání modulu. Chceme-li například provádět nějakou akci při změně dne, tak stačí pouze sledovat, zda 3. bit (číslováno od 0.bitu) je ve stavu "1".

**Parametry**

<b>Čas</b>	OUT	L	Obsahuje okamžitý čas v DB-Net formátu (počet sekund od 1.1.1980 0:0:0).
		ML	

<b>Složky</b>	OUT	MI	Okamžitý čas po položkách.
		NONE	

Proměnná - matice rozměru [8x1], která obsahuje okamžitý čas rozebraný na jednotlivé položky - den, měsíc, rok, hodina, sekunda, minuta:

Řádek	Význam
0	sekundy
1	minuty
2	hodiny
3	den v měsíci
4	měsíc
5	rok <i>do roku 2000 jsou hodnoty 80..99, počínaje rokem 2000 jsou hodnoty 00 a výše</i>
6	den týdne
7	den roku

Den týdne má následující význam:

Hodnota	Význam
0	Neděle
1	Pondělí
2	Úterý
3	Středa
4	Čtvrtek
5	Pátek
6	Sobota

<b>Změny</b>	OUT	I	Tato proměnná nastavuje příznakové bity podle události, ke které od posledního volání modulu došlo.
		NONE	

Bit	Změna
0	sekundy
1	minuty
2	hodiny
3	dne
4	měsíce
5	roku

**Příklad**

```
GetTime      Cas, Cas_polozky, Cas_zmena
If           Cas_zmena.3, :00001
... příkazy při změně dne
:00001 EndIf
```

Mezi příkazy **If** a **EndIf** jsou příkazy pro volání modulů, které se vykonají při změně dne.

**GGraf**

Vysílání hodnoty do grafu na grafickém terminálu

**Popis**

Modul slouží pro vyslání hodnoty do grafu na grafickém terminálu.

Graf musí být v terminálu nadefinován pomocí programu **WGradet**.

Vyvolání tohoto modulu nezpůsobí vykreslení grafu na displej terminálu. K tomu je nutno umístit do příslušné obrazovky v programu **LCDSHELL** prvek **GItem**.

Je-li graf v okamžiku vyvolání modulu **Ggraf** vykreslen na displeji terminálu, překreslí se jeho průběh tak, aby odpovídal nové tabulce hodnot.

Vysílání hodnot do grafu chceme zpravidla provádět buďto se zadanou periodou, nezávisle na okamžitém režimu zobrazování, nebo při vyhodnocení nějaké události. **LCDSHELL** nemá prostředky ani pro periodické vysílání hodnot, ani pro vyhodnocování událostí. Zároveň ale zpravidla chceme graf zobrazovat na displeji jen tehdy, je-li aktivní příslušná obrazovka **LCDSHELL**u. Proto je obsluha grafů rozdělena do dvou částí - vysílání hodnot řídí aplikace v **PSE**, zobrazování grafu řídí prvek v **LCDSHELL**u.

V aplikaci musí být použit modul pro obsluhu grafického terminálu, např. LCDGTTY, LCD2100.

**Parametry**

Režim	PAR	Konst	Režim činnosti modulu.
XY Graf		ANO	Vysílají se obě souřadnice $x$ i $y$ . Graf musí mít při definici v programu <b>Wgradet</b> zvolen typ "XY-graf".
		NE	Vysílá se pouze souřadnice $y$ , souřadnice $x$ je určena pořadím vzorku. Graf musí mít při definici v programu <b>Wgradet</b> zvolen typ "Y-graf".
WID	IN	Konst	Identifikátor, který byl grafu přidělen v programu <b>WGradet</b> .
		I	
		MI	
X	IN	Konst	Hodnota souřadnice $X$ . Zadává se v rozmezí, které bylo pro graf určeno při definici v programu <b>WGradet</b> .
		I	
		L	
		F	
		MI	
		ML	
		MF	
Y	IN	Konst	Hodnota souřadnice $Y$ . Zadává se v rozmezí, které bylo pro graf určeno při definici v programu <b>WGradet</b> .
		I	
		L	
		F	
		MI	
		ML	
		MF	
Ax	IN	Konst	Konstanta pro přepočít podle vzorce $Ax \cdot X + Bx$ .
		I	
		L	
		F	
		MI	
		ML	
		MF	

Bx	IN	Konst	Konstanta pro přepočet podle vzorce $Ax \cdot X + Bx$ .
		I	
		L	
		F	
		MI	
		ML	
		MF	

Ay	IN	Konst	Konstanta pro přepočet podle vzorce $Ay \cdot Y + By$ .
		I	
		L	
		F	
		MI	
		ML	
		MF	

By	IN	Konst	Konstanta pro přepočet podle vzorce $Ay \cdot Y + By$ .
		I	
		L	
		F	
		MI	
		ML	
		MF	

**Příklad**

GGraf                      0x0004, 3, PolohaX, PolohaY, 1.0, 0.0, 1.0, 0.0

Modul vyšle do XY-grafu s WIDem 3 souřadnice bodu PolohaX, PolohaY. Přepočet souřadnic se neprovádí.

**GGrafClr**

Vyprázdnění tabulky hodnot grafu na grafickém terminálu

**Popis**

Modul slouží pro vymazání všech hodnot, které byly vyslány do grafu použitím modulu **GGraf**.

Graf musí být v terminálu nadefinován pomocí programu **WGradet**.

Vyvolání tohoto modulu nezpůsobí odstranění grafu z displeje terminálu, je-li zobrazen. Způsobí pouze smazání křivky průběhu. Případné souřadnicové osy zůstanou zobrazeny.

V aplikaci musí být použit modul pro obsluhu grafického terminálu, např. LCDGTTY, LCD2100.

**Parametry**

<b>WID</b>	IN	Konst	Identifikátor, který byl grafu přidělen v programu <b>WGradet</b> .
		I	
		MI	

**Příklad**

```
GGrafClr      3
```

Modul vyprázdní tabulku hodnot grafu s WIDem 3.

<b>Holiday</b>	Definice prázdnin pro modul <b>DayPlan</b>
----------------	--

**Popis**

Ve dnech, které jsou nastaveny jako období prázdnin v matici Prázdniny, modul kopíruje každý den aktuální informaci do vybraného sloupce v matici svátků Svátky, kterou se řídí modul **DayPlan**. Svátky je matice svátků z modulu **DayPlan**. Prázdniny je matice obsahující období prázdnin.

**Parametry**

<b>Svátky</b>	OUT	MI	Sloupec databázové proměnné - matice [3xn] svátků pro modul <b>DayPlan</b> . Zadává se sloupec, do kterého modul <b>Holiday</b> zapisuje.
---------------	-----	----	---

<b>Prázdniny</b>	IN	MI	Matice rozměru [5xm] pro období prázdnin, kde m je počet období (úseků) prázdnin v roce.
------------------	----	----	--

Význam jednotlivých sloupců matice:

0	den začátku prázdnin
1	měsíc začátku prázdnin
2	den konce prázdnin
3	měsíc konce prázdnin
4	maska typu dne, za který se má období prázdnin považovat (např. neděle).

**Příklad**

```
HoliDay      Svatky[*,6], Prazdniny

DayPlan      0x0000, 2, 0x001F, 0x0060, 0x0000, 0x0000, 0x0000,
              0x0000, 0x0000, 0x0000, Svatky, PlanCasy, Plan,
              Teplota
```

Matice Prazdniny nechť je rozměru [5,1] a má hodnoty uvedené v následující tabulce. Hodnota v posledním řádku matice je maska dnů a má význam "sobota nebo neděle" (kód 32 + 64) - viz popis modulu **DayPlan**.

1
7
31
8
96

Modul **Holiday** bude v období prázdnin od 1.7. do 31.8. zapisovat do 6. sloupce matice Svátky aktuální den a typ dne "sobota nebo neděle". V období prázdnin tak bude modul **DayPlan** plánovat podle druhého řádku matic PlanCasy a Plan. Modul má totiž pro první řádek má nastaveny všední dny (parametr = 0x001F), kdežto pro druhý řádek má nastaveny dny "sobota nebo neděle" (parametr = 0x0060).

<b>HourRun</b>	Sledování provozních hodin zařízení
----------------	-------------------------------------

**Popis**

Pokud je zařízení zapnuto (signál `Zapnuto` je v "1"), měří se provozní hodiny a sekundy. Ty jsou pak uloženy v matici `Hodiny`.

**Parametry**

<b>Zapnuto</b>	IN	Bit	Bit databázové proměnné, který indikuje zda je zařízení zapnuto či ne.
----------------	----	-----	--

Hodiny	OUT	ML	Provozní hodiny. Řádek databázové proměnné rozměru $[n \times 2]$ , kde $n$ je libovolné, dle požadavků programátora. Význam sloupců:	
			0	1
			Hodiny	Sekundy

V 0. sloupci jsou uloženy hodiny a v 1. sloupci sekundy.

<b>Nuluj</b>	IN	Bit	Nulování počítadla provozních hodin ("1" = nuluj). Modul bit po zpracování vynuluje.
		NONE	

**Příklad**

```
HourRun Zapnuto.2, Provoz[3,*]
```

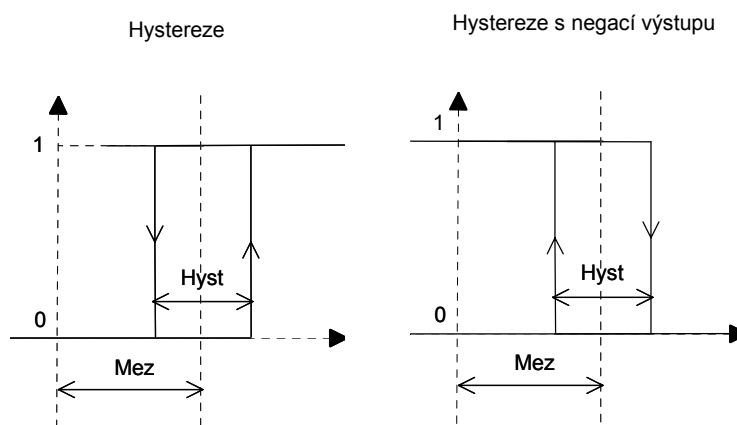
Pokud je 2. bit proměnné `Zapnuto` v "1", sumují se do buňky `Provoz[3,0]` provozní hodiny a do buňky `Provoz[3,1]` provozní sekundy zařízení.

**Hyst**

Test analogového údaje na mez s hysterezí

**Popis**

Modul **Hyst** umožňuje testování hodnoty analogového údaje na zadanou mez. Při testování je možné zadat pásmo necitlivosti (hysterezi) v okolí meze, která potlačí zbytečné mnohonásobné změny výsledku testu v případě, že se testovaná hodnota pohybuje v těsném okolí meze. Výsledek testování lze použít pro odvození alarmu nebo jiné následné akce.

**Parametry**

<b>Vstup</b>	IN	F	Proměnná s testovanou hodnotou.
		MF	
<b>Výstup</b>	OUT	BIT	Výsledek testu.
<b>Mez</b>	IN	Konst	Porovnávaná hodnota.
		F	
		MF	
<b>Hystereze</b>	IN	Konst	Hodnota hystereze.
		F	
		MF	
<b>Negace</b>	PAR	Výběr	ANO = negovat výsledek testu.

**Příklad**

```
Hyst      TestVstup, TestVystup.0, 100.0, 10.0, 0x0000
```

Modul porovnává hodnotu proměnné `TestVstup` s hodnotou `100.0` s uvažováním hystereze `10.0`. Výsledek porovnání je zapsán do bitu č. 0 proměnné `TestVystup`.

<b>If</b>	Podmíněný příkaz
-----------	------------------

**Popis**

Modul **If** realizuje podmíněný příkaz typu **If-EndIf** nebo **If-Else-EndIf**. Příkaz testuje podmínku (nastavení bitu v proměnné typu **I**) a je-li splněna, vykoná sérii příkazů resp. funkčních modulů za ním bezprostředně následující. Narazí-li na příkaz **Else**, přeskočí následující funkční moduly. Zpracování pokračuje prvním funkčním modulem za příkazem **EndIf**. Není-li podmínka splněna, pokusí se předat řízení prvnímu funkčnímu modulu za příkazem **Else**. Pokud tento příkaz není nalezen, předává řízení prvnímu funkčnímu modulu za příkazem **EndIf**. Funkční moduly "uvnitř" příkazu **If** se tedy v tomto případě nevykonají.

**Parametry**

<b>Podmínka</b>	IN	Bit	Podmínka provedení příkazů/modulů mezi <b>If</b> a následujícím <b>Else</b> nebo <b>EndIf</b> .
-----------------	----	-----	---

<b>Návěští</b>	PAR	Návěští	Návěští následujícího příkazu <b>Else</b> nebo <b>EndIf</b> - automatické, lokální, je tedy generované automaticky.
----------------	-----	---------	---

**Příklad**

```

                If      Test.0, :00000  Je-li digitální signál
                                   (bit) nastaven,
                . . .      proved' tuto sekvenci
                                   příkazů/modulů,
:00000 Else :00001      a není-li nastaven,
                . . .      proved' tuto sekvenci
:00001 EndIf

                                   Následující
                                   příkazy/moduly
                                   prováděj vždy

```



<b>Impln</b>	Ošetření šestnácti impulzních vstupů
--------------	--------------------------------------

### Popis

Modul **Impln** ošetřuje šestnácti impulzních vstupů, načítaných z logického kanálu DI. Předzpracované hodnoty neukládá do databáze, ale uchovává je ve vnitřních proměnných. Tyto hodnoty musejí být dále zpracovány modulem **DImp**. Modul **Impln** musí být umístěn v rychlém procesu (ProcQUICK).

Každý ze šestnácti signálů může mít aktivní vzestupnou nebo sestupnou hranu případně obě hrany. U netvarovaných signálů bývá většinou aktivní pouze vzestupná hrana.

Podrobný popis celého mechanismu ošetřování impulzních vstupů byl uveden v popisu modulu **DImp**.

### Parametry

<b>Kanál</b>	IN	DI16	Číslo čteného logického kanálu DI. Všechny signály tohoto kanálu budou ošetřovány jako impulzní. To však nebrání použití modulu <b>DigIn</b> nad tímtéž kanálem a ošetření vybraných signálů jako stavových.
<b>MinDélka</b>	PAR	Konst	Minimální přípustná délka mezi pulzy [ms]. Následuje-li pulz po předchozím pulzu v čase kratším než je tato minimální délka, je modulem ignorován.
<b>Timeout</b>	PAR	Konst	Maximální přípustná délka mezi pulzy [s]. Nepřijde-li po tuto dobu žádný impulz, chápe se měřená hodnota jako nulová, tzn. na impulz se "déle nečeká".
<b>Náběžná</b>	PAR	Výběr	Parametr udává pro každý ze šestnácti vstupních signálů, zda má aktivní vzestupnou hranu (strukturované číslo s položkami Bit0 až Bit 15).
<b>Sestupná</b>	PAR	Výběr	Parametr udává pro každý ze šestnácti vstupních signálů, zda má aktivní sestupnou hranu (strukturované číslo s položkami Bit0 až Bit 15).

### Příklad

```
ProcQUICK:
:1000 ImpIn #3, 100, 30, 0xFFFF, 0x0000
Proc00:
    Dimp      :1000,2,Delta,Pocet,Odhad,Konst,Sync.0, NONE
```

Modul **Impln** v rychlém procesu zpracovává 16 impulzních vstupů z logického kanálu DI č. 3. Minimální vzdálenost impulzů v čase je 100 ms, maximální vzdálenost impulzů v čase je 30 sekund, aktivní je pouze náběžná hrana každého impulzu.

Modul **DInput** v procesu č. 0 zpracovává signál č. 2 modulu **Impln**. Proměnná **Pocet** je průběžným počítadlem, proměnná **Delta** je přírůstek počítadla od minula, proměnná **Odhad** obsahuje odhad okamžité hodnoty, v proměnné **Konst** je konstanta měřidla (např. 0.125 m<sup>3</sup>/impulz). Nastavením bitu **Sync.0** do "1" se počítadlo vynuluje.

<b>IncDec</b>	Inkrementace/dekrementace proměnné
---------------	------------------------------------

**Popis**

Modul inkrementuje nebo dekrementuje proměnnou `Hodnota` o velikost `Přírůstek`. `Přírůstek` je hodnota typu `F`. Pokud je vstupní hodnota jiného typu (`I` nebo `L`), tak je hodnota `Přírůstek` automaticky konvertována na daný typ. Výsledkem konverze příliš velkých čísel (např. `1E15`) bude maximální číslo, které je možné v daném typu uchovat (pro `I` je to např. `32767`). Činnost modulu je blokována bitem `Běh`. Pokud je jeho hodnota nulová, modul nedělá nic.

**Parametry**

<b>Hodnota</b>	IN/OUT	I	Proměnná, která bude modulem inkrementována nebo dekrementována.
		L	
		F	
<b>Běh</b>	IN	Konst	Bit povoluje činnost modulu (1=běží).
		Bit	
		DI	
<b>Přírůstek</b>	IN	Konst	Velikost přírůstku při inkrementaci nebo dekrementaci.
		F	
		MF	

**Příklad**

```
IncDec      IVal, Beh, 1.0
```

Modul bude inkrementovat hodnotu `IVal` (předpokládejme, že je typu integer) o jedničku.

<b>Interpol</b>	Obecný funkční měnič (interpolace)
-----------------	------------------------------------

**Popis**

Modul zajišťuje lineární interpolaci vstupního signálu podle zadané tabulky. Tímto způsobem lze velmi snadno realizovat tzv. obecný funkční měnič, tedy modul realizující libovolnou funkci jednoho argumentu:  $y = f(x)$ .

**Parametry**

<b>Vstup</b>	IN	F	Vstupní proměnná (nezávislá proměnná $x$ ).
		MF	

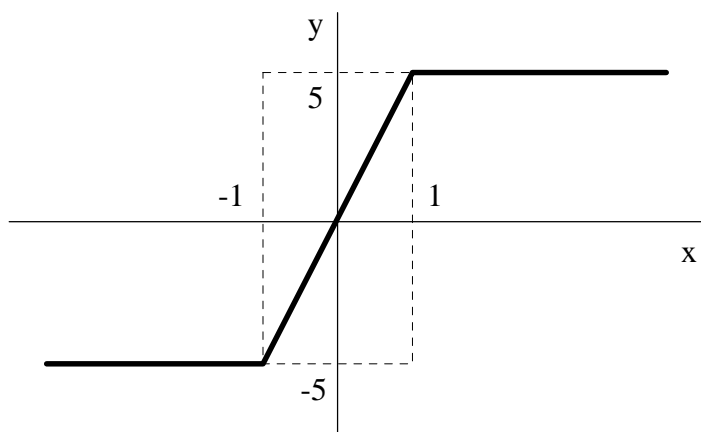
<b>Výstup</b>	OUT	F	Výstupní proměnná (funkční hodnota $y$ ).
		MF	

<b>Křivka</b>	IN	MF	Matice o rozměru $[N, 2]$ , která obsahuje jednotlivé body lomené čáry, definující funkci $y = f(x)$ . Ve sloupci 0 jsou hodnoty $x$ , ve sloupci 1 jim odpovídající hodnoty $y$ . Počet řádků matice je libovolný (tedy 2 až 999), podle potřebného počtu bodů na křivce $y = f(x)$ .
---------------	----	----	--

**Příklad**

Interpol Vstup, Výstup, Křivka

Je-li matice **Křivka** typu MF[2, 2] s hodnotami (po řádcích) -1.0, -5.0 a +1.0, +5.0, pak uvedený funkční modul realizuje nelinearitu typu nasycení se zesílením 5 a pásmem proporcionality <-1, +1> (viz obrázek). Vstupem modulu je proměnná **Vstup** a výstupem proměnná **Výstup**.



<b>InterXY</b>	Lineární interpolace $Z=F(X,Y)$
----------------	---------------------------------

**Popis**

Pro interpolaci ze dvou proměnných je potřeba mít správně zadanou interpolační matici `Plocha` s inicializovanými hodnotami, která představuje interpolovanou funkci dvou proměnných. Rozměry matice určují počet definičních bodů. Osa  $x$  jde ve směru sloupců, osa  $y$  ve směru řádků matice. Tabulka obsahuje v 0. řádku definiční hodnoty  $x_i$ . V 0. sloupci obsahuje definiční hodnoty  $y_j$ . Modul pak na základě proměnných `VstupX` a `VstupY` aproximuje dle tabulky `Plocha` výslednou hodnotu, kterou uloží do proměnné `Výstup`. Souřadnice `[0,0]` je nevyužitá. Hodnota interpolované funkce  $F$  v bodě  $(x_i, y_j)$  se v matici zadává do buňky `Plocha[i,j]`:

$$Plocha[i,j] = F(Plocha[0,i], Plocha[j,0]),$$

kde  $F$  je interpolovaná funkce.

	0	1	...	...	n
0		X1	...	...	Xn
1	Y1	F(X1,Y1)			F(Xn,Y1)
2	Y2	F(X1,Y2)			F(Xn,Y2)
.	.				
.	.				
.	.				
m	Ym	F(X1,Ym)			F(Xn,Ym)

V silně orámované části matice jsou funkční hodnoty.

**Parametry**

<b>VstupX</b>	IN	F	Proměnná X.
		MF	

<b>VstupY</b>	IN	F	Proměnná Y.
		MF	

<b>Výstup</b>	OUT	M	Výstupní proměnná - funkční hodnota.
		MF	

<b>Plocha</b>	IN	MF	Matice, která obsahuje definované hodnoty na osách $x$ a $y$ , a jim přiřazené hodnoty na ose $z$ . Musí to být inicializovaná proměnná, kterou lze za chodu dle potřeby modifikovat.
---------------	----	----	---

**Příklad**

InterXY  $X, Y, Z, Fce$

V proměnné  $z$  je interpolovaná hodnota funkce v bodě  $(x, y)$ . Funkce je zadaná maticí  $Fce$ .

Nechť matice  $Fce$  má hodnoty:

	0	1	2	3
0		1,5	1,6	1,7
1	0,1	100	122	135
2	0,2	114	220	298
3	0,3	131	318	411

Pak pro interpolovanou funkci platí:

$funkce(1.5, 0.1) = 100,$   
 $funkce(1.6, 0.3) = 318,$   
 atd.

<b>IRCI<sub>n</sub></b>	Čtení hodnoty z hardwareového čítačového vstupu / vstupu pro inkrementální čidla polohy
-------------------------	---

**Popis**

Modul zajišťuje načtení hodnoty z hardwareového čítačového vstupu resp. vstupu pro inkrementální čidla polohy. Zároveň zajišťuje přepočet počtu impulsů (kvant) na hodnotu fyzikální veličiny.

Pro přepočet na fyzikální hodnotu se použijí konstanty, které byly pro daný vstup nastaveny předchozím voláním funkčního modulu **IRCM<sub>ode</sub>**.

Modul se umísťuje do procesu, ve kterém se dále zpracovává hodnota polohy nebo jiného údaje měřeného pomocí čítačového vstupu.

**Parametry**

<b>Čítač</b>	IN	Konst	Číslo čítačového vstupu / vstupu pro inkrementální čidla. Číslování vstupů viz popis čítačových vstupů / vstupů pro inkrementální čidla polohy příslušného typu procesní stanice v dodatku "Čítačové vstupy / vstupy pro inkrementální čidla polohy"
		I	
		MI	
<b>Hodnota</b>	OUT	F	Výstupní proměnná (hodnota fyzikální veličiny).
		MF	

**Příklad**

ProcInit:

IRCM<sub>ode</sub> 0, IRC F1-F2, Obě hrany, 0.0, 0.025,  
NONE.0

ProcITR3:

IRCSet 0, KoncSpin

Proc00:

IRCI<sub>n</sub> 0, Hodnota

V procesu INIT je nastaven režim vstupu číslo 0 pro inkrementální čidlo polohy. Jedná se o inkrementální čidlo s výstupem ve formě dvou fázově posunutých signálů. Rozteč měřicích proužků je 0.1mm, z toho vyplývá krok 0.025mm (při posunu celého čidla o jeden proužek vzniknou celkem 4 přírůstky čítače). Hodnoty proměnných KoncSpin a Hodnota se udávají v milimetrech. Hodnota bitu Úspěch se nikam neukládá, předpokládá se, že zvolená kombinace čísla čítače, režimu a volby aktivních hran je na daném typu procesní stanice použitelná.

V procesu ITR3 se stav čítače nastavuje na hodnotu odpovídající poloze koncového spínače, konkrétní poloha v milimetrech je uložena v databázové proměnné KoncSpin. Předpokládá se, že k digitálnímu vstupu, odpovídajícímu interruptprocesu ITR3, je připojen signál od koncového spínače či jiného kalibračního čidla polohy.

V procesu 0 se získává okamžitá hodnota čidla polohy v milimetrech a ukládá se do databázové proměnné Hodnota.

**IRCMoDe**

Nastavení režimu a konstant hardwareového čítačového vstupu / vstupu pro inkrementální čidla polohy

**Popis**

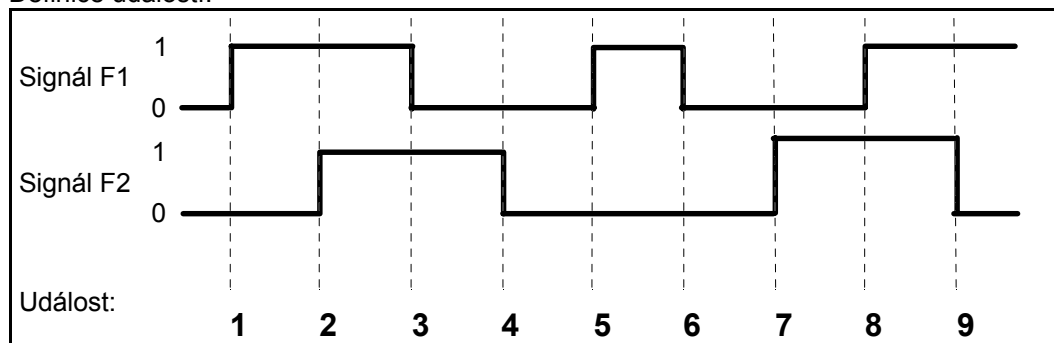
Modul zajišťuje nastavení režimu hardwareového čítačového vstupu resp. vstupu pro inkrementální čidla polohy. Zároveň umožňuje volbu aktivní hrany (náběžné / sestupné / obou hran) čítaných pulsů. Dále se jeho pomocí zadávají konstanty pro přepočet počtu impulsů (kvant) na hodnotu fyzikální veličiny.

**Režimy vstupu**

Pro popis režimů viz též popis parametru *Režim* níže.

Zde uvádíme přehlednou tabulku, jak vstup v jednotlivých režimech reaguje na různé kombinace vstupních signálů a hran.

Definice událostí:



Reakce na události:

(V tabulce "+" znamená zvýšení hodnoty čítače, "-" znamená snížení hodnoty čítače, prázdné políčko znamená, že čítač na událost nereaguje):

Režim vstupu	Volba aktivní hrany	Reakce na událost:								
		1	2	3	4	5	6	7	8	9
IRC F1-F2	nebere se v úvahu	+	+	+	+	+	-	-	-	-
IRC F2-F1	nebere se v úvahu	-	-	-	-	-	+	+	+	+
Směr +/-	Náběžná		-					+		
	Sestupná				+					-
	Obě hrany		-		+			+		-
Směr -/+	Náběžná		+					-		
	Sestupná				-					+
	Obě hrany		+		-			-		+
Nahoru	Náběžná		+					+		
	Sestupná				+					+
	Obě hrany		+		+			+		+
Dolů	Náběžná		-					-		
	Sestupná				-					-
	Obě hrany		-		-			-		-
F1+ F2-	Náběžná	+	-			+		-	+	
	Sestupná			+	-		+			-
	Obě hrany	+	-	+	-	+	+	-	+	-
F1- F2+	Náběžná	-	+			-		+	-	
	Sestupná			-	+		-			+
	Obě hrany	-	+	-	+	-	-	+	-	+

## Počáteční podmínky

Po startu procesní stanice jsou interní čítače, příslušející k čítačovým vstupům / vstupům pro inkrementální čidla polohy, vynulovány. Před prvním vyvoláním modulu **IRCMoDe** se případné pulsy na vstupech ignorují.

Po startu procesní stanice a vyvolání modulu **IRCMoDe** by se měla vždy zajistit synchronizace údaje čítače se skutečnou polohou, například rozjetím pohonu směrem ke koncovému spínači nebo k poziční značce - blíže viz oddíl "*Obsluha pozičních značek*" níže. Do provedení této synchronizace by aplikace neměla využívat hodnoty získané pomocí modulu **IRCIIn**.

Alternativou je průběžné ukládání polohy, získané voláním modulu **IRCIIn**, do databázové proměnné, která se neiniculuje po teplém startu, s následným použitím této proměnné pro vyvolání modulu **IRCSet** po teplém startu (z procesu **INIT**). Při této metodě se ovšem ztrácí přesnost neuvažováním pulsů, které přišly v době mezi posledním voláním modulu **IRCIIn** a výpadkem napájení procesní stanice. Je tedy zpravidla žádoucí i v tomto případě provést po náběhu systému rekaliibraci pomocí signálu indikujícího absolutní polohu.

## Obsluha pozičních značek

Inkrementální čidla polohy bývají zpravidla vybavena jednou nebo více pozičními (indexovými) značkami, které slouží k vynulování skutečné polohy, popřípadě nastavení skutečné polohy na předem určenou hodnotu, odpovídající dané značce. Vstupy pro tato čidla musejí poskytovat separátní vstupy pro poziční značky a umožňovat při aktivaci poziční značky nastavení hodnoty čítače na odpovídající předvolenou hodnotu (nejčastěji nulovou). Pomineme-li možnost manuální synchronizace polohy (kdy obsluha např. stiskem tlačítka vyvolá proces obsahující modul **IRCSet**), jsou možné dva režimy obsluhy pozičních značek:

- ♦ **Automatický režim** - vyžaduje speciální hardwareovou výbavu čítačového vstupu / vstupu pro inkrementální čidla polohy - dedikovaný signálový vstup (vstupy), po jehož aktivaci vstupní obvody bez účasti software samotného řídicího systému naplní interní čítač hodnotou, která byla předtím naplněna do dedikovaného registru vyvoláním modulu **IRCPreset**.
- ♦ **Poloautomatický režim** - umožňuje využít pro poziční značku v podstatě kterýkoliv digitální vstup, preferovaně takový, ke kterému je možno přiřadit přerušovací proces ProclTR0 až ProclTR15. Synchronizaci v tom případě provádí aplikační software řídicího systému vyvoláním modulu **IRCSet**. V tomto režimu bývá zpravidla nutno omezit rychlost pojezdu v okamžiku synchronizace tak, aby mezi okamžikem aktivace vstupu od poziční značky a okamžikem vyvolání modulu **IRCSet** nedošlo ke ztrátě impulzů na vlastních čítačích vstupech.

Která zařízení umožňují použití automatického režimu, popř. kolika dedikovanými vstupy pro poziční značky jsou vybavena, viz dodatek "*Čítačové vstupy / vstupy pro inkrementální čidla polohy*".

## Převod na fyzikální veličinu

Modulem **IRCMoDe** se také zadávají konstanty pro převod hodnoty interních čítačů na fyzikální veličinu (např. délku ve vhodných délkových jednotkách). Přepočet podle zadanych konstant probíhá v modulech **IRCIIn** a **IRCSet** podle vzorce:

$$\text{Hodnota} = \text{Počet} * \text{Krok} + \text{Nula}$$

Kde:

- ♦ Počet je stav interního čítače
- ♦ Krok, Nula jsou stejnojmenné parametry modulu **IRCMoDe**
- ♦ Hodnota je údaj v požadovaných fyzikálních jednotkách

Z tohoto vzorce vyplývá, že jelikož Počet je bezrozměrné číslo, vychází Hodnota ve stejných fyzikálních jednotkách, ve kterých je zadán Krok a Nula.

Modul **IRCSet** provádí výpočet hodnoty Počet ze zadané Hodnoty podle inverzního vzorce.

Při určování konstant v režimech vstupu IRC F1-F2 a IRC F2-F1 je třeba brát v úvahu, že jedné periodě kteréhokoliv signálu (což odpovídá jedné rozteči mezi měřicími proužky) odpovídá zpravidla větší množství přírůstků interního čítače. Počet přírůstků na jednu

periodu pro konkrétní vstup konkrétního typu procesní stanice najdete v dodatku “Čítačové vstupy / vstupy pro inkrementální čidla polohy”.

#### Umístění do procesu

Modul **IRCMoDe** se obvykle umísťuje do procesu **INIT**, ale je možné ho umístit i do periodického procesu, pokud chceme obsluhu umožnit změnu režimu nebo jiných parametrů za chodu aplikace (například z terminálu).

Při umístění do periodického procesu je nutno zabezpečit (např. pomocí modulu **If**), aby se modul vyvolával jen tehdy, jestliže dojde ke změně parametrů, a aby v okamžiku jeho vyvolání nebyly na vstup přivedeny pulsy, protože po dobu vykonávání modulu **IRCMoDe** může dojít ke ztrátě nebo chybnému vyhodnocení pulsů.

#### Změna parametrů za chodu

Voláním modulu **IRCMoDe** se nemění hodnota interních čítačů.

Použije-li se tedy tento modul k tomu, aby se za chodu aplikace změnila hodnota parametrů **Nula** a **Krok**, použije se stará hodnota interních čítačů s novými konstantami, takže při prvním následujícím volání modulu **IRCIIn** dojde ke skokové změně hodnoty, aniž by byly na vstup přivedeny jakékoliv pulsy. V případech, kdy je takovýto stav na závadu, je třeba po změně parametrů **Nula** a **Krok** zajistit správné nastavení hodnoty použitím modulu **IRCSet**.

#### Parametry

Čítač	IN	Konst	Číslo čítačového vstupu / vstupu pro inkrementální čidla. Číslování vstupů viz popis čítačových vstupů / vstupů pro inkrementální čidla polohy příslušného typu procesní stanice v dodatku “Čítačové vstupy / vstupy pro inkrementální čidla polohy”
		I	
		MI	



Režim	IN	Konst	Číselný kód pracovního režimu vstupu.		
		I	Kód	Režim	Popis
		MI	0	IRC F1-F2	Inkrementální čidlo polohy s výstupem ve formě dvou fázově posunutých signálů. Při pohybu směrem k vyšším hodnotám signál F1 předchází před signálem F2.
		1	IRC F2-F1	Inkrementální čidlo polohy s výstupem ve formě dvou fázově posunutých signálů. Při pohybu směrem k vyšším hodnotám je signál F1 opožděn za signálem F2.	
		2	Směr +/-	Obousměrný čítač impulsů na vstupu F2, směr je určen hodnotou na vstupu F1 tak, že při F1 v logické "0" a pulsech na F2 se hodnota zvyšuje, při F1 v logické "1" se snižuje.	
		3	Směr -/+	Obousměrný čítač impulsů na vstupu F2, směr je určen hodnotou na vstupu F1 tak, že při F1 v logické "0" a pulsech na F2 se hodnota snižuje, při F1 v logické "1" se zvyšuje.	
		4	Nahoru	Jednosměrný čítač impulsů na vstupu F2, při každém pulsu se hodnota zvyšuje.	
		5	Dolů	Jednosměrný čítač impulsů na vstupu F2, při každém pulsu se hodnota snižuje.	
		6	F1+ F2-	Obousměrný čítač se dvěma vstupy, při pulsech na vstupu F1 se hodnota zvyšuje, při pulsech na vstupu F2 se snižuje.	
		7	F1- F2+	Obousměrný čítač se dvěma vstupy, při pulsech na vstupu F1 se hodnota snižuje, při pulsech na vstupu F2 se zvyšuje.	
			Přiřazení signálů F1 a F2 ke konkrétním vstupům konkrétního typu procesní stanice najdete v dodatku "Čítačové vstupy /vstupy pro inkrementální čidla polohy"		

Hrany	IN	Konst	Číselný kód pro volbu aktivní hrany signálu.		
		I	Kód	Režim	Popis
		MI	1	Náběžná	Vstup reaguje na náběžnou hranu signálu, tedy na změnu signálu z logické "0" do logické "1"
			2	Sestupná	Vstup reaguje na sestupnou hranu signálu, tedy na změnu signálu z logické "1" do logické "0"
			3	Obě hrany	Vstup reaguje na náběžnou i na sestupnou hranu.
Pozn.: V režimech IRC F1-F2 a IRC F2-F1 se hodnota předaná za parametr Hrany nebere v úvahu, vstup reaguje na obě hrany bez ohledu na hodnotu parametru.					

Nula	IN	Konst	Hodnota fyzikální veličiny, které odpovídá nulová hodnota interního čítače. Tuto hodnotu bude vracet modul <b>IRCI</b> n před příchodem prvního pulsu.
		F	
		MF	

Krok	IN	Konst	Hodnota fyzikální veličiny, která odpovídá jednomu pulsu (kvantu). Při režimech IRC F1-F2 a IRC F2-F1 zpravidla jedné periodě vstupních signálů odpovídá větší množství kvant. Viz dodatek "Čítačové vstupy /vstupy pro inkrementální čidla polohy".
		F	
		MF	

Úspěch	OUT	I	Jméno databázové proměnné, jejíž jeden bit (0÷15) se nastaví na hodnotu 1, pokud nastavení režimu proběhne úspěšně. Volání modulu <b>IRCMoDe</b> může skončit neúspěchem v následujících případech: - Na daném typu procesní stanice neexistuje čítačový vstup / vstup pro inkrementální čidlo polohy s číslem předaným za parametr Čítač. - Na daném vstupu daného typu procesní stanice není podporována požadovaná kombinace režimu a volby aktivní hrany. Viz dodatek "Čítačové vstupy /vstupy pro inkrementální čidla polohy".  Tento parametr slouží jako pomůcka pro ladění aplikace. Pokud po odladění aplikace nechceme tento příznak používat, smí se za jméno databázové proměnné uvést "NONE".
--------	-----	---	--

#### Příklad

```

ProcInit:
                IRCMoDe    0, IRC F1-F2, Obě hrany, 0.0, 0.025,
                NONE.0

ProcITR3:
                IRCSet     0, KoncSpin

Proc00:
                IRCIn      0, Hodnota
  
```

V procesu **INIT** je nastaven režim vstupu číslo 0 pro inkrementální čidlo polohy. Jedná se o inkrementální čidlo s výstupem ve formě dvou fázově posunutých signálů. Rozteč měřicích proužků je 0.1mm, z toho vyplývá krok 0.025mm (při posunu celého čidla o jeden proužek vzniknou celkem 4 přírůstky čítače). Hodnoty proměnných **KoncSpin** a **Hodnota** se udávají v milimetrech. Hodnota bitu **Úspěch** se nikam neukládá, předpokládá se, že zvolená kombinace čísla čítače, režimu a volby aktivních hran je na daném typu procesní stanice použitelná.

V procesu **ITR3** se stav čítače nastavuje na hodnotu odpovídající poloze koncového spínače, konkrétní poloha v milimetrech je uložena v databázové proměnné **KoncSpin**. Předpokládá se, že k digitálnímu vstupu, odpovídajícímu interrupt procesu **ITR3**, je připojen signál od koncového spínače či jiného kalibračního čidla polohy.

V procesu **0** se získává okamžitá hodnota čidla polohy v milimetrech a ukládá se do databázové proměnné **Hodnota**.

<b>IRCPreset</b>	Nastavení přednastavené hodnoty (poziční značky) hardwareového čítačového vstupu / vstupu pro inkrementální čidla polohy
------------------	--

### Popis

Modul určuje hodnotu hardwareového čítačového vstupu resp. vstupu pro inkrementální čidla polohy, na kterou se tento vstup nastaví při aktivaci jednoho konkrétního vstupu pro poziční značku v režimu automatické obsluhy pozičních značek. V režimu poloautomatické obsluhy pozičních značek použijte modul **IRCSet**. Blíže viz oddíl “*Obsluha pozičních značek*” v popisu modulu **IRCMode**.

Modul zároveň zajišťuje přepočet hodnoty fyzikální veličiny na odpovídající počet impulsů (kvant). Pro přepočet fyzikální hodnoty na hodnotu interního čítače se použijí konstanty, které byly pro daný vstup nastaveny předchozím voláním funkčního modulu **IRCMode**.

Modul se umísťuje do procesu ProcInit (za/pod modul **IRCMode**), případně i do vhodného periodického procesu (pokud se poloha odpovídající poziční značce za chodu aplikace mění).

### Parametry

Čítač	IN	Konst	Číslo čítačového vstupu / vstupu pro inkrementální čidla. Číslování vstupů viz popis čítačových vstupů / vstupů pro inkrementální čidla polohy příslušného typu procesní stanice v dodatku “Čítačové vstupy / vstupy pro inkrementální čidla polohy”
		I	
		MI	

Značka	IN	Konst	Číslo vstupu pro poziční značku v rámci příslušného čítačového vstupu / vstupu pro inkrementální čidla. Číslování pozičních značek viz dodatek “Čítačové vstupy / vstupy pro inkrementální čidla polohy”
		I	
		MI	

Hodnota	IN	Konst	Hodnota fyzikální veličiny, na kterou se má vstup nastavit při aktivaci příslušného vstupu pro poziční značku.
		F	
		MF	

### Příklad

```
ProcInit:
    IRCMode      0, IRC F1-F2, Obě hrany, 0.0, 0.025, NONE.0
    IRCPreset    0, 0, KoncSpin
```

V procesu **INIT** je nastaven režim vstupu číslo 0 pro inkrementální čidlo polohy. Jedná se o inkrementální čidlo s výstupem ve formě dvou fázově posunutých signálů. Rozteč měřicích proužků je 0.1mm, z toho vyplývá krok 0.025mm (při posunu celého čidla o jeden proužek vzniknou celkem 4 přírůstky čítače).

Hodnota proměnné **KoncSpin** (v milimetrech) se nastaví jako aktuální hodnota vstupu, pokud dojde k aktivaci poziční značky číslo 0.

<b>IRCSet</b>	Nastavení hodnoty hardwareového čítačového vstupu / vstupu pro inkrementální čidla polohy
---------------	---

### Popis

Modul zajišťuje nastavení hodnoty hardwareového čítačového vstupu resp. vstupu pro inkrementální čidla polohy. Zároveň zajišťuje přepočítání hodnoty fyzikální veličiny na odpovídající počet impulsů (kvant).

Modul se používá hlavně v rámci zpracování signálu indexové značky v režimu poloautomatické obsluhy pozičních značek. Blíže viz oddíl “*Obsluha pozičních značek*” v popisu modulu **IRCMode**.

Pro přepočítání fyzikální hodnoty na hodnotu interního čítače se použijí konstanty, které byly pro daný vstup nastaveny předchozím voláním funkčního modulu **IRCMode**.

Modul se umísťuje do procesu, ve kterém se vyhodnocuje absolutní poloha, nejčastěji do interrupt procesu, příslušného k digitálnímu vstupu, na nějž je přiveden signál koncového spínače nebo jiného zdroje údaje o absolutní poloze.

*Pozn.: Při umísťování tohoto modulu do interrupt procesu, který musí být typu “Logický automat” nebo “Reléové schéma”, je nutno modul **IRCSet** vložit jako tzv. Systémový modul, tedy pomocí klávesové kombinace Ctrl-S.*

### Parametry

Čítač	IN	Konst	Číslo čítačového vstupu / vstupu pro inkrementální čidla. Číslování vstupů viz popis čítačových vstupů / vstupů pro inkrementální čidla polohy příslušného typu procesní stanice v dodatku “Čítačové vstupy / vstupy pro inkrementální čidla polohy”
		I	
		MI	

Hodnota	IN	Konst	Hodnota fyzikální veličiny, na kterou se má vstup nastavit.
		F	
		MF	

### Příklad

```
ProcInit:
    IRCMode    0, IRC F1-F2, Obě hrany, 0.0, 0.025,
    NONE.0

ProcITR3:
    IRCSet     0, KoncSpin

Proc00:
    IRCIn      0, Hodnota
```

V procesu **INIT** je nastaven režim vstupu číslo 0 pro inkrementální čidlo polohy. Jedná se o inkrementální čidlo s výstupem ve formě dvou fázově posunutých signálů. Rozteč měřicích proužků je 0.1mm, z toho vyplývá krok 0.025mm (při posunu celého čidla o jeden proužek vzniknou celkem 4 přírůstky čítače). Hodnoty proměnných **KoncSpin** a **Hodnota** se udávají v milimetrech. Hodnota bitu **Úspěch** se nikam neukládá, předpokládá se, že zvolená kombinace čísla čítače, režimu a volby aktivních hran je na daném typu procesní stanice použitelná.

V procesu **ITR3** se stav čítače nastavuje na hodnotu odpovídající poloze koncového spínače, konkrétní poloha v milimetrech je uložena v databázové proměnné **KoncSpin**. Předpokládá se, že k digitálnímu vstupu, odpovídajícímu interrupt procesu **ITR3**, je připojen signál od koncového spínače či jiného kalibračního čidla polohy.

V procesu 0 se získává okamžitá hodnota čidla polohy v milimetrech a ukládá se do databázové proměnné **Hodnota**.

<b>LCD1100</b>	Interpretr dat z LCDSHELLu pro terminál APT1100 připojený po lince RS232
----------------	--

**Popis**

Modul interpretuje data z LCDSHELLu a realizuje žádané zobrazovací a vstupní funkce. V parametru `Linka` se volí příslušný komunikační port. Typicky se volí linka 0 (linka 1 je RS485).

Je-li v systému ADiS zapojen HW modul seriové komunikace AD-UART, lze jej použít pro připojení LCDSHELLu. Číslo komunikačního portu je pak dáno V/V konfigurací modulu AD-UART.

Parametrem `Parita` se volí parita přenosu, `Rychlost` přenosová rychlost. Typická rychlost je 9600Bd. Parametr `StopBit` určuje počet stop bitů.

Modul se umísťuje do procesu `Idle`.

**Parametry**

<b>Linka</b>	PAR	Konst	Použitá linka pro přenos (typicky 0).
<b>Parita</b>	PAR	Konst	Parita pro přenos (0 = bez parity, 1 = sudá parita).
<b>Rychlost</b>	PAR	Konst	Přenosová rychlost v [Bd] (doporučená hodnota je 9600Bd).
<b>StopBit</b>	PAR	Konst	Počet stop bitů při přenosu (1 nebo 2).

**LCD1100\_4** Interpret dat z LCDSHELLu pro terminál APT1100 připojený po lince RS485**Popis**

Modul interpretuje data z LCDSHELLu a realizuje žádané zobrazovací a vstupní funkce. V parametru `Linka` se volí příslušný komunikační port. Typicky se volí linka 1 (linka 0 je RS232).

Je-li v systému ADiS zapojen HW modul seriové komunikace AD-UART s externím převodníkem RS232 na RS485 anebo modul AD-UART4, lze je použít pro připojení LCDSHELLu. Číslo komunikačního portu je pak dáno V/V konfigurací daného HW modulu.

Parametrem `Parita` se volí parita přenosu, `Rychlost` přenosová rychlost. Typická rychlost je 9600Bd. Parametr `StopBit` určuje počet stop bitů.

Modul se umísťuje do procesu `Idle`.

**Parametry**

<b>Linka</b>	PAR	Konst	Použitá linka pro přenos (typicky 1).
<b>Parita</b>	PAR	Konst	Parita pro přenos (0 = bez parity, 1 = sudá parita).
<b>Rychlost</b>	PAR	Konst	Přenosová rychlost v [Bd] (doporučená hodnota je 9600Bd).
<b>StopBit</b>	PAR	Konst	Počet stop bitů při přenosu (1 nebo 2).

<b>LCD200</b>	Interpretr dat z LCDSHELLu pro terminál APT200 připojený po lince RS232
---------------	---

**Popis**

Modul interpretuje data z LCDSHELLu a realizuje žádané zobrazovací a vstupní funkce. V parametru `Linka` se volí příslušný komunikační port. Typicky se volí linka 0 (linka 1 je RS485).

Je-li v systému ADiS zapojen HW modul seriové komunikace AD-UART, lze jej použít pro připojení LCDSHELLu. Číslo komunikačního portu je pak dáno V/V konfigurací modulu AD-UART.

Parametrem `Parita` se volí parita přenosu, `Rychlost` přenosová rychlost. Typická rychlost je 9600Bd. Parametr `StopBit` určuje počet stop bitů.

Modul se umísťuje do procesu `Idle`.

**Parametry**

<b>Linka</b>	PAR	Konst	Použitá linka pro přenos (typicky 0).
<b>Parita</b>	PAR	Konst	Parita pro přenos (0 = bez parity, 1 = sudá parita).
<b>Rychlost</b>	PAR	Konst	Přenosová rychlost v [Bd] (doporučená hodnota je 9600Bd).
<b>StopBit</b>	PAR	Konst	Počet stop bitů při přenosu (1 nebo 2).

<b>LCD200_4</b>	Interpretr dat z LCDSHELLu pro terminál APT200 připojený po lince RS485
-----------------	---

**Popis**

Modul interpretuje data z LCDSHELLu a realizuje žádané zobrazovací a vstupní funkce. V parametru `Linka` se volí příslušný komunikační port. Typicky se volí linka 1 (linka 0 je RS232).

Je-li v systému ADiS zapojen HW modul seriové komunikace AD-UART s externím převodníkem RS232 na RS485 anebo modul AD-UART4, lze je použít pro připojení LCDSHELLu. Číslo komunikačního portu je pak dáno V/V konfigurací daného HW modulu.

Parametrem `Parita` se volí parita přenosu, `Rychlost` přenosová rychlost. Typická rychlost je 9600Bd. Parametr `StopBit` určuje počet stop bitů.

Modul se umísťuje do procesu `Idle`.

**Parametry**

<b>Linka</b>	PAR	Konst	Použitá linka pro přenos (typicky 1).
<b>Parita</b>	PAR	Konst	Parita pro přenos (0 = bez parity, 1 = sudá parita).
<b>Rychlost</b>	PAR	Konst	Přenosová rychlost v [Bd] (doporučená hodnota je 9600Bd).
<b>StopBit</b>	PAR	Konst	Počet stop bitů při přenosu (1 nebo 2).



<b>LCD2100</b>	Interpretr dat z LCDSHELLu pro grafický terminál APT2100
----------------	--

**Popis**

Modul interpretuje data z LCDSHELLu v rámci řídicího terminálu APT2100 a realizuje žádané zobrazovací a vstupní funkce.

Modul se umísťuje do procesu `Idle`.

**Parametry**

Modulu se nepředávají žádné parametry.

<b>LCD485</b>	Interpretr dat z LCDSHELLu pro terminály řady APT připojené po RS485
---------------	--

**Popis**

Modul pro připojení terminálu řady APT po sériové lince RS485. Modul interpretuje data z LCDSHELLu a realizuje žádané zobrazovací a vstupní funkce. V parametru `Linka` se volí příslušný komunikační port. Typicky se volí linka 1 (linka 0 je RS232).

Je-li v systému ADiS zapojen HW modul seriové komunikace AD-UART s externím převodníkem RS232 na RS485 anebo modul AD-UART4, lze je použít pro připojení LCDSHELLu. Číslo komunikačního portu je pak dáno V/V konfigurací daného HW modulu.

Parametrem `Parita` se volí parita přenosu, `Rychlost` přenosová rychlost. Typická rychlost je 9600Bd. Parametr `StopBit` určuje počet stop bitů.

Modul se umísťuje do procesu `Idle`.

**Parametry**

<b>Linka</b>	PAR	Konst	Použitá linka pro přenos (typicky 1).
<b>Parita</b>	PAR	Konst	Parita pro přenos (0 = bez parity, 1 = sudá parita).
<b>Rychlost</b>	PAR	Konst	Přenosová rychlost v [Bd] (doporučená hodnota je 9600Bd).
<b>StopBit</b>	PAR	Konst	Počet stop bitů při přenosu (1 nebo 2).

**LCDG485**

Interpretr dat z LCDSHELLu pro grafický terminál APT2000 připojený po lince RS485

**Popis**

Modul pro připojení APT2000 po seriové lince RS485. Modul interpretuje data z LCDSHELLu a realizuje žádané zobrazovací a vstupní funkce. V parametru `Linka` se volí příslušný komunikační port. Typicky se volí linka 1 (linka 0 je RS232).

Je-li v systému ADiS zapojen HW modul seriové komunikace AD-UART s externím převodníkem RS232 na RS485 anebo modul AD-UART4, lze je použít pro připojení LCDSHELLu. Číslo komunikačního portu je pak dáno V/V konfigurací daného HW modulu.

Parametrem `Parita` se volí parita přenosu, `Rychlost` přenosová rychlost. Typická rychlost je 9600Bd. Parametr `StopBit` určuje počet stop bitů.

Modul se umísťuje do procesu `Idle`.

**Parametry**

<b>Linka</b>	PAR	Konst	Použitá linka pro přenos (typicky 1).
<b>Parita</b>	PAR	Konst	Parita pro přenos (0 = bez parity, 1 = sudá parita).
<b>Rychlost</b>	PAR	Konst	Přenosová rychlost v [Bd] (doporučená hodnota je 9600Bd).
<b>StopBit</b>	PAR	Konst	Počet stop bitů při přenosu (1 nebo 2).

<b>LCDADIR</b>	Interpretr dat z LCDSHELLu pro paralelní terminál 2x8 znaků s mini-klávesnicí (ADIR)
----------------	--

**Popis**

Modul interpretuje data z LCDSHELLu a realizuje žádané zobrazovací a vstupní funkce. Umísťuje se do procesu `Idle`.

Obsluha terminálu s miniklávesnicí se poněkud liší od obsluhy terminálů se standardními klávesnicemi. Odlišnosti jsou popsány v poznámkách u jednotlivých kapitol části "*Ovládání terminálů řady APT*".

Klávesnice systému ADIR je oproti ostatním miniklávesnicím ochuzena o klávesy *plus* a *minus*. Tento fakt společně s miniaturním displejem způsobuje určitá omezení zobrazení a editace. V systémovém menu jsou z toho důvodu zcela vypuštěny položky "Databáze" a "Provozní deník".

**Parametry**

Modul nemá žádné parametry.

<b>LCDART4KM</b>	Interpretr dat z LCDSHELLu pro paralelní terminál 4x20 znaků s mini-klávesnicí (ART4000M).
------------------	--

**Popis**

Modul interpretuje data z LCDSHELLu a realizuje žádané zobrazovací a vstupní funkce. Modul se umísťuje do procesu `idle`.

Obsluha terminálu s miniklávesnicí se poněkud liší od obsluhy terminálů se standardními klávesnicemi. Odlišnosti jsou popsány v poznámkách u jednotlivých kapitol části "*Ovládání terminálů řady APT*".

**Parametry**

Modul nemá žádné parametry.

<b>LCDDC300</b>	Interpretr dat z LCDSHELLu pro paralelní terminál 2x16 znaků procesní stanice DC300.
-----------------	--

**Popis**

Modul interpretuje data z LCDSHELLu a realizuje žádané zobrazovací a vstupní funkce.  
Modul se umísťuje do procesu `idle`.

**Parametry**

Modul nemá žádné parametry.

*Pozn.: Jedná se o nestandardní procesní stanici, která není v nabídce firmy AMiT spol. s r.o.*

**LCDGTTY**

Interpretr dat z LCDSHELLu pro grafický terminál APT2000 připojený po lince RS232

**Popis**

Modul pro připojení APT2000 po seriové lince RS232. Modul interpretuje data z LCDSHELLu a realizuje žádané zobrazovací a vstupní funkce. V parametru `Linka` se volí příslušný komunikační port. Typicky se volí linka 0 (linka 1 je RS485).

Je-li v systému ADiS zapojen HW modul seriové komunikace AD-UART, lze jej použít pro připojení LCDSHELLu. Číslo komunikačního portu je pak dáno V/V konfigurací modulu AD-UART.

Parametrem `Parita` se volí parita přenosu, `Rychlost` přenosová rychlost. Typická rychlost je 9600Bd. Parametr `StopBit` určuje počet stop bitů.

Modul se umísťuje do procesu `Idle`.

**Parametry**

<b>Linka</b>	PAR	Konst	Použitá linka pro přenos (typicky 0).
<b>Parita</b>	PAR	Konst	Parita pro přenos (0 = bez parity, 1 = sudá parita).
<b>Rychlost</b>	PAR	Konst	Přenosová rychlost v [Bd] (doporučená hodnota je 9600Bd).
<b>StopBit</b>	PAR	Konst	Počet stop bitů při přenosu (1 nebo 2).

<b>LCDK12</b>	Interpretr dat z LCDSHELLu pro paralelní terminál LCDK12
---------------	--

**Popis**

Modul interpretuje data z LCDSHELLu a realizuje žádané zobrazovací a vstupní funkce.  
Umísťuje se do procesu `Idle`.

**Parametry**

Modul nemá žádné parametry.



<b>LCDK14</b>	Interpretr dat z LCDSHELLu pro paralelní terminál LCDK14
---------------	--

**Popis**

Modul interpretuje data z LCDSHELLu a realizuje žádané zobrazovací a vstupní funkce.  
Umísťuje se do procesu `Idle`.

**Parametry**

Modul nemá žádné parametry.

**LCDMest**

Interpretr dat z LCDSHELLu pro paralelní terminál (8x21 znaků max.)  
stanice Mesterm.

**Popis**

Modul interpretuje data z LCDSHELLu a realizuje žádané zobrazovací a vstupní funkce. Umísťuje se do procesu `Idle`.

Terminál má nastavitelnou velikost zobrazovaných znaků, čemuž odpovídá i počet zobrazitelných řádků a sloupců na displeji. Velikost znaků se nastavuje pomocí LCDShell prvku `CharSize`.

Možné nastavení displeje:

Velikost znaku (šířka x výška)	Rozlišení displeje (řádky x sloupce)
6x8	8x21
8x10	6x16
10x16	4x12
16x30	2x8

Implicitní velikost znaku je 6x8 bodů, tj. rozlišení 8x21 znaků na displeji. V systémovém menu LCDShellu se používá vždy toto implicitní nastavení.

**Parametry**

Modul nemá žádné parametry.

<b>LCDMini</b>	Interpretr dat z LCDSHELLu pro paralelní terminál 2x16 znaků s mini-klávesnicí (ART267)
----------------	---

**Popis**

Modul interpretuje data z LCDSHELLu a realizuje žádané zobrazovací a vstupní funkce. Umísťuje se do procesu `Idle`.

Obsluha terminálu s miniklávesnicí se poněkud liší od obsluhy terminálů se standardními klávesnicemi. Odlišnosti jsou popsány v poznámkách u jednotlivých kapitol části "*Ovládání terminálů řady APT*".

**Parametry**

Modul nemá žádné parametry.

<b>LCD4x20</b>	Interpretr dat z LCDSHELLu pro paralelní terminál 4x20 znaků
----------------	--

Používá se pro systém ART400 nebo paralelní terminál APT130.

**Popis**

Modul interpretuje data z LCDSHELLu a realizuje žádané zobrazovací a vstupní funkce. Umísťuje se do procesu `Idle`.

**Parametry**

Modul nemá žádné parametry.

<b>LCDRfsh</b>	Občerstvování LCD displeje
----------------	----------------------------

**Popis**

Tento modul se používá v případech, kdy je terminál LCD připojen seriovou linkou a nachází se v prostředí se silnějším elektromagnetickým rušením. Modul periodicky obnovuje obsah displeje a tím občerstvuje nápisy na terminálu, které mohou být vlivem rušení poškozené.

Modul se umísťuje do periodického procesu s periodou 0.5 nebo 1s.

**Použití**

Používá se pouze v případech, kdy dochází k chybám zobrazení na displeji vlivem rušení, jinak je použití zbytečné.

**Parametry**

Modul nemá žádné parametry.

**LCDTTY**

Interpretr dat z LCDSHELL pro terminály řady APT připojené po RS232

**Popis**

Modul pro připojení terminálů řady APT po seriové lince RS232. Modul interpretuje data z LCDSHELLu a realizuje žádané zobrazovací a vstupní funkce. V parametru `Linka` se volí příslušný komunikační port. Typicky se volí linka 0 (linka 1 je RS485).

Je-li v systému ADiS zapojen HW modul seriové komunikace AD-UART, lze jej použít pro připojení LCDSHELLu. Číslo komunikačního portu je pak dáno V/V konfigurací modulu AD-UART.

Parametrem `Parita` se volí parita přenosu, `Rychlost` přenosová rychlost. Typická rychlost je 9600Bd. Parametr `StopBit` určuje počet stop bitů.

Modul se umísťuje do procesu `Idle`.

**Parametry**

<b>Linka</b>	PAR	Konst	Použitá linka pro přenos (typicky 0).
<b>Parita</b>	PAR	Konst	Parita pro přenos (0 = bez parity, 1 = sudá parita).
<b>Rychlost</b>	PAR	Konst	Přenosová rychlost v [Bd] (doporučená hodnota je 9600Bd).
<b>StopBit</b>	PAR	Konst	Počet stop bitů při přenosu (1 nebo 2).

**Popis**

Příkaz **Let** realizuje přiřazovací příkaz typu *proměnná = výraz*, který lze použít k veškerým výpočtům, logickým a srovnávacím operacím prováděným procesní stanicí. Tento příkaz je nejčastěji používaný příkaz pseudojazyka, neboť nahrazuje velké množství specializovaných aritmetických a logických funkčních modulů ze systémů jiných výrobců. Zápis výrazu je blízký obvyklému technickému chápání.

**Tvar příkazu**

Přiřazovací příkaz má formát *proměnná = výraz*. Levá strana specifikuje jednoduchou proměnnou nebo prvek matice, kterému se přiřadí hodnota výrazu z pravé strany.

**Odkaz na proměnnou**

Odkazujeme-li se na prvek matice, uvádí se oba indexy v hranatých závorkách, tedy *[řádek, sloupec]*. Je-li třeba pracovat s jednotlivým bitem proměnné typu `DBT_INT`, uvádí se číslo bitu za jménem (a indexy, jsou-li použity) za tečkou. Rozsah čísla bitu je 0 až 15. Je-li třeba pracovat s jednotlivým bitem proměnné typu `DBT_LONG`, uvádí se číslo bitu za jménem (a indexy, jsou-li použity) za tečkou. Rozsah čísla bitu je 0 až 31. Další možnosti odkazu na bit je odkaz pomocí alias jména. Uvedme několik příkladů:

- ♦ jednoduchá proměnná libovolného typu  
`Promenna`
- ♦ prvek matice libovolného typu  
`Matice[3,2]`
- ♦ alias (bit jednoduché proměnné)  
`@Bit`
- ♦ bit jednoduché proměnné typu `L`  
`Vlajky.12`
- ♦ bit prvku matice typu `ML`  
`Matice[2,35].30`
- ♦ prvek matice, sloupcovým indexem je hodnota proměnné `Y_INDEX`  
`Matice[1,Y_INDEX]`

**Výraz, operandy**

Pravá strana přiřazovacího příkazu je vyhodnocovaný výraz. Výraz má v zásadě obvyklou strukturu typu *operand operátor operand*, která se podle potřeby opakuje. Uvedme opět několik příkladů:

<code>13.6</code>	reálné číslo
<code>15</code>	celé číslo
<code>0x1234</code>	šestnáctkové celé číslo
<code>0o12345</code>	osmičkové celé číslo
<code>0b1011101000101</code>	dvojkové celé číslo
<code>Var + 13.6</code>	hodnota proměnné zvětšená o číslo
<code>Var1 / Var2</code>	podíl dvou proměnných
<code>Matice[1, Index]</code>	prvek maticové proměnné, řádek 1, sloupec daný obsahem jednoduché proměnné
<code>Var1+Var2*(Var3 - Var4[1, Index]/Var5)</code>	ukázka složitějšího výrazu

Poslední příklad je složitější aritmetický výraz, ze kterého je patrná další vlastnost, totiž že pořadí vyhodnocování výrazu lze upravit použitím závorek stejným způsobem jak jsme zvyklí z matematiky.

## Operátory

Priorita	Operátor	Význam	Příklad	Výsledek
4 (nejvyšší)	not nebo !	logická negace	not 1	0
	~	bitová negace	~0b10010	0b01101
3	*	násobení	4 * 3	12
	/	dělení	4 / 3	1,33
	div	celočíslné dělení	4 div 3	1
	mod	zbytek po dělení	4 mod 3	1
	and	logický součin	1 and 1	1
	&	bitový log. součin	0b0011 and 0b0101	0b0001
	<<	bitový posun doleva	0b0110 << 1	0b1100
	>>	bitový posun doprava	0b0110 >> 1	0b0011
2	+	součet	4 + 3	7
	-	rozdíl	4 - 3	1
	or	logický součet	1 or 0	1
		bitový log. součet	0b0011   0b0101	0b0111
	xor	log.výhradní součet	1 xor 1	0
	^	bit.výhradní součet	0b0011 ^ 0b0101	0b0110
1 (nejnižší)	>	větší	4 > 3	1
	>=	větší nebo rovno	4 >= 3	1
	==	rovno	4 == 3	0
	!=	nerovno	4 != 3	1
	<=	menší nebo rovno	4 <= 3	0
	<	menší	4 < 3	0

Operátory příkazu **Let** mohou být aritmetické, logické a relační. *Aritmetické operátory* zajišťují operace s čísly a hodnotami proměnných, tedy s analogovými údaji (např. +, -, \*, / apod.). *Logické operátory* zajišťují výpočty s logickými, tedy digitálními údaji (např. and, or, not apod.). *Relační operátory* zajišťují porovnávání čísel a proměnných. Výsledkem porovnání je logická hodnota (např. <, >= apod.). Pořadí vyhodnocování jednotlivých operátorů (tzv. *precedence*) je uvedeno v tabulce.

## Funkce

Kromě čísel a proměnných lze ve výrazech využívat i funkce. Je definováno pět standardních funkcí pro nejčastěji používané operace:

*Sqrt( výraz )*

Funkce vrací hodnotu druhé odmocniny výrazu.

*Log10( výraz )*

Funkce vrací hodnotu desítkového logaritmu výrazu.

*Log( výraz )*

Funkce vrací hodnotu přirozeného logaritmu výrazu.

*Sin( výraz )*

Funkce vrací hodnotu sinus výrazu. Výraz je v radiánech.

*Cos( výraz )*

Funkce vrací hodnotu kosinus výrazu. Výraz je v radiánech.

*Tan( výraz )*

Funkce vrací hodnotu tangens výrazu. Výraz je v radiánech.

*Abs( výraz )*

Funkce vrací absolutní hodnotu výrazu.

*Exp( výraz )*

Funkce vrací hodnotu exponenciální funkce  $e^x$ , kde  $x$  je hodnota výrazu.



*Pow( výraz1, výraz2 )*

Funkce vrací hodnotu mocniny  $X^Y$ , kde  $X$  je hodnota výrazu 1 a  $Y$  je hodnota výrazu 2.

*Avg( výraz1, výraz2 [, výraz3 [, výraz4 ... ]])*

Funkce vrací aritmetický průměr hodnot všech výrazů, které jsou argumenty funkce. Funkce může mít proměnný počet argumentů, minimální počet jsou dva.

*Min( výraz1, výraz2 [, výraz3 [, výraz4 ... ]])*

Funkce vrací minimální hodnotu z hodnot všech výrazů, které jsou argumenty funkce. Funkce může mít proměnný počet argumentů, minimální počet jsou dva.

*Max( výraz1, výraz2 [, výraz3 [, výraz4 ... ]])*

Funkce vrací maximální hodnotu z hodnot všech výrazů, které jsou argumenty funkce. Funkce může mít proměnný počet argumentů, minimální počet jsou dva.

*If( výraz1, výraz2, výraz3 )*

Funkce vyhodnotí *výraz1*. Je-li jeho hodnota nenulová, vrací funkce hodnotu výrazu *výraz2*, je-li nulová, vrací hodnotu výrazu *výraz3*.

#### Příklad

Uvedme opět několik příkladů použití funkcí:

```
Let Var1 = Sqrt( Var2 * Var3 + 16.5 )
Let Var1 = Pow( Var2 + Var3, Var4 )
Let Prumer = Avg( Matice[0,0],Matice[0,1], Matice[0,2])
Let Minimum = Min( Test, 12 )
Let Maximum = Max( Test, 13 )
Let Test = If( Var1, 13.8, Var2 + 11 )
```

Použití funkce **If** (viz poslední výraz) umožňuje značné zjednodušení zápisu výrazů, jejichž hodnota závisí na vyhodnocení nějaké podmínky. Stejně přiřazení s využitím příkazu **If-Else-EndIf** by zabralo v tomto případě volání pěti funkčních modulů a bylo by rozhodně méně přehledné.

#### Konverze typů

V tabulce precedence operátorů byly uvedeny dva příklady:  $4/3=1.3333$  a  $4 \text{ div } 3=1$ . V prvním případě je podílem dvou celých čísel číslo reálné, zatímco ve druhém případě číslo celé. Obdobná situace může nastat i v mnoha dalších situacích. Naskytá se tedy otázka, jak tyto situace zpracuje příkaz **Let**.

Zavedme nyní pojem *velikost typu* - jeden typ je větší než druhý, pokud je schopen pokrýt všechny hodnoty druhého typu a ještě nějaké další navíc. Na procesní stanici se setkáváme se čtyřmi typy hodnot. Jsou to logická hodnota (nazvěme ji typ *Bool*), krátké celé číslo (nazvěme jej typ *Int*), dlouhé celé číslo (nazvěme jej typ *Long*) a konečně reálné číslo (nazvěme jej typ *Float*). Z uvedených rozsahů plyne zřejmý výsledek srovnání velikosti typů:  $Bool < Int < Long < Float$ .

Pro vyhodnocování výrazů příkazem **Let** platí toto pravidlo: typ každé dílčí operace je určen buď operátorem (v případě operátoru s pevným typem) nebo nejvyšším typem použitých operandů (v případě operátoru s volným typem). V následující tabulce je uveden seznam operátorů rozdělený na pevné a volné typy.

	Operátor	Význam	Přípustný typ operandů	Výsledný typ
Pevný výsledný typ	/	dělení	Int, Long, Float	Float
	not nebo !	logická negace	Bool	Bool
	and	logický součin	Bool	Bool
	or	logický součet	Bool	Bool
	xor	log.výhradní součet	Bool	Bool
	>	větší	Int, Long, Float	Bool
	>=	větší nebo rovno	Int, Long, Float	Bool
	==	rovno	Int, Long, Float	Bool
	!=	nerovno	Int, Long, Float	Bool
	<=	menší nebo rovno	Int, Long, Float	Bool
	<	menší	Int, Long, Float	Bool
Volný výsledný typ	*	násobení	Int, Long, Float	*)
	+	součet	Int, Long, Float	*)
	-	rozdíl	Int, Long, Float	*)
	div	celočíslné dělení	Int, Long	*)
	mod	zbytek po dělení	Int, Long	*)
	~	bitová negace	Int, Long	*)
	&	bitový log. součin	Int, Long	*)
		bitový log. součet	Int, Long	*)
	^	bit.výhradní součet	Int, Long	*)
	<<	bitový posun doleva	Int, Long	*)
	>>	bitový posun doprava	Int, Long	*)

\*) Výsledný typ je dán nejvyšším typem použitých operandů

Proto je tedy výsledek operace  $4 / 3$  typu *Float*, protože jsou sice oba operandy *Int*, ale použitý operátor dělení vyžaduje pevný typ výsledku *Float*. Výsledek operace  $3 + 13.6$  je také typu *Float*. V tomto případě to není díky operátoru, ale díky tomu, že jeden z operandů je *Float*.

Poznámka:

U operandů s volným typem je třeba pamatovat na to, že výsledek je dán nejvyšším typem použitých operandů. Např. výraz  $20000 * 2$  se vyhodnotí špatně jako  $-25536$ , protože výsledný typ je *Int* a číslo  $40000$  se do tohoto typu "nevejde". Pro správné vyhodnocení je potřeba výraz zadat ve tvaru  $20000.0 * 2$ , který se vyhodnotí v typu *Float* a výsledek dopadne dobře.

Je-li konečně celý výraz vyhodnocen, je typ výsledku převeden na očekávaný typ. Je-li např. proměnná *Var* typu *I*, je jí výrazem `Let Var = 13 + 92 / 10` přiřazena hodnota  $22$ . Výsledek výrazu je totiž  $22.2$  (*Float*) a je konvertován před přiřazením na typ levé strany, tedy *Int*.

Veškeré uvedené konverze typů probíhají zcela automaticky a není třeba se o ně žádným způsobem starat. Popsaný mechanismus sice vypadá složitě, je však jednoduchý na používání a je velmi dobře prověřen. Může se však přesto stát, že z nějakého důvodu je nutné změnit typ výrazu jiným způsobem, než by to automaticky zajistil příkaz **Let**. V takovém případě jsou k dispozici *funkce pro přetypování*, které převádějí hodnotu výrazu jakéhokoli typu na hodnotu požadovaného typu:

*Bool*( výraz )

*Int*( výraz )

*Long*( výraz )

*Float*( výraz )

Příklad

Uveďme několik příkladů:

Výraz	Výsledek
Bool( 13.8 )	1
Int( 13.8 )	13
Long( 13.8 )	13
Float( 1 )	1

### V/V operace

Kromě proměnných umí příkaz **Let** pracovat i s V/V signály. Je-li V/V signál použit na pravé straně přiřazovacího příkazu, jedná se o vstupní operaci, je-li na levé straně, jedná se o výstupní operaci. Uvedme příklady, z nichž je patrná i syntaxe:

```
Let Vstup = #D:0
```

Příkaz je ekvivalentní volání modulu: DigIn #0, Vstup, 0x0000

```
Let Vstup = #D:0.1
```

Příkaz nemá přímý ekvivalent ve funkčních modulech - operaci je třeba realizovat např. takto:

```
DigIn #0, Vstup, 0x000
```

```
Let Vstup.0 = Vstup.1
```

Obdobně lze číst z analogových vstupních kanálů a zapisovat na digitální výstupní kanály. Používání V/V operací v příkazu **Let** však obecně nelze doporučit, i když se může na první pohled zdát jednodušší, protože vede obvykle k méně srozumitelnému zápisu a nelze takových zápisů použít pro analýzu datových toků.

### Závěrečné poznámky

Mnoha čtenářům se uvedený výklad o příkazu **Let** může zdát velmi složitý a proto nesrozumitelný. Není však třeba zoufat - uvedli jsme zde stručný výčet všech možností uvedeného příkazu. Běžný programátor (a to i velmi zkušený!) však ve své praxi vystačí s velmi jednoduchými konstrukcemi a zdaleka nevyužije všech možností. Nejprve proto využijte skutečné základy - sčítání, odčítání, násobení a dělení, jednoduché logické operace a závorky. S jejich pomocí realizujete prakticky jakoukoli aritmetickou a logickou funkci potřebnou pro řízení (viz např. vzorový příklad použití programu). Teprve když tento aparát nestačí, pokuste se proniknout o krůček dále do možností příkazu **Let**.

### Syntaktický diagram

Syntaktický diagram příkazu **Let** je zapsán v běžně používané formě BNF. Terminální symboly jsou označeny tučně, neterminální symboly jsou označeny kurzívou, nepovinné výrazy jsou ohraničeny hranatými závorkami, jednotlivé varianty rozvoje symbolu jsou vypsány na oddělených řádcích.

```

příkaz_let      :
                   let proměnná = výraz
proměnná      :
                   TID [ [ výraz, výraz ] ] [ . výraz ]
                   alias
                   signál
                   #A:LongInteger
                   #D:LongInteger [ . výraz ]
TID           :
                   jméno
alias         :
                   @jméno
signál       :
                   #jméno
jméno        :
                   znak
                   jméno znak
                   jméno číslice

```

znak:

jedna z následujících konstrukcí  
**\_ a b c d e f g h i j k l m n o p q r s t u v w x y z**  
**A B C D E F G H I J K L M N O P Q R S T U V W X Y Z**

cislice:

**0 1 2 3 4 5 6 7 8 9**

výraz :

*hodnota*  
*unární\_operátor* výraz  
*výraz* *operátor* *výraz*  
 ( *výraz* )  
 ( *výraz* ). *výraz*  
 if ( *výraz*, *výraz*, *výraz* )  
 funkce ( *výraz* )  
 funkce ( *seznam\_výrazů* )

*unární\_operátor*:

jedna z následujících konstrukcí  
**+ - ! not ~**

*operátor*:

jedna z následujících konstrukcí  
**= > | < | == | <= | >= | != | + | - | | | ^ | or**  
**| xor | \* | / | & | div | mod | and | << | >>**

*funkce* :

jedna z následujících konstrukcí  
**avg min max sqrt int long float bool**

*seznam\_výrazů*:

*seznam\_výrazů*, *výraz*  
*výraz*

*hodnota*:

*LongInteger*  
*Float*  
*proměnná*

*LongInteger*:

obvyklý zápis dekadického čísla  
 zápis v jazyce C (vedoucí kombinace 0x, 0X,  
 např. 0x0010, 0xFFFF, 0x80000000)

*Float*:

obvyklý zápis reálného čísla (např. 1.0, -6.1234,  
 -6.1234E-15, 12.0)

## Parametry

<b>Výraz</b>	PAR	Řetězec	Jediný parametr příkazu <b>Let</b> .
--------------	-----	---------	--------------------------------------

Syntaxe byla popsána výše.

## Příklad

Příklady byly uvedeny v popisu příkazu .

<b>Limiter</b>	Omezení proměnné na zadaných mezích
----------------	-------------------------------------

### Popis

Modul omezuje vstupní hodnotu na interval daný proměnnými `Dolní` a `Horní`. Pokud je hodnota proměnné mimo interval, bude proměnná nastavena na hodnotu bližší hranice. Modul provádí kontrolu správnosti zadání velikostí hranic. Pokud bude `Dolní` větší než `Horní`, nebude proměnná `Hodnota` měněna. Hranice `Dolní` a `Horní` jsou automaticky typově konvertovány podle typu vstupující proměnné `Hodnota`.

### Parametry

<b>Hodnota</b>	IN/OUT	I	Proměnná, která je modulem omezována.
		L	
		F	

<b>Dolní</b>	IN	Konst	Dolní hranice intervalu.
		F	
		MF	

<b>Horní</b>	IN	Konst	Horní hranice intervalu.
		F	
		MF	

### Příklad

```
Limiter          IVal, 0, 100.0
```

Omezení hodnoty `IVal` na interval 0 až 100.

**Limits**

Testování, zda je proměnná v daných mezích

**Popis**

Modul umožňuje testování hodnoty analogového údaje na dvojici заданých mezí. Při testování je možné zadat pásmo necitlivosti (hysterezi) v okolí meze, která potlačí zbytečné mnohonásobné změny výsledku testu v případě, že se testovaná hodnota pohybuje v těsném okolí meze.

**Parametry**

<b>Vstup</b>	IN	F	Vstupní proměnná.
		MF	
<b>VýstupNad</b>	OUT	Bit	Výsledek testu "nad horní mezí".
<b>VýstupPod</b>	OUT	Bit	Výsledek testu "pod dolní mezí".
<b>DolníMez</b>	IN	Konst	Hodnota dolní meze.
		F	
		MF	
<b>HorníMez</b>	IN	Konst	Hodnota horní meze.
		F	
		MF	
<b>Hystereze</b>	IN	Konst	Hodnota hystereze.
		F	
		MF	
<b>Negace</b>	PAR	Konst	ANO = negovat výsledek testu.

Do výstupních parametrů **VýstupNad** a **VýstupPod** lze zadat stejný bit. Modul pak detekuje stav "V mezích ano/ne", viz příklady d),e),f) v tabulce.

Příklady použití modulu:

<p>a) Dva výstupy, hystereze, Negace NE</p>	<p>b) Dva výstupy, hystereze, Negace ANO</p>	<p>c) Dva výstupy, bez hystereze, Negace NE</p>
<p>d) Společný výstup, hystereze, Negace NE</p>	<p>e) Společný výstup, hystereze, Negace ANO</p>	<p>f) Společný výstup, bez hystereze, Negace ANO</p>

**Příklad**

Limits      Vstup, Nad.0, Pod.0, 50.0, 150.0, 0.0, 0x0000

Modul porovnává hodnotu proměnné **Vstup** s mezemi 50 a 150. Porovnání se provádí bez hystereze. Bit **Nad.0** se nastaví do "1", je-li **Vstup** > 150. Bit **Pod.0** se nastaví do "1", je-li **Vstup** < 50.

<b>LogPrint</b>	Tisk provozního deníku na tiskárně
-----------------	------------------------------------

**Popis**

Modul tiskne průběžně hlášení provozního deníku, tak jak vznikají. Modul se zpravidla umísťuje do IDLE procesu. Spolupracuje s modulem **PrintMgr**, který obsluhuje tiskárnu. Pokud je tiskárna vypnutá, modul "drží" nevytisknutá hlášení. Při zapnutí (připojení) tiskárny se všechna nevytisknutá hlášení vytisknou naráz. Hloubka "pozdřezných" hlášení je dána velikostí bufferu provozního deníku (50 hlášení). Po resetu procesní stanice se vytisknou všechna hlášení v bufferu provozního deníku.

Modul umí implicitně vytisknout pouze systémová hlášení (např. *Výpadek napájení*). Pokud je třeba tisknout uživatelská hlášení (generovaná v procesu např. modulem **Report**), je nutné definovat formátovací texty pro jednotlivá uživatelská hlášení. Tyto texty se definují pomocí programu DTE.

**Parametry**

<b>pPrintMgr</b>	PAR	Návěští	Návěští spolupracujícího modulu <b>PrintMgr</b> .
------------------	-----	---------	---

<b>Index</b>	IN	I	Index provozního deníku. Jde o speciální proměnnou, která má <i>WID xx900</i> (xx je číslo stanice). Nemusí se uvádět ani nemusí být v databázi.
		NONE	

<b>Bufer</b>	IN	MI	Bufer provozního deníku. Jde o speciální proměnnou, která má <i>WID xx901</i> (xx je číslo stanice). Nemusí se uvádět ani nemusí být v databázi.
		NONE	

**Příklad**

ProcIdle:

```
:01000 PrintMgr 0x0F00, 2, 9600, 8, 0, 1
```

```
LogPrint :01000, NONE, NONE
```

Modul **LogPrint** spolupracuje s modulem **PrintMgr**, který má návěští :1000. Oba moduly jsou umístěny v IDLE procesu.

<b>MDImp</b>	Čtení a přepočítání šestnáctice impulzních vstupů
--------------	---

**Popis**

Modul čte údaj z impulzního vstupu a přepočte jej na fyzikální rozměr. Modul spolupracuje s modulem pro obsluhu 16 impulzních vstupů **MImpIn**. Činnost modulu je obdobná jako u modulu **DImp** ze standardní knihovny PSE. Rozdíl spočívá v tom, že modul **MDImp** umožňuje zároveň provést přepočítání celé šestnáctice vstupů (nebo její části) do jednotlivých řádků prvního sloupce databázové matice.

**Parametry**

<b>Impulz</b>	PAR	Návěští	Návěští použitého modulu <b>MImpIn</b> . Návěští je globální (modul <b>MImpIn</b> je zpravidla v jiném procesu než modul <b>MDImp</b> ) a je ho tedy třeba zadat ručně.
<b>Indexy</b>	IN	MI	Matice, v níž prvky v jednotlivých řádcích prvního sloupce udávají, do kterých řádků ostatních matic se mají ukládat hodnoty jednotlivých vstupů.
<b>Posun</b>	PAR	Konst	Číslo, které udává, kolikátý řádek matice <b>Indexy</b> odpovídá nultému vstupu z modulu <b>MImpIn</b> . Má-li matice <b>indexy</b> méně řádků než <b>Posun</b> + 16, nepoužije se celá šestnáctice vstupů, ale jen odpovídající část.
<b>Delt</b>	OUT	MI	Matice, do jejíhož nultého sloupce se do řádků určených maticí <b>Indexy</b> ukládají hodnoty přírůstku měřené veličiny od posledního volání modulu <b>MDImp</b> . Nechceme-li počítat přírůstky, lze parametr zadat <b>NONE</b> .
		ML	
		MF	
		NONE	
<b>Sumy</b>	OUT	MI	Matice, do jejíhož nultého sloupce se do řádků určených maticí <b>Indexy</b> ukládají hodnoty průběžných počítadel. Nechceme-li udržovat hodnoty počítadel, lze parametr zadat <b>NONE</b> .
		ML	
		MF	
		NONE	
<b>Okamžité</b>	OUT	MI	Matice, do jejíhož nultého sloupce se do řádků určených maticí <b>Indexy</b> ukládají odhady okamžitých hodnot. Nechceme-li počítat okamžité hodnoty, lze parametr zadat <b>NONE</b> .
		ML	
		MF	
		NONE	
<b>Konstanty</b>	IN	MI	Matice, obsahující konstanty měřidel, tedy N jednotek na impuls. Zadá-li se parametr <b>NONE</b> , modul dosadí za všechny konstanty hodnotu 1.
		ML	
		MF	
		NONE	
<b>SyncIn</b>	IN	Bit	Bit databázové proměnné, který se použije jako synchronizační, to znamená, že je-li v něm při vstupu do modulu jednička, změní modul jeho hodnotu na nulu a vynulují se vnitřní čítače.
<b>SyncOut</b>	OUT	Bit	Bit databázové proměnné, který se nastaví do jedničky při příchodu synchronizačního pulzu v parametru <b>SyncIn</b> . Je-li tedy po provedení modulu <b>MDImp</b> v tomto bitu jednička, obsahuje matice <b>Sumy</b> definitivní načítané hodnoty za poslední periodu synchronizace. Parametr lze zadat <b>NONE</b> .
		NONE	

Matice **Sumy** obsahuje hodnotu načítanou od předchozího volání modulu, ale od příštího volání se začne přičítat znovu od nuly.

**Poznámka**

Pro standardní případ, kdy chceme zpracovávat všech 16 vstupů a hodnoty chceme ukládat do matic na řádky odpovídající pořadí vstupů, budou mít matice **Indexy**, **Delt**,



Sumy, Okamžité a Konstanty rozměry [16,1]. Do matice Indexy zadáme na jednotlivých řádcích hodnoty 0, 1, 2, ..., 15.

Pokud chceme zpracovávat méně než 16 vstupů, musí se zadat matice s odpovídajícím počtem řádků.

Za parametr Delt, Sumy nebo Okamžité může být dosazeno NONE, v tom případě se příslušné matice negenerují.

#### Příklad

ProcQUICK:

```
:00001      MImpIn   #0, M_TO, 0, 0xFFFF, 0x0000, NONE, NONE
```

Proc00:

```
MDImp      :00001, M_Ind, 2, NONE, M_Sumy, NONE, M_K, SYNC.0, SYNC.1
```

Předpokládáme, že matice M\_Sumy a M\_K jsou matice typu například F, s rozměrem ŘxS 10x1. Matice M\_Ind typu I obsahuje hodnoty:

0
1
4
5

Vzhledem k parametru Posun rovnému 2 se uplatní silně orámovaná část matice, to znamená, že modul **MDImp** uloží sumy z prvních dvou vstupů načtených modulem **MImpIn** s návěštím 00001 do čtvrtého a pátého řádku (viz indexy v silně orámované části matice M\_Ind) matice M\_Sumy s použitím konstant v odpovídajících prvcích matice M\_K. Okamžité hodnoty ani přírůstky se negenerují (za příslušné parametry je dosazeno NONE).

<b>MemCheck</b>	Výpočet kontrolního součtu proměnné nebo paměti FLASH
-----------------	---

**Popis**

Modul vypočte kontrolní součet (CRC16) paměťové oblasti využívané databázovou proměnnou, předanou za parametr.

Je-li za parametr *Proměnná* předáno *NONE*, vypočte se kontrolní součet paměti FLASH, ve které je uložen operační systém NOS, uživatelská aplikace a neměnitelná konstantní data NOSu i aplikace. Tato speciální funkce funguje s operačním systémem NOS počínaje V3.25, se staršími verzemi vrací v parametru *CRC* vždy hodnotu 65535, resp. -1, bez ohledu na obsah FLASH.

**Parametry**

Proměnná	IN	I	Jméno databázové proměnné, jejíž kontrolní součet chceme spočítat. Lze zadat i <i>NONE</i> pro vyvolání speciální funkce modulu viz Popis.
		L	
		F	
		MI	
		ML	
		MF	
		NONE	

CRC	OUT	I	Jméno databázové proměnné, do které modul uloží vypočtený kontrolní součet (CRC16). Lze zadat jednoduchou proměnnou nebo prvek matice.
		L	
		MI	
		ML	

**Příklad**

```
MemCheck Matice, Kontrola
```

Do proměnné *Kontrola* se uloží kontrolní součet proměnné *Matice*. Později můžeme obdobným voláním modulu **MemCheck** uložit kontrolní součet do jiné proměnné, a jejím porovnáním s proměnnou *Kontrola* zjistit, zda nedošlo k narušení paměti (RAM). Nesmíme zapomenout aktualizovat proměnnou *Kontrola* po každé změně obsahu proměnné *Matice*.

```
ProcINIT:
    if @FirstRun
        MemCheck NONE, FLASH_CRC
        Let      @FirstRun=0
    else
        MemCheck NONE, FLASH_CMP
        Let      @FLASH_Err = Bool(if(FLASH_CMP==FLASH_CRC,0,1))
    endif
```

Po prvním startu systému po nahrání aplikace se uloží kontrolní součet celé oblasti FLASH do proměnné *FLASH\_CRC*. Při následujících startech se s touto hodnotou porovnává nový kontrolní součet uložený do proměnné *FLASH\_CMP* pro ověření, že nedošlo k narušení kódu či konstant NOSu či aplikace. Podle výsledku se nastavuje bit *@FLASH\_Err*. Pomocný bit *@FirstRun* musí být inicializován na jedničku a bit *@FLASH\_Err* na nulu (oba pouze při studeném startu).

<b>MImpln</b>	Ošetření šestnácti impulzních vstupů
---------------	--------------------------------------

**Popis**

Modul má stejné vlastnosti jako modul **Impln** ze standardní knihovny PSE, rozdíl spočívá jen v tom, že místo jediného *timeoutu* a jediné *minimální délky* společné pro všechny vstupy se zadávají matice, které mohou obsahovat pro každý kanál jiný *timeout* a jinou *minimální délku*. Modul se používá ve spojení s modulem **MDImp**.

**Parametry**

<b>Kanál</b>	IN	DI16	Číslo logického kanálu DI.
--------------	----	------	----------------------------

Všechny signály tohoto kanálu budou ošetřovány jako impulzní. To však nebrání použití modulu **DigIn** nad tímtož kanálem a ošetření vybraných signálů jako stavových.

<b>MinDélky</b>	IN	MI	Jméno databázové matice minimálních přípustných délek mezi pulzy [ms]. Následuje-li pulz po předchozím pulzu v čase kratším než je minimální délka, je tento pulz modulem ignorován.
		NONE	

V úvahu se bere jen první sloupec matice, má-li matice méně než 16 řádků, použijí se jednotlivé řádky opakovaně v pořadí 1 až N, 1 až N, ... tolikrát, kolikrát je to třeba pro pokrytí všech šestnácti signálů. Lze tedy použít matici s jediným prvkem jako společný *timeout* pro všechny signály.

Parametr lze zadat **NONE**. Považuje se to jako by byla zadána hodnota 0 pro všechny kanály. V tom případě modul minimální délku pulzů nehlídá.

<b>Timeouty</b>	IN	MI	Matice <i>timeoutů</i> pro jednotlivé signály logického kanálu [s].
-----------------	----	----	---

V úvahu se bere jen první sloupec matice, má-li matice méně než 16 řádků, použijí se jednotlivé řádky opakovaně v pořadí 1 až N, 1 až N, ... tolikrát, kolikrát je to třeba pro pokrytí všech šestnácti signálů. Lze tedy použít matici s jediným prvkem jako společný *timeout* pro všechny signály.

<b>Posun</b>	PAR	Konst	Udává posunutí v matici <i>Timeouty</i> .
--------------	-----	-------	---

*Timeouty* pro jednotlivé signály se berou počínaje řádkem **Posun**. To umožňuje použít jednu matici typu ŘxS = 32x1 pro dva moduly **DImpln** s tím, že se jednomu modulu nastaví **Posun** 0 a druhému 16.

<b>Náběžná</b>	PAR	Výběr	Parametr udává, který ze šestnácti vstupů má aktivní vzestupnou hranu (strukturované číslo s položkami <b>Bit0</b> až <b>Bit15</b> ).
----------------	-----	-------	---

<b>Sestupná</b>	PAR	Výběr	Parametr udává, který ze šestnácti vstupů má aktivní sestupnou hranu (strukturované číslo s položkami <b>Bit0</b> až <b>Bit15</b> ).
-----------------	-----	-------	--

<b>hNáběžná</b>	IN	I	Jméno databázové proměnné, jejíž jednotlivé bity (0 až 15) udávají, jestli má příslušný vstup aktivní náběžnou hranu. Za tento parametr lze dosadit <b>NONE</b> , v tom případě se použije hodnota parametru <b>Náběžná</b> .
-----------------	----	---	---

<b>hSestupná</b>	IN	I	Jméno databázové proměnné, jejíž jednotlivé bity (0 až 15) udávají, jestli má příslušný vstup aktivní sestupnou hranu. Za tento parametr lze dosadit <b>NONE</b> , v tom případě se použije hodnota parametru <b>Sestupná</b> .
------------------	----	---	---

**Příklad**

ProcQUICK:

```
:00001 MImpln #0,NONE,M_TO, 0, 0xFFFF, 0x0000, NONE, NONE
```

Proc00:

```
MDImp :00001, M_Ind, 2, NONE, M_Sumy, NONE, M_K, SYNC.0, SYNC.1,
```

Předpokládáme, že matice  $M\_Sumy$  a  $M\_K$  jsou matice typu například  $\mathbb{F}$ , s rozměrem  $\check{R} \times S$   $10 \times 1$ . Matice  $M\_Ind$  typu  $\mathbb{I}$  obsahuje hodnoty:

0
1
4
5

Vzhledem k parametru `Posun` rovnému 2 se uplatní silně orámovaná část matice, to znamená, že modul **MDImp** uloží sumy z prvních dvou vstupů načtených modulem **MlmpIn** s návěštím `00001` do čtvrtého a pátého řádku (viz indexy v silně orámované části matice  $M\_Ind$ ) matice  $M\_Sumy$  s použitím konstant v odpovídajících prvcích matice  $M\_K$ . Všechny vstupy mají aktivní náběžnou a neaktivní sestupnou hranu. Okamžité hodnoty ani přírůstky se negenerují (za příslušné parametry je dosazeno `NONE`).

**MKO**

Monostabilní klopný obvod

**Popis**

Modul generuje pulz do výstupní proměnné na základě náběžné resp. sestupné hrany vstupního signálu.

Délka pulzu je dána násobkem periody procesu, ve kterém je modul definován. Hodnotu násobku určuje parametr *Délka*.

Parametrem *Režim* se určuje, zda se má pulz generovat na základě náběžné nebo sestupné hrany, případně, zda má být výstupní signál negován a jak se má modul zachovat po výpadku napájení.

Chování po výpadku napájení ovlivňuje parametr *Init*. Jsou tři možné hodnoty parametru:

0	Je nastaven vnitřní stav na "0". To znamená, že MKO je ve stabilním (základním stavu).
1	Je nastaven vnitřní stav na "1". To znamená, že MKO je ve vybuzeném stavu.
2	Nastaví se vnitřní stav podle okamžitého stavu vstupní proměnné.

Přijde-li náběžná resp. sestupná hrana vstupního signálu v době, kdy je modul ve vybuzeném stavu, délka pulzu se začne počítat znovu od začátku.

**Parametry**

<b>Režim</b>	PAR	Výběr	Nastavení režimu
--------------	-----	-------	------------------

<b>Sestupná</b>	Výběr	Je-li příznak nastaven, modul reaguje na sestupnou hranu, jinak reaguje na náběžnou hranu.
-----------------	-------	--

<b>Negace</b>	Výběr	Je-li příznak nastaven, výstupní signál je negován.
---------------	-------	---

<b>Init</b>	Výběr	Příznak chování po výpadku napájení viz popis modulu. Možné hodnoty jsou 0, 1 a 2.
-------------	-------	--

<b>Vstup</b>	IN	Bit	Bit databázové proměnné - vstup MKO. Podle tohoto bitu se vyhodnocuje náběžná nebo sestupná hrana.
--------------	----	-----	--

<b>Výstup</b>	OUT	Bit	Bit databázové proměnné - výstup MKO. Do tohoto bitu se generuje pulz.
---------------	-----	-----	--

<b>Délka</b>	PAR	Konst	Počet period generování pulzu.
--------------	-----	-------	--------------------------------

**Příklad**

MKO 0x0000, MKO1.0, MKO1.1, 10

Pulz se generuje na náběžnou hranu proměnné MKO1.0 do výstupní proměnné MKO1.1. Délka pulzu je 10 period procesu, ve kterém je modul definován.

<b>MtxAdd</b>	Součet dvou matic podle pravidel maticového počtu.
---------------	--

**Popis**

Modul sečte dvě matice libovolného (případně i nestejného) maticového typu a výsledek uloží do třetí matice. Je povoleno použít jako cílovou matici pro výsledek kteroukoliv ze zdrojových matic (sčítanců).

**Parametry**

<b>hDst</b>	OUT	MI	Cílová matice, do které se uloží výsledek. Jméno proměnné libovolného maticového typu. Musí mít stejný rozměr jako oba sčítance, jinak modul v inicializační fázi nahlásí systémovou chybu "špatný počet řádků" či "špatný počet sloupců" a dále neprovádí žádné operace.
		ML	
		MF	
<b>hSrc1</b>	IN	MI	První sčítanec. Jméno proměnné libovolného maticového typu.
		ML	
		MF	
<b>hSrc2</b>	IN	MI	Druhý sčítanec. Jméno proměnné libovolného maticového typu. Musí mít stejný rozměr jako první sčítanec, jinak modul v inicializační fázi nahlásí systémovou chybu "špatný počet řádků" či "špatný počet sloupců" a dále neprovádí žádné operace.
		ML	
		MF	

**Příklad**

```
MtxAdd M_A, M_A, M_B
```

Přičte matici `M_B` k matici `M_A`.

<b>MtxCopy</b>	Kopie matice
----------------	--------------

**Popis**

Modul provede kopii matice nebo její části na libovolné místo cílové matice. Přitom je možné provést transpozici. Lze též vyplnit matici nebo její část hodnotou jednoduché databázové proměnné libovolného typu. Rovněž lze kopírovat prvek matice do jednoduché proměnné nebo jednoduché proměnné navzájem, ale v těchto případech modul **MtxCopy** nepřináší žádnou výhodu oproti příkazu **Let**.

**Parametry**

<b>hDst</b>	OUT	MI	Cílová matice. Jméno proměnné libovolného maticového typu. Lze použít i jednoduchou proměnnou, i když to nepřináší žádnou výhodu.
		ML	
		MF	

<b>hDRow</b>	IN	MI	popsán dále
		ML	
		MF	
		NONE	

<b>hDCol</b>	IN	MI	popsán dále
		ML	
		MF	
		NONE	

Řádek a sloupec v cílové matici, počínaje kterým se do cílové matice kopíruje matice zdrojová. Je-li **hDst** jednoduchá proměnná, nemají tyto parametry význam. Za oba parametry lze dosadit **NONE**, v tom případě se oba berou, jako by byly nulové.

<b>hSrc</b>	IN	I	Zdrojová matice. Jméno maticové nebo jednoduché proměnné libovolného typu. Může být použita stejná matice, jaká je dosazena za cílovou, pro přesun části matice na jiné místo.
		L	
		F	
		MI	
		ML	
		MF	

Modul správně zvolí správný směr přesouvání tak, aby se data nepoškodila v případě, že se zdrojová a cílová oblast překrývají. Není ovšem možné rotovat celou matici, protože modul nedokáže vytvořit dočasné proměnné pro zdrojové proměnné, které se přepíší přesunutými daty. Ze stejného důvodu modul ignoruje parametr **hTransp** (viz níže), pokud **hSrc** i **hDst** odkazují na stejnou matici.

<b>hSRow1</b>	IN	MI	popsán dále
		ML	
		MF	
		NONE	

<b>hSCol1</b>	IN	MI	popsán dále
		ML	
		MF	
		NONE	

<b>hSRow2</b>	IN	MI	popsán dále
		ML	
		MF	
		NONE	

<b>hSCol2</b>	IN	MI	popsán dále
		ML	
		MF	
		NONE	

Počáteční a koncový řádek/sloupec. Tyto parametry definují rozsah ve zdrojové matici, který se má přenést do cílové matice. Za všechny tyto parametry lze dosadit `NONE`, v tom případě se kopíruje celá matice. Dosadí-li se `NONE` pouze za `hSRow1` nebo `hSCol1`, je výsledek stejný, jako by se dosadila proměnná s hodnotou 0. Dosadí-li se `NONE` pouze za `hSRow2` nebo `hSCol2`, kopíruje se do konce matice.

`hSRow2` může být menší než `hSRow1`, `hSCol2` může být menší než `hSCol1`, v tom případě se rozsah bere přes příslušný okraj matice. To umožňuje kopírování hodnot z kruhových bufferů, kdy požadované hodnoty mohou začínat před koncem bufferu a končit za začátkem.

<b>hTransp</b>	IN	I	Jméno databázové proměnné, jejíž jeden bit (0..15) definuje, zda se má při přesunu provádět transpozice matice. Hodnota 1 tohoto bitu znamená, že se matice transponuje, 0 znamená prosté kopírování.
		NONE	

Transpozici nelze provádět v rámci jedné matice. Víme-li již v čase návrhu aplikace, že budeme vždy kopírovat bez transpozice, můžeme za jméno proměnné dosadit `NONE`.

### Příklad

```
MtxCopy      M_A, NONE, NONE, M_B, R1, C1, R2, C2, FLAGS.0
```

Obsahuje-li bit `FLAGS.0` nulu, proměnné `R1` a `C1` 2 a proměnné `R2` a `C2` 3, přesune se oblast matice `M_B` mezi prvky `[2,2]` a `[3,3]` tedy část o rozměru `2x2` do matice `M_A` počínajíc souřadnicemi `[0,0]` (`hDRow` a `hDCol` jsou `NONE`). Jedná se o prostý přesun bez transpozice.



**MtxMul**

Násobení matic mezi sebou nebo matice číslem podle pravidel maticového počtu.

**Popis**

Modul provede násobení matice maticí. Pokud se za první činitel dosadí jednoduchá databázová proměnná, provede se násobení druhého činitele (matice) hodnotou této proměnné. Matice musejí mít správné rozměry podle pravidel maticového počtu, jinak modul v inicializační fázi běhu aplikace generuje systémovou chybu "špatný počet řádků" nebo "špatný počet sloupců" a dále neprovádí žádné výpočty.

**Parametry**

<b>hDst</b>	OUT	MI	Cílová matice, do které se má uložit výsledek násobení. Jméno databázové proměnné libovolného maticového typu.
		ML	
		MF	

Při násobení matice maticí se jako cílová matice nesmí použít žádný z činitelů, protože by v průběhu výpočtu docházelo k přepisování prvků maticových operandů, které ještě budou potřebné pro další výpočet prvků výsledné matice.

Při násobení matice maticí musí mít cílová matice stejný počet řádků jako první činitel a stejný počet sloupců jako druhý činitel.

Při násobení matice číslem musí mít cílová matice stejný rozměr jako druhý činitel.

<b>hSrc1</b>	IN	I	První (levý) činitel. Jméno databázové proměnné libovolného typu, maticového nebo jednoduchého.
		L	
		F	
		MI	
		ML	
		MF	

<b>hSrc2</b>	IN	MI	Druhý (pravý) činitel. Je-li za prvního činitele dosazena matice, musí mít druhý činitel tolik řádků, kolik má první činitel sloupců.
		ML	
		MF	

**Příklad**

MtxMul M\_C, M\_A, M\_B

Provede se maticové násobení matice M\_A a matice M\_B, M\_A je levý činitel (dle pravidel maticového počtu). Výsledek součinu se uloží do matice M\_C.

**MtxSub**

Rozdíl dvou matic podle pravidel maticového počtu

**Popis**

Modul odečte dvě matice libovolného (případně i nestejného) maticového typu a výsledek uloží do třetí matice. Je povoleno použít jako cílovou matici pro výsledek kteroukoliv ze zdrojových matic (menšence nebo menšitele).

**Parametry**

<b>hDst</b>	OUT	MI	Cílová matice, do které se uloží výsledek. Jméno proměnné libovolného maticového typu. Musí mít stejný rozměr jako menšenec i menšitel, jinak modul v inicializační fázi nahlásí systémovou chybu "špatný počet řádků" či "špatný počet sloupců" a dále neprovádí žádné operace.
		ML	
		MF	
<b>hSrc1</b>	IN	MI	Menšenec. Jméno proměnné libovolného maticového typu.
		ML	
		MF	
<b>hSrc2</b>	IN	MI	Menšitel. Jméno proměnné libovolného maticového typu. Musí mít stejný rozměr jako první menšenec, jinak modul v inicializační fázi nahlásí systémovou chybu "špatný počet řádků" či "špatný počet sloupců" a dále neprovádí žádné operace.
		ML	
		MF	

**Příklad**

```
MtxAdd M_A, M_A, M_B
```

Odečte matici `M_B` od matice `M_A`.

<b>MtxVMul</b>	Násobení matice vektorem po řádcích nebo po sloupcích.
----------------	--

**Popis**

Druhý operand musí být vektor (řádkový nebo sloupcový), to jest matice libovolného typu s počtem řádků nebo sloupců rovným jedné. V případě sloupcového vektoru se každý prvek prvního řádku matice předané jako první operand vynásobí prvkem v prvním řádku druhého operandu. Prvky druhého řádku druhým atd. pro celou matici. V případě řádkového vektoru je postup obdobný po sloupcích.

**Parametry**

<b>hDst</b>	OUT	MI	Cílová matice. Jméno proměnné libovolného maticového typu, do které se uloží výsledek operace. Musí mít stejný rozměr jako první operand, jinak modul v inicializační fázi běhu aplikace hlásí systémovou chybu "špatný počet řádků" nebo "špatný počet sloupců" a dále neprovádí žádné výpočty. Může být použita stejná matice, která se předá jako první zdrojový operand.
		ML	
		MF	
<b>hSrc1</b>	IN	MI	Matice, která se má roznásobit. Jméno proměnné libovolného maticového typu.
		ML	
		MF	
<b>hSrc2</b>	IN	MI	Vektor koeficientů, kterými se mají násobit jednotlivé řádky nebo sloupce matice <code>hSrc1</code> . Musí mít buďto jeden řádek a tolik sloupců, kolik má <code>hSrc1</code> , nebo jeden sloupec a tolik řádků, kolik má <code>hSrc1</code> . V opačném případě modul v inicializační fázi běhu aplikace hlásí systémovou chybu "špatný počet řádků" nebo "špatný počet sloupců" a dále neprovádí žádné výpočty.
		ML	
		MF	

**Příklad**

MtxVMul M\_A, M\_A, M\_B

Provede násobení prvků matice `M_A` koeficienty z matice `M_B`. Jsou-li před provedením modulu hodnoty matic:

M_A:	<table><tr><td>1</td><td>2</td></tr><tr><td>3</td><td>4</td></tr></table>	1	2	3	4	M_B:	<table><tr><td>5</td></tr><tr><td>6</td></tr></table>	5	6
1	2								
3	4								
5									
6									

bude po provedení modulu hodnota matice `M_A`:

5	10
18	24

<b>MuxAnIn</b>	Čtení hodnoty z AI kanálu multiplexeru s převodem na fyzikální veličinu
----------------	---

**Popis**

Modul čte analogový údaj z multiplexeru a přepočítává jej na fyzikální rozměr.

**Parametry**

<b>Mux</b>	PAR	Návěští	Návěští modulu, obsluhujícího multiplexer (např. <b>DM_MUX3</b> ).
<b>Kanál</b>	IN	Konst	Číslo kanálu na vstupu multiplexeru v rozsahu <0;n), kde n je počet vstupů multiplexeru.
		I	
		MI	
<b>Hodnota</b>	OUT	F	Přepočítaná hodnota ve fyzikálních jednotkách.
		MF	
<b>Rozsah</b>	IN	Konst	Horní hranice měřicího rozsahu HW modulu v elektrických jednotkách.
		F	
		MF	
<b>EIMin</b>	IN	Konst	Dolní mez signálu v elektrických jednotkách.
		F	
		MF	
<b>EIMax</b>	IN	Konst	Horní mez signálu v elektrických jednotkách.
		F	
		MF	
<b>FyzMin</b>	IN	Konst	Dolní mez signálu ve fyzikálních jednotkách.
		F	
		MF	
<b>FyzMax</b>	IN	Konst	Horní mez signálu ve fyzikálních jednotkách.
		F	
		MF	

Přepočet na fyzikální veličinu

Probíhá identicky s modulem **AnIn**.

**Umístění modulu**

Modul pro obsluhu multiplexeru je třeba periodicky vyvolávat za dodržení podmínek uvedených v jeho popisu.

Moduly **MuxAnIn/MuxNi1000**, sloužící k načtení hodnot jednotlivých multiplexovaných vstupů, lze potom umístit do libovolně rychlého procesu. Musíme si ovšem uvědomit, že jejich umístěním do seberychejšího procesu se nic nemění na tom, že vždy po dobu uvedenou v dokumentaci obslužného modulu zůstane hodnota získaná pomocí těchto modulů neměnná.

**Příklad**

```
Proc01 (perioda: 1 s)
  10001:  DM_MUX3      #0.0, MUX_Ctrl, NONE.0, NONE.0
          AnOut        #0.0, MUX_Ctrl, 10V, 0.0, 10.0, 0.0, 10.0

Proc02 (perioda: 100 ms)
          MuxAnIn      10001:, 0, InpA0B0, 5V, 0.0, 5.0, 0.0, 5.0
```

```
MuxAnIn      10001: , 1, InpA1B1, 5V, 1.0, 5.0, 30.0, 150.0  
MuxAnIn      10001: , 2, InpA2B2, 5V, 1.0, 5.0, 30.0, 150.0
```

Výše uvedený příklad demonstruje analogové řízení multiplexeru DM-MUX3/1 s periodou přepínání 1 sekunda. Multiplexer je připojen na vstup AI.0, jeho řízení je prováděno výstupem AO.0.

V jiném procesu jsou načítány hodnoty jednotlivých vstupů multiplexeru do proměnných InpA0B0, InpA1B1 a InpA2B2. První ze vstupů je ukládán přímo ve voltech (0 až 5 V), ostatní jako hodnota fyzikální veličiny před (nějakým) připojeným převodníkem v rozsahu 30 (odpovídá napětí 1 V) až 150 (odpovídá napětí 5 V).

Je třeba upozornit, že vzhledem ke skutečnosti diskutované v oddíle "Umístění modulu" se hodnota každé z výše uvedených proměnných bude měnit jen cca při každém třicátém vyvolání procesu `Proc02`, mezitím zůstane konstantní - viz též popis funkčního modulu **DM-MUX3**.

<b>MuxNi1000</b>	Čtení teploty z odporového snímače Ni1000 připojeného přes multiplexer
------------------	--

**Popis**

Modul čte analogový údaj ze vstupu multiplexeru a přepočítává jej na teplotu měřenou odporovým snímačem teploty Ni1000. Postup přepočtu je uveden v popisu funkčního modulu **Ni1000**.

**Parametry**

Mux	PAR	Návěští	Návěští modulu, obsluhujícího multiplexer (např. <b>DM_MUX3</b> ).
-----	-----	---------	--

Kanál	IN	Konst	Číslo kanálu na vstupu multiplexeru v rozsahu <0;n), kde n je počet vstupů multiplexeru. Pozor: Jsou-li na daném HW typu stanice v konfiguraci V/V kanálů vyhrazeny zvláštní kanály pro snímače Ni1000, je potřeba jako parametr <u>obslužného modulu multiplexeru</u> používat tyto. Operační systém na těchto kanálech zajišťuje, že měřicí napětí se připojuje pouze na dobu nezbytnou pro měření vstupu a po zbytek času se napětí odpojí. Tímto mechanismem se odstraňuje chyba snímače, která by jinak vznikala vlivem vlastního zahřívání snímače.
		I MI	

Teplota	OUT	F MF	Naměřená teplota [°C]

Citlivost	PAR	Konst	Citlivost snímače [ppm] (6180 nebo 5000)
-----------	-----	-------	--

**Umístění modulu**

Modul pro obsluhu multiplexeru je třeba periodicky vyvolávat za dodržení podmínek uvedených v jeho popisu.

Moduly **MuxAnIn/MuxNi1000**, sloužící k načtení hodnot jednotlivých multiplexovaných vstupů, lze potom umístit do libovolně rychlého procesu. Musíme si ovšem uvědomit, že jejich umístěním do seberychlejšího procesu se nic nemění na tom, že vždy po dobu uvedenou v dokumentaci obslužného modulu zůstane hodnota získaná pomocí těchto modulů neměnná.

**Příklad**

```

Proc01 (perioda: 1 s)
    10001:    DM_MUX3      #1.0, NONE, DOBits.0, DOBits.1
              DigOut      DOBits, #0, 0000

Proc02 (perioda: 10 s)
    MuxNi1000 10001:, 0, TempA0B0, 6180
    MuxNi1000 10001:, 1, TempA1B1, 5000
    MuxNi1000 10001:, 2, TempA2B2, 5000

```

Výše uvedený příklad demonstruje digitální řízení multiplexeru DM-MUX3/1 s periodou přepínání 1 sekunda. Multiplexer je připojen na vstup AI.0, jeho řízení je prováděno výstupy DO.0, DO.1.

V jiném procesu jsou načítány teploty z jednotlivých vstupů multiplexeru do proměnných TempA0B0, TempA1B1 a TempA2B2. První ze snímačů má citlivost 6180 ppm, ostatní 5000 ppm.

<b>NETStat</b>	Informace o stavu komunikační sítě DB-Net
----------------	---

**Popis**

Modul získá informace o stavu komunikační sítě. Používá se pro kontrolu spojení mezi stanicemi v případech aplikací sestávajících z více stanic, kdy si procesní stanice navzájem předávají data.

**Parametry**

<b>LifeList</b>	<b>OUT</b>	<b>MI</b>	Matice rozměru [32,6]. Obsahuje tzv. <i>LifeList</i> , což je struktura popisující stav všech 32 stanic zapojených v síti. Každý řádek popisuje jednu stanic a odpovídá číslu stanice.
		<b>NONE</b>	

Význam jednotlivých sloupců:

Sloupec	Význam	Popis
0	wStStatus	Stav přípojného místa - stanice sítě (nepřipojena, pasivní, aktivní off-line, aktivní on-line): NET_NONE ( -1 ) Stanice sítě není připojena. NET_PASSIVE ( 0 ) Připojená pasivní stanice sítě. NET_ACT_NOREADY ( 1 ) Připojená aktivní stanice sítě je ve stavu prvotního poslechu sítě před připojením se na síť (neřídí provoz sítě). NET_ACT_READY ( 2 ) Připojená aktivní stanice sítě je ve stavu čekání na připojení se na síť (neřídí provoz sítě). NET_ACT_TOKEN ( 3 ) Aktivní stanice sítě je připojena na síť a řídí provoz sítě.
1	wRqStatus	Rezerva pro vyšší verze systému.
2	wTokenStatus	Stav oběhu tokenu (nemám token, mám token a vyřizuji požadavky aplikace, mám token a udržuji chod sítě), má význam pouze při dotazu na stav sám sebe: TOKEN_IDLE ( 0 ) Stanice sítě nemá token. TOKEN_REQUEST ( 1 ) Stanice sítě má token a vyřizuje aplikaci její požadavky na přenos dat po síti. TOKEN_LIFELIST ( 2 ) Stanice sítě má token a udržuje síť (oživení <i>LifeListu</i> ).
3	nOk	Počet správně vyřízených požadavků od okamžiku spuštění stanice sítě, má význam pouze při dotazu na stav sám sebe.
4	nToken	Počet tokenů od okamžiku spuštění stanice sítě, má význam pouze při dotazu na stav sám sebe.
5	nErr	Počet chybně vyřízených požadavků od okamžiku spuštění stanice sítě, má význam pouze při dotazu na stav sám sebe.

<b>Status</b>	<b>OUT</b>	<b>MI</b>	Matice rozměru [32,1] obsahující stavy stanic. Číslo stanice odpovídá číslu sloupce. Význam stavu je stejný jako parametru wStStatus v <i>LifeListu</i> . Liší se pouze číselnými kódy, které jsou voleny tak, aby se jedním bitem dalo testovat, zda je stanice přítomná na síti resp. zda je to aktivní nebo pasivní stanice.
		<b>NONE</b>	

NET\_NONE ( 0x01 )

Stanice sítě není připojena.

NET\_PASSIVE ( 0x02 )

Připojená pasivní stanice sítě.

NET\_ACT\_NOREADY ( 0x24 )

Připojená aktivní stanice sítě je ve stavu prvotního poslechu sítě před připojením se na síť (neřídí provoz sítě).

NET\_ACT\_READY ( 0x28 )

Připojená aktivní stanice sítě je ve stavu čekání na připojení se na síť (neřídí provoz sítě).

NET\_ACT\_TOKEN ( 0x30 )

Aktivní stanice sítě je připojena na síť a řídí provoz sítě.

<b>Token</b>	OUT	F	Doba oběhu jednoho tokenu v sekundách.
		NONE	

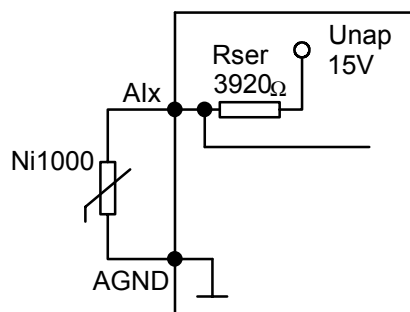


**Ni1000**

## Čtení teploty z odporového snímače Ni1000

**Popis**

Modul **Ni1000** čte analogový údaj z logického kanálu AI a přepočítává jej na teplotu měřenou odporovým snímačem teploty Ni1000 za předpokladu zapojení vstupního obvodu dle následujícího schématu:



Pro správnou funkci vstupů se snímači Ni1000 je nutné nastavit příslušné HW propojky na procení stanici dle technické dokumentace.

Přepočet odporu snímače  $R$  [ $\Omega$ ] na teplotu  $v$  [ $^{\circ}\text{C}$ ] probíhá podle vztahu:

$$v = C_1 \cdot \Delta R + C_2 \cdot \Delta^2 R + C_3 \cdot \Delta^3 R + C_4 \cdot \Delta^4 R, \text{ kde } \Delta R = R - R_0, R_0 = 1000\Omega.$$

Výše uvedený vztah je přibližnou inverzní náhradou vztahu:

$$R = R_0 \cdot (1 + A \cdot v + B \cdot v^2 + C \cdot v^3 + D \cdot v^4),$$

převzatého z materiálů výrobce snímačů - firmy Sensit.

Největší chyba inverzní náhrady je  $\Delta_{\max}$  [ $^{\circ}\text{C}$ ], viz níže uvedenou tabulku.

Výše uvedený přepočtení vztah se použije pro známé hodnoty citlivosti 6180ppm nebo 5000ppm. Pokud se při parametrizaci modulu **Ni1000** zadá jakákoliv jiná hodnota citlivosti, probíhá přepočet odporu snímače  $R$  [ $\Omega$ ] na teplotu  $v$  [ $^{\circ}\text{C}$ ] podle lineárního vztahu:

$$v = \frac{R - R_0}{R_0 \cdot \text{Citlivost} \cdot 10^{-6}}.$$

Tabulka koeficientů snímačů:

Koeficient	Typ snímače	
	Ni1000/5000ppm	Ni1000/6180ppm
$R_0$ [ $\Omega$ ]	1000	1000
Citlivost [ppm]	5000	6180
$C_1$ [ $^{\circ}\text{C} \cdot \Omega^{-1}$ ]	$2.259 \cdot 10^{-1}$	$1.82447 \cdot 10^{-1}$
$C_2$ [ $^{\circ}\text{C} \cdot \Omega^{-2}$ ]	$-5.957 \cdot 10^{-5}$	$-4.077 \cdot 10^{-5}$
$C_3$ [ $^{\circ}\text{C} \cdot \Omega^{-3}$ ]	$1.700 \cdot 10^{-8}$	$1.628 \cdot 10^{-8}$
$C_4$ [ $^{\circ}\text{C} \cdot \Omega^{-4}$ ]	$-2.890 \cdot 10^{-12}$	$-6.720 \cdot 10^{-12}$
$A$ [ $^{\circ}\text{C}^{-1}$ ]	$4.427 \cdot 10^{-3}$	$5.485 \cdot 10^{-3}$
$B$ [ $^{\circ}\text{C}^{-2}$ ]	$5.172 \cdot 10^{-6}$	$6.650 \cdot 10^{-6}$
$C$ [ $^{\circ}\text{C}^{-3}$ ]	$5.585 \cdot 10^{-6}$	0
$D$ [ $^{\circ}\text{C}^{-4}$ ]	0	$0.02805 \cdot 10^{-9}$
$\Delta_{\max}$ [ $^{\circ}\text{C}$ ]	0.00579	0.0195

## Parametry

<b>Kanál</b>	IN	AI	Vstupní logický kanál AI Jsou-li na daném HW typu stanice v konfiguraci V/V kanálů vyhrazeny zvláštní kanály pro snímače Ni1000, je potřeba používat tyto. Operační systém na těchto kanálech zajišťuje, že měřicí napětí se připojuje pouze na dobu nezbytnou pro měření vstupu a po zbytek času se napětí odpojí. Tímto mechanismem se odstraňuje chyba snímače, která by jinak vznikala vlivem vlastního zahřívání snímače. Perioda měření těchto kanálů operačním systémem je 200 ms nezávisle na periodě procesu, ve kterém je umístěn modul Ni1000. Modul tedy načítá hodnotu, která může být nejvýše 200 ms stará.
--------------	----	----	--

<b>Teplota</b>	OUT	F	Naměřená teplota [°C]
		MF	

<b>Citlivost</b>	PAR	Konst	Citlivost snímače [ppm] (6180 nebo 5000)
------------------	-----	-------	--

## Příklad

Ni1000 #1.0, TEPLOTA, 6180

Do proměnné TEPLOTA se uloží teplota měřená odporovým snímačem teploty Ni1000/6180ppm, připojeným ke vstupu namapovanému na logický kanál 1, signál 0.

## Ni1000U2T Převod napětí na snímači Ni1000 na teplotu

### Popis

Modul **Ni1000U2T** je obdobou funkčního modulu **Ni1000**.

Používá se v případech, kdy měřená hodnota vstupu není přiřazena do logického kanálu. Tento případ nastává tehdy, kdy se neměří standardní interní vstup řídicího systému. S tím se setkáváme zejména v případě měření na vzdáleném analogovém vstupu, připojeném na komunikační sběrnici (např. sběrnice **ARION**, **CAN**).

V takovém případě se měřený vstup načte do databázové proměnné jako napěťový (zpravidla v rozsahu 0÷5 V), a následně se pomocí modulu **Ni1000U2T** toto napětí převede na odpovídající teplotní údaj.

Předpokládá se stejné schéma měřicího obvodu, jako u modulu **Ni1000**, pro převod platí stejné rovnice.

U vstupních modulů, vyráběných firmou AMiT, jsou hodnoty konstant  $U_{\text{nap}}$  a  $R_{\text{ser}}$  vždy 15 V a 3920  $\Omega$ , jak je naznačeno ve schématu v popisu modulu **Ni1000**. Modul **Ni1000U2T** je ovšem možno použít i pro vstupy, vyráběné jinými firmami, proto je možno tyto konstanty zadat jako parametry. Pro vstupy, vyráběné firmou AMiT, je možno nechat těmto parametrům jejich implicitní hodnoty.

### Parametry

Napětí	IN	F	Naměřená napětí na snímači [V]
		MF	

Teplota	OUT	F	Naměřená teplota [°C]
		MF	

Citlivost	PAR	Konst	Citlivost snímače [ppm] (6180 nebo 5000). Viz rozbor v popisu modulu <b>Ni1000</b> .
-----------	-----	-------	--

Unap	PAR	Konst	Napájecí napětí měřicího děliče [V] - viz schéma v popisu modulu <b>Ni1000</b> . Pro vstupy, vyráběné firmou AMiT, vyhoví implicitní hodnota 15 V.
------	-----	-------	--

Rser	PAR	Konst	Odpor sériového rezistoru v měřicím děliči [ $\Omega$ ] - viz schéma v popisu modulu <b>Ni1000</b> . Pro vstupy, vyráběné firmou AMiT, vyhoví implicitní hodnota 3920 $\Omega$ .
------	-----	-------	--

### Příklad

```
ARN_AI      :17002, 1, NONE.0, 0, Napeti, 5, 0, 5, 0, 5
Ni1000U2T   Napeti, TEPLOTA, 6180, 15.0, 3920.0
```

Do proměnné **TEPLOTA** se uloží teplota měřená odporovým snímačem teploty Ni1000/6180ppm, připojeným ke vstupu 0 zařízení, připojeného na sběrnici **ARION**. Vstup byl nejprve načten jako napěťový v rozsahu 0÷5 V pomocí modulu **ARN\_AI**.

*Pozn.: Teoreticky je možné používat pro dočasné uložení vstupního napětí stejnou proměnnou jako pro uložení převedené teploty, zde proměnnou **TEPLOTA**. Z praktických důvodů je ovšem nutno tento postup důrazně nedoporučit - v proměnné **TEPLOTA** by se na krátké okamžiky objevovalo místo teploty napětí ve voltech, což by mohlo působit problémy v nezávislých procesech nebo při vizualizaci, vč. **LCDSHELLU**.*

<b>Nop</b>	Prázdná operace
------------	-----------------

**Popis**

Modul slouží k vizuálnímu oddělení skupin řádků uvnitř procesu při tvorbě aplikace.

**Parametry**

Žádné.

<b>ParseTime</b>	Převod času z formátu systému DB-Net do položek
------------------	---

**Popis**

Modul dostává čas ve formátu systému DB-Net (počet sekund od 1.1.1980 00:00:00) v proměnné *Čas*. Rozloží jej na položky do proměnné *DMYhms*.

**Parametry**

<b>Čas</b>	IN	L	Obsahuje čas ve formátu systému DB-Net.
		ML	
		NONE	

Pokud je čas v matici a nechceme měnit souřadnice za běhu, stačí zadat souřadnice pomocí čísel v tomto parametru a parametry *Řádek* resp. *Sloupec* zadat *NONE*. Pokud je potřeba měnit souřadnice (souřadnici) za běhu, musí se souřadnice zadat pomocí databázových proměnných v parametrech *Řádek*, *Sloupec*. Souřadnice zadaná proměnnou má přednost před souřadnicí zadanou číslem.

Zadá-li se parametr *NONE*, modul načte čas z proměnné, ale načítá vnitřní čas stanice.

<b>Řádek</b>	IN	I	Řádek v matici <i>Čas</i> . Lze zadat <i>NONE</i> .
		NONE	

<b>Sloupec</b>	IN	I	Sloupec v matici <i>Čas</i> . Lze zadat <i>NONE</i> .
		NONE	

<b>DMYhms</b>	OUT	MI	Proměnná - matice rozměru [8x1], která obsahuje okamžitý čas rozložený na jednotlivé položky - den, měsíc, rok, hodina, sekunda, minuta, den týdne a den roku. Pozor, den roku začíná nulou (1.ledna).
---------------	-----	----	--

Řádek	Význam
0	sekundy
1	minuty
2	hodiny
3	den v měsíci
4	měsíc
5	rok
6	den týdne
7	den roku

Den týdne má následující význam:

Hodnota	Význam
0	Neděle
1	Pondělí
2	Úterý
3	Středa
4	Čtvrtek
5	Pátek
6	Sobota

**Příklad**

```
ParseTime    Cas[2,0], NONE, Cas_S, Cas_položky
```

Modul převede čas z časové matice *Cas* na položky do matice *Cas\_položky*. Řádek je zadán napevno jako číslo 2 a sloupec je dán proměnnou *Cas\_S*.

<b>PhaseCtrl</b>	Fázové řízení (triaku) - nastavování
------------------	--------------------------------------

**Popis**

Modul slouží k procesu fázového řízení (typicky triaků) jako modul provádějící výpočetně náročné přípravné práce pro vlastní výkonný modul **PhaseRun**. Fázové řízení znamená, že v určený časový okamžik každé periody se sepne příslušný digitální výstup a těsně před koncem periody se stejný digitální výstup zase rozepte. Úsek, kdy je digitální výstup v sepnutém stavu se pak nazývá *dobou otevření* nebo *úhlem otevření* (úhel 0 odpovídá nulovému otevření, úhel 180° nebo-li  $\pi$  radiánů pak plnému otevření).

Modul je určen pro umístění do řádných procesů eventuelně do QUICK procesu. S jeho pomocí se definuje řízený digitální výstup a požadovaná doba (úhel) otevření.

Pokud se více instancí modulu **PhaseCtrl** odkazuje na stejný modul **PhaseRun** (což odpovídá současnému řízení více výstupů), pak **musí** být umístěny ve stejném procesu a **musí** ovládat digitální výstupy ve stejném logickém kanálu.

**Parametry**

<b>PhaseRun</b>	PAR	Návěští	Návěští modulu <b>PhaseRun</b> , který je použit pro fyzické fázové řízení.
-----------------	-----	---------	---

<b>Vstup</b>	IN	Konst	Vstupní hodnota s dobou nebo úhlem otevření. Podle parametru <i>Jednotka</i> se určuje v jakých jednotkách byla hodnota <i>Vstup</i> zadána. Je-li hodnota mimo meze pro zvolený rozsah, řízený výstup se buď naplno otevře (více jak maximum) nebo se neotevře vůbec (méně jak minimum).
		I	
		L	
		F	
		MI	
		ML	
		MF	

<b>Jednotka</b>	PAR	Výběr	Konstanta určující jednotku parametru <i>Vstup</i> . Může nabývat hodnot:		
			Hodnota	Konst	Popis
			ms	0	Vstup se zadává v milisekundách. Povolený rozsah je od 0 po hodnotu periody.
			deg	1	Vstup se zadává v úhlových stupních. Povolený rozsah je od 0 do 180.
			rad	2	Vstup se zadává v radiánech. Povolený rozsah je od 0 do $\pi$ .
			%	3	Vstup se zadává v procentech periody. Povolený rozsah je od 0 do 100.

<b>Vystup</b>	OUT	DO	Výstupní řízený bit.
---------------	-----	----	----------------------

**Příklad**

Viz příklad u modulu **PhaseRun**.

<b>PhaseRun</b>	Fázové řízení (triaku)
-----------------	------------------------

**Popis**

Modul slouží k procesu fázového řízení (typicky triaků) jako výkonný řídicí modul. Fázové řízení znamená, že v určený časový okamžik každé periody se sepne příslušný digitální výstup a těsně před koncem periody se stejný digitální výstup zase rozepne. Úsek, kdy je digitální výstup v sepnutém stavu se pak nazývá *dobou otevření* nebo *úhlem otevření* (úhel 0 odpovídá nulovému otevření, úhel 180° nebo-li  $\pi$  radiánů pak plnému otevření). Oblasti okolo začátku a konce periody nejdou vlivem zpoždění programu a HW korektně řídit. Nejrychlejší sepnutí je možno cca 160 $\mu$ s od počátku periody, nejpozdější sepnutí záleží na nastavení parametrů a minimálně je cca 300 $\mu$ s od konce periody.

Modul je určen pro umístění do interrupt procesu, čímž se definuje vazba na jeden digitální vstup a jeho logika (reakce na náběžnou nebo sestupnou hranu). Modul **PhaseRun** vyžaduje, aby svázaný digitální vstup generoval hranu vždy na začátku řízené periody (pro řízení triaku tomu odpovídá průchod sítě nulovým napětím). Od tohoto okamžiku modul **PhaseRun** odpočítává řízení výstupů po dobu jedné periody.

Modul **PhaseRun** není LA modul. Při vkládání do interrupt procesu typu LA se musí vkládat jako systémový modul, při vkládání do interrupt procesu typu RS se musí vkládat do systémového sloupce.

Modul svými parametry v zásadě popisuje časové konstanty řízené periody a svým umístěním v interrupt procesu vstupní (synchronizační) digitální vstup. Při řízení modul pracuje s přesností 1/128 periody, takže například pro periodu 10ms je přesnost vystavení výstupu  $\pm 39\mu$ s.

Na modul se pak odkazují moduly **PhaseCtrl** umístěné v řádných procesech, které definují konkrétní řízené výstupy a k nim odpovídající doby (úhly) otevření.

**Parametry**

<b>Perioda</b>	PAR	Konst	V milisekundách délka řízené periody. Přípustný rozsah periody je od 0.1ms do 2000ms.
----------------	-----	-------	--

Pro periody pod cca 8ms však dochází k vlivu trvání délky modulu, který se projeví zejména pokud je řízeno více výstupů na blízké hodnoty otevření. V takových případech dochází vlivem "nestíhání programu" ke zpoždování otevírání.

<b>Zpozdění</b>	PAR	Konst	V milisekundách hodnota zpoždění reakce modulu a digitálních výstupů. Implicitní hodnota 0,160 ms vychází z naměřených hodnot pro AMiRiS99, AMAP99 a ART267. V dalším textu je uveden návod, jak provést vlastní nastavení tohoto parametru.
-----------------	-----	-------	--

<b>ZpozdRozd</b>	PAR	Konst	V milisekundách hodnota zpoždění, která se přidá k hodnotě Zpozdění pro sestupnou hranu řídicích signálů před koncem periody. Implicitní hodnota 0,080 ms vychází z naměřených hodnot pro AMiRiS99, AMAP99 a ART267. V dalším textu je uveden návod, jak provést vlastní nastavení tohoto parametru.
------------------	-----	-------	--

Parametr **ZpozdRozd** definuje dvě důležité hodnoty:

1. V řídicích systémech firmy AMiT jsou digitální výstupy opticky odděleny. Z fyzikálních vlastností používaných součástek vychází, že doba rozepnutí je delší (řádově o desítky  $\mu$ s) než doba sepnutí.
  2. Je třeba zabezpečit, aby řídicí signály vypnuly bezpečně před koncem periody, jinak hrozí plné otevření triaků na následující periodu.
- Součet obou těchto hodnot je třeba zadat jako **ZpozdRozd**, čímž se modulu určuje předstih pro bezpečné rozepnutí řídicích signálů.

## Neregulovatelné oblasti

Vlivem zpoždění programu a HW nejdou oblasti okolo začátku a konce periody korektně řídit.

Doba  $Z_{\text{pozdeni}}$  udává, s jakým předstihem musí řídicí systém vydávat povel k sepnutí, takže doba  $\text{Perioda} - Z_{\text{pozdeni}}$  je maximální doba otevření.

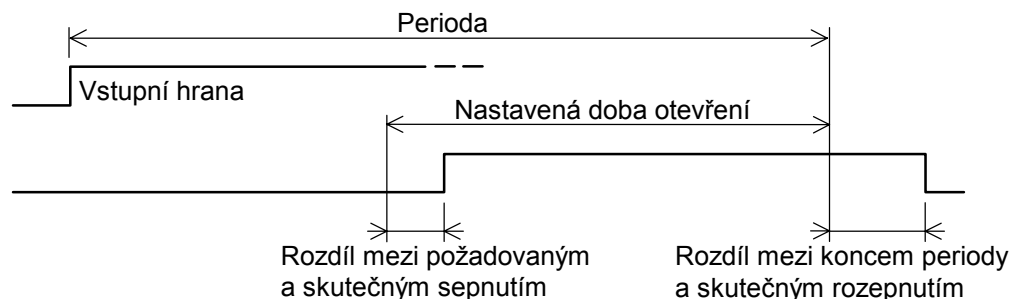
Doba  $Z_{\text{pozdeni}}$  a  $Z_{\text{pozdBzd}}$  udává, s jakým předstihem musí řídicí systém vydávat povel k rozepnutí, aby se stihlo fyzicky provést před koncem periody. Vzhledem použité přesnosti  $1/128$  periody je minimální doba otevření  $Z_{\text{pozdeni}} + Z_{\text{pozdBzd}} + 1/128 * \text{Perioda}$ .

## Nastavení parametrů

Přednastavené konstanty  $Z_{\text{pozdeni}}$  a  $Z_{\text{pozdBzd}}$  vychází z naměřených hodnot pro systémy AMiRiS99, AMAP99 a ART267. Nastavení konstant pro jiný typ systému nebo pro kalibraci konkrétního kusu jsou zapotřebí dostatečné technické znalosti a vybavení. Bezpečně lze pouze zvyšovat hodnotu  $Z_{\text{pozdBzd}}$ .

Princip nastavení parametrů je:

- 1) Vytvořte aplikaci, s uvažovaným počtem řízených výstupů. Konstanty  $Z_{\text{pozdeni}}$  a  $Z_{\text{pozdBzd}}$  nastavte na 0.
- 2) Jeden z výstupů postupně nastavujte na náhodně zvolené doby otevření v rozsahu 10% až 90% periody.
- 3) Pro každou nastavenou hodnotu změřte dobu
  - mezi požadovaným okamžikem otevření a skutečným okamžikem sepnutí výstupu a
  - mezi okamžikem konce periody a skutečným okamžikem rozepnutí výstupu. Pro každou nastavenou hodnotu toto měření proveďte vícekrát.
 Při měření negenerujte vstupní pulzy s požadovanou periodou, ale používejte osamělé pulzy.



- 4) Průměrnou hodnotu naměřených rozdílů mezi požadovaným a skutečným sepnutím zadejte jako parametr  $Z_{\text{pozdeni}}$ .
- 5) Maximální hodnota z naměřených rozdílů mezi koncem periody a skutečným rozepnutím zmenšená o minimální hodnotu z naměřených rozdílů mezi požadovaným a skutečným sepnutím je základ parametru  $Z_{\text{pozdBzd}}$ . K tomu je třeba přičíst dobu (dle uvážení) o kterou je třeba výstupy bezpečně rozepnout před koncem periody, aby nedošlo k přesahu. Tato korekce by měla vycházet z naměřené odchylky periody vstupních pulzů.



**Příklad**

```
ProcITR00

:01111          PhaseRun 10.0, 0.160, 0.080

Proc00          1.0 s

                PhaseCtrl :01111, WOUT0, ms, #OUT0
                PhaseCtrl :01111, WOUT1, %, #OUT1
```

Fázové řízení je nastaveno na interrupt proces 0 (u AMiRiS99 odpovídá DI0.0). Řízená perioda je 10ms, konstanty zpoždění jsou implicitní.

Řídíme dva výstupy: #OUT0 na hodnotu dle proměnné WOUT0 zadávanou v milisekundách a #OUT1 na hodnotu dle proměnné WOUT1 zadávanou v procentech.

<b>PID</b>	Regulátor PID
------------	---------------

**Popis**

Modul **PID** realizuje regulátor s regulačními algoritmy P, I, PI, PD a PID. Lze u něj nastavit řadu provozních parametrů a přizpůsobit detailní vlastnosti modulu konkrétním potřebám. Režim práce a vlastnosti regulátoru se nastavují jako logické příznaky v proměnné typu  $\mathbb{I}$ . Význam jednotlivých bitů je tento:

**Bit 0**Nulový akční zásah

Během inicializace procesní stanice nebo během jiných činností, kdy není žádoucí pohybovat akčním členem (ventil apod.) se tento příznak nastavuje do 1. Regulátor pak udržuje akční zásah v "neutrálním stavu" (viz dále). Hodnota příznaku 0 znamená odblokování regulátoru, povolení jeho činnosti.

**Bit 1**Režim AUT/MAN

Má-li tento příznak hodnotu 0, pracuje regulátor v autonomním režimu, tj. reguluje podle zadanych parametrů. Nastavením příznaku do 1 se přepíná do manuálního režimu, kdy výsledný akční zásah nezapisuje do výstupní proměnné - očekává, že je do ní akční zásah zapisován "ručně" (jiným způsobem), např. z ovládacího terminálu nebo po síti. Zpětné přepnutí na autonomní provoz probíhá beznárazově, tj. nedojde ke skokové, ale plynulé změně akčního zásahu - toto opatření podstatně snižuje namáhání akčních členů při přepínání režimu regulátoru. Beznárazové přepínání funguje pouze u regulátorů s integrační složkou (PI nebo PID).

**Bit 2**Konstanty PSD/PID

Je-li tento příznak nastaven, zadávají se běžné konstanty regulátoru PID tedy  $K$ ,  $T_i$  a  $T_d$  odpovídající vzorci pro výpočet akčního zásahu:

$$y = K \cdot \left( x + \frac{1}{T_i} \int_0^t x \cdot dt + T_d \cdot \frac{dx}{dy} \right)$$

Je-li tento příznak roven 0, zadávají se přímo konstanty tzv. *ekvivalentního PSD regulátoru*. Tyto konstanty odpovídají vzorci pro výpočet akčního zásahu:

$$y = y_{i-1} + b_0 \cdot x_i + b_1 \cdot x_{i-1} + b_2 \cdot x_{i-2}$$

Kde index  $i$  značí okamžitou hodnotu příslušné veličiny,  $i-1$  hodnotu minulou a  $i-2$  předminulou.

**Bit 3**Nevyužit**Bit 4**Změna konstant

Změna konstant regulátoru je spojena se značným množstvím přepočtů a kontrol. Není proto vhodné, aby regulátor prováděl tyto přepočty opakovaně. Měníme-li proto konstanty regulátoru, pak po změně poslední z nich zapíšeme do tohoto bitu hodnotu 1. Regulátor pak provede jednorázově všechny potřebné operace a vrátí hodnotu tohoto bitu na 0. Tato hodnota setrvá po celou dobu až do další změny parametrů.

**Konstanty regulátoru**

Činnost regulátoru lze dále ovlivnit nastavením osmi konstant. Tyto konstanty jsou uloženy v databázové matici typu MF o rozměru 8 řádků a 1 sloupec. Jednotlivé prvky mají tento význam:

**[0, 0]**

V režimu PID:                      proporcionální konstanta  $K$   
V režimu PSD:                      konstanta  $b_0$

**[1, 0]**

V režimu PID:                      Integrační konstanta  $T_i$   
V režimu PSD:                      konstanta  $b_1$

**[2, 0]**

V režimu PID:                      derivační konstanta  $T_d$   
V režimu PSD:                      konstanta  $b_2$

**[3, 0]**

Dolní mez akčního zásahu

[4, 0]

Horní mez akčního zásahu

[5, 0]

Neutrální stav akčního zásahu (posunutí nuly, bias). Je to hodnota, která se přičítá k vypočtenému akčnímu zásahu.

Příklad

Pokud se má výstup pohybovat v rozmezí 0% až 100% a neutrální stav (počáteční hodnota výstupu) má být 50%, pak se zadají hodnoty:

Dolní mez: -50

Horní mez: 50

Neutrální stav: 50

[6, 0]

Pásmo necitlivosti regulátoru na regulační odchylku. Je-li absolutní hodnota regulační odchylky menší než zadaná hodnota, neprovádí regulační algoritmus žádné změny akčního zásahu. Tak lze v ustáleném stavu podstatně omezit namáhání akčních členů nepřetržitými drobnými pohyby způsobenými vlivem derivační složky regulátoru.

[7, 0]

Zpoždění derivace. Časová konstanta filtru 1. řádu na odfiltrování šumu, aby se derivační složka nezahlucovala šumem. Hodnota 0 má význam "bez zpoždění". Tento parametr má smysl nastavovat tehdy, když kolísání měřené hodnoty vlivem šumu se projevuje na derivační složce natolik, že značně zkresluje výstup regulátoru.

Vliv šumu lze obecně eliminovat dvěma způsoby:

1) zvětšením periody spouštění modulu (zvětšením periody procesu, ve kterém je definován).

Derivační složka je totiž dána vztahem:

$$D = K \frac{T_d}{\Delta T} \Delta x$$

kde

 $\Delta T$  je perioda $\Delta x$  je difference současné a minulé regulační odchylky.

Zvětšením periody se zmenší velikost derivační složky, tedy i výsledný projev šumu.

Zvětšení periody je ovšem možné jen u relativně "pomalých" regulací, v jiných případech se musí použít druhá metoda.

2) zpožděním derivace

Difference regulační odchylky se filtruje filtrem 1. řádu. Časová konstanta filtru by měla být co nejmenší, jinak by se eliminoval nejen vliv šumu ale i vliv derivační složky vůbec. Většinou se zadává časová konstanta řádově několik vteřin.

Pokud je hodnota časové konstanty menší než je perioda modulu, případně rovna nule, zpoždění derivace se neuplatní.

## Parametry

<b>Žádaná</b>	IN	F	Proměnná s žádanou hodnotou, na kterou se reguluje.
		MF	

<b>Měření</b>	IN	F	Proměnná s měřenou hodnotou, která se reguluje.
		MF	

Je-li potřeba zaměnit smysl regulace, provede se to záměnou proměnných dosazených za parametry Žádaná a Měření.

<b>Výstup</b>	OUT	F	Proměnná, do níž zapisuje regulátor hodnotu akčního zásahu.
---------------	-----	---	---

<b>Režim</b>	IN/OUT	I	Proměnná, která obsahuje režim činnosti regulátoru a volby (viz popis). Doporučujeme dosadit za parametr <b>inicializovanou</b> databázovou proměnnou. Hodnotu proměnné lze sice také nastavovat až v <code>INIT</code> procesu (příp. i jindy), ale v tom
--------------	--------	---	---

			případě se nesmí zapomenout nastavit <b>bit č. 4 na "1"</b> , jinak se inicializace modulu neprovede správně.
<b>Parametry</b>	IN	MF	Matice o rozměru [8, 1] s parametry regulátoru.

**Příklad**

PID    ZadanaH, MerenaH, Zasah, Rezim, Parametry

Regulátor reguluje hodnotu v proměnné MerenaH na žádanou hodnotu v proměnné ZadanaH. Potřebné akční zásahy zapisuje do proměnné Zasah. Řídící proměnnou (režim) regulátoru je Rezim, parametry jsou nastaveny v matici Parametry.

Je-li obsah proměnné Rezim roven 100 (bitově) a obsah matice Parametry roven 1.0, 120.0, 35.0, -50.0, 50.0, 50.0, 1.0, 0, znamená to, že regulátor řádně pracuje v autonomním režimu, konstanty odpovídají verzi PID a jsou nastaveny na  $K = 1$ ,  $T_i = 120$  a  $T_d = 35$ . Akční zásah se bude pohybovat od 0 do 100% s "nulou" (neutrálním stavem) na 50%. Pásmo necitlivosti regulátoru na odchylku je nastaveno na  $\pm 1$  (např. °C). Derivační složka je bez zpoždění.

**PPlan**

Periodické plánování hodnot po etapách

**Popis**

Modul realizuje časový plán pro jednu proměnnou, který je rozdělen na jednotlivé etapy. Pomocí řídicích proměnných lze ovládat start nebo ukončení etapy, přeskočení na zadanou etapu, start etapy od zadaného časového okamžiku apod. Modul může realizovat několik různých plánů. Každý plán je představován řádkem v matici časů Časy a řádkem v matici hodnot zlomů Plán. Volbou řádku lze tedy přepínat jednotlivé plány. **Všechny parametry a proměnné plánovače s významem času se zadávají v násobcích běhů procesu, ve kterém je modul umístěn.** Pokud je tedy modul umístěn v procesu s periodou 1s, jsou všechny parametry v sekundách.

Etapa (i-tá) je představována trojicí hodnot:

délka etapy Časy[ŘádekČasů, i]  
počáteční hodnota Plán[ŘádekPlán, i]  
koncová hodnota Plán[ŘádekPlán, i+1]

Hodnota v etapě je buď lineárně interpolovaná z těchto dvou hodnot, nebo je rovna počáteční hodnotě (tzv úrovnové plánování).

Plánovač může běžet buď v automatickém režimu, kdy sám přechází na další etapu, nebo vyžaduje spuštění další etapy pomocí řídicí proměnné.

Plán končí po proběhnutí všech etap tj. všech sloupců matice Časy, nebo končí na prvním nulovém sloupci matice Časy.

Počet sloupců matice hodnot Plán musí být stejný jako počet sloupců matice Časy. V případě úrovnového plánování je maximální počet etap dán počtem sloupců matic. U lineárního plánování je maximální počet etap o jedna menší kvůli poslední koncové hodnotě v matici Plán. Hodnota v posledním sloupci matice časů se v tomto režimu ignoruje.

Po skončení plánu může plánovač automaticky spustit celý cyklus znovu. Ukončení etapy nebo ukončení celého cyklu indikuje plánovač v řídicí proměnné.

**Parametry**

Režim	PAR	Výběr	Režim činnosti.
Lineární	Výběr	ANO: Plánovač uvnitř etapy provádí lineární aproximaci mezi počáteční a koncovou hodnotou etapy. NE: Celou etapu se plánuje počáteční hodnota.	
Periodic	Výběr	ANO: Plánovač je periodický. Po skončení poslední etapy automaticky pokračuje od začátku první etapy. NE: Po skončení poslední etapy plánovač skončí. Znovu se dá spustit pomocí řídicí proměnné.	
AktivVýst	Výběr	ANO: Mimo aktivní dobu, plánovač nezapisuje do výstupní proměnné. NE: Plánovač přepisuje výstupní proměnnou vždy.	
Auto-Etapa	Výběr	ANO: Po ukončení etapy plánovač automaticky přechází na novou etapu. NE: Plánovač nepřechází automaticky na novou etapu. Nová etapa se spouští bitem Řízení. 2=-1.	
Řízení	IN/OUT	I	Řídicí proměnná.

Význam jednotlivých bitů:

0.bit - vstupní bit, spuštění plánovače

Nastavením bitu do "1" se spustí plánovač. Plánovač tento bit po spuštění vynuluje.

1.bit - vstupní bit, povolení spuštění další etapy

Je-li bit nastaven do "1", tak po ukončení etapy plánovač automaticky přechází na novou etapu. Plánovač tento bit nenuluje.

Požívá se v případě, když není nastaven automatický přechod mezi etapami (AutoEtapa = NE).

2.bit - vstupní bit, žádost o přeskočení na novou etapu

Nastavením bitu do "1" se okamžitě ukončí právě zpracovávaná etapa, a přejde se na novou etapu, jejíž číslo musí být zadáno v proměnné NováEtapa. Po přechodu na novou etapu je bit vynulován.

3.bit - vstupní bit, vypnutí plánovače

Nastavením bitu do "1" se plánovač vypne. Plánovač přestane prepisovat výstupní proměnnou Výstup, nedělá nic.

4.bit - vstupní bit, spuštění etapy od zadaného času

Nastavením bitu do "1" se spustí etapa od zadaného času uvnitř etapy (nikoli od začátku). "Startovací" čas je v zadán v proměnné NovýČas. Po spuštění je bit vynulován. Je-li zadaný startovací čas větší než délka etapy, spustí se etapa od začátku. Číslo etapy se tímto řídicím bitem nemění. Bit se používá v kombinaci s bitem číslo 2.- žádost o přeskočení na novou etapu.

8.bit - výstupní bit, plánovač dojel na konec

Nastavením do "1" plánovač oznamuje ukončení celého cyklu. Bit se vynuluje po spuštění nového cyklu.

9.bit - výstupní bit, ukončení etapy

Nastavením do "1" plánovač oznamuje ukončení etapy. Bit se vynuluje po spuštění další etapy příp. celého cyklu.

10. bit - výstupní bit, plánovač je aktivní ("běží")

"1"= plánovač je aktivní a plánuje.

"0"= plánovač stojí.

<b>Časy</b>	IN	ML	Matice časů. Řádek matice představuje časové délky jednotlivých etap (násobky běhu procesu). Číslo aktuálního řádku určuje proměnná ŘádekČasů.
-------------	----	----	--

<b>ŘádekČasů</b>	IN	I	Proměnná, která určuje číslo aktuálního řádku matice časů.
------------------	----	---	--

<b>Plán</b>	IN	MF	Matice hodnot zlomů. Řádek matice představuje počáteční a koncové hodnoty jednotlivých etap. Počáteční hodnota následující etapy je shodná s koncovou hodnotou etapy předchozí. Aktuální řádek matice určuje proměnná ŘádekPlán.
-------------	----	----	--

<b>ŘádekPlán</b>	IN	I	Proměnná, která určuje číslo aktuálního řádku matice hodnot zlomů Plán.
------------------	----	---	---

<b>Výstup</b>	OUT	F	Výstupní proměnná, která představuje plánovanou hodnotu.
---------------	-----	---	--

<b>Etapa</b>	OUT	I	Proměnná, do které plánovač zapisuje číslo právě probíhající etapy (0 až n-1, kde n je počet sloupců matice Časy).
--------------	-----	---	--

<b>ČasEtapy</b>	OUT	L	Proměnná, do které plánovač zapisuje čas uvnitř etapy v násobcích běhu procesu.
-----------------	-----	---	---

<b>NováEtapa</b>	IN	I	Proměnná, ve které plánovač očekává číslo další etapy, viz.
------------------	----	---	---

			bit Řízení.2.
<b>NovýČas</b>	IN	I	Proměnná, ve které plánovač očekává startovací čas nově spuštěné etapy, viz. bit Řízení.4.

**Příklad**

Let                      PlanInfoT1.3 = ! PlanOn1.0

PPlan                  0x000D, PlanInfoT1, PlanTime1, PlanRow1,  
                          PlanT1,PlanRow1, T1req, Section1, TimeInSectT1,  
                          NewSection1, NewTimeScT1

Plánovač plánuje žádanou teplotu v místnosti T1 v proměnné T1req. Plánuje se lineární interpolací, po poslední etapě plánovač skončí, mimo aktivitu nepíše do plánu. Spouštění etap není automatické, mezi etapami se přechází bitem PlanInfoT1.2, číslo nové etapy je v proměnné NewSection1. Plánovač se vypíná bitem PlanOn1.0.

**PrintBar**

Tisk obdélníku, jehož šířka závisí na hodnotě databázové proměnné, na sériové tiskárně

**Popis**

Modul slouží ke grafické indikaci hodnoty číselné proměnné na tiskárně. Opakovaným voláním tohoto modulu s různými hodnotami lze vytvořit graf průběhu veličiny.

Šířka obdélníku je závislá na hodnotě databázové proměnné (lze použít i prvek matice). Binární kódy, které je třeba vyslat na tiskárnu pro tisk každého bodu obdélníku, jakož i inicializační a ukončovací sekvence, jsou uloženy v databázové matici.

**Parametry**

<b>PrintMgr</b>	PAR	Návěští	Návěští modulu <b>PrintMgr</b> , který je použit pro fyzické vyslání dat na sériovou tiskárnu.
-----------------	-----	---------	--

<b>Hodnota</b>	IN	I	Jméno databázové proměnné libovolného (i maticového) typu.
		L	
		F	
		MI	
		ML	
		MF	

<b>Řádek Sloupec</b>	IN	I	Řádek a sloupec, které odkazují na prvek matice <b>Hodnota</b> , který se má použít. Je-li <b>Hodnota</b> jednoduchá proměnná, může se za <b>Řádek</b> i <b>Sloupec</b> dosadit <b>NONE</b> .
		NONE	

<b>DolníMez HorníMez</b>	IN	I	Meze veličiny <b>Hodnota</b> , které odpovídají nulové a plné šířce obdélníku. Jména databázových proměnných libovolného jednoduchého typu.
		L	
		F	

<b>Kódy</b>	IN	MI	Jméno matice typu, jejíž jednotlivé řádky obsahují binární kódy, které se mají vyslat na tiskárnu pro provedení určité akce.
-------------	----	----	--

Každý řádek popisuje jednu akci, Je-li třeba k provedení akce vyslat méně znaků, než je počet sloupců matice, je třeba vložit za poslední znak sekvence číslo větší než 255. Význam jednotlivých řádků:

Řádek	Význam
0	Inicializace grafického režimu tiskárny - je-li třeba
1	Kód, který se má vyslat pro levý okraj obdélníku
2	Kód, který se má vyslat pro střední bod(y) obdélníku
3	Kód, který se má vyslat pro pravý okraj obdélníku
4	Kód, který se má vyslat pro bod(y) prázdné části plochy indikátoru
5	Kód, který se má vyslat pro ukončení prázdné plochy indikátoru.
6	Ukončení grafického režimu tiskárny - je-li třeba.

<b>Šířka</b>	IN	I	Jméno databázové proměnné typu, která obsahuje plnou šířku indikátoru v bodech.
--------------	----	---	---

<b>Úspěch</b>	OUT	Bit	Bit databázové proměnné, který se nastaví na 1, pokud se tisk podařilo uskutečnit, nebo na 0, pokud tisk nebyl možný (plný výstupní tiskový buffer).
		NONE	

Modul tiskne na principu "všechno nebo nic", takže pokud je po návratu z modulu v bitu **Úspěch** nula, je možné (a zpravidla potřebné) opakovaným voláním tisk opakovat dokud se nepodaří.



Nechceme-li testovat výsledek tisku, můžeme za jméno proměnné dosadit `NONE`.

**Příklad**

`ProcIDLE:`

`:00001 PrintMgr 0xF00,1,4800,8,0,1`

`Proc00:`

`PrintBar :00001,Hodnota,NONE,NONE,Low,High,Kody,Sirka,  
FLAGS.0`

Provede se tisk obdélníku podle hodnoty databázové proměnné `Hodnota`.

<b>PrintBin</b>	Tisk binárních dat na sériové tiskárně
-----------------	--

**Popis**

Modul vytiskne na sériové tiskárně data obsažená v zadaném řádku matice typu MI.

**Parametry**

<b>PrintMgr</b>	PAR	Návěští	Návěští modulu <b>PrintMgr</b> , který je použit pro fyzické vyslání dat na sériovou tiskárnu.
-----------------	-----	---------	--

<b>Data</b>	IN	MI	Jméno databázové matice, jejíž jeden řádek se vyšle na tiskárnu. Je-li třeba vyslat menší počet znaků, než je počet sloupců matice, je třeba vložit do matice za požadovaná data hodnotu větší než 255.
		ML	
		MF	

<b>hŘádek</b>	IN	I	Jméno databázové proměnné, která obsahuje číslo řádku matice, který se má použít. Víme-li už v čase návrhu aplikace, že budeme tímto modulem tisknout vždy stejný řádek, můžeme dosadit NONE a číslo řádku zadat do následujícího parametru.
		NONE	

<b>Řádek</b>	PAR	Konst	Hodnota, která se použije, je-li hŘádek=NONE.
--------------	-----	-------	---

<b>Úspěch</b>	OUT	Bit	Jméno databázové proměnné typu, jejíž jeden bit (0 až 15) se nastaví na 1, pokud se tisk podařilo uskutečnit, nebo na 0, pokud tisk nebyl možný (plný výstupní tiskový buffer).
		NONE	

Modul tiskne na principu "všechno nebo nic", takže pokud je po návratu z modulu v bitu Úspěch nula, je možné (a zpravidla potřebné) opakovaným voláním tisk opakovat dokud se nepodaří.

Nechceme-li testovat výsledek tisku, dosadíme za jméno proměnné NONE.

**Příklad**

ProcIDLE:

```
:00001 PrintMgr 0xF00,1,4800,8,0,1
```

Proc00:

```
PrintBin :00001, Tisky, NONE, 2, FLAGS.0
```

Provede se tisk řádku 2 matice Tisky.

<b>PrintMgr</b>	Modul zajišťuje komunikaci se sériovou tiskárnou nebo podobným zařízením
-----------------	--

**Popis**

Modul se zpravidla umísťuje do IDLE procesu stanice, podobně jako například modul **LCDDTY**. Zajišťuje základní obsluhu sériové linky pro potřeby ostatních modulů knihovny **PRINT**. V jeho parametrech se udává požadované nastavení komunikačního kanálu, (komunikační rychlost, parita, počet stopbitů apod.) Tomuto modulu musí být přiřazeno návěští, které potom musíme uvést v příslušném parametru všech modulů **Print...**, jež provádějí vlastní tisk dat.

Aktivaci tisku lze vázat na nastavení **DIP přepínače** na panelu procesní stanice v případě, že chceme na stejném sériovém kanálu jak komunikovat s tiskárnou či podobným zařízením, tak tuto linku využívat při servisních zásazích pro komunikaci pomocí protokolu DB-Net. V tomto případě lze modul nastavit tak, že po resetu/zapnutí procesní stanice testuje stav některého **DIP přepínače** a je-li **OFF**, obsluha tiskárny se nenainstaluje a použitý kanál je možno využívat protokolem DB-Net. Máme-li dostatek volných komunikačních kanálů, můžeme také nastavit modul **PrintMgr** tak, že se obsluha nainstaluje bez ohledu na stav jakýchkoliv přepínačů.

**Parametry**

Typ	PAR	Výběr	Strukturované číslo:
<b>Typ tisk.</b>	Konst		Typ připojené tiskárny, rezerva pro budoucí změny, v této verzi modulu nemá žádný význam, zadávejte 0.
<b>DIP</b>	Konst		Číslo <b>SW DIP přepínače</b> 1 až 15, který musí být při resetu/zapnutí procesní stanice v poloze <b>ON</b> , aby se obsluha tiskárny nainstalovala. Je-li zadána hodnota 0, nainstaluje se obsluha tiskárny vždy, bez ohledu na stav <b>DIP přepínačů</b> .
<b>Přímo</b>	Výběr		"NE" znamená, že má být pro komunikaci použit <b>XON/XOFF protokol</b> , "ANO" znamená komunikaci bez jakéhokoli <b>handshake</b> .
<b>Kanál</b>	PAR	Konst	Číslo komunikačního kanálu, který se má pro komunikaci použít (např. na procesní stanici ART400: 0=RS232, 1=RS485, na procesní stanici ADiS lze navíc použít pro komunikaci HW modul AD_UART).
<b>Rychlost</b>	PAR	Konst	Komunikační rychlost, na jaké se mají data přenášet. Zadá-li se rychlost, jakou hardware procesní stanice nepodporuje, použije se nejbližší podporovaná rychlost. Např. zadáme-li 9610Bd, použije se 9600.
<b>Délka (7 až 8)</b>	PAR	Konst	Počet datových bitů v komunikačním slově.
<b>Parita (0 až 1)</b>	PAR	Konst	Parita - zadáme-li nulu, přenáší se bez parity, při jakékoliv jiné hodnotě se sudou paritou.
<b>Stopbity (1 až 2)</b>	PAR	Konst	Počet stopbitů, ukončujících komunikační slovo.

**Příklad**

ProcIDLE:

:00001      PrintMgr 0xF00,1,9600,8,0,1

Proc00:

PrintBin :00001, Tisky, 2, FLAGS.0

Tisk (nebo komunikace se zařízením, které může být považováno za tiskárnu) probíhá na lince 1 (RS232), rychlostí 9600Bd, 8 bitů dat, 1 stopbit, žádná parita, nepoužívá se *XON/XOFF protokol*, obsluha se nainstaluje vždy bez ohledu na stav *DIP přepínačů*.

<b>PrintStr</b>	Tisk řetězce na sériové tiskárně
-----------------	----------------------------------

**Popis**

Na sériovou tiskárnu se vytiskne textový řetězec, který je přímým parametrem modulu. Řetězec je ukončen znakem s kódem 0 (to zajistí program PSE), z čehož vyplývá, že nemůže tento znak obsahovat.

**Parametry**

<b>PrintMgr</b>	PAR	Návěští	Návěští modulu <b>PrintMgr</b> , který je použit pro fyzické vyslání dat na sériovou tiskárnu.
-----------------	-----	---------	--

<b>Řetězec</b>	PAR	Řetězec	Řetězec znaků, které se mají vytisknout. Řetězec může obsahovat řídicí znaky, viz dodatek "Řídicí znaky v řetězcích".
----------------	-----	---------	---

<b>Úspěch</b>	OUT	Bit	Bit databázové proměnné, který se nastaví na 1, pokud se tisk podařilo uskutečnit, nebo na 0, pokud tisk nebyl možný (plný výstupní tiskový buffer).
		NONE	

Modul tiskne na principu "všechno nebo nic", takže pokud je po návratu z modulu v bitu **Úspěch** nula, je možné (a zpravidla potřebné) opakovaným voláním tisk opakovat dokud se nepodaří.

Nechceme-li testovat výsledek tisku, dosadíme za jméno proměnné **NONE**.

**Příklad**

```
ProcIDLE:
```

```
:00001 PrintMgr 0xF00, 1, 4800, 8, 0, 1
```

```
Proc00:
```

```
PrintStr :00001, "Ahoj\n", FLAGS.0
```

Vytiskne na tiskárně řetězec "Ahoj" a odřádkuje.

Formát tisknutého data a času se zadává pomocí formátovacího řetězce. Proměnná s datem a časem může být prvek matice. Řádek a sloupec je potom zadán databázovými proměnnými. Pokud je proměnná s datem a časem jednoduchého typu, nemusí se řádek a sloupec proměnné vyplňovat.

<b>PrintMgr</b>	PAR	Návěští	Návěští modulu <b>PrintMgr</b> , který je použit pro fyzické vyslání dat na sériovou tiskárnu.
-----------------	-----	---------	--

<b>Formát</b>	PAR	Řetězec	Formátovací řetězec. Při formátování tisknutého řetězce se speciální formátovací makra nahrazují položkami data a času. Formátovací makro začíná znakem '%'.  
---------------	-----	---------	--

<b>Makro</b>	<b>Význam</b>	<b>Šířka textu</b>
%D	Den	2 znaky doplněné nulou
%M	Měsíc	2 znaky doplněné nulou
%Y	Rok	2 znaky doplněné nulou
%h	Hodiny	2 znaky doplněné nulou
%m	Minuty	2 znaky doplněné nulou
%s	Sekundy	2 znaky doplněné nulou
%d	Den v týdnu	7 znaků ("Pondeli".. "Nedele ")

Kromě toho může formátovací řetězec obsahovat řídicí znaky - viz dodatek "Řídicí znaky v řetězcích".

Čas	IN	L	Databázová proměnná s datem a časem.
		ML	

<b>Řádek</b>	IN	I	Databázová proměnná - řádek v matici Čas. Pokud se neuvede žádná proměnná, bere se to, jako by byl zadán řádek 0.
		NONE	

<b>Sloupec</b>	IN	I	Databázová proměnná - sloupec v matici čas. Pokud se neuvede žádná proměnná, bere se to, jako by byl zadán sloupec 0.
		NONE	

<b>Úspěch</b>	OUT	Bit	Bit databázové proměnné, který se nastaví na 1, pokud se tisk podařilo uskutečnit, nebo na 0, pokud tisk nebyl možný (plný výstupní tiskový buffer).
		NONE	

Modul tiskne na principu "všechno nebo nic", takže pokud je po návratu z modulu v bitu Úspěch nula, je možné (a zpravidla potřebné) opakovaným voláním tisk opakovat, dokud se nepodaří.

Nechceme-li výsledek tisku testovat, dosadíme za jméno proměnné `NONE`.

**Příklad**

ProcIdle:

:18000 PrintMgr 0x1F00, 2, 19200, 8, 0, 1

Proc00:

PrintTM :18000,"%D.%M.%Y %d %h:%m:%s ", Time, NONE, NONE,  
TiskOk.1

Tisk data a času pro příklad hodnoty proměnné Time vypadá následovně: "10.04.97  
Čtvrtek 16:07:15".

<b>PrintVar</b>	Tisk číselné hodnoty na sériové tiskárně
-----------------	--

**Popis**

Číselná hodnota zformátovaná podle zadaného formátovacího řetězce se vypíše na tiskárně.

**Parametry**

<b>PrintMgr</b>	PAR	Návěští	Návěští modulu <b>PrintMgr</b> , který je použit pro fyzické vyslání dat na sériovou tiskárnu.
-----------------	-----	---------	--

<b>Formát</b>	PAR	Řetězec	Formátovací řetězec ve formátu běžném u funkcí <b>..printf()</b> v jazyce "C".
---------------	-----	---------	--

Nejčastější jednoduché formáty:

Formát	Význam
%d	Celé číslo (16 bitů) se znaménkem
%u	Celé číslo (16 bitů) bez znaménka
%ld	Celé číslo (32 bitů) se znaménkem
%lu	Celé číslo (32 bitů) bez znaménka
%f	Číslo float ve formátu 123.456
%g	Číslo float ve formátu 1.234e56

Formátovací řetězec umožňuje stanovit přesnost výpisu, počet cifer, zarovnání vpravo nebo vlevo, atd. Detailní popis viz popis funkcí **..printf()** v jazyce "C". Pozor, formát musí odpovídat typu použité proměnné, není například dovoleno použít formát "%f" pro proměnnou typu `int` nebo `long`, v obou případech bude vytištěno nesmyslné číslo.

Formátovací řetězec může obsahovat libovolný text před i za vlastním formátovacím polem. Např. "Příkon: %d kW" je platný zápis formátovacího řetězce.

Formátovací řetězec může obsahovat i řídicí znaky, viz dodatek "Řídicí znaky v řetězcích".

<b>Proměnná</b>	IN	I	Jméno databázové proměnné libovolného (i maticového) typu, jejíž hodnota se má vytisknout.
		L	
		F	
		MI	
		ML	
		MF	

<b>Řádek Sloupec</b>	IN	I	Řádek a sloupec prvku matice, který se má vytisknout. Je-li <b>Proměnná</b> jednoduchá proměnná, nemají tyto parametry význam a může se za ně dosadit <b>NONE</b> . Je-li <b>Proměnná</b> maticová proměnná, má dosazení <b>NONE</b> za <b>Řádek</b> nebo <b>Sloupec</b> stejný efekt, jako dosazení proměnné s nulovou hodnotou.
		NONE	

<b>Úspěch</b>	OUT	Bit	Jméno databázové proměnné, jejíž jeden bit (0 až 15) se nastaví na 1, pokud se tisk podařilo uskutečnit, nebo na 0, pokud tisk nebyl možný (plný výstupní tiskový buffer).
		NONE	

Modul tiskne na principu "všechno nebo nic", takže pokud je po návratu z modulu v bitu **Úspěch** nula, je možné (a zpravidla potřebné) opakovaným voláním tisk opakovat dokud se nepodaří.

Nechceme-li testovat výsledek tisku, dosadíme za jméno proměnné **NONE**.



**Příklad**

ProcIDLE:

:00001 PrintMgr 0xF00,1,4800,8,0,1

Proc00:

PrintVar :00001, "Prikon %5d kW\n", Prikon, NONE, NONE,  
FLAGS.0

Obsahuje-li proměnná `Prikon` (typu `I`) hodnotu 1234, vytiskne se na tiskárně text:  
`Prikon: 1234 kW` a odřádkuje se.

<b>PrtDbStr</b>	Tisk databázového řetězce na sériové tiskárně
-----------------	---

**Popis**

Modul vytiskne na sériové tiskárně data obsažená v zadaném řádku matice typu MI. Funkce je obdobná jako u modulu **PrintBin**, data se ale chápou jako nulou ukončený řetězec, tisk se tedy ukončí, vyskytne-li se v dotyčném řádku matice hodnota 0 nebo hodnota větší než 255.

**Parametry**

<b>PrintMgr</b>	PAR	Návěští	Návěští modulu <b>PrintMgr</b> , který je použit pro fyzické vyslání dat na sériovou tiskárnu.
-----------------	-----	---------	--

<b>Data</b>	IN	MI	Jméno databázové matice, jejíž jeden řádek se vyšle na tiskárnu. Je-li třeba vyslat menší počet znaků, než je počet sloupců matice, je třeba vložit do matice za požadovaná data hodnotu 0 nebo hodnotu větší než 255.
		ML	
		MF	

<b>hŘádek</b>	IN	I	Jméno databázové proměnné, která obsahuje číslo řádku matice, který se má použít. Víme-li už v čase návrhu aplikace, že budeme tímto modulem tisknout vždy stejný řádek, můžeme dosadit NONE a číslo řádku zadat do následujícího parametru.
---------------	----	---	--

<b>Řádek</b>	PAR	Konst	Hodnota, která se použije, je-li hŘádek=NONE.
--------------	-----	-------	---

<b>Úspěch</b>	OUT	Bit	Jméno databázové proměnné, jejíž jeden bit (0 až 15) se nastaví na 1, pokud se tisk podařilo uskutečnit, nebo na 0, pokud tisk nebyl možný (plný výstupní tiskový buffer).
		NONE	

Modul tiskne na principu "všechno nebo nic", takže pokud je po návratu z modulu v bitu Úspěch nula, je možné (a zpravidla potřebné) opakovaným voláním tisk opakovat dokud se nepodaří.

Nechceme-li testovat výsledek tisku, dosadíme za jméno proměnné NONE.

**Příklad**

```
ProcIDLE:
```

```
:00001 PrintMgr 0xF00,1,4800,8,0,1
```

```
Proc00:
```

```
PrtdbStr :00001, Tisky, NONE, 2, FLAGS.0
```

Provede se tisk řádku 2 matice Tisky.

<b>Pt100R2T</b>	Přepoččet odpor -> teplota pro čidlo Pt100
-----------------	--

**Popis**

Modul realizuje přepoččet hodnoty elektrického odporu platinového teploměru PT100 na teplotu ve stupních Celsia.

**Parametry**

<b>Odpor</b>	IN	F	Elektrický odpor v ohmech.
		MF	

<b>Teplota</b>	OUT	F	Přepočítaná teplota ve stupních Celsia.
		MF	

**Příklad**

```

AnIn          #0.0, R, 10, 0, 10, 5, 313.65
Pt100R2T      R, T
Filtr1R       T, Tfiltr, 0.80

```

Nejprve se změří elektrický odpor platinového teploměru pomocí modulu `AnIn`, odpor se měří v rozmezí  $100\text{--}313.65\Omega$ , čemuž odpovídá teplota  $0\text{--}600^\circ\text{C}$ . Přepočítaná teplota se poté ještě filtruje.

<b>PulseOut</b>	Impluzní výstup na digitálním výstupu
-----------------	---------------------------------------

**Popis**

Modul umožňuje použít některé digitální výstupy jako impulzní výstupy. Umožňuje definovat časové poměry impulzů, vysílat impulzy v dávkách, zahájit další dávku impulzů přesto, že předchozí dávka ještě nebyla celá vyslána.

Zpravidla nechceme vysílat stejný počet impulzů při každém vyvolání periodického procesu, do něž je modul **PulseOut** vložen, ale chceme vysílat při splnění nějaké podmínky. Proto je nutno použít jednu ze dvou následujících technik:

```
If @Podminka
    PulseOut #0.0, 10, 10.0, 100.0, BezInverze, @PulseRun
EndIf
```

nebo (předpokládáme, že při vzniku podmínky pro vysílání se proměnná `Pocet` v jiné části aplikace naplní požadovaným počtem impulzů):

```
PulseOut #0.0, Pocet, 10.0, 100.0, BezInverze, @PulseRun
Let      Pocet = 0
```

Druhá technika je výhodnější a doporučovaná, protože při ní je modul **PulseOut** vyvoláván i v situaci, kdy není požadováno vysílání impulzů, což umožňuje správné vynulování bitu `@PulseRun` po odvysílání všech požadovaných impulzů. Tato druhá technika využívá toho, že při vyvolání modulu **PulseOut** s nulovou hodnotou parametru `Pocet` nemá modul žádný vliv na činnost výstupu, pouze naplní bit předaný za parametr `Běh` správnou hodnotou podle toho, jestli se pulsy ještě vysílají, nebo už skončily.

Při první technice bude mít bit `@PulseRun` stále hodnotu 1, protože bezprostředně po vyvolání modulu **PulseOut** s parametrem `Pocet` rovným 10 bude výstup zákonitě v běhu, po odvysílání požadovaných impulzů už zase není vyvoláván modul **PulseOut**, takže bit `@PulseRun` nemá kdo vynulovat.

Viz též dodatek “*Použití digitálních výstupů jako frekvenčních nebo impulzních výstupů*”.

**Parametry**

<b>Signál</b>	OUT	DO	Číslo zapisovaného signálu DO.
---------------	-----	----	--------------------------------

<b>Počet</b>	IN	Konst	Počet impulzů, které se mají vyslat. Pokud ještě neskončilo vysílání impulzů vyžádaných v předchozím vyvolání modulu <b>PulseOut</b> , nové impulzy se přičtou k počítadlu zbývajících impulzů, takže se žádné impulzy “neztratí”. Je-li hodnota parametru <code>Pocet</code> nulová, nebude činnost výstupu nijak ovlivněna, pouze se naplní bit předaný za parametr <code>Běh</code> .
		I	
		L	
		MI	
		ML	

<b>DélkaOn</b>	IN	Konst.	Délka aktivní části každého impulzu. Nesmí být delší než <code>MaxDélka</code> , jinak se žádné impulzy nevyšlou.
		F	
		MF	

<b>DélkaOff</b>	IN	Konst.	Délka mezery mezi impulzy. Nesmí být delší než <code>MaxDélka</code> , jinak se žádné impulzy nevyšlou.
		F	
		MF	

Pokud je modul **PulseOut** vyvolán v době, kdy se ještě vysílají impulzy vyvolané předchozím vyvoláním modulu **PulseOut**, a liší-li se hodnota některého z parametrů `DélkaOn`, `DélkaOff` od předchozího vyvolání, budou zbývající impulzy z předchozího vyvolání vyslány již s novým nastavením časových poměrů.

<b>MaxDélka</b>	PAR	Konst	Maximální délka generovaných pulzů i prodlev - viz rozbor v dodatku "Použití digitálních výstupů jako frekvenčních nebo impulzních výstupů". Je-li za některý z parametrů DélkaOn, DélkaOff předána větší hodnota, vyvolání modulu <b>PulseOut</b> nemá vliv na činnost výstupu.
-----------------	-----	-------	---

<b>Inverze</b>	PAR	Výběr	Udává vztah mezi stavy ON/OFF a fyzickým stavem výstupu.	
			Hodnota	Význam
			0	BezInverze - ON odpovídá logické jedničce, OFF odpovídá logické nule. Po odvysílání impulzů zůstane výstup ve stavu logické nuly.
			1	Inverzní - ON odpovídá logické nule, OFF odpovídá logické jedničce. Po odvysílání impulzů zůstane výstup ve stavu logické jedničky.

Pozn.: V případě, že je v aplikaci použito více modulů **FreqOut** nebo **PulseOut** se stejnou hodnotou parametru Signál, je výsledné chování jejich společného výstupu určeno takto:

Má-li v  kterémkoliv  z dotyčných modulů parametr Inverze hodnotu Inverzní, výstup funguje inverzně.

Má-li ve  všech  dotyčných modulech parametr Inverze hodnotu BezInverze, výstup funguje normálně.

<b>Běh</b>	OUT	Bit	Výstupní bit, indikující, že na příslušném výstupu probíhá generování impulzů. Pokud vysílání všech vyžádaných impulzů již skončilo, bit se při nejbližším vyvolání modulu <b>PulseOut</b> vynuluje. Nechceme-li stav vysílání impulzů testovat, lze za tento parametr dosadit NONE.
------------	-----	-----	---

### Příklad

```
If @Podminka
    Let Pocet = 10
EndIf
PulseOut #0.0, Pocet, 10.0, 100.0, BezInverze, @PulseRun
Let Pocet = 0
```

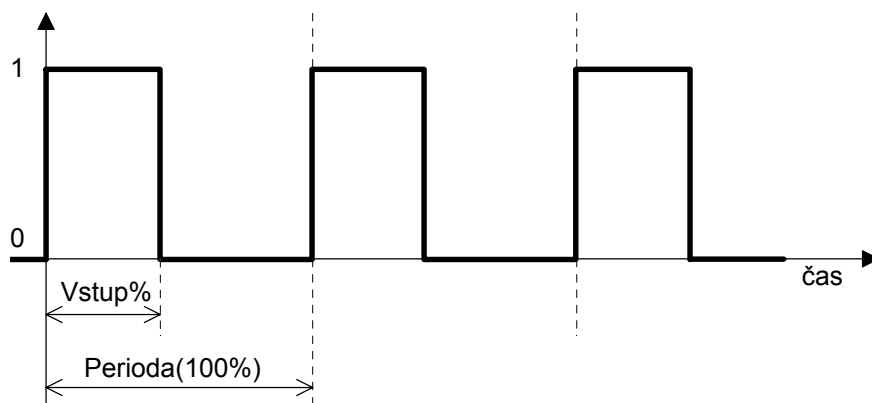
Je-li nastaven bit @Podminka, je na digitální signál DO0.0 vysílána sekvenčně deseti impulzů v úrovni logické jedničky o délce 10 ms s mezerami 100 ms. Po ukončení vysílání je výstup nastaven do logické nuly. Bit @PulseRun je možno použít ke zjištění, jestli se pulzy ještě vysílají (hodnota 1), nebo jejich vysílání již skončilo (hodnota 0).

**PWM**

Pulzně šířková modulace

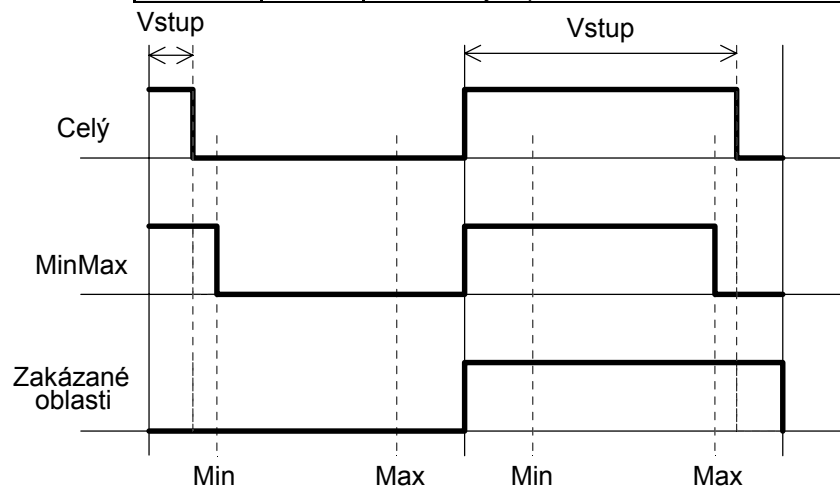
**Popis**

Modul převede proměnnou  $V_{\text{stup}}$  v rozsahu 0..100% na pulzně šířkovou modulaci, která má periodu  $Perioda$ . Rozlišení velikosti střídý je dané periodou procesu, ve kterém se modul nachází.

**Parametry**

Režim	PAR	Výběr	Režim činnosti
-------	-----	-------	----------------

Interval	Konst	Rozsah a omezení pulzu.
		<p>0 <b>MinMax</b> Velikost pulzu (doba, ve které je výstup v "1") je modulem omezena na interval <math>\langle Min, Max \rangle</math>. Pokud by měl být pulz kratší než je velikost <math>Min</math>, tak je omezen na velikost <math>Min</math>. Pokud by měl být pulz delší než je velikost <math>Max</math>, tak je omezen na velikost <math>Max</math>.</p> <p>1 <b>Celý</b> Pulz může mít délku v celém rozsahu PWM tj. v intervalu <math>\langle 0, Perioda \rangle</math>. Délka 0 se projeví tak, že výstup je trvale v "0". Jestliže má délka hodnotu <math>Perioda</math>, projeví se to tak, že výstup je trvale v "1".</p> <p>2 <b>Zakázané oblasti</b> Pulz nesmí být kratší než <math>Min</math> a zároveň nesmí být delší než <math>Max</math>. Pokud by měl být pulz kratší než je velikost <math>Min</math>, tak je na výstupu trvale "0". Pokud by měl být pulz delší než je velikost <math>Max</math>, tak je na výstupu trvale "1".</p>



<b>Plynulá</b>	Výběr	ANO=povolení plynulé změny PWM. Střída se v tomto případě nepočítá pouze na začátku periody PWM, ale může se přepočítávat a měnit i uvnitř periody.
----------------	-------	---

<b>Začíná_1</b>	Výběr	ANO=perioda PWM začíná log. "1" a končí log. "0", NE=perioda PWM začíná log. "0" a končí log. "1".
-----------------	-------	--

<b>Vstup</b>	IN	F	Vstupní proměnná s rozsahem 0..100%
		MF	

<b>Perioda</b>	IN	Konst	Perioda PWM [s].
		F	
		MF	

<b>Min</b>	IN	Konst	Minimální délka pulzu [s].
		F	
		MF	

<b>Max</b>	IN	Konst	Maximální délka pulzu [s].
		F	
		MF	

<b>Výstup</b>	OUT	Bit	Výstupní bit.
		DO	

#### Upozornění

Modul se musí umístit do procesu s takovou periodou, aby perioda PWM (parametr Perioda) byla vždy menší než je 65535 násobek periody procesu.

#### Příklad

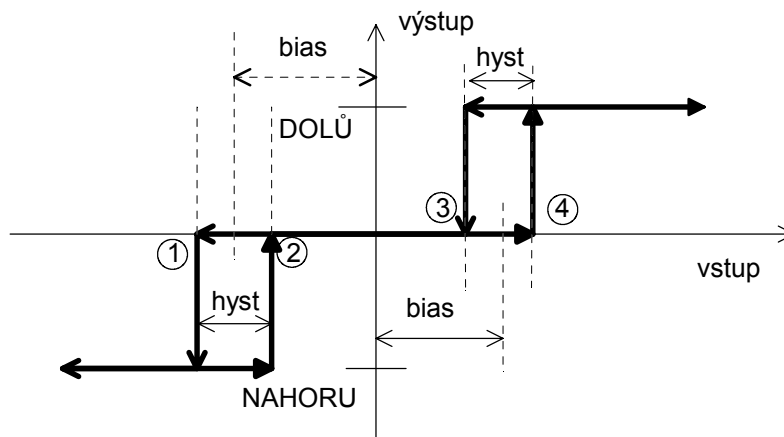
PWM 0x0008, Tlaction, 15, 2, 15, PWM1.0

Modul je umístěn v procesu s periodou 0.5s. Realizuje PWM výstup do 0. bitu proměnné PWM1. PWM má periodu 15s, minimální pulz je 2s, maximální pulz není omezen (15s = periodě PWM).

<b>Relay</b>	Obecné (třístavové) relé s hysterezí - reléový regulátor
--------------	--

**Popis**

Modul **Relay** pracuje jako obecné třístavové relé s hysterezí, vhodné pro realizaci jednodušších regulací, porovnávacích operací (analogie modulu **Hyst**).



Při narůstání vstupního signálu relé překlápí ze stavu NAHORU do VYPNUTO v bodě 2 a dále do stavu DOLŮ v bodě 4. Při poklesu vstupního signálu překlápí zpět do stavu VYPNUTO v bodě 3 a do stavu NAHORU v bodě 1.

Konkrétní poloha bodů 1 až 4 je určena dvěma konstantami - konstanta *bias* (posun) určuje polohu a konstanta *hyst* (hystereze) vzdálenost bodů. Z obrázku je zřejmé, že nastavením *hyst* na 0 získáme relé bez hystereze, nastavením *bias* na 0 získáme dvoustavové relé. Vhodnou volbou těchto dvou parametrů lze tedy získat jakoukoli charakteristiku reléového typu.

V modulu **Relay** se vyhodnocují dvě vstupní veličiny - žádaná hodnota a měřená (regulovaná) veličina. Z těchto hodnot se vypočítá regulační odchylka a ta se teprve použije pro pohyb po přepínací charakteristice relé. Výstupem modulu jsou pak přímo povely NAHORU a DOLŮ pro ovládací ventil.

**Parametry**

<b>Vstup</b>	IN	F	Proměnná se vstupní (žádanou) hodnotou.
		MF	
<b>Měření</b>	IN	F	Proměnná s měřenou (regulovanou) hodnotou.
		MF	
<b>VýstupON</b>	OUT	Bit	Bit pro povel NAHORU.
		DO	
<b>VýstupOFF</b>	OUT	Bit	Bit pro povel DOLŮ.
		DO	
<b>Hystereze</b>	IN	Konst	Velikost hystereze <i>hyst</i> .
		F	
		MF	
<b>Bias</b>	IN	Konst	Velikost posunu <i>bias</i> .
		F	
		MF	



**Příklad**

Relay                    ZadanaH, MerenaH, Ovladani.0, Ovladani.1, 5, 15

Modul pracuje jako reléový regulátor s žádanou hodnotou v proměnné ZadanaH, regulovanou veličinou v proměnné MerenaH a výstupem do bitů č. 0 a 1 a proměnné Ovladani. Hystereze má velikost 5 a bias má velikost 15.

REM	Komentář
-----	----------

**Popis**

Modul představuje komentář, kterým lze označovat a oddělovat od sebe úseky programu a zpřehlednit tak výsledný zápis programu. Modul se nijak neprojeví na výsledném generovaném kódu aplikace. Lze jej vkládat zcela libovolně.

**Parametry**

Komentář	PAR	String	Řetězec znaků komentáře.
----------	-----	--------	--------------------------

**Příklad**

```
REM      "--- Nacteni teplot z analogovych kanalu -----"
AnIn     #0.0, T1, 10.0, 0.0, 10.0, -20.0, 50.0
AnIn     #0.1, T2, 10.0, 0.0, 10.0, -20.0, 50.0
REM      "--- Filtrace -----"
Filtr1R  T1, T1f, 0.50
Filtr1R  T2, T2f, 0.50
REM      "-----"
```

<b>Repeat</b>	Cyklus s podmínkou na konci
---------------	-----------------------------

**Popis**

Modul **Repeat** realizuje cyklus s podmínkou na konci. Příkazy/moduly těla cyklu se vykonávají tak dlouho, dokud není splněna podmínka v následujícím příkazu **Until**.

**Parametry**

Návěští	PAR	Návěští	Návěští následujícího příkazu <b>Until</b> , který ohraničuje cyklus <b>Repeat</b> a obsahuje rovněž podmínku ukončení cyklu. Návěští je automatické a lokální, lze je tedy generovat automaticky.

**Příklad**

```

Repeat      :00000
. . .
Let      Konec.0 = . . .
:00000  Until      Konec.0

```

Moduly těla cyklu se vykonávají tak dlouho, dokud se v příkazu **Let** nepřihadí podmínkovému bitu hodnota 1, která činnost cyklu ukončí. Poté se provádí modul bezprostředně následující za příkazem **Until**.

## Report

Zápis alarmu/hlášení/informace do provozního deníku

## Popis

Modul **Report** umožňuje zapsat alarmové nebo jiné hlášení nebo obecně jakoukoli informaci do provozního deníku v závislosti na vzniku jakékoli události v procesní stanici. Další zpracování zapsaných informací (např. kvitace alarmů) nezajišťuje sama procesní stanice, nýbrž vizualizační zařízení, tedy ovládací terminál nebo specializovaný programový modul na vizualizační stanici.

## provozní deník

Práce systému DB-Net je v zásadě periodická - periodicky se sbírají a zpracovávají data z technologického procesu, periodicky se archivují, periodicky se obnovují obrazovky na vizualizačních stanicích atd. Při práci systému však dochází k neplánovaným významným událostem, které je třeba podchytit a "zaprotokolovat". Jde například o výpadky a obnovu napájení procesních stanic, vybočení významných technologických veličin mimo meze nebo návrat do mezí apod. Vzhledem k tomu, že se jedná o neplánované a nepravdělné události, není vhodné využít pro jejich záznam běžné archivy, protože charakter jejich práce je periodický. Pro záznam takových událostí je v systému DB-Net k dispozici *provozní deník*.

Procesní stanice detekuje vznik významné události a vkládá informaci o ní do speciálního "archivu" - provozního deníku. Kapacita tohoto archivu je omezená (standardně na 50 záznamů). Nová hlášení pak přepisují nejstarší záznamy. Záznamy z provozního deníku je možné dekódovat, převést všechny nebo jen některé do textové podoby a zobrazit na obrazovce uživatelské stanice nebo terminálu. Poté je možné je vytisknout nebo jiným způsobem zpracovat.

Vkládání záznamů do provozního deníku zajišťuje operační systém NOS procesní stanice. Záznam obsahuje datum a čas, kdy byl záznam vložen, kód hlášení, označení místa v programovém vybavení (funkčního modulu, fáze zpracování apod.), kde hlášení vzniklo, a doplňující informaci, závislou na kódu hlášení.

Systém DB-Net definuje cca 10 systémových hlášení, které není dovoleno používat v uživatelských programech. Lze však definovat prakticky libovolné množství uživatelských hlášení, přiřadit jim text a formát výpisu a interpretace doplňující informace. Tak lze uspokojit prakticky jakékoli požadavky na protokolování výjimečných událostí.

Provozní deník je blíže popsán v části "Stavové a chybové informace, provozní deník".

## Parametry

<b>Podmínka</b>	IN	Bit	Proměnná a číslo bitu proměnné. Přechod tohoto bitu z 0 na 1 detekuje vznik události, kterou je třeba zaprotokolovat. Opačný přechod žádnou událost nevyvolá.
-----------------	----	-----	---

<b>Kód</b>	IN	Konst	Celé číslo označující kód události. Na úrovni procesní stanice je lhostejné, který kód reprezentuje systémové hlášení, který alarm nebo uživatelské hlášení apod., protože procesní stanice není vybavena prostředky pro další zpracování těchto informací. Toto přiřazení "kód - význam" musí být provedeno v terminálu nebo uživatelské stanici.
		I	
		L	
		F	
		MI	
		ML	
		MF	

<b>WID/ Data1</b>	PAR	Konst	Datový údaj č. 1. Celé číslo v rozsahu 0 až 65535 jehož význam určuje aplikátor. Jedná-li se ale o alarm/hlášení týkající se nějaké databázové proměnné (např. překročení mezí), uvádí se na tomto místě jméno příslušné proměnné.
		I	
		L	
		F	
		MI	
		ML	
		MF	

Data2	PAR	Konst	Datový údaj č. 2. Celé číslo typu $\mathbb{L}$ (rozsah cca -2000000000 až +2000000000) jehož význam určuje programátor.
		I	
		L	
		F	
		MI	
		ML	
		MF	

Zdroj	PAR	Konst	Celočíselný údaj, charakterizující místo výskytu hlášení v procesní stanici. Význam a způsob kódování není přesně stanoven, lze použít např. $1000 \cdot \text{číslo\_procesu} + \text{číslo\_řádku}$ nebo jakýkoli jiný způsob. Hodnota se ukládá do položky <code>Segment</code> hlášení (viz. kap. <i>DTE - Editor formátovacích řetězců provozního deníku</i> ).
		I	
		L	
		F	
		MI	
		ML	
		MF	

**Příklad**

```
Report Test.1, 1000, 04000, 0, 0
```

Přejde-li bit č. 1 proměnné `Test` z 0 do 1, bude protokolováno hlášení s kódem 1000. Tomuto hlášení přisoudíme význam např. "proměnná Data1 porušila mez určenou Data2 (0: dolní, 1: horní)". Pak tedy víme, že se jedná o proměnnou s kódem 04000 a porušení dolní meze. Místo výskytu hlášení není specifikováno. Čas výskytu události a číslo procesní stanice, na níž událost nastala jsou vždy doplněny automaticky.

**ReqDb**

Žádost o přenos databázové proměnné po síti DB-Net

**Popis**

Při každém spuštění modulu se přenáší žádaná proměnná buď směrem do vzdálené stanice (zápis) nebo směrem ven ze vzdálené stanice (čtení). Pokud je typ proměnné matice, lze přenášet jak celou matici, tak i její výřez.

Číslo stanice, na kterou nebo ze které se proměnná přenáší, je dáno číslem stanice, která je zdrojem proměnné a v modulu se nezadá.

Pokud se přenáší matice, lze přenášet jen část matice. Maximální velikost výřezu matice, kterou je modul schopen v současné verzi přenést je 240B.

Tabulka udává maximální velikost výřezu (počet řádků x počet sloupců) pro jednotlivé databázové typy:

Typ	poč.řádků x poč. sloupců
MI	120
ML	60
MF	60

Zadá-li se výřez větší, modul jej interpretuje jako chybu parametrů. Chybový kód se objeví v proměnné **Vložení**.

Přenos větší matice se musí rozdělit na části a postupně použít více modulů **ReqDb**.

**Parametry**

<b>Zapisovat</b>	PAR	Výběr	Nastavení příznaku znamená zápis proměnné, v opačném případě se jedná o čtení.
------------------	-----	-------	--

<b>Proměnná</b>	IN/OUT	I	Pokud se přenáší matice, lze přenášet jen část matice. Tento "výřez" pak začíná na pozici dané řádkem a sloupcem a má rozměry zadané parametry <b>Řádků</b> a <b>Sloupců</b> .
		L	
		F	
		MI	
		ML	
		MF	

<b>Řádků</b>	PAR	Konst	Počet řádků přenášené části matice.
--------------	-----	-------	-------------------------------------

<b>Sloupců</b>	PAR	Konst	Počet sloupců přenášené části matice.
----------------	-----	-------	---------------------------------------

<b>Vložení</b>	OUT	I	Proměnná - chybový kód vložení požadavku. Lze zadat NONE. Popis kódů viz. dodatek "Stavy sériové komunikace".
		NONE	

<b>Stav</b>	IN/OUT	I	Proměnná - stav přenosového požadavku. Lze zadat NONE. Popis kódů viz. dodatek "Stavy sériové komunikace".
		NONE	

**Příklad**

ReqDb 0x0001, Stav[2, 3], 1, 2, Succ, Res

Jde o zápis výřezu matice **Stav** na stanici danou číslem stanice databázové proměnné **Stav**. Výsledek vložení požadavku je v proměnné **Succ**, stav požadavku je v proměnné **Res**. V tabulce je znázorněn výřez matice.

řádek\sloupec	0	1	2	3	4	5
0						
1						
2						
3						
4						

<b>ReqDbDir</b>	<b>Žádost o přímý přenos databázové proměnné po síti</b>
-----------------	--

**Popis**

Přímý přenos znamená, že v databázi vlastní stanice nemusí být kopie vzdálené proměnné, jak je tomu u modulů DbReq a ReqDb. Číslo vzdálené stanice se nebere z definice zdroje proměnné v databázi, ale určuje se explicitně parametrem modulu. Lze tedy přenášet vlastní lokální proměnnou do libovolné jiné vzdálené proměnné nebo opačně vzdálenou libovolnou proměnnou do lokální. Navíc lze parametry přenosu zadávat v proměnných a měnit je tak za běhu.

Při každém spuštění modulu se přenáší žádaná proměnná buď směrem do vzdálené stanice (zápis) nebo směrem ze vzdálené stanice (čtení). Pokud je typ proměnné matice, lze přenášet jak celou matici, tak i její výřez.

Typ lokální a vzdálené proměnné musí být buď shodný, nebo jedna proměnná může být matice a druhá jednoduchá přičemž musí být shodný jejich básový typ. Povolené kombinace jsou tedy např. I-I, I-MI, MI-I apod. Příklad zakázaných kombinací je I-L, MI-F apod.

Pokud se přenáší matice, lze přenášet také jen část matice. Zapisovaný výřez matice se musí celý "vejít" do vzdálené matice, stejně tak čtený výřez matice se musí celý "vejít" do lokální matice. Maximální velikost výřezu matice, kterou je modul schopen v současné verzi přenést je 240B.

Tabulka udává maximální velikost výřezu (počet řádků x počet sloupců) pro jednotlivé databázové typy:

Typ	poč.řádků x poč. sloupců
MI	120
ML	60
MF	60

Zadá-li se výřez větší, modul jej interpretuje jako chybu parametrů. Chybový kód se objeví v proměnné vložení.

Přenos větší matice se musí rozdělit na části a postupně použít více modulů **ReqDbDir**.

**Parametry**

Většinu parametrů lze zadat dvojím způsobem: buď jako číslo - odpovídající proměnná pak musí být zadána jako NONE, nebo jako proměnnou - v tom případě číselný parametr nemá význam.

<b>Zapisovat</b>	PAR	Výběr	Nastavení příznaku znamená zápis proměnné, v opačném případě se jedná o čtení.
------------------	-----	-------	--

<b>Stanice</b>	PAR	Konst	Číslo vzdálené stanice.
----------------	-----	-------	-------------------------

<b>hStanice</b>	IN	I	Proměnná - číslo vzdálené stanice. Je-li NONE, bere se hodnota z Stanice.
		NONE	

<b>WID</b>	PAR	Konst	WID vzdálené proměnné. Typ vzdálené proměnné musí být shodný s typem lokální proměnné.
------------	-----	-------	--

<b>hWID</b>	IN	I	Proměnná - WID vzdálené proměnné. Je-li NONE, bere se hodnota z WID.
		NONE	

<b>Typ</b>	PAR	Konst	Databázový typ vzdálené proměnné.
------------	-----	-------	-----------------------------------

Typ	Číslo
I	0
L	1
F	2

MI	3
ML	4
MF	5

<b>hTyp</b>	IN	I	Proměnná - databázový typ vzdálené proměnné. Je-li NONE, bere se hodnota z Typ.
		NONE	
<b>Řádek</b>	PAR	Konst	Počáteční řádek výřezu vzdálené matice.
<b>hŘádek</b>	IN	IN	Proměnná - počáteční řádek výřezu vzdálené matice. Je-li NONE, bere se hodnota z Řádek.
		NONE	
<b>Sloupec</b>	PAR	Konst	Počáteční sloupec výřezu vzdálené proměnné.
<b>hSloupec</b>	IN	I	Proměnná - počáteční sloupec výřezu vzdálené matice. Je-li NONE, bere se hodnota z Sloupec.
		NONE	
<b>Řádků</b>	PAR	Konst	Počet řádků výřezu vzdálené matice.
<b>hŘádků</b>	IN	I	Proměnná - počet řádků výřezu vzdálené matice. Je-li NONE, bere se hodnota z Řádků.
		NONE	
<b>Sloupců</b>	PAR	Konst	Počet sloupců výřezu vzdálené matice.
<b>hSloupců</b>	IN	I	Proměnná - počet sloupců výřezu vzdálené matice. Je-li NONE, bere se hodnota z Sloupců.
		NONE	
<b>LokProm</b>	IN/OUT	I	Lokální databázová proměnná libovolného typu shodného s typem vzdálené proměnné.
		L	
		F	
		MI	
		ML	
		MF	
<b>LokŘ</b>	PAR	Konst	Počáteční řádek výřezu lokální matice.
<b>hLokŘ</b>	IN	I	Proměnná - počáteční řádek výřezu lokální matice. Je-li NONE, bere se hodnota z LokŘ.
		NONE	
<b>LokSI</b>	PAR	Konst	Počáteční sloupec výřezu lokální matice.
<b>hLokSI</b>	IN	I	Proměnná - počáteční sloupec výřezu lokální matice. Je-li NONE, bere se hodnota z LokSl.
		NONE	
<b>Vložení</b>	OUT	I	Proměnná - chybový kód vložení požadavku. Lze zadat NONE. Popis kódů viz. dodatek "Stavy sériové komunikace".
		NONE	
<b>Stav</b>	IN/OUT	I	Proměnná - stav přenosového požadavku. Popis kódů viz. dodatek "Stavy sériové komunikace".
		NONE	

### Příklad

ReqDbDir 0x0000, 7, NONE, 7002, NONE, 2, NONE, 0, NONE, 0, NONE, 1, NONE, 1, NONE, S7\_Tlak, 0, NONE, 0, NONE, NONE, NONE



Jde o čtení jednoduché proměnné typu  $\mathbb{F}$  s WIDem 7002 ze stanice číslo 7 do proměnné S7\_Tlak. Výsledek vložení požadavku ani stav požadavku se v aplikaci nekontrolují.

<b>ReqTime</b>	Přenos času po síti
----------------	---------------------

**Popis**

Při každém spuštění modulu se přenáší čas buď směrem do vzdálené stanice (zápis) nebo směrem ven ze vzdálené stanice (čtení).

Čas musí být udržován v databázové proměnné typu L ve formátu systému DB-Net (např. modulem **GetTime**).

**Parametry**

<b>Zapisovat</b>	PAR	Výběr	Nastavení příznaku znamená zápis času do sítě, v opačném případě se jedná o čtení ze sítě.
------------------	-----	-------	--

<b>Stanice</b>	PAR	Konst	Číslo stanice, na kterou se zapisuje nebo ze které se čte. Lze zadat číslo 0..31.
----------------	-----	-------	---

<b>Čas</b>	IN/OUT	L	Databázová proměnná, ze které se získá čas při zápisu, nebo do které se načte čas při čtení. V případě úspěšného zápisu se nastaví na daný čas RTC obvod (systémový čas) vzdálené stanice.
------------	--------	---	--

<b>Vložení</b>	OUT	Bit	Proměnná - chybový kód vložení požadavku. Lze zadat NONE. Popis kódů viz. dodatek "Stavy sériové komunikace".
		NONE	

<b>Stav</b>	IN/OUT	Bit	Proměnná - stav přenosového požadavku. Lze zadat NONE. Popis kódů viz. dodatek "Stavy sériové komunikace".
		NONE	

**Příklad**

```
Proc00(1s): GetTime    TIME, TIMEm, TIMEd
```

```
Proc01(30s): ReqTime  0x0001, 7, TIME, Succ, Res
```

Jde o zápis času na stanici číslo 7 periodicky každých 30s. Čas se získá z proměnné TIME, kterou udržuje modul **GetTime** v procesu s periodou 1s. Výsledek vložení požadavku je v proměnné Succ, stav požadavku je v proměnné Res.

**Popis**

Modul umožňuje měřit otáčky 8 kanálů. Otáčky se měří pomocí impulzních snímačů, jejichž signály jsou přivedeny na V/V modul rychlých digitálních vstupů **AD-FDI8** řídicího systému ADiS (pozor na fyzické umístění modulu v sestavě, viz popis vlastního V/V modulu). Měření má krátkou dobu odezvy, provádí se přepočtem přes periodu otáčení.

Modul lze umístit do libovolného procesu.

Měřicí rozsah je zhruba 40 až 100000 pulzů za minutu. Perioda pulzů se měří s rozlišením od 51.2μs až do 400ns pomocí 16-ti bitového čítače (15 bitů pro vlastní měření). Modul volí rozlišení automaticky dle zadaných minimálních otáček, tak aby se zaručila maximální možná přesnost měření.

Pokud se liší měřené otáčky příp. počty impulzů na jednu otáčku u jednotlivých kanálů, liší se také přesnosti měření kanálů. Necht  $n$  jsou okamžité otáčky a  $k$  je počet impulzů na jednu otáčku. Největší přesnost měření má potom ten kanál, jehož násobek  $n \times k$  je nejvyšší.

**Pro dosažení maximální přesnosti je vhodné stanovit minimální otáčky kanálů pokud možno co nejvyšší. Zároveň není vhodné měřit otáčky více kanálů, jejichž násobky  $n \times k$  se příliš liší (více než o řád).**

Pro každý kanál se zadávají minimální měřené otáčky  $n_{\min}$  a počet impulzů  $k$  na jednu otáčku. Pokud se minimální otáčky zadají nulové, modul dosadí implicitní hodnotu 5060 ot./min. Ze všech osmi kanálů se vybere minimum násobku  $n_{\min} \times k$ .

Následující tabulka udává rozlišení pro zadaný minimální násobek (rozlišení platí vždy od hodnoty násobku výše):

$n_{\min} \times k$ [min <sup>-1</sup> ]	Rozlišení
od 5036	400ns
2518	800ns
1259	1.6μs
630	3.2μs
315	6.4μs
158	12.8μs
79	25.6μs
40	51.2μs

Otáčky, nižší než je stanovená mez  $n_{\min} \times k$ , se vyhodnotí jako nulové.

**Pozor:** Modul spotřebovává vyčerpatelný systémový prostředek (hardwareový časovač procesoru). Při vyčerpání prostředků tohoto typu může dojít ke konfliktu s jinými funkčními moduly, které tyto prostředky rovněž využívají. Určení povolených a zakázaných kombinací těchto modulů závisí na typu použité procesní stanice, proto před současným použitím jiných modulů s tímto varováním kontaktujte technickou podporu.

**Parametry**

Pro každý z osmi kanálů se zadávají dva parametry:

Režim	PAR	Výběr	Režim kanálu.
Skládá se ze tří hodnot:			
Imp./ot.	Konst	Počet impulzů na jednu otáčku. Pokud se zadá 0, modul daný kanál vůbec neměří.	
Vzorků	Konst	Počet vzorků klouzavého průměru (1..8). Délky měřených pulzů lze filtrovat klouzavým průměrem z maximálně osmi vzorků.	
Náběžná	Výběr	ANO=modul reaguje na náběžnou hranu pulzu. NE=modul reaguje na sestupnou hranu.	
Min	PAR	Konst	Minimální otáčky. Minimální otáčky spolu s počtem impulzů

			na jednu otáčku určují největší měřenou periodu.
--	--	--	--

Zadá-li se do parametru 0, modul dosadí implicitní hodnotu 5060 ot./min.

Naměřené nižší otáčky, než je zadaná minimální hodnota, se vyhodnotí jako nulové.

Po definici jednotlivých kanálů následuje parametr:

<b>Otáčky</b>	OUT	MF	Naměřené otáčky [ot./min.]. Je to databázová maticová proměnná rozměru [8,1]. Každému kanálu odpovídá jeden řádek matice.
---------------	-----	----	---

### Příklad

RPM0x8804, 200.0, 0x8802, 700.0, 0x8801, 1500.0, 0x8800, 0.0, 0x8800, 0.0, 0x8800, 0.0, 0x8800, 0.0, 0x8800, 0.0, Otacky

Modul měří otáčky na prvních třech kanálech, ostatní kanály se neměří.

Stanovení rozlišení měření:

Kanál	Min.otáčky	Poč. impulzů	nzk
1	200	4	800
2	700	2	1400
3	1500	1	1500

Minimum násobku je 800, čemuž odpovídá rozlišení 3.2μs.

<b>RS</b>	Klopný obvod typu RS
-----------	----------------------

**Popis**

Hodnota výstupu v závislosti na vstupech je dána tabulkou:

Reset	Set	Výstup
1	0	0
0	1	1
0	0	předchozí stav
1	1	nedefinováno - 0

**Parametry**

<b>Set</b>	IN	Bit	Bit databázové proměnné, který vstupuje do RS klopného obvodu jako signál <i>Set</i> .
------------	----	-----	--

<b>Reset</b>	IN	Bit	Bit databázové proměnné, který vstupuje do RS klopného obvodu jako signál <i>Reset</i> .
--------------	----	-----	--

<b>Výstup</b>	OUT	Bit	Bit výstupní databázové proměnné, který vystupuje z RS klopného obvodu jako signál <i>Výstup</i> .
---------------	-----	-----	--

**Příklad**

RS RS\_par.0, RS\_par.1, RS\_Par.2

Signál *Set* se bere z 0. bitu, signál *Reset* z 1. bitu proměnné RS\_par a výstup je na bitu č. 2 téže proměnné.

**RS485Rx**

Přepnutí RS485 na příjem (uživatelská komunikace)

**Popis**

Modul přepne kanál RS485 na příjem, tj. vypne budič linky. Používá se v případech, kdy je potřeba řídit směr komunikace "ručně" namísto automatického řízení směru modulem ComInit.

Modul ComInit provádí odbuzení linky (přepnutí na příjem) v přerušení od posledního vyslaného znaku. Na některých komunikačních obvodech (momentálně u modulů řídicího systému ADiS AD-UART4 a AD-UART) nastává problém v tom, že přerušení od vyslaného znaku obvod generuje okamžitě po zapsání znaku do vysílacího registru a nikoliv až po dokončení vysílání znaku. Modul ComInit tak provede odbuzení dřív, než se stačí poslední znak odvysílat. Poslední znak se k příjemci již nedostane. Tento HW problém se řeší tak, že se modulu ComInit zakáže řízení směru a místo toho se použijí moduly RS485Rx a RS485Tx. Modulem RS485Tx se přepíná na vysílání. Provádí se to obvykle před zápisem vysílaného rámce do vysílacího bufru modulem ComWrite. Na příjem se přepíná modulem RS485Rx. Neprovádí se to přímo v přerušení od posledního znaku, ale přepnutí se oproti tomuto přerušení musí zpoždit o čas potřebný na odvysílání znaku. Velikost zpoždění je závislá na komunikační rychlosti a celkovém počtu vysílaných bitů na jeden znak (délka slova + startbit + paritní bit + stopbity). Zpoždění se provádí např. použitím modulů pro timeout (Tmo, TmoStart, ...).

*Pozn.: Druhou možností jak řešit tento HW problém je řídit směr modulem ComInit a vysílané rámce zvětšit o jeden znak navíc. Tento znak navíc modul již nestihne odvysílat, takže se odvysílá pouze vlastní rámec. Nevýhodou této metody je to, že poslední znak modul někdy "nespolkne" celý, ale začátek znaku (startbit) se přeci jen na linku dostane. To protistrana může vyhodnotit jako chybu linky (parita, nepřišel stopbit, ...). Pokud je daný komunikační protokol na chyby linky nějak "citlivý", mohla by tato metoda působit problémy.*

**Parametry**

<b>ComInit</b>	PAR	Návěští	Návěští modulu ComInit.
----------------	-----	---------	-------------------------

**Příklad**

Komunikujeme na kanálu RS485. Přepínání směru linky budeme provádět ručně pomocí modulů RS485Rx a RS485Tx. Na vysílání budeme přepínat bezprostředně před zahájením vysílání rámce. Na příjem budeme přepínat se zpožděním v délce jednoho znaku, které odvodíme od přerušení od posledního vyslaného znaku. Parametry komunikační linky budou 9600 Bd, délka 8 bitů, sudá parita, jeden stopbit. Na jeden znak se tak bude vysílat 11 bitů: 1 startbit, 8 bitů znak, 1 paritní bit, 1 stopbit. Při rychlosti 9600 Bd bude odvysílání 11 znaků trvat cca 1.15 ms. Pro zpoždění použijeme tedy timeout 2 ms.

**ProcINIT:**

```
:01000 SubInst 100 Instance podprog. přeruš. od přij. znaku
:01001 SubInst 101 Instance podprog. přeruš. od vysl. znaku
:02000 ComInit 0x0001, 0, 9600, 8, Sudá, 1, :01000, :01001, :NONE, :NONE,
PrijsBuf, VyslBuf
:01002 SubInst 102 Instance podprog. přeruš. od timeoutu
:03000 Tmo :01002, 1, 2 Definice kanálu timeoutu
```

**Proc00:**

```
Periodický proces 5s - vysílání
RS485Tx :02000 Přepnutí na vysílání
... vysílání rámce pomocí modulu ComWrite ...
```

**Lib101:**

```
Podprogram přer. od vyslaného znaku
TmoStart :03000, 2 Nastartování TMO 2 ms
```

**Lib102:**

```
Přerušení od timeoutu
RS485Rx :02000 Přepnutí RS485 na příjem
```

**RS485Tx**

Přepnutí RS485 na vysílání (uživatelská komunikace)

**Popis**

Modul přepne kanál RS485 na vysílání, tj. zapne budič linky. Používá se v případech, kdy je potřeba řídit směr komunikace "ručně" namísto automatického řízení směru modulem ComInit, viz popis modulu RS485Rx.

**Parametry**

<b>ComInit</b>	PAR	Návěští	Návěští modulu ComInit.
----------------	-----	---------	-------------------------

**Příklad**

Viz popis modulu RS485Rx.

---

## SeqStep

Řízení sekvence

## Popis

Modul je určen pro řízení sekvence kroků. Každému kroku je přiřazen jeden bit v proměnných **Krok** a **Přechod**. Hlavním výstupem modulu je proměnná **Krok**, ve které modul nastavuje vždy pouze jeden bit, což je právě prováděný krok. Na jednotlivé bity této proměnné se obvykle navazují další moduly nebo celé větve programu a tyto potom realizují činnosti pro jednotlivé kroky. Do dalšího kroku lze přejít, pokud je nastaven bit odpovídající právě prováděnému kroku v proměnné **Přechod**. Parametrem **Maska** lze vybrat, které kroky z celkových šestnácti se budou provádět. Ostatní kroky se přeskakují. Sekvence se startuje náběžnou hranou vstupního bitu **Běh**. Nastavením bitu do "0" se sekvence přeruší a **Krok** se vynuluje. V sekvenci lze pokračovat opětovným nastavením bitu do "1". Byla-li již sekvence ukončena, spustí se tímto způsobem znovu.

Přechodem z posledního kroku dál se sekvence ukončí. Konec je signalizován bitem **Konec**, proměnná **Krok** se vynuluje. Sekvenci lze znovu spustit nastavením bitu **Reset**, anebo se musí nastavit alespoň na jeden běh modulu bit **Běh** do "0" a potom opět do "1". Bitem **Reset** lze sekvenci kdykoli přerušit a spustit znovu.

Změnit krok je možné ještě jiným způsobem a to zápisem do proměnné **Krok**. V takovém případě modul akceptuje tuto novou hodnotu kroku, pokud je daný krok povolen v parametru **Maska**. Nově zvolený krok se spustí, přičemž přechod do dalšího kroku je umožněn až v dalším běhu modulu.

Modul umožňuje realizovat maximálně 16 kroků. Je-li potřeba více kroků, lze moduly **SeqStep** zřetěžit. Provede se to tak, že do parametru **Konec** prvního modulu a parametru **Běh** následujícího modulu se zadá stejný bit databázové proměnné.

## Parametry

<b>Běh</b>	IN	Bit	Aktivace (start) sekvence. Náběžnou hranou se startuje sekvence. Nastavením do "0" se sekvence přeruší. Po ukončení celé sekvence lze sekvenci spustit znovu náběžnou hranou bitu.
		NONE	NONE má význam, jakoby byl bit trvale v "1".
<b>Reset</b>	IN	Bit	Reset sekvence - spuštění znovu od začátku. Modul tento bit poté vynuluje. Modul reaguje na tento bit, pouze pokud je aktivován bitem <b>Běh</b> . Bit <b>Reset</b> lze nastavit i v neaktivním stavu modulu, modul však na tento bit zareaguje až při své aktivaci.
		NONE	NONE má význam, jakoby byl bit trvale v "0".
<b>Maska</b>	IN	Výběr	Výběr, které kroky se budou provádět. Maximum je 16 kroků odpovídající 16-ti bitům. Je-li bit v "1", tak se daný krok bude provádět, je-li bit v "0", tak se daný krok bude přeskakovat.
		I	
		MI	
<b>Přechod</b>	IN	I	Povolení přechodu do dalšího kroku. Nastavením jednotlivých bitů do "1" se modulu povoluje přechod z odpovídajících kroků do kroků následujících. Číslo bitu odpovídá číslu bitu kroku, ze kterého se přechází. Přechodem z posledního kroku dál se sekvence ukončí.
		MI	
		Výběr	
<b>Krok</b>	IN/OUT	I	Modul v proměnné nastavuje vždy pouze jeden bit, což je právě prováděný krok. Je-li sekvence ukončena nebo přerušena bitem <b>Běh</b> , tak modul proměnnou nuluje. Zápisem do této proměnné mimo modul lze "vnutit" modulu nový krok. Nově zvolený krok musí být povolený v parametru <b>Maska</b> a modul musí být aktivní ( <b>Běh</b> = "1").
<b>Konec</b>	OUT	Bit	Příznak ukončení sekvence. Po ukončení modul nastaví bit na "1". Po novém spuštění sekvence modul bit vynuluje.
		NONE	



**Příklad**

Realizujeme cyklickou sekvenci, která se skládá ze tří kroků. Cykličnost se zajistí zadáním stejného bitu @Res do parametrů Reset a Konec.

```
SeqStep      @Beh, @Res, 0x0007, Prech, Krok, @Res
If           Krok.0
...činnost v kroku 0. Přejchod do dalšího kroku se provede nastavením bitu Prech.0.
EndIf
If           Krok.1
...činnost v kroku 1. Přejchod do dalšího kroku se provede nastavením bitu Prech.1.
EndIf
If           Krok.2
...činnost v kroku 2. Přejchod do dalšího kroku se provede nastavením bitu Prech.2.
EndIf
```

---

**SetPwd**

Nastavení přístupového hesla LcdShellu

**Popis**

Modul je určen pro nastavování (změnu) přístupového hesla do LcdShellu.

**Parametry**

<b>Uživatel</b>	IN	Konst	Číslo uživatele, jemuž se mění heslo. Číslo se zadává v rozsahu 1 až 3 a odpovídá číslování při zadávání jmen a hesel v aplikaci LcdShellu.
		I	
		MI	
<b>Heslo</b>	IN	Konst	Zapisované heslo - číslo v rozsahu 0 až 65535.
		I	
		MI	
<b>Zapiš</b>	IN	Bit	Příkaz pro zápis hesla. Nastavením do "1" se provede zápis hesla a modul následně bit vynuluje. Zadá-li se do parametru NONE, tak se provádí zápis v každém běhu modulu.
		NONE	

**Příklad**

Nastavením bitu `Set.0` do "1" provedeme změnu hesla. Číslo uživatele zadáváme v proměnné `User`, hodnotu nového hesla v proměnné `Password`.

```
SetPwd      User, Password, Set.0
```

<b>SetTime</b>	Nastavení času procesní stanice
----------------	---------------------------------

**Popis**

Modul nastavuje čas procesní stanice. Čas z databázové proměnné `Čas` se zapíše do RTC obvodu procesní stanice. Modul testuje příznak proměnné v databázi, zda do ní bylo zapsáno. Zápis času modul provede pouze v případě, že je příznak nastaven a poté příznak vynuluje.

**Parametry**

Čas	IN	L	Nový čas v DB-Net formátu
-----	----	---	---------------------------

**Příklad**

`SetTime`                      `ZadanyCas`

Do proměnné `ZadanyCas` se zadává čas, například na LCD terminálu nebo na jiném místě v programu aplikace. Tento čas se po každé změně zapíše do RTC stanice.

<b>ShortCut</b>	Regulace teploty vratu na základě simulace zkratu
-----------------	---

Modul je určen ke zlepšení regulace vratu v kotelnách (vrat je přívod topné vody zpět do kotle z vytápěné soustavy).

**Popis**

Pokud v kotelně chybí samostatně realizovaný tepelný zkrat, který je samostatně regulovatelný, je třeba hlídat výstupní teplotu vody do topných větví. Tepelný zkrat slouží k tomu, aby rozdíl mezi teplotou topné vody z kotle do větve a vody přiváděné zpět do kotle (vratu) nepřekročil určitou mez, jinak by byl kotel nadměrně namáhán. Pomocí tepelného zkratu se dá mísit přiváděná voda zpátky z větve s ohřátou vodou do větve a zvyšovat tak uměle teplotu vratu.

Tento modul se používá v případech, kdy není fyzicky realizovaný tepelný zkrat a rozdíl teplot je tak třeba regulovat v mezích pomocí regulace (snižováním) teploty do větve. Modul uzpůsobuje žádanou teplotu do větve podle vratu. V normálním stavu modul ponechá žádanou teplotu beze změny (žádaná teplota vody je např. dána ekvitemní křivkou). Pokud rozdíl teplot narůstá na kritickou mez, modul začne snižovat žádanou teplotu. Pokud musí modul po dlouhou dobu "držet žádanou teplotu níž", znamená to, že daný počet kotlů nestačí k vytopení soustavy a je potřeba připnout další kotel.

**Určení žádané teploty vratu**

Nelze přesně stanovit předpis, jak určit správnou hodnotu, protože závisí na tom o kolik stupňů je kotel nebo skupina kotlů schopná ohřát přiváděnou vodu. To většinou závisí na konkrétních podmínkách technologie. Z našich zkušeností se osvědčilo určovat požadovanou teplotu vratu o 8 až 12 C° menší, než je žádaná teplota výstupní vody z kotle do větve.

**Princip činnosti**

Do modulu vstupují dvě žádané hodnoty - teplota topné vody a teplota vratu. Výstupem je modifikovaná žádaná teplota topné vody, která je většinou žádanou hodnotou pro regulaci teploty z kotle do větve.

Je-li žádaná hodnota  $VratZad$  vratu větší než skutečně měřená hodnota, je snížena hodnota  $OTVZad$  o hodnotu

$$(VratZad - VratMěř) * K$$

Je-li hodnota vratu větší než žádaná, je naopak hodnota žádané teploty ve větvi zvýšena o

$$(VratMěř - VratZad) * K$$

Programátor pak musí regulovat na hodnotu

$$\min(OTVZad, OTV)$$

Modul tedy provádí pouze

$$OTV = OTVZad + (VratMěř - VratZad) * K$$

Rozdíly  $OTVZad$  a  $OTV$  lze vyhodnocovat a na jejich základě připojovat nebo odpojovat další kotle.

**Parametry**

<b>VratZad</b>	IN	F	Vstupní databázová proměnná, která obsahuje žádanou hodnotu teploty vratu.
		MF	
<b>VratMěř</b>	IN	F	Vstupní databázová proměnná, která obsahuje měřenou hodnotu teploty vratu.
		MF	

<b>K</b>	IN	Konst	Koeficient zesílení regulace vratu (zpravidla volen na hodnotu 1).
		F	
		MF	

<b>OTVZad</b>	IN	F	Vstupní databázová proměnná, která obsahuje doporučenou teplotu z ekvitermu.
		MF	

<b>OTV</b>	OUT	F	Výstupní databázová proměnná, která obsahuje modifikovanou žádanou teplotu do větve.
		MF	

**Příklad**

Let                      VratZadana = OTVZadana - 8.0

ShortCut              VratZadana,VratMerena,K,OTVZadana, OTVModif

Žádaná teplota vratu je o 8 stupňů menší než žádaná teplota topné vody.

<b>Stand_in</b>	Sledování provozních hodin dvou zařízení, která se pravidelně střídají
-----------------	--

**Popis**

Modul sleduje provoz zařízení A a B, která nikdy nepracují najednou. U každého zařízení jsou sledovány dva typy poruch (I. a II.). Jakmile je jedno zařízení v poruše, je spuštěno druhé. Spouštění a vypínání zařízení modul ovládá pomocí proměnných VýstupA a VýstupB. Požadavek běhu některého zařízení se nastavuje proměnnou Start. Modul vybere zařízení a do výstupní proměnné VýstupA nebo VýstupB je uložena "1" pro chod příslušného zařízení. Je-li proměnná ChybaA\_I nebo ChybaA\_II v "1", je zařízení A v poruše a poběží zařízení B. Je-li proměnná ChybaB\_I nebo ChybaB\_II v "1", je zařízení B v poruše a poběží zařízení A. Pro každé zařízení se sledují provozní hodiny. Ty jsou uloženy v matici o velikosti  $[n \times 2]$  typu ML. Zařízení A přísluší řádek ŘádekA této matice. Zařízení B přísluší řádek ŘádekB.

Při použití modulu **HourRun** se typicky používá stejná matice, takže pro aplikaci může být jedna matice provozních hodin. Proměnná Den nastavuje den, kdy se střídají zařízení, Čas je čas v sekundách od začátku dne, kdy se střídají zařízení.

**Typy poruch**

Zařízení podporuje dva typy poruch (I. a II.). Jejich logický význam je v rukou programátora. Většinou má porucha typu I. význam vážné poruchy (např. zaplavení kotelny) a typ II. je porucha, která není nebezpečná.

**Parametry**

<b>Start</b>	IN	Bit	Je-li "1", je zařízení v chodu. Je-li "0", je zařízení vypnuto.
<b>VýstupA</b>	OUT	Bit	Výstup - udává, zda má být zařízení A v chodu.
<b>VýstupB</b>	OUT	Bit	Výstup - udává zda má být zařízení B v chodu.
<b>ChybaA_I</b>	IN	Bit	Je-li "1", je zařízení A v poruše typu I.
<b>ChybaA_II</b>	IN	Bit	Je-li "1", je zařízení A v poruše typu II.
<b>ChybaB_I</b>	IN	Bit	Je-li "1", je zařízení B v poruše typu I.
<b>ChybaB_II</b>	IN	Bit	Je-li "1", je zařízení B v poruše typu II.
<b>ProvozHod</b>	IN/OUT	ML	Databázová proměnná rozměru $[n, 2]$ pro sledování provozních hodin. Nultý sloupec je počet hodin, první sloupec je počet sekund (pomocný sloupec pro vnitřní účely modulu).
<b>ŘádekA</b>	PAR	Konst	Řádek v matici ProvozHod pro sledování provozních hodin zařízení A.
<b>ŘádekB</b>	PAR	Konst	Řádek v matici ProvozHod pro sledování provozních hodin zařízení B.
<b>Den</b>	PAR	Výběr	Maska dnů v týdnu, ve kterých se mohou střídát zařízení podle počtu provozních hodin. Pokud není důvod ke střídání - vypnuté zařízení má více provozních hodin, zařízení se nestřídají. Čas střídání je v Čas.
<b>Čas</b>	PAR	Konst	Počet sekund od počátku dne. V tomto čase se střídají zařízení.

**Příklad**

```
Stand_in      Start.0, A.0, B.0, EA.0, EA.1, EB.0, EB.1, T_Run,  
              0, 1, 0x0001, 0
```

Zařízení se střídají o půlnoci z neděle na pondělí (maska = 0x0001, čas = 0). Provozní hodiny zařízení jsou v matici T\_Run na řádcích 0 a 1.

<b>StartType</b>	Údaje o startu systému
------------------	------------------------

**Popis**

Získá údaje o tom, jakým způsobem byla aplikace rozběhnuta. Umožňuje určit, zda se jednalo o studený či teplý start, případně zda byl start vyvolán hlídacím časovačem, v případě uživatelského hlídacího časovače také kterým.

Modul se zpravidla vkládá do procesu INIT, ale může být vložen i v kterémkoliv jiném procesu.

**Parametry**

Druh	OUT	I	Do proměnné předané za tento parametr se uloží hodnota dle následující tabulky, určující druh startu systému.							
		MI								
		NONE								
				<table><tr><th>Hodnota</th><th>Význam</th></tr><tr><td>0</td><td>První start po zavedení aplikace. Všechny proměnné byly inicializovány včetně neinicializovaných, které byly vynulovány.</td></tr><tr><td>1</td><td>Studený start aplikace (např. po zjištěném narušení obsahu RAM). Všechny proměnné byly inicializovány včetně neinicializovaných, které byly vynulovány.</td></tr><tr><td>2</td><td>Teplý start aplikace. Inicializovány (popř. vynulovány) byly pouze proměnné s nastaveným příznakem “Inicializace při teplém startu”.</td></tr></table>	Hodnota	Význam	0	První start po zavedení aplikace. Všechny proměnné byly inicializovány včetně neinicializovaných, které byly vynulovány.	1	Studený start aplikace (např. po zjištěném narušení obsahu RAM). Všechny proměnné byly inicializovány včetně neinicializovaných, které byly vynulovány.
Hodnota	Význam									
0	První start po zavedení aplikace. Všechny proměnné byly inicializovány včetně neinicializovaných, které byly vynulovány.									
1	Studený start aplikace (např. po zjištěném narušení obsahu RAM). Všechny proměnné byly inicializovány včetně neinicializovaných, které byly vynulovány.									
2	Teplý start aplikace. Inicializovány (popř. vynulovány) byly pouze proměnné s nastaveným příznakem “Inicializace při teplém startu”.									
Nechceme-li zjišťovat druh startu systému, lze za tento parametr dosadit NONE.										

Hlídač	OUT	Bit	Výstupní bit, indikující, že systém nastartoval v důsledku restartu vyvolaného hlídacím časovačem. Hodnota 1 znamená restart vyvolaný hlídacím časovačem, 0 znamená jinou příčinu startu systému. Nechceme-li testovat vyvolání restartu hlídacím časovačem, lze za tento parametr dosadit NONE.
--------	-----	-----	--

Kanál	OUT	I	Do proměnné předané za tento parametr se uloží číslo uživatelského hlídacího časovače (totožné s parametrem Kanál funkčního modulu <b>Watchdog</b> ), který vyvolal restart systému, nebo hodnota 32 (neplatné číslo uživatelského hlídacího časovače). Nechceme-li zjišťovat číslo uživatelského hlídacího časovače, který vyvolal restart, lze za tento parametr dosadit NONE.
		MI	
		NONE	

*Pozn.: Hodnota v tomto parametru je platná pouze v případě, že hodnota parametru Hlídač je 1. V opačném případě je v parametru Kanál vrácena hodnota 32.*

*Došlo-li před restartem k vypršení limitu více uživatelských hlídacích časovačů, je vráceno nejnižší z jejich čísel.*

*Pokud byl restart vyvolán hlídacím časovačem (hodnota parametru Hlídač je 1), a přitom je v parametru Kanál vrácena hodnota 32, znamená to, že příčinou restartu bylo vypršení limitu některého systémového logického hlídacího časovače, nikoli uživatelského.*

Viz též principiální popis systému logických hlídacích časovačů v dodatku "Zabezpečení systému hlídacím časovačem (watchdog)".

**Příklad**

```
StartType NONE, @Watchdog, WhichWD
```

Uloží do databázového bitu @Watchdog příznak, že došlo k restartu od hlídacího časovače. Pokud ano, je možné podle hodnoty v proměnné WhichWD testovat, zda to byl některý z uživatelských hlídacích časovačů, a podle toho přijmout odpovídající opatření.



<b>Station</b>	Zjistí číslo stanice
----------------	----------------------

**Popis**

Získá číslo své vlastní stanice nastavené DIP přepínačem.

**Parametry**

<b>Stanice</b>	OUT	I	Jméno proměnné, do které se načítá číslo stanice.
----------------	-----	---	---

## StopWatch "Stopky" pro měření času procesoru

### Popis

Modul umožňuje měřit dobu provádění úseku kódu aplikace. Modul představuje klasické stopky, které se dají spouštět, zastavovat a nulovat. Měřit se dá na 16-ti kanálech. Měření se provádí tak, že na začátek měřeného úseku se vloží modul s nastaveným příznakem `Start` pro spuštění stopek a na konec se vloží modul s příznakem `Stop` a `Reset` pro zastavení a vynulování stopek. Tomuto ukončujícímu modulu se zadá jméno databázové proměnné, do které bude modul ukládat naměřený čas. Modul může také ještě vyhodnocovat minimální a maximální naměřený čas.

### Parametry

<b>Kanál</b>	PAR	Konst	Číslo kanálu (0..15). Kanály jsou vzájemně nezávislé. Lze použít jakoby 16 nezávislých stopek.
--------------	-----	-------	--

<b>Akce</b>	PAR	Výběr	Prováděná akce.
-------------	-----	-------	-----------------

<b>Start</b>	ANO	Spuštění stopek. Začne se měřit čas.
	NE	-

<b>Stop</b>	ANO	Zastavení stopek. Přestane se měřit čas. Lze využít i jako dočasné zastavení stopek pro přeskočení neměřené části kódu a dalším modulem s příznakem <code>Start</code> stopky zase spustit.
	NE	-

<b>Reset</b>	ANO	Nulování stopek. Vnitřní počítadlo se vynuluje.
	NE	-

Časy	OUT	F	Poslední naměřený čas [ms].	
		MF	Matice naměřených časů rozměru [3,1].	
			Řádek	Význam
			0	Poslední naměřený čas [ms].
			1	Nejdelší naměřený čas [ms].
			2	Nejkratší naměřený čas [ms].
		NONE		

Modulu na začátku měřeného úseku se zadává parametr `NONE`. Čas se ukládá až modulem na konci úseku, kterému se zadá jméno proměnné. Je-li mezi začátkem a koncem úseku vložen nějaký další modul `StopWatch` (se stejným kanálem), kterému se zadá jméno nějaké jiné proměnné, bude tato proměnná obsahovat "mezičas" v daném bodě úseku. Uvedme, že vložený modul uvnitř úseku nemusí provádět žádnou akci (`Akce = 0x0000`).

### Příklad

Změříme délku vykonávání procesu. Na začátek procesu vložíme modul pro nastartování stopek a na konci procesu vložíme modul pro zastavení a vynulování stopek a pro zapsání naměřených časů do matice `CasProc`.

```
StopWatch      0, 0x0001, NONE
```

```
... moduly, které proces obsahuje ...
```

```
StopWatch      0, 0x0006, CasProc
```

<b>StrFormat</b>	Formátování hodnoty (uživatelská komunikace).
------------------	---

**Popis**

Modul slouží k převodu hodnoty databázové proměnné do řetězce v zadaném formátu. Používá se v uživatelské komunikaci při vytváření vysílaného rámce.

**Parametry**

<b>Data</b>	OUT	MI	Výstupní řetězec - řádková matice $MI[1, n]$ , do které se zapisuje formátovaná hodnota. Jedna buňka matice odpovídá jednomu znaku řetězce.
-------------	-----	----	---

<b>IndexIn</b>	IN	Konst	Index v matici <i>Data</i> , od kterého se začínají zapisovat znaky.
		I	
		MI	

<b>IndexOut</b>	OUT	I	Index v matici <i>Data</i> posunutý o počet zapsaných znaků.
		NONE	

<b>TextMaska</b>	PAR	Řetězec	Textová maska. Text, který se do výstupního řetězce zapíše před formátovanou hodnotou. Řetězec může obsahovat řídicí znaky, viz dodatek "Řídicí znaky v řetězcích". Délka řetězce nesmí přesáhnout 64 znaků.
------------------	-----	---------	--

<b>Formát</b>	IN	Konst	Formát dat.
		I	
		MI	

Tabulka jednotlivých formátů:

(U některých formátů je popisováno pořadí bytů ve výstupním řetězci. Jednotlivé byty jsou označeny B0, B1, B2, .... B0 je nejnižší významový byte. Pořadí bytů je označeno BE - "big endian", nebo LE - "little endian")

Číslo	Typ	Popis	Typ proměnné Hodnota
-1	TextMaska	Do výstupního řetězce se zapíše pouze textová maska. Parametr <i>Hodnota</i> může být v tomto případě NONE.	NONE
0	BinInt	Binární integer (2 byty). Byty jdou v pořadí LE (B0, B1). Např. hodnota 0x1234 se zformátuje na (0x34, 0x12).	I (L)
1	BinIntBE	Binární integer (2 byty). Byty jdou v pořadí BE (B1, B0). Např. hodnota 0x1234 se zformátuje na (0x12, 0x34).	I (L)
2	BinLong	Binární long (4 byty). Byty jdou v pořadí LE (B0, B1, B2, B3). Např. hodnota 0x12345678 se zformátuje na (0x78, 0x56, 0x34, 0x12).	L (I)
3	BinLongBE	Binární long (4 byty). Byty jdou v pořadí BE (B3, B2, B1, B0). Např. hodnota 0x12345678 se zformátuje na (0x12, 0x34, 0x56, 0x78).	L (I)
4	Bin3	Binární celočíselná hodnota na 3 bytech. Byty jdou v pořadí LE (B0, B1, B2). Např. hodnota 0x123456 se zformátuje na (0x56, 0x34, 0x12).	L (I)
5	Bin3BE	Binární celočíselná hodnota na 3 bytech. Byty jdou v pořadí BE (B2, B1, B0). Např. hodnota 0x123456 se zformátuje na (0x12, 0x34, 0x56).	L (I)
6	BinFlt	Binární float (4 byty). Byty jdou v pořadí LE (B0, B1, B2, B3). Např. hodnota 123.456 (binárně 0x42F6E979) se zformátuje na (0x79, 0xE9, 0xF6, 0x42).	F

7	BinFltBE	Binární float (4 byty). Byty jdou v pořadí BE (B3, B2, B1, B0). Např. hodnota 123.456 (binárně 0x42F6E979) se zformátuje na (0x42, 0xF6, 0xE9, 0x79).	F
8	Hex	Hexadecimální číslo zapsané textově - malá písmena. Např. hodnota 0x89AB se zformátuje na "89ab".	I, L
9	HEX	Hexadecimální číslo zapsané textově - velká písmena. Např. hodnota 0x89AB se zformátuje na "89AB".	I, L
10	BCD	BCD kódované číslo. Jedna buňka matice představuje 2 číslice kódované BCD. Např. hodnota 12345 se zformátuje na (0x01, 0x23, 0x45).	I, L
11	Dec	Celé číslo zapsané textově - kladné i záporné. Např. hodnota -12345 se zformátuje na "-12345". Pomocí tohoto formátu lze zformátovat hodnotu typu $\mathbb{I}$ v rozsahu -32768 .. 32767 a hodnotu typu $\mathbb{L}$ v rozsahu -2147483648.. 2147483647.	I, L
12	UDec	Celé číslo zapsané textově - bez znaménka. Např. hodnota 12345 se zformátuje na "12345". Pomocí tohoto formátu lze zformátovat hodnotu typu $\mathbb{I}$ v rozsahu 0 .. 65535 a hodnotu typu $\mathbb{L}$ v rozsahu 0 .. 4294967295.	I, L
13	Flt_f	Float číslo zapsané textově - neexponenciální tvar. Např. hodnota 123.456 se implicitně zformátuje na "123.456000".	F
14	Flt_e	Float číslo zapsané textově - exponenciální tvar, malé písmeno exponentu. Např. hodnota 123.456 se implicitně zformátuje na "1.234560e+02".	F
15	Flt_E	Float číslo zapsané textově - exponenciální tvar - velké písmeno exponentu. Např. hodnota 123.456 se implicitně zformátuje na "1.234560E+02".	F
16	Flt_g	Float číslo zapsané textově (malé písmeno exponentu) - exponenciální nebo neexponenciální tvar se volí automaticky dle velikosti proměnné a zadané šířky řetězce. Např. hodnota $10^5$ se implicitně zformátuje na "100000.000000", zatímco hodnota $10^{10}$ se zformátuje na "1.000000e+10".	F
17	Flt_G	Float číslo zapsané textově (velké písmeno exponentu) - exponenciální nebo neexponenciální tvar se volí automaticky dle velikosti proměnné a zadané šířky řetězce. Např. hodnota $10^5$ se implicitně zformátuje na "100000.000000", zatímco hodnota $10^{10}$ se zformátuje na "1.000000e+10".	F
18	String	Textový řetězec. Do výstupního řetězce se přidá řetězec z databázové proměnné. Za parametr <i>Hodnota</i> musí být v tomto případě dosazena databázová matice $MI[n, m]$ , jejíž jeden řádek představuje výstupní textový řetězec. Číslo řádku se volí při zadávání parametru <i>Hodnota</i> . Jednomu znaku řetězce odpovídá jedna buňka matice. Textový řetězec je ukončen nulou, nebo je omezen počtem sloupců matice.	MI
19	BinString	Binární řetězec. Do výstupního řetězce se přidá řetězec z databázové proměnné. Za parametr <i>Hodnota</i> musí být v tomto případě dosazena databázová matice $MI[n, m]$ , jejíž jeden řádek představuje výstupní textový řetězec. Číslo řádku se volí při zadávání parametru <i>Hodnota</i> . Jednomu znaku řetězce odpovídá jedna buňka matice. Binární řetězec není ukončen nulou, nuly se v něm mohou volně vyskytovat. Je omezen pouze počtem sloupců matice.	MI
50	DD.MM	Datum ve formátu <i>den.měsíc</i> . Např. "30.08".	L

51	DD.MM.YY	Datum ve formátu <i>den.měsíc.rok</i> . Např. "30.08.00".	L
52	DD.MM.Y4	Datum ve formátu <i>den.měsíc.rok</i> (rok má 4 cifry). Např. "30.08.2000".	L
53	hh:mm	Čas ve formátu <i>hodiny:minuty</i> . Např. "09:38".	L
54	hh:mm:ss	Čas ve formátu <i>hodiny:minuty:sekundy</i> . Např. "09:38:07".	L
55	DMYhms	Datum a čas ve formátu <i>den.měsíc.rok hodiny:minuty:sekundy</i> . Např. "30.08.00 09:38:07".	L
56	DMY4hms	Datum a čas ve formátu <i>den.měsíc.rok hodiny:minuty:sekundy</i> (rok má 4 cifry). Např. "30.08.2000 09:38:07".	L
57	DD	Datum ve formátu <i>den</i> . Např. "30".	L
58	MM	Datum ve formátu <i>měsíc</i> . Např. "08".	L
59	YY	Datum ve formátu <i>rok</i> . Např. "00".	L
60	YYYY	Datum ve formátu <i>rok</i> (4 cifry). Např. "2000".	L
61	hh	Čas ve formátu <i>hodiny</i> . Např. "09".	L
62	mm	Čas ve formátu <i>minuty</i> . Např. "38".	L
63	ss	Čas ve formátu <i>sekundy</i> . Např. "07".	L

Ve sloupci "Typy proměnné Hodnota" tabulky jsou povolené typy databázové proměnné, kterou lze dosadit za parametr *Hodnota*. Je-li typ uveden v závorce, znamená to, že není pro daný formát přirozený a před formátováním se přetypuje na přirozený typ. Konkrétně, dosadí-li se u formátů *BinInt* a *BinIntBE* proměnná typu *L*, tak se berou pouze dolní 2 byty. Dosadí-li se u formátů *BinLong* a *BinLongBE* proměnná typu *I*, jsou horní 2 byty nulové. Dosadí-li se u formátů *Bin3* a *Bin3BE* proměnná typu *I*, je nejvyšší byte nulový.

SpecForm	IN	Konst	Specifikátor formátu. Ovlivňuje zarovnání a doplnění nulami.	
		I	Význam hodnot:	
		MI	Hodnota	Popis
			-1	Nic, implicitní nastavení
			32	Mezera Nezáporná čísla začínají mezerou, záporná znakem '-'. Má význam pouze u formátů <i>Dec</i> , <i>UDec</i> , <i>Flt_f</i> , <i>Flt_e</i> , <i>Flt_E</i> , <i>Flt_g</i> a <i>Flt_G</i> .
			43	VždyZnam Čísla se formátují vždy se znaménkem ('+' nebo '-'). Implicitně mají znaménko pouze záporná čísla. Má význam pouze u formátů <i>Dec</i> , <i>UDec</i> , <i>Flt_f</i> , <i>Flt_e</i> , <i>Flt_E</i> , <i>Flt_g</i> a <i>Flt_G</i> .
			45	Vlevo Zarovnání vlevo (implicitně se zarovnává doprava). Má význam pouze tehdy, když je hodnota parametru <i>MinDélka</i> větší než je délka zformátovaného řetězce. Nemá význam u formátů <i>BinInt</i> , <i>BinIntBE</i> , <i>BinLong</i> , <i>BinLongBE</i> , <i>Bin3</i> a <i>Bin3BE</i> .
			48	Nuly Čísla jsou doplněna nulami. Má význam pouze u formátů <i>Hex</i> , <i>HEX</i> , <i>Dec</i> , <i>UDec</i> , <i>Flt_f</i> , <i>Flt_e</i> , <i>Flt_E</i> , <i>Flt_g</i> a <i>Flt_G</i> .

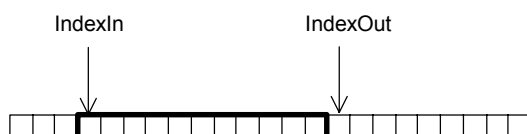
MinDélka	IN	Konst	Minimální délka zapisovaného řetězce. Hodnota -1 má význam "bez omezení". Je-li zformátovaný řetězec kratší, doplní se na tuto délku. Implicitně se doplňuje mezerami, zarovnání vpravo. Volbou parametru <i>SpecForm</i> lze zvolit doplňování čísel nulami příp. zarovnávání vlevo. Nemá význam u formátů <i>BinInt</i> , <i>BinIntBE</i> , <i>BinLong</i> , <i>BinLongBE</i> , <i>Bin3</i> a <i>Bin3BE</i> .
		I	
		MI	

Přesnost	IN	Konst	Počet desetinných míst / maximální délka řetězce. Hodnota -1 má význam "bez omezení".						
		I	Význam pro jednotlivé formáty:						
		MI	<table><tr><th>Formát</th><th>Význam parametru</th></tr><tr><td><i>Flt_f, Flt_e, Flt_E, Flt_g, Flt_G</i></td><td>Počet desetinných míst.</td></tr><tr><td><i>String, BinString</i></td><td>Maximální délka řetězce. Je-li řetězec v matici hodnota delší, zapíše se do výstupního řetězce pouze prvních <i>n</i> znaků, kde <i>n</i> je hodnota parametru.</td></tr></table>	Formát	Význam parametru	<i>Flt_f, Flt_e, Flt_E, Flt_g, Flt_G</i>	Počet desetinných míst.	<i>String, BinString</i>	Maximální délka řetězce. Je-li řetězec v matici hodnota delší, zapíše se do výstupního řetězce pouze prvních <i>n</i> znaků, kde <i>n</i> je hodnota parametru.
			Formát	Význam parametru					
			<i>Flt_f, Flt_e, Flt_E, Flt_g, Flt_G</i>	Počet desetinných míst.					
<i>String, BinString</i>	Maximální délka řetězce. Je-li řetězec v matici hodnota delší, zapíše se do výstupního řetězce pouze prvních <i>n</i> znaků, kde <i>n</i> je hodnota parametru.								
U ostatních formátů nemá parametr význam.									

Hodnota	IN	I	Vstupní formátovaná hodnota. Typ proměnné se volí v závislosti na zadaném formátu (viz popis parametru Formát).
		L	
		F	
		MI	
		NONE	

OK	OUT	Bit	Příznak úspěchu ("1" = úspěch, "0" chyba). Převod skončí chybou, pokud se zformátovaný řetězec "nevejde" do výstupního řetězce.
		NONE	

Obrázek objasňující význam některých parametrů. Je na něm znázorněna matice Data. V silně orámované části je zapsaný řetězec.



Maximální délka výsledného zformátovaného řetězce (včetně textové masky), který modul zapisuje do proměnné Data, **nesmí přesáhnout 64 znaků**. V opačném případě se zformátuje pouze část řetězce a příznak úspěchu OK se nastaví na "0".

Je-li potřeba formátovat řetězec delší než 64 znaků, je nutné použít více modulů StrFormat tak, aby každý jednotlivý modul formátoval pouze část řetězce, která nepřesáhne 64 znaků. Proměnná Data bude v tomto případě větší než 64 znaků a moduly ji budou sdílet.

#### Příklad

Chceme zformátovat float hodnotu proměnné T1 (typ F) do vysílaného telegramu OutTlg (MI[1, 40]). Hodnotě předchází text "Y=". Hodnota je formátovaná na délku 7 znaků, na jedno desetinné místo a je zarovnaná vlevo.

```
StrFormat OutTlg, 0, NdxOut, "Y=", Flt_f, Vlevo, 7, 1, T1[0,0], @OK
```

Je-li hodnota T1 např. 123.456, bude v proměnné OutTlg zapsán řetězec:

"Y=123.4 ". Proměnná NdxOut bude mít hodnotu 9, bit @OK bude mít hodnotu 1.

<b>StrParse</b>	Převod formátovaných dat (řetězce) na hodnotu (uživatelská komunikace).
-----------------	---

**Popis**

Modul slouží k převodu z řetězce v zadaném formátu na hodnotu. Používá se v uživatelské komunikaci při dekódování přijatého rámce.

**Parametry**

<b>Data</b>	IN	MI	Řádková matice $MI[1, n]$ obsahující vstupní převáděný řetězec. Jedna buňka matice odpovídá jednomu znaku řetězce. Převáděný řetězec začíná buňkou $Data[0, IndexIn]$ .
-------------	----	----	---

<b>IndexIn</b>	IN	Konst	Index v matici <i>Data</i> , od kterého se začíná převádět.
		I	
		MI	

<b>IndexOut</b>	OUT	I	Index v matici <i>Data</i> posunutý o počet zpracovaných znaků. Skončí-li převod chybou z toho důvodu, že nesouhlasí textová maska (viz dále), nastaví se parametr na hodnotu <i>IndexIn</i> .
		NONE	

<b>TextMaska</b>	PAR	Řetězec	Textová maska. Text, který musí předcházet vlastní hodnotě v převáděném řetězci. Text začíná buňkou $Data[0, IndexIn]$ . Neshoduje-li se počáteční text v převáděném řetězci s textovou maskou, skončí převod neúspěchem. Musí odpovídat i velikost písmen. Řetězec může obsahovat řídicí znaky, viz dodatek "Řídicí znaky v řetězcích". Délka řetězce <b>nesmí přesáhnout 64 znaků</b> .
------------------	-----	---------	---

<b>Formát</b>	IN	Konst	Formát dat.
		I	
		MI	

Tabulka jednotlivých formátů:

(U některých formátů je popisováno pořadí bytů ve výstupním řetězci. Jednotlivé byty jsou označeny B0, B1, B2, .... B0 je nejnižší významový byte. Pořadí bytů je označeno BE - "big endian", nebo LE - "little endian")

Číslo	Typ	Popis	Typ proměnné Hodnota
-1	TextMaska	Pouze se kontroluje textová maska, tj. zda začátek řetězce je shodný s textovou maskou. Parametr <i>Hodnota</i> (viz dále) může být v tomto případě NONE.	NONE
0	BinInt	Binární integer (2 byty). Byty jdou v pořadí LE (B0, B1). Např. řetězec (0x12, 0x34) se převede na hodnotu 0x3412.	I (L, F)
1	BinIntBE	Binární integer (2 byty). Byty jdou v pořadí BE (B1, B0). Např. řetězec (0x12, 0x34) se převede na hodnotu 0x1234.	I (L, F)
2	BinLong	Binární long (4 byty). Byty jdou v pořadí LE (B0, B1, B2, B3). Např. řetězec (0x12, 0x34, 0x56, 0x78) se převede na hodnotu 0x78563412.	L (I, F)
3	BinLongBE	Binární long (4 byty). Byty jdou v pořadí BE (B3, B2, B1, B0). Např. řetězec (0x12, 0x34, 0x56, 0x78) se převede na hodnotu 0x12345678.	L (I, F)
4	Bin3	Binární celočíselná hodnota na 3 bytech. Byty jdou v pořadí LE (B0, B1, B2). Např. řetězec (0x12, 0x34, 0x56) se převede na hodnotu 0x563412.	L (I, F)

5	Bin3BE	Binární celočíselná hodnota na 3 bytech. Byty jdou v pořadí BE (B2, B1, B0). Např. řetězec (0x12, 0x34, 0x56) se převede na hodnotu 0x123456.	L (I, F)
6	BinFlt	Binární float (4 byty). Byty jdou v pořadí LE (B0, B1, B2, B3). Např. řetězec (0x79, 0xE9, 0xF6, 0x42) se převede na hodnotu 123.456 (binárně 0x42F6E979).	F (I, L)
7	BinFltBE	Binární float (4 byty). Byty jdou v pořadí BE (B3, B2, B1, B0). Např. řetězec (0x42, 0xF6, 0xE9, 0x79) se převede na hodnotu 123.456 (binárně 0x42F6E979).	F (I, L)
8	Hex	Hexadecimální číslo zapsané textově - na velikosti písmen nezáleží. Délka načítaného čísla je omezena délkou řetězce nebo prvním znakem, který není hexadecimální. Např. řetězec "89AB-Ano" se převede na hodnotu 0x89AB.	I, L (F)
10	BCD	BCD kódované číslo. Jedna buňka matice představuje 2 číslice kódované BCD. Délka načítaného čísla je omezena délkou řetězce nebo prvním znakem, který není BCD. Např. řetězec (0x01, 0x23, 0x45, 0xA0) se převede na hodnotu 12345.	I, L (F)
11	Dec	Celé číslo zapsané textově - kladné i záporné. Délka načítaného čísla je omezena délkou řetězce nebo prvním nečíselným znakem. Např. řetězec "-12345m" se převede na hodnotu -12345.	I, L (F)
13	Flt	Float číslo zapsané textově - může být i v exponenciálním tvaru. Délka načítaného čísla je omezena délkou řetězce nebo prvním nepatřičným znakem. Např. řetězec "1.456e2Pa" se převede na hodnotu 145.6.	F (I, L)
18	String	Textový řetězec. Výstupní proměnná je v tomto případě řádková matice MI představující řetězec. Modul ze vstupního převáděného řetězce vypreparuje textový podřetězec, který zapíše do výstupní proměnné - řetězce. Podřetězec je omezen maximální délkou převáděného řetězce (parametr <i>Délka</i> ) nebo je ukončen nulou případně oddělovačem. Oddělovače se zadávají v parametru <i>Oddělovač</i> (viz dále). Např. převáděný řetězec je "X1:Alfa,Beta,Gama", textová maska je "X1:" a oddělovačem je znak ';'. Do výstupní matice <i>Hodnota</i> se pak zapíše řetězec "Alfa".	MI
19	BinString	Binární řetězec. Výstupní proměnná je v tomto případě řádková matice MI představující binární řetězec. Modul ze vstupního převáděného řetězce vypreparuje binární podřetězec, který zapíše do výstupní proměnné - řetězce. Podřetězec je omezen maximální délkou převáděného řetězce (parametr <i>Délka</i> ) nebo je ukončen oddělovačem. Nula binární řetězec neukončuje. Oddělovače se zadávají v parametru <i>Oddělovač</i> (viz dále).	MI
50	DD.MM	Datum ve formátu <i>den.měsíc</i> . Např. "30.08", "30.8", "30.8", apod.	L
51	DD.MM.YY	Datum ve formátu <i>den.měsíc.rok</i> . Např. "30.08.00", "30.8.00", "30. 8.00", apod.	L
52	DD.MM.Y4	Datum ve formátu <i>den.měsíc.rok</i> (rok má 4 cifry). Např. "30.08.2000", "30.8.2000", "30. 8.2000", apod.	L
53	hh:mm	Čas ve formátu <i>hodiny:minuty</i> . Např. "09:38", "9:38", " 9:38".	L
54	hh:mm:ss	Čas ve formátu <i>hodiny:minuty:sekundy</i> . Např. "09:38:07", "9:38:7", " 9:38: 7".	L



55	DMYhms	Datum a čas ve formátu <i>den.měsíc.rok</i> <i>hodiny:minuty:sekundy</i> . Např. "30.08.00 09:38:07", "30.8.00 9:38:7", "30. 8.00 9:38: 7".	L
56	DMY4hms	Datum a čas ve formátu <i>den.měsíc.rok</i> <i>hodiny:minuty:sekundy</i> (rok má 4 cifry). Např. "30.08.2000 09:38:07", "30.8.2000 9:38:7", "30. 8.2000 9:38: 7".	L
57	DD	Datum ve formátu <i>den</i> . Např. "30", "9", " 9".	L
58	MM	Datum ve formátu <i>měsíc</i> . Např. "08", "8", " 8".	L
59	YY	Datum ve formátu <i>rok</i> . Např. "00", "0".	L
60	YYYY	Datum ve formátu <i>rok</i> (4 cifry). Např. "2000".	L
61	hh	Čas ve formátu <i>hodiny</i> . Např. "09", "9", " 9".	L
62	mm	Čas ve formátu <i>minuty</i> . Např. "38", "06", " 6".	L
63	ss	Čas ve formátu <i>sekundy</i> . Např. "07", "7", " 7".	L

Pozn. u formátů data a času, které neobsahují kompletní datum a čas, je výsledkem datum 1.1.1980 00:00:00 posunuté o převedenou část.

Např. formát je DD a převáděný řetězec je "15", pak výsledné datum je 15.1.1980 00:00:00. Z toho vyplývá, že lze datum a čas převádět po částech do jednotlivých proměnných použitím více modulů StrParse za sebou a výsledné datum potom získat součtem jednotlivých proměnných.

Ve sloupci "Typ proměnné Hodnota" tabulky jsou uvedeny povolené typy databázové proměnné, kterou lze dosadit za parametr Hodnota. Je-li typ uveden v závorce, znamená to, že není pro daný formát přirozený a hodnota se z přirozeného typu na tento typ přetypovává. Vyplývá z toho, že může dojít ke ztrátě přesnosti.

Délka	IN	Konst	Maximální délka převáděného řetězce. Je-li převáděný řetězec delší, převádí se pouze část řetězce omezená hodnotou parametru. Hodnota -1 má význam "bez omezení".
		I	
		MI	

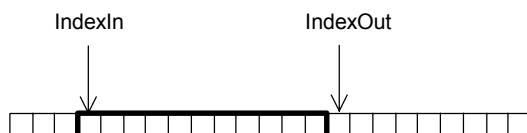
Oddělovač	PAR	Řetězec	Oddělovače slov v textu. Zadávají se bezprostředně za sebou. Máme-li např. slova odděleny buď čárkou nebo mezerou nebo tečkou, zadáme do parametru řetězec " , . ". Parametr má význam u formátů <i>String</i> a <i>BinString</i> , viz popis parametru Formát.
-----------	-----	---------	---

Hodnota	OUT	I	Výstupní převedená hodnota. Typ proměnné se volí v závislosti na zadaném formátu (viz popis parametru Formát).
		L	
		F	
		MI	
		NONE	

OK	OUT	Bit	Příznak úspěchu ("1" = úspěch, "0" chyba). K chybě může dojít z těchto příčin: - nesouhlasí textová maska - převáděný řetězec je příliš krátký - špatný formát řetězce - hodnotu nelze v tomto formátu převést
		NONE	

Neplatná	IN	Konst	Hodnota zapisovaná do výstupní proměnné Hodnota při neúspěchu. Skončí-li převod chybou (bit OK = 0), zapíše se do výstupní proměnné tato hodnota. U řetězcových formátů <i>String</i> a <i>BinString</i> se celá výstupní matice vyplní touto hodnotou.
		I	
		L	
		F	
		MI	
		ML	
		MF	
		NONE	

Obrázek objasňující význam některých parametrů. Je na něm znázorněna matice Data. V silně orámované části je převáděný řetězec.



Modul je schopen převést hodnotu, která je ve vstupním řetězci zapsána **maximálně na 64 znaků** (textová maska se do toho nepočítá). V opačném případě převod skončí neúspěchem (OK se nastaví na "0"). Problém příliš dlouhého převáděného řetězce může nastat pouze u formátů s variabilní délkou (*Hex*, *BCD*, *Dec*, *Flt*, *String* a *BinString*)

### Příklad

Chceme převést hexadecimální číslo do proměnné `HodnotaHex`. Vstupní řetězec má formát pevné délky "`Y=0xhhh`", kde `hhh` je převáděné hexadecimální číslo. Řetězec začíná od indexu 6 matice `Data`.

```
StrParse    Data, 6, NONE, "Y=0x", Hex, 8, "", HodnotaHex, @OK, 65535
```

Je-li proměnná `Data` např. zadaná takto "`A=567 Y=0x45FE Z=2`", zapíše se do proměnné `HodnotaHex` hodnota `0x45FE`. V případě, že se nepodaří číslo převést (např. "`A=567 X=0x45FE Z=2`"), bude mít proměnná `HodnotaHex` hodnotu `65535` (`0xFFFF`) a bit `@OK` bude mít hodnotu 0.

<b>StrSubst</b>	Náhrada znaků v řetězci (uživatelská komunikace)
-----------------	--

**Popis**

Modul slouží k nahrazení určitých znaků řetězce znaky jinými. Používá se v uživatelské komunikaci, pokud se např. začátek a konec rámce rozpoznává dle nějakých speciálních znaků a v datové části se tyto znaky již nesmí vyskytnout, aby přijímací strana špatně nevyhodnotila začátek nebo konec rámce. Je-li datová část prokolu binární a mohou se v ní tudíž vyskytnout libovolné znaky, musí se případné vyskytnuvší se speciální znaky nahradit nějak jinak. Obvykle se provádí náhrada dvěma znaky, přičemž první znak má význam přesmykače, a druhý znak je definovaným způsobem překódovaný původní znak. Když se potom rámec rozebírá, tak přesmykač oznamuje, že následný znak je překódován a je jej tedy nutno překódovat zpět.

**Parametry**

<b>DataVst</b>	IN	MI	Řádková matice $MI[1, n]$ obsahující vstupní řetězec. Jedna buňka matice odpovídá jednomu znaku řetězce.
<b>DataVýst</b>	OUT	MI	Řádková matice $MI[1, n]$ , do které se запиše výstupní řetězec. Jedna buňka matice odpovídá jednomu znaku řetězce.
<b>DélkaCelá</b>	IN	Konst	Délka celého vstupního řetězce, tj. počet buňek matice $DataVst$ představující řetězec.
		I	Maximální délka je 256 znaků. Řetězec začíná vždy od indexu 0, tj. první znak je v buňce $DataVst[0, 0]$ .
		MI	
<b>IndexSubs</b>	IN	Konst	Index v matici $DataVst$ , od kterého se začíná nahrazovat.
		I	
		MI	
<b>DélkaSubs</b>	IN	Konst	Délka oblasti v matici $DataVst$ , ve které se nahrazuje.
		I	
		MI	
<b>DélkaVýst</b>	OUT	I	Výsledná délka výstupního řetězce v matici $DataVýst$ .
		NONE	Výstupní řetězec začíná vždy od indexu 0, tj. první znak je v buňce $DataVýst[0, 0]$ .

Následuje 10 dvojic parametrů popisujících jednotlivé skupiny nahrazovaných znaků a jejich náhrady (X je číslo skupiny):

<b>SkupinaX</b>	PAR	Řetězec	Skupina znaků, které se nahrazují. Řetězec může obsahovat řídicí znaky, viz dodatek "Řídicí znaky v řetězcích". Je možné zadat maximálně 10 znaků.
<b>NáhradaX</b>	PAR	Řetězec	Skupina znaků, kterými se nahrazuje. Řetězec může obsahovat řídicí znaky, viz dodatek "Řídicí znaky v řetězcích". Je možné zadat maximálně 10 znaků.

**Modul nahrazuje jednotlivé skupiny postupně.** Nejprve nahradí všechny výskyty první skupiny v celém řetězci. V tomto výsledku potom nahradí výskyty druhé skupiny, atd. Na tento mechanismus se musí brát ohled při zadávání jednotlivých skupin znaků. Tyto skupiny se musí zadat ve vhodném pořadí, aby se nestalo to, že se v jednom kroku nahradí něco, co již bylo vloženo jako náhrada v krocích předchozích.

Ukážeme to na příkladu:

Znak "a" chceme nahradit znaky "xA" a znak "x" chceme nahradit znaky "xB". Kdybychom skupiny zadali v pořadí Skupina1 = "a", Náhrada1 = "xA", Skupina2 = "x", Náhrada2 = "xB", tak se stane následující:

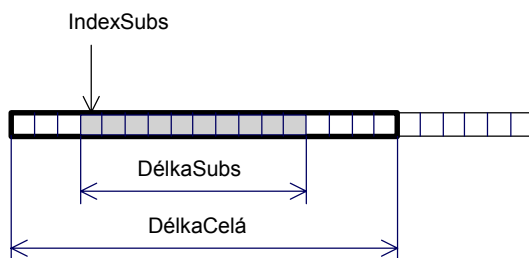
Pokud bychom např. převáděli řetězec "ax", tak se v prvním kroku převede na "xAx" a v druhém kroku se převede na "xBxAxB". Výsledek je tedy špatný. Proto musíme zadat skupiny v obráceném pořadí: Skupina1 = "x", Náhrada1 = "xB", Skupina2 = "a", Náhrada2 =

“xA”. Potom v prvním kroku se řetězec převede na “axB” a v druhém kroku na “xAxB”. Tento výsledek je již správný.

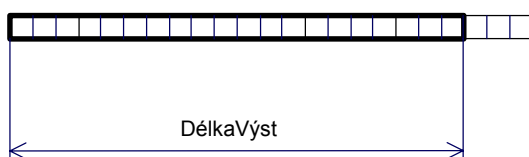
Při sestavování rámce z toho vyplývá pravidlo, že nejprve se musí nahradit přesmykač a potom teprve ostatní řídicí znaky. Při rozebírání přijatého rámce to platí obráceně - přesmykač se musí převést zpět jako poslední.

Obrázky na objasnění významu některých parametrů:

Na prvním obrázku je znázorněna vstupní matice DataVst. Silně orámovaná část představuje celý vstupní řetězec (začíná vždy od indexu 0). Tmavá část je podřetězec, ve kterém se provádí náhrada.



Na druhém obrázku je znázorněna matice DataVýst. V silně orámované části je výstupní řetězec (začíná vždy od indexu 0).



Délka vstupního řetězce ani délka výstupního řetězce **nesmí přesáhnout 256 znaků**, jinak je řetězec na tuto délku oříznut.

#### Příklad

Vysíláme rámce, které mají tuto strukturu:

- 0x02 ... 1 byte začínající rámec
- LEN ... 1 byte určující délku dat rámce
- DATA ... data rámce o délce LEN
- CHK ... 2 byty zabezpečení typu CRC-16
- 0x03 ... 1 byte ukončující rámec

Vlastní data jsou binární, mohou se v nich tedy objevit znaky libovolné hodnoty. Jelikož se začátek a konec rámce rozpoznává podle speciálního počátečního (0x02) a koncového (0x03) znaku, je nutno zabezpečit, aby se tyto znaky nevyskytly uvnitř rámce. Budeme tedy provádět náhradu těchto znaků uvnitř rámce. Náhrada se musí provádět pro celou oblast <LEN, DATA, CHK>. Znaky budeme nahrazovat přesmykačem a původním znakem s nastaveným nejvyšším bitem. Jako přesmykač použijeme speciální znak 0x1A. Znak 0x02 se tedy bude nahrazovat dvojicí znaků 0x1A, 0x82 a podobně znak 0x03 se bude nahrazovat dvojicí 0x1A, 0x83. Vyskytne-li se v datové části rámce znak 0x1A, musíme jej také nahradit, jinak by byl chybně dekodován jako přesmykač. Budeme jej

tedy nahrazovat dvojicí znaků 0x1A, 0x9A. Nahrazení přesmykače musíme provést jako první před ostatními znaky.

Vysílaný rámec je připraven v matici Tlg, v proměnné DelkaData je délka datové části rámce. Rámec s nahrazenými znaky budeme ukládat do proměnné TlgOut. Do proměnné DelkaOut budeme zapisovat délku výsledného řetězce.

```
LET          DelkaSubst = DelkaData + 3
LET          DelkaCela = DelkaData + 5
StrSubst     Tlg, TlgOut, DelkaCela, 1, DelkaSubst, DelkaOut, "\x1A", "\x1A\x9A",
              "\x02", "\x1A\x82", "x03", "\x1A\x83", "", "", "", "", "", "", "", "",
              "", "", "", "", "", ""
```

<b>SubInst</b>	Instance podprogramu
----------------	----------------------

**Popis**

Modul slouží k vytvoření instance podprogramu. Používá se ve spolupráci s jinými moduly (např. `ComInit`), které vyžadují odkaz na instanci podprogramu.

Podprogram může být spouštěn dvěma způsoby:

- ♦ Pomocí modulu `Call`  
V tomto případě není potřeba modulu `SubInst`, protože instance podprogramu je nahrazena modulem `Call`.
- ♦ Pomocí jiného modulu (např. `ComInit`)  
V tomto případě je modul `SubInst` nutný. Do jeho parametru se zadá odpovídající číslo podprogramu a dále se modulu zadá návěští. Modul odkazující se na instanci podprogramu (např. `ComInit`) se potom odkazuje přes toto návěští.

Modul se vkládá do procesu `INIT`.

**Parametry**

Číslo	PAR	Konst	Číslo instancovaného podprogramu (100 až 999).
-------	-----	-------	--

**Příklad**

Máme uživatelskou komunikaci po seriové lince. Hlavní modul komunikace `ComInit` bude vyvolávat podprogram při přerušení od přijatého znaku. V podprogramu se budou znaky z přijímacího buferu načítat do proměnné `RdTlg` představující přijímaný telegram.

```
ProcINIT:
:01000  SubInst  100
:02000  ComInit  0x0000, 0, 9600, 8, Sudá, 1, :01000, :NONE, :NONE, :NONE,
                               RecvBuf, SendBuf

Lib100:
      ComRead  :02000, RdTlg, RdNdx, RdNdx, NONE
```

V procesu `INIT` je pomocí modulu `SubInst` vytvořena instance podprogramu `Lib100`. Modul `ComInit` se na tuto instanci odkazuje přes návěští `:01000`. V podprogramu zpracovávajícím přerušení od přijatého znaku je vložen modul `ComRead`, který postupně zapisuje přijaté znaky do matice `RdTlg`. Po přijetí celého telegramu je třeba index `RdNdx` vynulovat, což již v příkladu není.

**Switch**

Přepínač: rozskok podle hodnoty

**Popis**

Příkaz **Switch** umožňuje realizaci přepínače - vykonání různých funkčních modulů pro různé hodnoty řídicí proměnné přepínače. Dvojice příkazů **Switch-EndSwitch** omezuje oblast přepínače a specifikuje řídicí proměnnou. V této oblasti jsou definovány jednotlivé větve přepínače pomocí dvojice příkazů **Case-EndCase**. Každá dvojice specifikuje jednu hodnotu řídicí proměnné, pro kterou budou vykonány funkční moduly uvnitř této větve. Mezi poslední příkaz **EndCase** a příkaz **EndSwitch** lze také vkládat funkční moduly. Tyto moduly pak tvoří tzv. "implicitní větev", která se bude provádět tehdy, když řídicí proměnná bude mít hodnotu nerovnající se ani jedné z možností "case".

**Parametry**

Proměnná	IN	I	Řídící proměnná přepínače. Podle konkrétních hodnot této proměnné bude vykonána jedna (nebo také žádná) z větví přepínače.
----------	----	---	--

Návěští	PAR	Návěští	Návěští následujícího příkazu <b>EndSwitch</b> - automatické a lokální, je tedy generováno automaticky.
---------	-----	---------	---

**Příklad**

```

Switch Test, :00010      Přepínač podle hodnoty
                           proměnné Test
      Case 0, :00000      Větev pro hodnotu
                           proměnné = 0
      . . .               Příkazy/moduly větve
:00000      EndCase       Konec větve
      Case 1, :00001      Větev pro hodnotu
                           proměnné = 1
      . . .
:00001      EndCase
      Case 2, :00002      Větev pro hodnotu
                           proměnné = 2
      . . .
:00002      EndCase

                           Implicitní větev pro všechny
                           ostatní hodnoty proměnné
      . . .
:00010EndSwitch          Konec přepínače

```

<b>SyncArch</b>	Archiv se synchronizací
-----------------	-------------------------

**Popis**

Modul umožňuje archivaci údajů v databázi v okamžicích definovaných synchronizačním bitem. Ke každému vzorku se ve zvláštní matici typu **ML** udržuje okamžitý kalendářní čas, ve kterém byl tento vzorek zaarchivován, tedy čas příchodu příslušného synchronizačního pulzu. Podle tohoto času lze také hodnoty v archivu vyhledávat, například pomocí modulu **FindWDay**. Prvkem archivu je sloupcový vektor libovolného množství hodnot, limitem je jen maximální rozměr databázové maticové proměnné.

**Parametry**

<b>Vstup</b>	<b>IN</b>	<b>I</b>	Zdrojová proměnná a číslo řádku. Jméno: Jméno databázové proměnné libovolného (zpravidla maticového) typu, jejíž hodnoty se při příchodu synchronizačního pulzu mají vložit do archivu. V úvahu se bere jen první sloupec matice, je-li <b>Vstup</b> maticového typu, musí mít nejméně stejný počet řádků jako archivní matice. Řádek: První řádek matice, od kterého se jednotlivé prvky vkládají do archivu.
		<b>L</b>	
		<b>F</b>	
		<b>MI</b>	
		<b>ML</b>	
		<b>MF</b>	

<b>Počet</b>	<b>PAR</b>	<b>Konst</b>	Počet řádků matice <b>Vstup</b> , které se mají do archivu vložit.
--------------	------------	--------------	--

<b>Archiv</b>	<b>OUT</b>	<b>MI</b>	Jméno: Archivní matice. Jméno databázové proměnné libovolného maticového typu. Typ nemusí být stejný jako u proměnné <b>Vstup</b> , modul automaticky zajistí typovou konverzi. Počet sloupců matice určuje hloubku archivu. Počet řádků určuje počet údajů archivovaných v jednom vzorku. Řádek: První řádek matice, od kterého se začnou ukládat jednotlivé řádky matice <b>Vstup</b> .
		<b>ML</b>	
		<b>MF</b>	

<b>Čas</b>	<b>OUT</b>	<b>ML</b>	Archivní matice časů.
------------	------------	-----------	-----------------------

Jméno databázové matice, do které modul ukládá časy archivovaných vzorků. Počet řádků může být libovolný, ale modul pracuje jen s prvním řádkem, takže více řádků matice časů nemá smysl. Počet sloupců by měl být stejný jako u archivní matice. Pokud ne, nehlásí se žádná chyba, ale hloubka archivu bude rovna menšímu z těchto dvou počtů. Je-li počet sloupců matice časů větší než u archivní matice, nebude správně pracovat modul **FindDay**.

<b>SyncIn</b>	<b>IN</b>	<b>Bit</b>	Vstupní synchronizační bit. Je-li při vstupu do modulu hodnota tohoto bitu 1, zapíše se nový vzorek do archivu a hodnota synchronizačního bitu se vynuluje.
---------------	-----------	------------	---

<b>SyncOut</b>	<b>OUT</b>	<b>Bit</b>	Výstupní synchronizační bit. Dojde-li k synchronizaci vstupním bitem <b>SyncIn</b> , modul zřetězí tuto synchronizaci do výstupního bitu tak, že na jeden běh procesu se tento výstupní bit nastaví do "1". Do obou parametrů <b>SyncIn</b> a <b>SyncOut</b> lze zadat stejný bit a navázat tak skupinu modulů <b>SyncArch</b> za sebou na jeden společný synchronizační bit.
----------------	------------	------------	---

<b>Krok</b>	<b>PAR</b>	<b>Konst</b>	Toto číslo definuje, kolik prvků se z archivu vypustí, pokud dojde k jeho zaplnění a je třeba udělat místo pro nové vzorky.
-------------	------------	--------------	---



Význam hodnot:

Hodnota Krok	Postup vypouštění
0	Vypustí se vždy všechny vzorky ze stejné sekundy
1	Vypustí se vždy všechny vzorky ze stejné minuty
2	Vypustí se vždy všechny vzorky ze stejné hodiny
3	Vypustí se vždy všechny vzorky ze stejného dne
4	Vypustí se vždy všechny vzorky ze stejného měsíce
5	Vypustí se vždy všechny vzorky ze stejného roku

Index	IN/OUT	I	Index archivu. Proměnná, která určuje index příště archivované položky v archivní a časové matici.
-------	--------	---	--

Indexuje	PAR	Výběr	Jestliže je jedna archivní matice vytvářena pomocí více volání modulu <b>SyncArch</b> (se stejnou archivní i časovou maticí), například proto, že se do archivu ukládají hodnoty z různých databázových proměnných, musí hodnotu archivního indexu měnit až poslední modul <b>SyncArch</b> . V tomto parametru se nastavuje, zdali dotýčný modul <b>SyncArch</b> je poslední (Indexuje=ANO), nebo zda budou následovat další moduly archivující do téhož archivu (Indexuje=NE).
----------	-----	-------	---

#### Příklad

```
SyncArch   HodnotyA[5,*], 2, Archiv[0,*], Casy, Sync.0, Sync.0, 3, Index, 0x0000
SyncArch   HodnotyB[0,*], 2, Archiv[2,*], Casy, Sync.0, Sync.0, 3, Index, 0x0001
```

1. modul archivuje hodnoty z řádků 5 a 6 vektoru *HodnotyA* do prvních dvou řádků matice *Archiv*.
2. modul archivuje hodnoty z řádků 0 a 1 vektoru *HodnotyB* do řádků 2 a 3 matice *Archiv*. Časy vzorků se ukládají do matice *Casy*. Oba moduly jsou synchronizovány společným bitem *Sync.0*. Archivní index se udržuje v proměnné *Index*, hodnotu archivního indexu mění až 2. modul (protože oba moduly pracují se stejnou archivační maticí). Při zaplnění archivu se vypouštějí hodnoty z celého dne, to znamená, že v archivu bude vždy běžný den (rozpracovaný) a tolik dní, kolik se do archivní matice vejde se zárukou, že všechny dny (pochopitelně s výjimkou běžného) budou v archivu vždy celé.

<b>SyncMark</b>	Generátor časových značek pro modul <b>SyncArch</b>
-----------------	---

**Popis**

Modul na základě vstupních hodnot generuje časovou značku nastavením bitu *Sync*. Časová značka je generována při změně zadané jednotky (vteřina, minuta, hodina, den, týden, měsíc) s volitelným posunem dopředu i dozadu. Změna jednotky den je dána půlnocí, jednotky týden půlnocí z neděle na pondělí, jednotky měsíc půlnocí posledního dne v aktuálním měsíci. Modul musí být umístěn v procesu s takovou periodou, která odpovídá nejmenšímu časovému členění požadovaných značek. Pokud je například umístěn v procesu s periodou 1 minuta a chceme generovat časovou značku každý den o půlnoci, bude časová značka umístěna náhodně v intervalu 00:00:00 až 00:01:00. Pokud je požadavek na přesnější umístění časové značky, nezbyvá než modul umístit do procesu s periodou jedné vteřiny. Navazující modul (nejčastěji **SyncArch**) pak musí zabezpečit využití tohoto bitu (nesmí mu "utéci"), protože v dalším běhu procesu je bit modulem **SyncMark** nulován.

**Parametry**

Jednotka	IN	Konst- výběr	Jednotka časové periody.
		I	
		MI	

Hodnota	Jednotka
1	Vteřina
2	Minuta
3	Hodina
4	Den
5	Týden
6	Měsíc

Perioda	IN	Konst	Udává počet jednotek, po kterých bude generována časová značka. Pokud například bude zvolena jednotka den a perioda bude 3, znamená to, že časová značka bude generována o půlnoci každý třetí den od spuštění procesu.
		I	
		MI	

PosunHod PosunMin PosunSec	IN	Konst	Parametry udávají posun (v kladném i záporném směru) od časové značky odpovídající jednotky. Posun se vždy uvažuje v minimálně o "řád" nižší jednotce, tedy například jednotka hodina může mít posun daný součtem <i>PosunSec</i> a <i>PosunMin</i> , ale hodnota <i>PosunHod</i> se ignoruje.
		I	
		MI	

Sync	OUT	Bit	Pokud dojde ke splnění časové podmínky, je tento bit nastaven na jedničku. Pokud podmínka splněna není, je tento bit nulován. Tedy jednička se objeví vždy pouze na jeden běh procesu a v příštím běhu je zase bit vynulován.
------	-----	-----	---

Další	OUT	L	Čas příští synchronizace.
		NONE	

**Příklad**

SyncMark      Den, 1, 0, 11, 59, 59, Sync.0, NONE

Modul nastaví bit *Sync.0* do jedničky každý den v 11:59:59.

<b>SyncSum</b>	Údržba sum a maxim v maticích se synchronizací
----------------	--

**Popis**

Činnost modulu se řídí dvěma bity. Bitem `Zpracuj` se určují vzorky. Suma se počítá jako součet vzorků vstupní proměnné. Maximum je maximální hodnota vstupní proměnné porovnávaná v době vzorku. K hodnotě maxima se udržuje i čas, ve kterém maximum nastalo.

Modul pracuje maticově, tj. obsluhuje tolik sum a maxim, kolik je řádků vstupní proměnné.

**Parametry**

<b>Zpracuj</b>	IN	Bit	Bit databázové proměnné - vzorkování. Nastavením bitu na "1" se do výstupních sum připočtou hodnoty vstupní matice a zkontroluje se, zda některé hodnoty nedosáhly maxima.
----------------	----	-----	--

<b>NullIn</b>	IN	Bit	Bit databázové proměnné - nulování. Nastavením bitu do "1" se v dalším běhu modulu vynulují výstupní sumy a maxima. Modul bit vynuluje.
---------------	----	-----	---

<b>NullOut</b>	OUT	Bit	Bit databázové proměnné - zřetězení nulování. Nastaví-li modul bit na "1", znamená to, že na výstupu modulu jsou konečné hodnoty a v dalším kroku se vynulují. Na tento bit je vhodné navázat např. synchronizovaný archiv modulem <b>SyncArch</b> .
----------------	-----	-----	--

Pokud je třeba navázat víc modulů za sebou na stejný synchronizační bit, je vhodné vyplnit stejný bit do parametrů `NullIn` a `NullOut`.

<b>Přírůstky</b>	IN	MI	Vstupní maticová proměnná libovolného typu. Modul automaticky přetypuje výpočet dle výstupních proměnných. Modul používá pouze první sloupec z matice.
		ML	
		MF	

<b>Sumy</b>	OUT	MI	Matice sum libovolného typu. Počet řádků matice musí být stejný nebo větší než počet řádků vstupní proměnné. Pokud nechceme počítat sumy, lze do parametru zadat NONE.
		ML	
		MF	
		NONE	

<b>Maxima</b>	OUT	MI	Matice maxim, lze použít libovolný typ. Počet řádků matice musí být stejný nebo větší než počet řádků vstupní proměnné. Pokud nechceme počítat maxima, lze do parametru zadat NONE.
		ML	
		MF	
		NONE	

<b>Čas</b>	IN	L	Vstupní proměnná - čas. Nastane-li maximum, zapíše se tento čas do proměnné časů maxim <code>Maxima</code> .
------------	----	---	--

<b>ČasyMaxim</b>	OUT	ML	Výstupní matice - časy, kdy nastala maxima. Počet řádků matice musí být stejný nebo větší než počet řádků vstupní proměnné. Pokud nechceme udržovat časy maxim, lze do parametru zadat NONE.
		NONE	

**Příklad**

`SyncSum      SyncQtr.1, SyncMon.0, SyncMon.0, QtrA, MonA1, MonMax1, EnTime, NONE`

Je-li bit `SyncQtr.1` čtvrt hodinová synchronizace, bit `SyncMon.0` měsíční synchronizace a proměnná `QtrA` práce ve čtvrt hodině, která se každou čtvrt hodinu vynuluje, pak modul udržuje měsíční sumu práce `MonA1` a měsíční čtvrt hodinové maximum `MonMax1`.

<b>Timeout</b>	Údržba <i>Timeoutů</i> pro komunikace
----------------	---------------------------------------

**Popis**

Modul umožňuje generovat až 16 různých timeoutů, na které se jiné moduly odkazují přes návěští modulu **Timeout** a index *timeoutu*. Modul musí mít návěští.

**Parametry**

<b>Instal</b>	PAR	Výběr	V bitové masce <i>ANO</i> znamená, že tento timeout se udržuje a obsluhuje.
---------------	-----	-------	---

<b>Tim0.. Tim15</b>	PAR	Konst	Délka timeoutu v násobcích periody procesu, ve kterém je modul umístěn.
-------------------------	-----	-------	---

**Timer**

Časovač

**Popis**

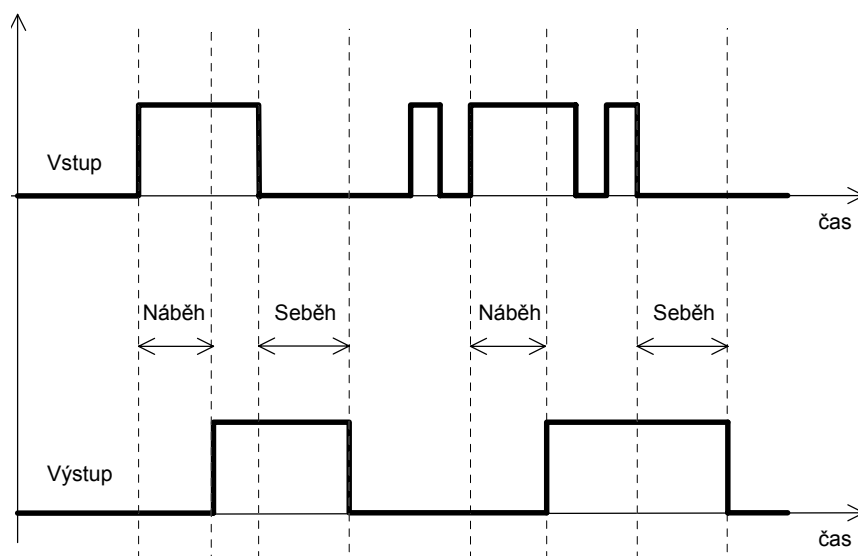
Modul umožňuje realizovat zpoždění digitálního signálu. Výstupní signál modulu "v zásadě kopíruje" vstupní signál. Náběžná hrana výstupního signálu je zpožděna vůči vstupnímu signálu o zadanou dobu *Náběh* a sestupná hrana je zpožděna o dobu *Seběh*. Navíc je možné vstupní signál před použitím negovat.

Druhou možností modulu je vytvářet pulzy odvozené od hran vstupního signálu.

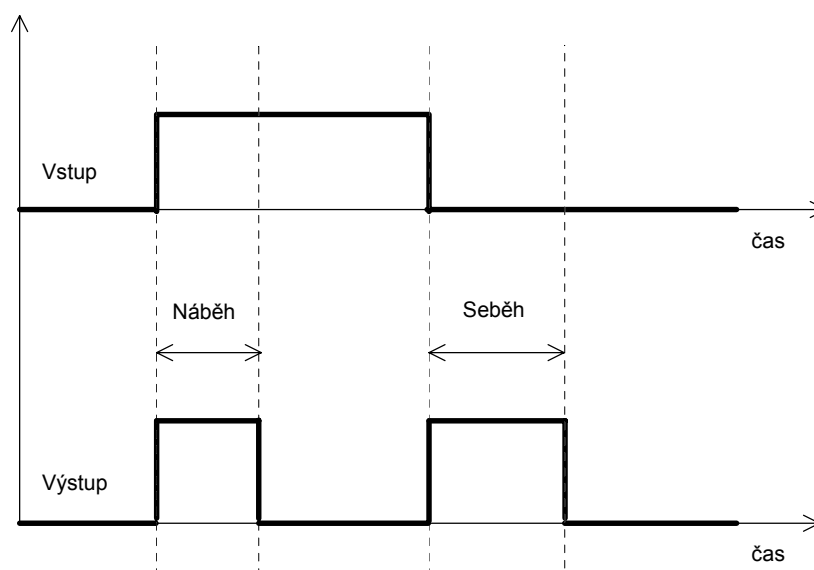
**Parametry**

Režim	PAR	Výběr	Nastavení režimu a negace.
-------	-----	-------	----------------------------

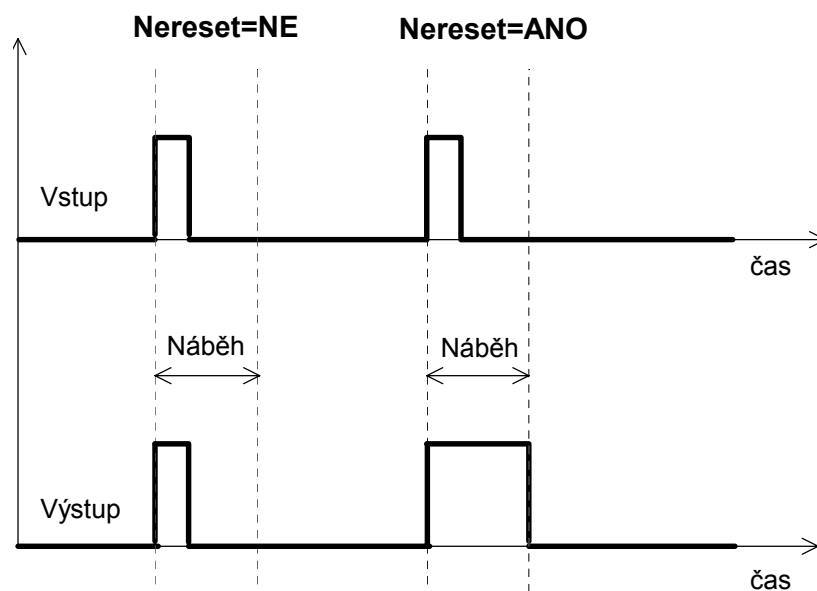
<b>0-Normal</b>	Konst	(zpožďovač) Výstupní signál kopíruje vstupní. Náběžná hrana je zpožděna vůči vstupnímu signálu o <i>Náběh</i> . Sestupná hrana je zpožděna vůči vstupnímu signálu o <i>Seběh</i> .
-----------------	-------	---



<b>1-Pulzy</b>	Konst	Na náběžnou hranu generuje pulz o délce <i>Náběh</i> . Na sestupnou hranu generuje pulz o délce <i>Seběh</i> .
----------------	-------	--



<b>Nereset</b>	Výběr	ANO=neresetovaný režim. Má význam pouze pro režim generace pulzů ( $Typ=Pulzy$ ). V neresetovaném režimu se výstupní pulz vždy dokončí celý nezávisle na změně vstupního signálu.
----------------	-------	---

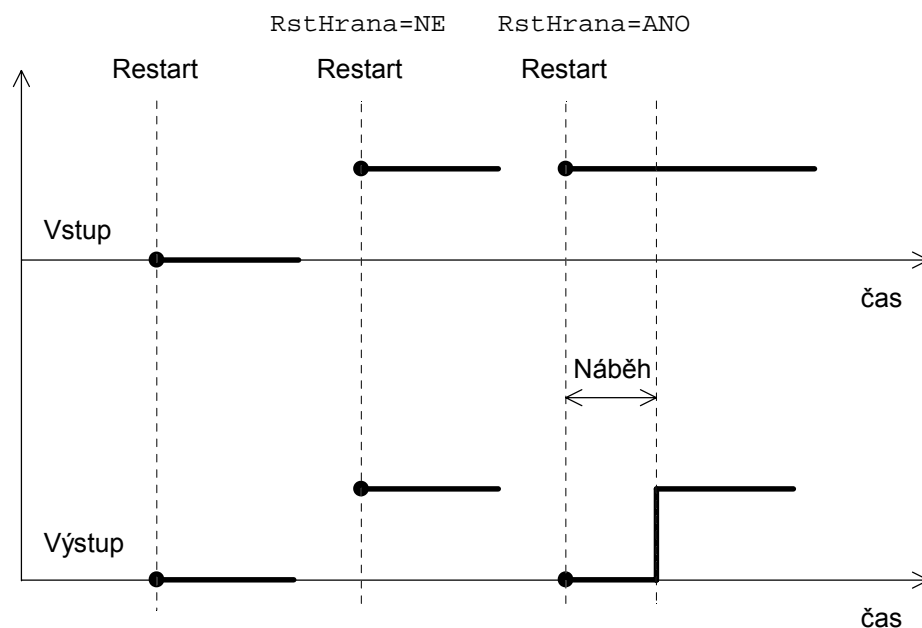


<b>NegVstup</b>	Výběr	Negace vstupního signálu.
-----------------	-------	---------------------------

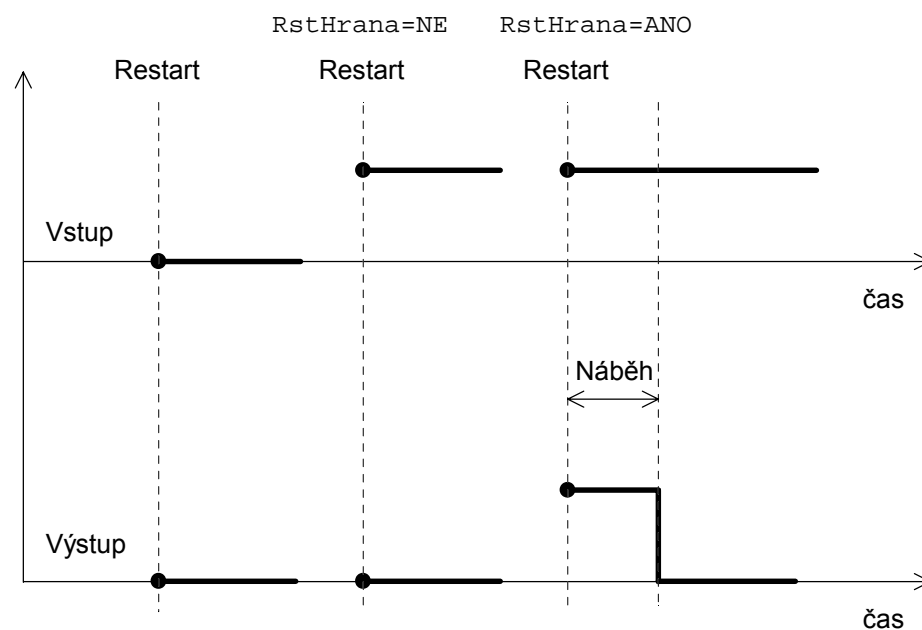
<b>Neg-Výstup</b>	Výběr	Negace výstupního signálu.
-------------------	-------	----------------------------

<b>RstHrana</b>	Výběr	<p>Chování po restartu:</p> <p>ANO = Je-li po restartu na vstupu "1", vyhodnotí se to jako náběžná hrana. V režimu <i>Normal</i> se provede zpoždění signálu. V režimu <i>Pulzy</i> se vygeneruje pulz od náběžné hrany.</p> <p>NE = Je-li po restartu na vstupu "1", modul zůstává v klidovém stavu. V režimu <i>Normal</i> je výstup nastaven na "1". V režimu <i>Pulzy</i> je výstup nastaven na "0".</p> <p>Je-li po restartu na vstupu "0", modul zůstává v klidovém stavu nezávisle na hodnotě parametru. V režimu <i>Normal</i> i v režimu <i>Pulzy</i> je výstup nastaven na "0".</p>
-----------------	-------	---

Na obrázku je znázorněno chování po restartu v režimu *Normal*:



Na dalším obrázku je znázorněno chování po restartu v režimu *Pulzy*:



<b>Vstup</b>	IN	Bit	Vstupní signál.
<b>Výstup</b>	OUT	Bit	Výstupní signál.
<b>Náběh</b>	IN	I	Zpoždění náběžné hrany / délka pulzu od náběžné hrany (v násobcích period procesu).

<b>Seběh</b>	IN	I	Zpoždění sestupné hrany / délka pulzu od sestupné hrany (v násobcích period procesu).
--------------	----	---	--

**Příklad**

Timer                      0x0000, X, Y, UpTime, DownTime

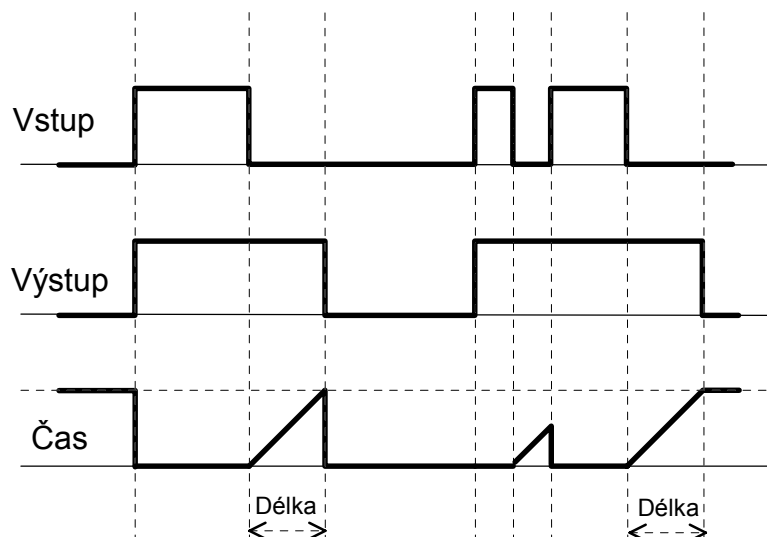
Modul zpožďuje náběžnou hranu signálu X o UpTime period a sestupnou hranu zpožďuje o DownTime period. Výstup se ukládá do proměnné Y.



<b>TimerOff</b>	Zpoždění sestupné hrany
-----------------	-------------------------

**Popis**

Modul realizuje zpoždění sestupné hrany digitálního signálu. Výstupní signál modulu "v zásadě kopíruje" vstupní signál. Sestupná hrana výstupního signálu je zpožděna vůči vstupnímu signálu o zadanou dobu *Délka*.

**Parametry**

<b>Vstup</b>	IN	Bit	Vstupní signál.
<b>Délka</b>	IN	Konst	Délka zpoždění [ms].
		L	
		ML	
<b>Výstup</b>	OUT	Bit	Výstupní signál.
<b>Čas</b>	OUT	L	Průběžný čas od poslední sestupné hrany vstupního signálu. Hodnota se pohybuje v intervalu 0 až <i>Délka</i> .
		NONE	

**Chování po restartu:**

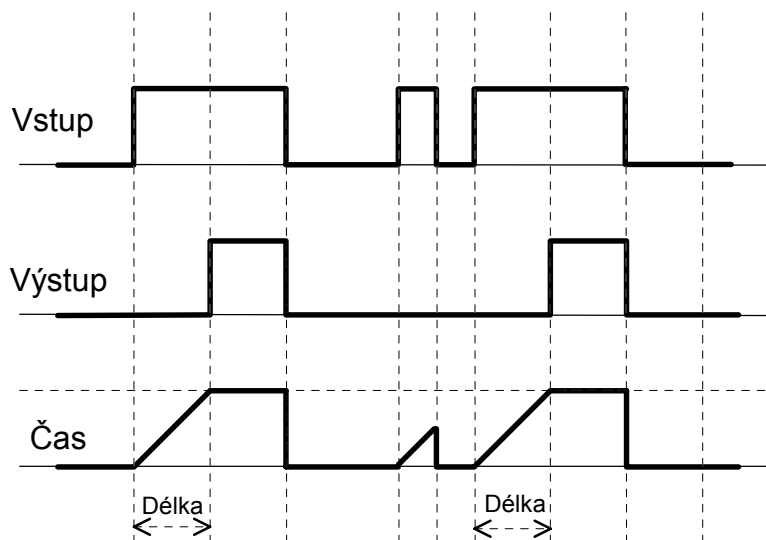
Je-li vstup v "0", je výstup nastaven rovněž na "0".

Je-li vstup v "1", je výstup nastaven rovněž na "1".

<b>TimerOn</b>	Zpoždění náběžné hrany
----------------	------------------------

**Popis**

Modul realizuje zpoždění náběžné hrany digitálního signálu. Výstupní signál modulu "v zásadě kopíruje" vstupní signál. Náběžná hrana výstupního signálu je zpožděna vůči vstupnímu signálu o zadanou dobu *Délka*.

**Parametry**

<b>Vstup</b>	IN	Bit	Vstupní signál.
<b>Délka</b>	IN	Konst	Délka zpoždění [ms].
		L	
		ML	
<b>Výstup</b>	OUT	Bit	Výstupní signál.
<b>Čas</b>	OUT	L	Průběžný čas od poslední náběžné hrany vstupního signálu. Hodnota se pohybuje v intervalu 0 až <i>Délka</i> .
		NONE	

**Chování po restartu:**

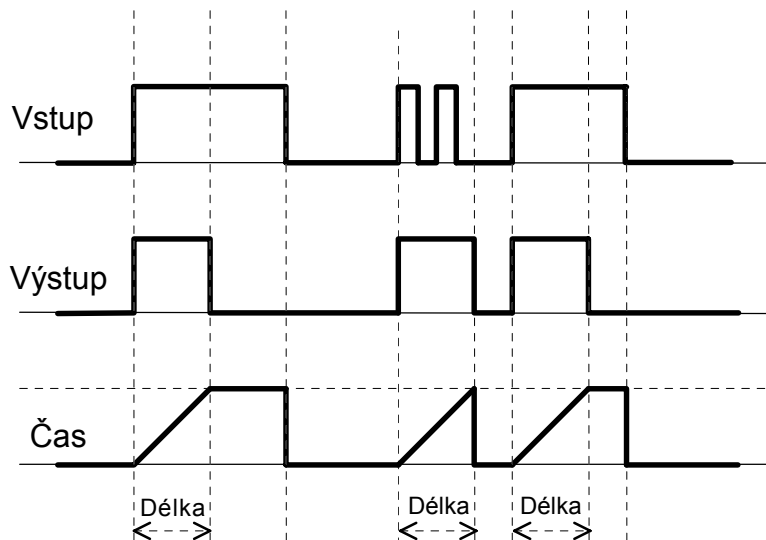
Je-li vstup v "0", je výstup nastaven rovněž na "0".

Je-li vstup v "1", modul to vyhodnotí jako náběžnou hranu a provede zpoždění náběžné hrany výstupu.

## TimerPuls Generování pulzu od náběžné hrany

### Popis

Modul generuje pulz stanovené délky odvozený od náběžné hrany vstupního digitálního signálu.



### Parametry

<b>Vstup</b>	IN	Bit	Vstupní signál.
<b>Délka</b>	IN	Konst	Délka zpoždění [ms].
		L	
		ML	
<b>Výstup</b>	OUT	Bit	Výstupní signál.
<b>Čas</b>	OUT	L	Průběžný čas od poslední náběžné hrany vstupního signálu. Hodnota se pohybuje v intervalu 0 až Délka.
		NONE	

### Chování po restartu:

Je-li vstup v "0", je výstup nastaven rovněž na "0".

Je-li vstup v "1", modul to vyhodnotí jako náběžnou hranu a vygeneruje pulz.

<b>Tmo</b>	Hlavní modul hlídání timeoutu
------------	-------------------------------

**Popis**

Definice timeoutu.

**Parametry**

<b>TmoItr</b>	PAR	Návěští	Návěští modulu <code>SubInst</code> s instancí podprogramu na zpracování přerušení od vypršení timeoutu.
---------------	-----	---------	--

<b>Rozlišení</b>	IN	Konst	Interní rozlišení [ms] s jakým modul nastavuje délku timeoutu. Při nastavování délky (modulem <code>TmoSet</code> ) se skutečná délka zaokrouhlí nahoru na celý násobek rozlišení (zhruba). Je vhodné nastavit rozlišení jako nejvyšší možné. Se snižováním rozlišení totiž vzrůstá zatížení systému. Změní-li se hodnota parametru za běhu, musí se provést reset stanice, jinak se změna neprojeví.
		I	
		MI	

<b>Nejdelší</b>	IN	Konst	Délka nejdelšího uvažovaného timeoutu [ms]. Změní-li se hodnota parametru za běhu, musí se provést reset stanice, jinak se změna neprojeví.
		L	
		ML	

Modul se umísťuje se do procesu `INIT`.

**Používání timeoutů:**

Základním modulem je modul `Tmo`, který definuje jeden timeoutový kanál. Ostatní moduly se na tento kanál vážou přes návěští. Timeout se spouští modulem `TmoStart`, ve kterém se nastaví také délka timeoutu. Běžící timeout lze zrušit modulem `TmoStop` nebo jej lze restartovat znovu od začátku pomocí modulu `TmoStart`. Nedojde-li u běžícího timeoutu k jeho zrušení nebo restartování do nastavené délky, dojde k vypršení timeoutu. Tato událost se dá testovat dvěma způsoby:

- a) pomocí přerušení  
Modulu `Tmo` se v tomto případě zadá do parametru `TmoItr` návěští modulu `SubInst` s instancí podprogramu, který bude při vypršení timeoutu vyvolán. Kód na obsluhu timeoutu se pak píše do tohoto podprogramu.
- b) periodické testování  
Testování se provádí vyvoláváním modulu `TmoCheck` umístěného nejčastěji do periodického procesu. Modulu `Tmo` se v tomto případě zadá do parametru `TmoItr` hodnota `NONE`.

Jeden timeoutový kanál může sice obsluhovat timeouty s různou délkou, avšak současně může obsluhovat pouze jeden běžící timeout. Pokud je potřeba hlídat více timeoutů běžících souběžně, je třeba pro každý z nich definovat zvláštní timeoutový kanál (tedy více modulů `Tmo`).

**Příklad**

Chceme používat hlídání timeoutem, budeme nastavovat délky 20 a 50 ms. Timeouty se nebudou hlídat souběžně. Dále chceme, aby se při vypršení timeoutu spustil podprogram číslo 100.

```
ProcINIT:
:01000   Tmo       :02000, 5, 50
:02000   SubInst   100
```

Rozlišení jsme zvolili 5 ms, nejdelší délku timeoutu 50 ms. Na druhém řádku je definice instance podprogramu číslo 100. Do něj napíšeme obsluhu události vypršení timeoutu.

<b>TmoCheck</b>	Kontrola stavu timeoutu
-----------------	-------------------------

**Popis**

Modul kontroluje, zda již timeout nastal nebo ne.

**Parametry**

Tmo	PAR	Návěští	Návěští modulu Tmo.
-----	-----	---------	---------------------

Nastal	OUT	Bit	Výsledek testu. "1" = nastal, "0" = nenastal.
--------	-----	-----	---

Modul nuluje interní příznak timeoutu. Znamená to, že při prvním běhu modulu po vypršení timeoutu, modul vrátí příznak "nastal" a v dalších bězích již vrací "nenastal", dokud se timeout znovu nespustí a znovu nevypřší.

Interní příznak timeoutu zůstává nastaven, dokud jej nevyčteme pomocí modulu TmoCheck nebo dokud není timeout znovu spuštěn. Interní příznak se rovněž nuluje vyvoláním modulu TmoStop.

**Příklad**

Realizujeme uživatelskou komunikaci po seriové lince. Znaky přijímaného telegramu načítáme v podprogramu přerušení, který se vyvolává po každém přijetí znaku. Komunikační protokol stanovuje, že délka mezi znaky nesmí přesáhnout 50 ms, jinak je telegram neplatný. Tuto délku budeme tedy hlídat pomocí timeoutu, který budeme testovat modulem TmoCheck. Dojde-li k timeoutu mezi znaky, nastavíme bit @ChybaTlg.

Příklad popíšeme zjednodušeně, abychom se zbytečně nezabývali komunikačními moduly. Přerušení od přijatého znaku bude vyvolávat podprogram číslo 100.

```

ProcINIT:
:01000   Tmo           :NONE, 10, 50           Definice timeoutu
        ... definice hl. komunikačního modulu nepopisujeme ...

Proc100:                                     Přerušení od přijatého znaku
        ... načtení znaku pomocí modulu ComRead ...
        TmoStart :01000, 50                   Spuštění timeoutu
        ... testování posledního znaku telegramu ...
        If          @PoslZnak, :NONE
        |   TmoStop :01000                     Zrušení timeoutu po posl. znaku
        EndIf

Proc00:                                     Periodický proces s periodou 0.5 s.
        TmoCheck :01000, @TmoNastal
        If          @TmoNastal, :NONE
        |   LET     @ChybaTlg = 1              Nastavení chyby
        EndIf

```

<b>TmoStart</b>	Spuštění timeoutu
-----------------	-------------------

**Popis**

Modul nastartuje timeout.

**Parametry**

<b>Tmo</b>	PAR	Návěští	Návěští modulu Tmo.
<b>Délka</b>	IN	Konst	Délka timeoutu [ms].
		L	
		ML	

**Příklad**

Realizujeme uživatelskou komunikaci po sériové lince. Znaky přijímaného telegramu načítáme v podprogramu přerušení, který se vyvolává po každém přijetí znaku. Komunikační protokol stanovuje, že délka mezi znaky nesmí přesáhnout 50 ms, jinak je telegram neplatný. Tuto délku budeme tedy hlídat pomocí timeoutu. Dojde-li k timeoutu mezi znaky, nastavíme bit @ChybaTlg.

Příklad popíšeme zjednodušeně, abychom se zbytečně nezabývali komunikačními moduly. Přerušení od přijatého znaku bude vyvolávat podprogram číslo 100. Vypršení timeoutu způsobí vyvolání podprogramu číslo 101.

ProcINIT:

```
:01000  Tmo      :02000, 10, 50      Definice timeoutu
:02000  SubInst  101                  Instance podprog. přerušení od timeoutu
      ... definice hl. komunikačního modulu nepopisujeme ...
```

Proc100:

```
      Přerušení od přijatého znaku
      ... načtení znaku pomocí modulu ComRead ...
      TmoStart :01000, 50              Spuštění timeoutu
      ... testování posledního znaku telegramu ...
      If      @PoslZnak, :NONE
      | TmoStop :01000                  Zrušení timeoutu po posl. znaku
      EndIf
```

Proc101:

```
      Přerušení od timeoutu
      LET      @ChybaTlg = 1           Nastavení chyby
```

<b>TmoStop</b>	Zrušení timeoutu
----------------	------------------

**Popis**

Modul zruší běžící timeout.

**Parametry**

<b>Tmo</b>	PAR	Návěští	Návěští modulu Tmo.
------------	-----	---------	---------------------

**Příklad**

Viz příklad u modulu TmoStart.

<b>Until</b>	Ukončení cyklu s podmínkou na konci
--------------	-------------------------------------

**Popis**

Příkaz **Until** ukončuje cyklus **Repeat-Until** s podmínkou na konci. Podrobný popis tohoto cyklu viz příkaz **Repeat**.

**Parametry**

Podmínka	IN	Bit	Cyklus bude ukončen, bude-li se hodnota podmínky rovnat 1.
----------	----	-----	--

**Příklad**

```
Repeat      :00000
. . .
Let        Konec.0 = . . .
:00000 Until Konec.0
```

Moduly těla cyklu se vykonávají tak dlouho, dokud se v příkazu **Let** nepřihadí podmínkovému bitu hodnota 1, která činnost cyklu ukončí. Poté se provádí modul bezprostředně následující za příkazem **Until**.



<b>ValAct1</b>	Buzení ventilu se stavovými koncovými spínači
----------------	---

**Popis**

Funkční modul **ValAct1** umožňuje buzení ventilu se stavovými koncovými spínači. Tento modul spolupracuje s modulem pro řízení obecného ventilu **ValCtrl**. U popisu modulu **ValCtrl** je popsána podrobně celá činnost dvojice funkčních modulů.

**Parametry**

<b>Vstup</b>	IN	I	Proměnná, která obsahuje povel pro ventil od modulu <b>ValCtrl</b> .
<b>Konc.OFF</b>	IN	DI	Signál logického kanálu DI - sepnutí koncového spínače stavu ventilu 0%. Signál musí být přiveden v pozitivní logice, tedy 1 znamená sepnuto.
<b>Konc.ON</b>	IN	DI	Signál logického kanálu DI - sepnutí koncového spínače stavu ventilu 100%. Signál musí být přiveden v pozitivní logice, tedy 1 znamená sepnuto.
<b>Info</b>	IN/OUT	I	Jméno proměnné, do níž zapisuje modul <b>ValAct1</b> zpětnou informaci pro modul <b>ValCtrl</b> .
<b>AkceOFF</b>	OUT	DO	Signál logického kanálu DO, na nějž je připojen povel pro zavírání ventilu.
<b>Akce ON</b>	OUT	DO	Signál logického kanálu DO, na nějž je připojen povel pro otevírání ventilu.
<b>Parametry</b>	IN	MI	Proměnná o rozměru [3, 1] s parametry ventilu.

**Příklad**

ValCtrl                      Poloha, OdhadPolohy, Info, Akce, Parametry

ValAct1                      Akce, #0.0, #0.1, Info, #0.8, #0.9, Parametry

Modul **ValCtrl** řídí pohyb ventilu na polohu (v %) danou proměnnou *Poloha*. Odhad skutečné polohy ventilu je v proměnné *OdhadPolohy*. Informace o poloze a stavu ventilu se přenáší z modulu **ValAct1** do modulu **ValCtrl** v proměnné *Info*, řídící informace pro pohyb ventilu opačným směrem v proměnné *Akce*. Koncové spínače jsou přivedeny (předpokládáme-li pouze standardní vstupy procesní stanice) na DI.0 a DI.1, povely pro zavírání a otevírání ventilu jsou připojeny na DO.8 a DO.9. Parametry ventilu jsou specifikovány v matici *Parametry*.

<b>ValAct2</b>	Buzení ventilu s impulzními koncovými spínači
----------------	---

**Popis**

Funkční modul **ValAct2** umožňuje buzení ventilu s impulzními koncovými spínači. Tento modul spolupracuje s modulem pro řízení obecného ventilu **ValCtrl**. U popisu modulu **ValCtrl** je popsána podrobně celá činnost dvojice funkčních modulů.

**Parametry**

<b>Vstup</b>	IN	I	Proměnná, která obsahuje povel pro ventil od modulu <b>ValCtrl</b> .
<b>Konc.OFF</b>	IN	DI	Signál logického kanálu DI, na nějž je připojen signál o sepnutí koncového spínače stavu ventilu 0%. Signál musí být přiveden v pozitivní logice, tedy 1 znamená sepnuto.
<b>Konc.ON</b>	IN	DI	Signál logického kanálu DI, na nějž je připojen signál o sepnutí koncového spínače stavu ventilu 100%. Signál musí být přiveden v pozitivní logice, tedy 1 znamená sepnuto.
<b>Info</b>	IN/OUT	I	Proměnná, do níž zapisuje modul <b>ValActx</b> zpětnou informaci pro modul <b>ValCtrl</b> .
<b>Akce OFF</b>	OUT	DO	Signál logického kanálu DO, na nějž je připojen povel pro zavírání ventilu.
<b>Akce.ON</b>	OUT	DO	Signál logického kanálu DO, na nějž je připojen povel pro otevírání ventilu.
<b>Parametry</b>	IN	MI	Proměnná o rozměru [3, 1] s parametry ventilu.

**Příklad**

ValCtrl                      Poloha, OdhadPolohy, Info, Akce, Parametry

ValAct2                      Akce, #0.0, #0.1, Info, #0.8, #0.9, Parametry

Modul **ValCtrl** řídí pohyb ventilu na polohu (v %) danou proměnnou *Poloha*. Odhad skutečné polohy ventilu je v proměnné *OdhadPolohy*. Informace o poloze a stavu ventilu se přenáší z modulu **ValAct2** do modulu **ValCtrl** v proměnné *Info*, řídící informace pro pohyb ventilu opačným směrem v proměnné *Akce*. Koncové spínače jsou přivedeny (předpokládáme-li pouze standardní vstupy procesní stanice) na DI.0 a DI.1, povel pro zavírání a otevírání ventilu jsou připojeny na DO.8 a DO.9. Parametry ventilu jsou specifikovány v matici *Parametry*.

<b>ValAct3</b>	Buzení ventilu bez koncových spínačů
----------------	--------------------------------------

**Popis**

Funkční modul **ValAct3** umožňuje buzení ventilu bez koncových spínačů. Tento modul spolupracuje s modulem pro řízení obecného ventilu **ValCtrl**. U popisu modulu **ValCtrl** je popsána podrobně celá činnost dvojice funkčních modulů.

**Parametry**

<b>Vstup</b>	IN	I	Proměnná, která obsahuje povel pro ventil od modulu <b>ValCtrl</b> .
<b>Info</b>	IN/OUT	I	Proměnná, do níž zapisuje modul <b>ValActx</b> zpětnou informaci pro modul <b>ValCtrl</b> .
<b>AkceOFF</b>	OUT	DO	Signál logického kanálu DO, na nějž je připojen povel pro zavírání ventilu.
<b>Akce ON</b>	OUT	DO	Signál logického kanálu DO, na nějž je připojen povel pro otevírání ventilu.
<b>Parametry</b>	IN	MI	Proměnná o rozměru [3, 1] s parametry ventilu.

**Příklad**

ValCtrl                      Poloha, OdhadPolohy, Info, Akce, Parametry

ValAct3                      Akce, Info, #0.8, #0.9, Parametry

Modul **ValCtrl** řídí pohyb ventilu na polohu (v %) danou proměnnou `Poloha`. Odhad skutečné polohy ventilu je v proměnné `OdhadPolohy`. Informace o poloze a stavu ventilu se přenáší z modulu **ValAct3** do modulu **ValCtrl** v proměnné `Info`, řídicí informace pro pohyb ventilu opačným směrem v proměnné `Akce`. Povely pro zavírání a otevírání ventilu jsou připojeny na DO.8 a DO.9. Parametry ventilu jsou specifikovány v matici `Parametry`.

**Popis**

Modul **ValCtrl** umožňuje řízení obecného ventilu. Převod obecných signálů a vlastností na konkrétní typ ventilu zajišťuje spolupracující modul **ValAct1**, **ValAct2** nebo **ValAct3**.

Mezi oběma spolupracujícími moduly dochází k výměně řídicích dat. Tato data se přenášejí přes proměnnou typu  $\mathbb{I}$  (jednu pro každý směr) a jsou bitově kódovaná. Pracuje-li ventil v automatickém režimu, není obsah těchto slov významný. Ventil však lze přepnout i do ručního režimu a používat řadu dodatečných funkcí modulu **ValCtrl**. Proto nyní popíšeme nejdůležitější bity obou proměnných.

## Proměnná Akce

Obsah této proměnné určuje modul **ValCtrl**. Pokud tento modul přepneme do ručního režimu (viz popis proměnné **Info**), lze použít proměnnou **Akce** k ručnímu řízení ventilu prostřednictvím modulu **ValActx**.

## Bit 0

Zavírání ventilu

Hodnota 1 v tomto bitu znamená pro **ValCtrl** povel "zavírej ventil".

## Bit 1

Otevírání ventilu

Hodnota 1 v tomto bitu znamená pro **ValCtrl** povel "otevírej ventil".

## Proměnná Info

## Bit 0

Koncový spínač 0%

Hodnota 1 v tomto bitu znamená, že se ventil nachází na koncovém dorazu 0%. Tento signál je stavový bez ohledu na typ skutečně použitých koncových spínačů. Lze jej využít především pro účely vizualizace na uživatelské stanici.

## Bit 1

Koncový spínač 100%

Hodnota 1 v tomto bitu znamená, že se ventil nachází na koncovém dorazu 100%. Tento signál je stavový bez ohledu na typ skutečně použitých koncových spínačů. Lze jej využít především pro účely vizualizace na uživatelské stanici.

## Bit 4

Režim ventilu AUT/MAN

Je-li hodnota tohoto bitu nulová, pracuje dvojice modulů **ValCtrl/ValActx** autonomně. Nastavením bitu do hodnoty 1 se zablokuje zápis modulu **ValCtrl** do proměnné **Akce** a lze provádět ruční řízení ventilu.

## Bit 5

Automatická identifikace ventilu

Nastavením tohoto bitu do 1 žádáme modul **ValCtrl** o provedení identifikace ventilu. Modul **ValCtrl** převede ventil nejdříve do stavu 0% (detekováno koncovým spínačem), pak přejede s ventilem do stavu 100% (detekováno koncovým spínačem) a nakonec se vrátí s ventilem do původní polohy. Přitom se měří doba přestavení ventilu z 0 na 100%. Naměřená doba se zapíše do bloku parametrů na souřadnici [1, 0] (viz dále) a nuluje se požadavek identifikace (bit 5). Automatická identifikace je použitelná pouze pro ventily vybavené koncovými spínači.

## Bit 6

Změna konstant

Změna konstant ventilu je spojena se značným množstvím přepočtů a kontrol. Není proto vhodné, aby modul **ValCtrl** prováděl tyto přepočty opakovaně. Měníme-li proto konstanty ventilu, pak po změně poslední z nich zapíšeme do tohoto bitu hodnotu 1. Modul pak provede jednorázově všechny potřebné operace a vrátí hodnotu tohoto bitu na 0. Tato hodnota setrvá po celou dobu až do další změny parametrů.

Na základě výše popsaných řídicích dat je schopen modul **ValCtrl** odhadovat skutečnou polohu ventilu, která je díky malé rychlosti přestavování často ve skluzu za požadovanou polohou. Pokud je ventil vybaven koncovými spínači, synchronizuje se odhadovaná poloha ventilu se skutečnou při každém dojezdu ventilu na koncový spínač.

## Bit 7

Prvotní inicializace ventilu

Bit signalizuje (hodnota "1"), že se provádí prvotní inicializace ventilu. Tuto inicializaci modul provádí po restartu systému. Jelikož modul po restartu nezná skutečnou polohu ventilu, probíhá inicializace tak, že se ventil nejprve zcela zavře a po té najede na požadovanou polohu. U ventilu bez koncových spínačů modul zavírá po dobu rovnou cca 120 % délky přeběhu, u ventilu s koncovými spínači zavírá, dokud nesepne koncový spínač. Během prvotní inicializace má odhadovaná poloha hodnotu 9999.0 %.

Bit 15

#### Vypnutí inicializace ventilu

Nastavením bitu do "1" lze vypnout prvotní inicializaci ventilu. Modul potom neprovádí prvotní zavření ventilu, ale přímo najede na požadovanou polohu. Jako výchozí odhad polohy načte po restartu hodnotu z buňky [2, 0] matice *Parametry* (viz dále).

Ostatní bity proměnné *Info* jsou určeny pro vnitřní účely modulu. Je nepřípustné je jakkoliv měnit nebo využívat.

#### Parametry

Pro každý typ ventilu lze nastavit základní časové parametry, které se používají při jeho řízení. Tyto parametry jsou uloženy v maticové proměnné typu *MI* o rozměru [3, 1]. Všechny časy se udávají v násobcích periody procesu.

[0, 0]

Prvek definuje dobu minimálního trvání budícího impulsu servomotoru v násobcích periody procesu.

[1, 0]

Prvek definuje dobu přeběhu ventilu ze stavu 0% do stavu 100% v násobcích periody procesu.

[2, 0]

Inicializační poloha ventilu [%]. Má význam pouze v režimu vypnutí inicializace ventilu. V tomto režimu modul po restartu načte hodnotu jako výchozí odhad polohy ventilu.

#### Parametry

<b>Vstup</b>	IN	F	Proměnná, která obsahuje požadovanou polohu ventilu (rozsah 0 až 100%).
<b>Poloha</b>	OUT	F	Proměnná, která obsahuje odhadnutou skutečnou polohu ventilu (rozsah 0 až 100%). Probíhá-li po restartu inicializace ventilu, má proměnná po tuto dobu hodnotu 9999.0. Probíhá-li automatická identifikace ventilu, neodpovídá hodnota proměnné odhadované poloze. Proměnná má po tuto dobu spíše význam "počítadla" pro stanovení doby přeběhu.
<b>Info</b>	IN/OUT	I	Proměnná, do níž zapisuje modul <b>ValActx</b> zpětnou informaci pro modul <b>ValCtrl</b> .
<b>Akce</b>	OUT	I	Proměnná, která obsahuje povel pro ventil z modulu <b>ValActx</b> .
<b>Parametry</b>	IN	MI	Proměnná o rozměru [3, 1] s parametry ventilu (viz popis).

#### Příklad

ValCtrl                      Poloha, OdhadPolohy, Info, Akce, Parametry

ValAct1                      Akce, #0.0, #0.1, Info, #0.8, #0.9, Parametry

Modul **ValCtrl** řídí pohyb ventilu na polohu (v %) danou proměnnou *Poloha*. Odhad skutečné polohy ventilu je v proměnné *OdhadPolohy*. Informace o poloze a stavu ventilu se přenáší z modulu **ValAct1** do modulu **ValCtrl** v proměnné *Info*, řídicí informace pro pohyb ventilu opačným směrem v proměnné *Akce*. Koncové spínače jsou přivedeny (předpokládáme-li pouze standardní vstupy procesní stanice) na DI.0 a DI.1, povel pro zavírání a otvírání ventilu jsou připojeny na DO.8 a DO.9. Parametry ventilu jsou

specifikovány v matici `Parametry`. Je-li např. modul umístěn v procesu s periodou 0.5 s a obsah matice je 10, 360 a 0, znamená to, že minimální délka budícího pulzu servomotoru bude 5 sekund a doba přeběhu ventilu z jednoho dorazu na druhý je 3 minuty.

**Valve**

Ovládání jednoduchého ventilu bez koncových spínačů

**Popis**

Ventil se řídí pomocí dvou výstupních bitů **Dolů** a **Nahoru**. Modul zajišťuje, aby nikdy nebyly současně oba bity v "1". Poloha ventilu se odhaduje na základě zadané doby přeběhu. Modul pohybuje s ventilem tak, aby se odhadovaná poloha blížila žádané. Poloha ventilu se pohybuje v rozsahu 0..100%.

**Dorazy**

Je-li žádaná poloha  $\geq 100\%$ , tak je signál **Nahoru** trvale v "1" (kvůli synchronizaci polohy).

Je-li žádaná poloha  $\leq 0\%$ , tak je signál **Dolů** trvale v "1" (kvůli synchronizaci polohy).

**Inicializace**

Po resetu se modul přepne do režimu inicializace, kdy zavírá ventil po dobu 1,5 násobku zadané doby přeběhu (signál **Dolů** je v "1"). Odhadovaná poloha má v tomto režimu speciální hodnotu 9999,0%. Dle této hodnoty lze režim inicializace detekovat.

**Reverzace**

Aby se zamezilo technologickým problémům při reverzaci směru chodu, je zajištěno, že mezi přepnutím z jednoho směru na druhý se vždy na dobu jedné periody procesu ventil zastaví - oba bity **Dolů** i **Nahoru** se nastaví na "0".

**Minimální krok**

Minimální časový krok je dán periodou procesu. Liší-li se žádaná a odhadovaná poloha méně než o kolik se dá změnit poloha za dobu periody, tak se s ventilem "nehýbe".

**Ruční řízení**

Typicky se modul používá ve spolupráci s modulem **PID**, který určuje žádanou polohu pro ventil. Ruční řízení se provádí tak, že se do ručního režimu přepne modul **PID**, a ručně se mění žádaná poloha. Modul **PID** tyto změny "registruje" a je pak schopen se znovu přepnout do automatického režimu beznárazově.

**Parametry**

<b>Žádaná</b>	IN	F	Žádaná poloha ventilu 0..100%.
		MF	
<b>Přeběh</b>	IN	Konst	Zadaná doba přeběhu [s]. Hodnota je určena technickými parametry ventilu.
		F	
		MF	
<b>Poloha</b>	OUT	F	Odhadovaná poloha 0..100%. V režimu inicializace má hodnotu 9999,0.
		MF	
<b>Dolů</b>	OUT	Bit	Povel "jed' dolů" (1=jed').
		DO	
<b>Nahoru</b>	OUT	Bit	Povel "jed' nahoru" (1=jed').
		DO	

**Příklad**

Valve    Zadana, 60.0, Poloha, Rizeni.0, Rizeni.1

Zadaná doba přeběhu je 60 s, v proměnné **Zadana** modul dostává žádanou polohu, do proměnné **Poloha** modul zapisuje odhad polohy. Bit **Rizeni.0** je povel "dolů" a bit **Rizeni.1** je povel "nahoru".

<b>VarWStat</b>	Test příznaku zápisu databázové proměnné
-----------------	--

**Popis**

Operační systém NOS udržuje pro každou proměnnou *příznak zápisu*, který se při **každém** zápisu nastaví. Modul vrací nastavení tohoto příznaku a zároveň umožňuje jeho nulování či nastavení.

**Parametry**

Proměnná	IN	I	Testovaná proměnná.
		L	
		F	
		MI	
		ML	
		MF	

Zapsáno	OUT	Bit	Bit v databázové proměnné, který se nastaví podle příznaku zápisu testované proměnné.
---------	-----	-----	---

Význam bitu:

- 0 - do testované proměnné nebylo zapsáno
- 1 - do testované proměnné bylo zapsáno

Nastavit	PAR	Konst	Typ požadované operace s příznakem zápisu u testované proměnné:
----------	-----	-------	---

- 0 - příznak se po testu nuluje
- 1 - příznak se po testu nastaví
- 2 - příznak se po testu nezmění

**Příklad**

```
VarWStat T11_zad, TEMP.0, 0
```

Modul testuje příznak zápisu proměnné `T11_zad` a výsledek testu uloží do nultého bitu proměnné `TEMP`. Příznak zápisu proměnné `T11_zad` je následně vynulován.



<b>Watchdog</b>	Obsluha uživatelského hlídacího časovače
-----------------	--

**Popis**

Provede obsluhu uživatelského hlídacího časovače, a zároveň nastaví časový limit, do kterého musí být provedena nová obsluha.

Zápornou hodnotou parametru `Limit` lze dosáhnout deaktivace uživatelského hlídacího časovače.

Po startu či restartu systému jsou všechny uživatelské hlídací časovače deaktivovány, tj. nemusí být vůbec obsluhovány. Bude-li ovšem některý obsloužen, musí již být obsluhován trvale, není-li následně deaktivován. Jinak dojde k restartu systému.

Viz též principiální popis systému logických hlídacích časovačů v dodatku "*Zabezpečení systému hlídacím časovačem (watchdogem)*".

**Parametry**

Kanál	IN	Konst.	Číslo uživatelského hlídacího časovače, který má být obsloužen, popř. deaktivován. Číslo musí být v rozsahu 0 až 31. Není-li, pracuje se pouze se zbytkem po dělení předané hodnoty třiceti dvěma.
		I	
		L	
		MI	
		ML	

Limit	IN	Konst.	Doba v milisekundách, do které musí být provedena nová obsluha příslušného uživatelského hlídacího časovače, jinak bude vyvolán restart systému. Záporná hodnota parametru způsobí deaktivaci příslušného uživatelského hlídacího časovače (limit nikdy nevyprší).
		I	
		L	
		MI	
		ML	

Parametr `Limit` může být nastaven maximálně na 1048527 ms (cca 1048 s, tj. cca 17.5 min). Předání větší hodnoty způsobí deaktivaci příslušného uživatelského hlídacího časovače, stejně jako předání záporné hodnoty.

Skutečná doba, po které vypršení limitu vyvolá restart systému, může být až o 70 ms delší, než je hodnota parametru `Limit`. V této závěrečné fázi před restartem (po vypršení zadaného limitu) ovšem nelze zaručit, že případná obsluha vypršeného hlídacího časovače zabrání restartu systému.

**Příklad**

```
Watchdog 0, 1000
```

Provede obsluhu uživatelského hlídacího časovače číslo 0. Do jedné sekundy musí být znovu vyvolán funkční modul **Watchdog** s parametrem `Kanál` rovným 0, jinak někdy během intervalu 1000 až 1070 ms od posledního vyvolání dojde k restartu systému.

```
Watchdog 1, -1
```

Deaktivuje uživatelský hlídací časovač číslo 1. Tento uživatelský hlídací časovač již nemusí být nikdy obsloužen, aniž by došlo k restartu systému.

<b>While</b>	Cyklus s podmínkou na začátku
--------------	-------------------------------

**Popis**

Příkaz **While** realizuje cyklus s podmínkou na začátku. Dokud je splněna podmínka uvedená v příkazu **While**, vykonávají se příkazy/moduly těla cyklu. Konec těla cyklu je vymezen následujícím příkazem **EndWhile**.

**Parametry**

<b>Podmínka</b>	IN	Bit	Bit proměnné, která obsahuje podmínku vykonání těla cyklu. Tělo se vykoná, je-li příslušný bit roven 1, jinak se pokračuje prvním modulem/příkazem následujícím za příkazem <b>EndWhile</b> .
<b>Návěští</b>	PAR	Návěští	Návěští následujícího příkazu <b>EndWhile</b> , který ukončuje cyklus. Návěští je automatické a lokální, lze ho generovat automaticky.

**Příklad**

```
While    Test.1, :00000
. . .
:00000EndWhile
```

Pokud je bit 1 proměnné `Test` nastaven, provádí příkaz procesní stanice moduly/příkazy uzavřené mezi příkazy **While** a **EndWhile**.

# Dodatek: Stavy sériové komunikace

## Vkládání požadavku na sériový přenos

Při vkládání požadavku na sériový přenos dat může dojít k chybám, jejichž seznam je v následující tabulce:

Hodnota	Význam
1	Požadavek byl vložen - žádná chyba.
2	Požadavek nebyl vložen, protože již dříve byl vložen identický požadavek a ještě nebyl vyřízen.
4	Požadavek nebyl vložen, protože buffer požadavků je plný.
8	Požadavek nebyl vložen, protože: 1) vlastníkem požadované databázové proměnné je tato stanice nebo 2) je požadován přenos výřezu databázové matice který přesahuje rozměry matice nebo 3) lokální proměnná je jiného datového typu než definuje komunikační modul (jen pro moduly pro přímý přenos proměnných) nebo 4) je požadován přenos přes modem voláním modulu <b>MdmRg...</b> a vlastníkem požadované databázové proměnné není vzdálená stanice definovaná modulem <b>MStation</b> .
16	Požadavek nebyl vložen, protože: 1) je požadován přenos přes modem a žádný modem není volný nebo 2) je požadován přenos po Ethernetu a není k dispozici žádná funkční rozhraní (Ethernetová karta)

## Stav vyřízení požadavku

Výsledek zpracování požadavku se průběžně ukládá do databázové proměnné typu **I**, jejíž jednotlivé bity popisují stav přenosu následujícím způsobem:

Bit	Význam
0	Má hodnotu 1, pokud právě probíhá komunikace.
1	Má hodnotu 1, pokud poslední ukončená komunikace skončila úspěšně.
2	Má hodnotu 1, pokud poslední ukončená komunikace skončila chybou.
3	Má hodnotu 1, pokud žádost byla rozložena na více rámců, přenos některého z nich skončil chybou a ostatní se ještě komunikují.
12÷15	Pokud komunikace skončila chybou (bit 0 má hodnotu 0 a bit 2 má hodnotu 1), obsahují tyto bity kód chyby komunikace podle následující tabulky. Jinak není hodnota těchto bitů definována.

Kódy chyb v bitech 12÷15:

Hodnota	Význam
1	Chyba přenosu (kontrolní součet, neplatný řídicí znak rámce, ...)
2	Špatný WID, typ nebo rozměr databázové proměnné vzdálené stanice. Chyba parametrizace databáze, definice proměnné na této a vzdálené stanici se liší.
3	Byla požadována funkce, kterou vzdálená stanice nepodporuje.
4	Neznámá chyba.
5	Vzdálená stanice neodpověděla.
7	Vzdálená stanice odpověděla rámcem nesprávného typu.

# Dodatek: Řídicí znaky v řetězcích

V řetězcových konstantách, které se předávají jako parametr některých funkčních modulů, se mohou uvádět některé speciální znakové sekvence, které se interpretují jako speciální řídicí znaky. Většina těchto sekvencí vychází z řídicích znaků, které jsou standardní v jazyce "C", na základě našich zkušeností, zejména s tiskem, byly přidány další sekvence.

Znakové sekvence z následující tabulky se smějí používat v řetězcových parametrech jen u těch modulů, u kterých je v popisu modulu uveden odkaz na tento dodatek.

Tabulka řídicích znaků

Znaková sekvence	Standard "C"	Převede se na kód		Význam
		Dekadicky	HEXA	
<code>\a</code> <code>\A</code>	ANO	7	07	"alert" - zvuková výstraha
<code>\b</code> <code>\B</code>	ANO	8	08	"backspace" - návrat o jeden znak
<code>\e</code> <code>\E</code>	NE	27	1B	"Escape"
<code>\f</code> <code>\F</code>	ANO	12	0C	"Form feed" - přechod na další stránku
<code>\l</code> <code>\L</code>	NE	10	0A	"Line feed" - přechod na další řádek (bez návratu na začátek řádku)
<code>\n</code> <code>\N</code>	ANO	13 10	0D 0A	"CRLF" - přechod na začátek dalšího řádku
<code>\r</code> <code>\R</code>	ANO	13	0D	"Carriage return" - návrat na začátek řádku
<code>\t</code> <code>\T</code>	ANO	9	09	"Horizontal tab" - přechod na další pozici tabulátoru na řádku
<code>\v</code> <code>\V</code>	ANO	11	0B	"Vertical tab" - přechod na další pozici tabulátoru na stránce
<code>\xnn</code> <code>\Xnn</code>	ANO		nn	Libovolný znak HEXa
<code>\nnn</code>	ANO	nnn		Libovolný znak dekadicky.

Následuje-li za zpětným lomítkem jiný znak než znaky uvedené v tabulce, vyhodnotí se řetězec, jako by v něm zpětné lomítko nebylo. Řetězec "Hello, \world!" se vyhodnotí jako "Hello, world!".

Chceme-li do řetězce opravdu vložit zpětné lomítko, je třeba ho zapsat dvakrát. Řetězec "He\\llo, world!" se vyhodnotí jako "He\llo, world!".

Vzhledem k tomu, že znak s kódem nula se interně využívá pro ukončení řetězce, není možné tento kód do řetězce žádným způsobem (ani pomocí "\x00" nebo "\0") vložit. Řetězec "Hello\0, world!" se vyhodnotí pouze jako "Hello". Potřebujeme-li například na tiskárnu skutečně vyslat řídicí znak NUL, musíme použít modul pracující místo s řetězci s binárními daty (v tomto případě **PrintBin**).

# Dodatek: Čítačové vstupy / vstupy pro inkrementální čidla polohy

---

Pro rychlé (hardwareově podporované) čítačové vstupy a pro vstupy pro inkrementální čidla polohy se používá stejný typ vstupů a stejná sada funkčních modulů **IRCIn**, **IRCMode** a **IRCSet**.

Pomocí parametrů **Režim** a **Hrany** funkčního modulu **IRCMode** se volí, pro jaký konkrétní účel je dotyčný vstup používán, popřípadě jaký typ čidla je k němu připojen.

Různé typy procesních stanic jsou vybaveny různým počtem čítačových vstupů resp. vstupů pro inkrementální čidla polohy, některé typy procesních stanic těmito vstupy nejsou vybaveny vůbec.

Některé vstupy na některých typech procesních stanic nepodporují všechny režimy nebo některé podporují s jistými omezeními.

Tyto skutečnosti jsou popsány v následujícím přehledu.

Hardwareové parametry vstupů, jako jsou napěťové úrovně, vstupní odpory a maximální frekvence vstupních signálů, nejsou předmětem této příručky - najdete je v technické příručce příslušného typu procesní stanice.

## **Procesní stanice typu ADiS (CPU-jednotka AD-CPU166)**

---

Tento typ procesní stanice není vybaven čítačovými vstupy / vstupy pro inkrementální čidla polohy.

## **Procesní stanice typu ADiS-F (CPU-jednotka AD-CPU166F)**

---

Tento typ procesní stanice není vybaven čítačovými vstupy / vstupy pro inkrementální čidla polohy.

## **Procesní stanice typu ADiS167 (CPU-jednotky AD-CPU167 a AD-CPU167/I)**

---

Tyto typy procesních stanic se vybavují čítačovými vstupy, resp. vstupy pro inkrementální čidla polohy, prostřednictvím modulů AD-IRC2. Každý tento modul doplní systém o dva čítačové vstupy, resp. vstupy pro inkrementální čidla polohy. Jejich čísla v rozsahu 0÷15 určuje uživatel v rámci "Konfigurace V/V" v PSE3. Celkem je tedy možno systém vybavit až šestnácti čítačovými vstupy, resp. vstupy pro inkrementální čidla polohy ve formě až osmi modulů AD-IRC.

Upozornění: Modul AD-IRC2, na rozdíl od většiny ostatních modulů systému ADiS167, není možno používat v režimu implicitní konfigurace V/V kanálů.

Kanál "0" modulu AD-IRC2 (horní konektor CANON)		
Číslo vstupu	n - určeno v "Konfiguraci V/V" v PSE3	
Signál F1	signál B, resp. dvojice signálů B-B'	
Signál F2	signál A, resp. dvojice signálů A-A'	
Rozsah interního čítače	32 bitů (-2.147.483.648 ÷ 2.147.483.647)	
Vstupy pro poziční (indexové) značky v automatickém režimu	Jeden (číslo značky 0): signál I, resp. dvojice signálů I-I'	
Podporované kombinace režimu a volby aktivních hran:		
Režim	Hrany	Poznámka
IRC F1-F2	nebere se v úvahu	Jedné úplné periodě vstupních signálů odpovídají 4 jednotkové změny interního čítače. (Čítač se mění při každé hraně kteréhokoliv vstupního signálu)
IRC F2-F1	nebere se v úvahu	
Směr +/-	Náběžná Sestupná	
Směr -/+	Náběžná Sestupná	
Nahoru	Tyto režimy nejsou na tomto typu procesní stanice podporovány. Použijte režimy Směr +/-, resp. Směr -/+, a zkratujte mezi sebou signály B-B'.	
Dolů		
F1+ F2-	Tyto režimy nejsou na tomto typu procesní stanice podporovány.	
F1- F2+		

Kanál "1" modulu AD-IRC2 (spodní konektor CANON)		
Číslo vstupu	n+1 - "n" je určeno v "Konfiguraci V/V" v PSE3	
Signál F1	signál B, resp. dvojice signálů B-B'	
Signál F2	signál A, resp. dvojice signálů A-A'	
Rozsah interního čítače	32 bitů (-2.147.483.648 ÷ 2.147.483.647)	
Vstupy pro poziční (indexové) značky v automatickém režimu	Jeden (číslo značky 0): signál I, resp. dvojice signálů I-I'	
Podporované kombinace režimu a volby aktivních hran:		
Režim	Hrany	Poznámka
Stejně jako u vstupu číslo 0		

Upozornění: V závislosti na zvoleném režimu a aktivních hranách může na každém kanálu LED Bx, popř. i Ax svítit inverzně, tzn. svítit při nepřítomnosti signálu, zhasínat při přítomnosti signálu.

### Procesní stanice typu AMAP98

Tento typ procesní stanice není vybaven čítačovými vstupy / vstupy pro inkrementální čidla polohy.

### Procesní stanice typu AMAP99

Tento typ procesní stanice není vybaven čítačovými vstupy / vstupy pro inkrementální čidla polohy.

### Procesní stanice typu AMiRiS-CA a AMiRiS-X

Tento typ procesní stanice není vybaven čítačovými vstupy / vstupy pro inkrementální čidla polohy.

## Procesní stanice typu AMiRiS99

Tento typ je vybaven dvěma čítačovými vstupy resp. vstupy pro inkrementální čidla polohy, očíslovanými **0** a **1**.

Číslo vstupu	0	
Signál F1	DI0.0	
Signál F2	DI0.1	
Rozsah interního čítače	32 bitů (-2.147.483.648 ÷ 2.147.483.647)	
Vstupy pro poziční (indexové) značky v automatickém režimu	Nejsou. Použijte libovolný volný digitální vstup pro poloautomatický režim obsluhy pozičních značek.	
Podporované kombinace režimu a volby aktivních hran:		
Režim	Hrany	Poznámka
IRC F1-F2	nebere se v úvahu	Jedné úplné periodě vstupních signálů odpovídají 4 jednotkové změny interního čítače. (Čítač se mění při každé hraně kteréhokoliv vstupního signálu)
IRC F2-F1	nebere se v úvahu	
Směr +/-	Náběžná Sestupná Obě hrany	
Směr -/+	Náběžná Sestupná Obě hrany	
Nahoru	Náběžná Sestupná Obě hrany	Vstupní signál F1 nemá vliv na činnost vstupu, odpovídající digitální vstup může být použit obvyklým způsobem.
Dolů	Náběžná Sestupná Obě hrany	
F1+ F2-	Náběžná Sestupná	V okamžiku aktivní hrany jednoho vstupního signálu musí být druhý signál v úrovni logické "0". Jinak vstup reaguje na opačnou hranu, než je požadováno. To může vést ke ztrátě pulsů v případě, že pulsy přicházejí na obou vstupních signálech současně.
F1- F2+	Náběžná Sestupná	

Číslo vstupu	1	
Signál F1	DI0.2	
Signál F2	DI0.3	
Rozsah interního čítače	32 bitů (-2.147.483.648 ÷ 2.147.483.647)	
Vstupy pro poziční (indexové) značky v automatickém režimu	Nejsou. Použijte libovolný volný digitální vstup pro poloautomatický režim obsluhy pozičních značek.	
Podporované kombinace režimu a volby aktivních hran:		
Režim	Hrany	Poznámka
Stejně jako u vstupu číslo 0		

*Pozn.: Bez ohledu na to, jestli se použijí funkční moduly pro využití vstupů DI0.0÷DI0.3 jako čítačových vstupů / vstupů pro inkrementální čidla polohy, jsou signály z těchto vstupů přístupné v odpovídajících logických kanálech DI0 a DI0AC (čísla 0 a 1). Nic tedy nebrání využití těchto signálů jako obvyklých digitálních vstupů obvyklým způsobem.*

## Procesní stanice typu APT2100

Tento typ procesní stanice není vybaven čítačovými vstupy / vstupy pro inkrementální čidla polohy.

## Procesní stanice typu ART267, ART267A

Tento typ je vybaven dvěma čítačovými vstupy, očíslovanými **0** a **1**.

Číslo vstupu	0		
Signál F1	DI0.1		
Signál F2	DI0.0		
Rozsah interního čítače	32 bitů (-2.147.483.648 ÷ 2.147.483.647)		
Vstupy pro poziční (indexové) značky v automatickém režimu	Nejsou. Použijte libovolný volný digitální vstup pro poloautomatický režim obsluhy pozičních značek.		
Podporované kombinace režimu a volby aktivních hran:			
Režim	Hrany	Poznámka	
IRC F1-F2	Tento režim není na tomto typu procesní stanice podporován		
IRC F2-F1	Tento režim není na tomto typu procesní stanice podporován		
Směr +/-	Náběžná Sestupná Obě hrany	Vstupní signál F1 nemá vliv na činnost vstupu, odpovídající digitální vstup může být použit obvyklým způsobem.	
Směr -/+	Náběžná Sestupná Obě hrany		
Nahoru	Náběžná Sestupná Obě hrany		
Dolů	Náběžná Sestupná Obě hrany		
F1+ F2-	Tento režim není na tomto typu procesní stanice podporován		
F1- F2+	Tento režim není na tomto typu procesní stanice podporován		

Číslo vstupu	1	
Signál F1	DI0.3	
Signál F2	DI0.2	
Rozsah interního čítače	32 bitů (-2.147.483.648 ÷ 2.147.483.647)	
Vstupy pro poziční (indexové) značky v automatickém režimu	Nejsou. Použijte libovolný volný digitální vstup pro poloautomatický režim obsluhy pozičních značek.	
Podporované kombinace režimu a volby aktivních hran:		
Režim	Hrany	Poznámka
Stejně jako u vstupu číslo 0		

*Pozn.: Bez ohledu na to, jestli se použijí funkční moduly pro využití vstupů DI0.0÷DI0.3 jako čítačových vstupů / vstupů pro inkrementální čidla polohy, jsou signály z těchto vstupů přístupné v odpovídajících logických kanálech DI0 a DI0AC (čísla 0 a 1). Nic tedy nebrání využití těchto signálů jako obvyklých digitálních vstupů obvyklým způsobem.*



## Procesní stanice typu ART4000, ART4000F a ART4000M

Tyto typy jsou vybaveny dvěma čítačovými vstupy resp. vstupy pro inkrementální čidla polohy, očíslovanými **0** a **1**.

Číslo vstupu	0	
Signál F1	DI0.0	
Signál F2	DI0.1	
Rozsah interního čítače	32 bitů (-2.147.483.648 ÷ 2.147.483.647)	
Vstupy pro poziční (indexové) značky v automatickém režimu	Nejsou. Použijte libovolný volný digitální vstup pro poloautomatický režim obsluhy pozičních značek.	
Podporované kombinace režimu a volby aktivních hran:		
Režim	Hrany	Poznámka
IRC F1-F2	nebere se v úvahu	Jedné úplné periodě vstupních signálů odpovídají 4 jednotkové změny interního čítače. (Čítač se mění při každé hraně kteréhokoliv vstupního signálu)
IRC F2-F1	nebere se v úvahu	
Směr +/-	Náběžná Sestupná Obě hrany	
Směr -/+	Náběžná Sestupná Obě hrany	
Nahoru	Náběžná Sestupná Obě hrany	Vstupní signál F1 nemá vliv na činnost vstupu, odpovídající digitální vstup může být použit obvyklým způsobem.
Dolů	Náběžná Sestupná Obě hrany	
F1+ F2-	Náběžná Sestupná	V okamžiku aktivní hrany jednoho vstupního signálu musí být druhý signál v úrovni logické "0". Jinak vstup reaguje na opačnou hranu, než je požadováno. To může vést ke ztrátě pulsů v případě, že pulsy přicházejí na obou vstupních signálech současně.
F1- F2+	Náběžná Sestupná	

Číslo vstupu	1	
Signál F1	DI0.2	
Signál F2	DI0.3	
Rozsah interního čítače	32 bitů (-2.147.483.648 ÷ 2.147.483.647)	
Vstupy pro poziční (indexové) značky v automatickém režimu	Nejsou. Použijte libovolný volný digitální vstup pro poloautomatický režim obsluhy pozičních značek.	
Podporované kombinace režimu a volby aktivních hran:		
Režim	Hrany	Poznámka
Stejně jako u vstupu číslo 0		

*Pozn.: Bez ohledu na to, jestli se použijí funkční moduly pro využití vstupů DI0.0÷DI0.3 jako čítačových vstupů / vstupů pro inkrementální čidla polohy, jsou signály z těchto vstupů přístupné v odpovídajících logických kanálech DI0 a DI0-AC (čísla 0 a 1). Nic tedy nebrání využití těchto signálů jako obvyklých digitálních vstupů obvyklým způsobem.*

## Procesní stanice typu AMiNi, AMiNi-E, AMiNi2D, AMiNi-T a AMiNi-TE

Tyto typy jsou vybaveny třemi čítačovými vstupy resp. vstupy pro inkrementální čidla polohy, očíslovanými **0**, **1** a **2**.

Číslo vstupu	0	
Signál F1	AMiNi, AMiNi-E: DI0.1 AMiNi-T, AMiNi-TE: DI0.6	
Signál F2	AMiNi, AMiNi-E: DI0.0 AMiNi-T, AMiNi-TE: DI0.7	
Rozsah interního čítače	32 bitů (-2.147.483.648 ÷ 2.147.483.647)	
Vstupy pro poziční (indexové) značky v automatickém režimu	Nejsou. Použijte libovolný volný digitální vstup pro poloautomatický režim obsluhy pozičních značek.	
Podporované kombinace režimu a volby aktivních hran:		
Režim	Hrany	Poznámka
IRC F1-F2	nebere se v úvahu	Jedné úplné periodě vstupních signálů odpovídají 4 jednotkové změny interního čítače. (Čítač se mění při každé hraně kteréhokoliv vstupního signálu)
IRC F2-F1	nebere se v úvahu	
Směr +/-	Náběžná Sestupná Obě hrany	Vstupní signál F1 nemá vliv na činnost vstupu, odpovídající digitální vstup může být použit obvyklým způsobem.
Směr -/+	Náběžná Sestupná Obě hrany	
Nahoru	Náběžná Sestupná Obě hrany	
Dolů	Náběžná Sestupná Obě hrany	
F1+ F2-	Tento režim není na tomto typu procesní stanice podporován	
F1- F2+	Tento režim není na tomto typu procesní stanice podporován	

Číslo vstupu	1	
Signál F1	AMiNi, AMiNi-E: DI0.3 AMiNi-T, AMiNi-TE: DI0.4	
Signál F2	AMiNi, AMiNi-E: DI0.2 AMiNi-T, AMiNi-TE: DI0.5	
Rozsah interního čítače	32 bitů (-2.147.483.648 ÷ 2.147.483.647)	
Vstupy pro poziční (indexové) značky v automatickém režimu	Nejsou. Použijte libovolný volný digitální vstup pro poloautomatický režim obsluhy pozičních značek.	
Podporované kombinace režimu a volby aktivních hran:		
Režim	Hrany	Poznámka
Stejně jako u vstupu číslo 0		

Číslo vstupu	2	
Signál F1	AMiNi, AMiNi-E: DI0.5 AMiNi-T, AMiNi-TE: DI0.2	
Signál F2	AMiNi, AMiNi-E: DI0.4 AMiNi-T, AMiNi-TE: DI0.3	
Rozsah interního čítače	32 bitů (-2.147.483.648 ÷ 2.147.483.647)	
Vstupy pro poziční (indexové) značky v automatickém režimu	Nejsou. Použijte libovolný volný digitální vstup pro poloautomatický režim obsluhy pozičních značek.	
Podporované kombinace režimu a volby aktivních hran:		
Režim	Hrany	Poznámka
Stejně jako u vstupu číslo 0		

*Pozn.: Bez ohledu na to, jestli se použijí funkční moduly pro využití vstupů DI0.0÷DI0.5 jako čítačových vstupů / vstupů pro inkrementální čidla polohy, jsou signály z těchto vstupů přístupné v odpovídajících logických kanálech DI0 a DI0-AC (čísla 0 a 1). Nic tedy nebrání využití těchto signálů jako obvyklých digitálních vstupů obvyklým způsobem.*

---

### **Procesní stanice typu MEST100**

---

Tento typ procesní stanice není vybaven čítačovými vstupy / vstupy pro inkrementální čidla polohy.

---

### **Procesní stanice typu ADOS100/200**

---

Tento typ procesní stanice není vybaven čítačovými vstupy / vstupy pro inkrementální čidla polohy.

---

### **Procesní stanice typu APT3000**

---

Tento typ procesní stanice není vybaven čítačovými vstupy / vstupy pro inkrementální čidla polohy.

---

### **Procesní stanice typu ADiR**

---

Tento typ procesní stanice není vybaven čítačovými vstupy / vstupy pro inkrementální čidla polohy.

# Dodatek: Obsluha sběrnice CAN

---

Sběrnice **CAN** je univerzální rychlá třívodičová sériová sběrnice, primárně určená pro připojování vstupně/výstupních zařízení k řídicím systémům. Lze ji ovšem (v omezené míře, dané malým datovým rozsahem rámců) použít i pro komunikaci mezi řídicími systémy navzájem.

Dvě nejnižší komunikační vrstvy sedmivrstvého referenčního modelu ISO/OSI (fyzická a linková) jsou zajištěny obvodově speciálními řadiči. Třetí až šestá vrstva (síťová, transportní, relační a prezentační) jsou vynechány (jak je u jednoduchých lokálních sběrnic obvyklé). Pro sedmou komunikační vrstvu (aplikační) sběrnice CAN existuje několik úrovní standardizace, z nichž nejvyšší představuje standard **CANopen**. Tento standard určuje jednotné komunikační prostředí pro zařízení různých výrobců. Komunikační knihovna **CAN** (součást standardní instalace PSP3 počínaje verzí 3.20) je postavena právě na standardu CANopen, což zajišťuje možnost propojení se širokou škálou zařízení jak z produkce firmy AMiT, tak i jiných výrobců. Nutnou a postačující podmínkou připojitelnosti jakéhokoliv zařízení k řídicím systémům firmy AMiT, obsahujícím řadič sběrnice CAN, je podpora standardu CANopen dotyčným zařízením, a to alespoň na úrovni Minimum Capability Device.

Nižší úroveň standardizace protokolu aplikační vrstvy představuje standard **CAL** (CAN Application Layer). Zařízení, u nichž je specifikována pouze podpora standardu CAL a nikoliv CANopen, zpravidla nelze pomocí knihovny **CAN** k řídicím systémům firmy AMiT připojit, nebo jen v omezené míře (díky specifickým vlastnostem implementace CANopen na zařízeních firmy AMiT, viz níže). Každý konkrétní případ je vhodné předem konzultovat s odborníkem z firmy AMiT.

(Jelikož protokoly standardu CANopen představují nadmnožinu protokolů standardu CAL, můžeme se ve specifikaci zařízení podporujících CANopen setkat s inzerováním podpory obou těchto standardů.)

Představitelem standardů aplikační vrstvy sběrnice CAN nezávislých na CAL a CANopen je např. **DeviceNet**. Zařízení odpovídající tomuto nebo jiným standardům nelze připojit k řídicím systémům firmy AMiT pomocí knihovny **CAN**.

## Základy sběrnice CAN

---

Jednotlivá zařízení na sběrnici se nazývají **uzly (node)** a mají přiděleny číselné identifikátory **Node-Id**.

Datové rámce přenášené po sběrnici se nazývají **komunikační objekty (communication object)** a mají rovněž přiděleny číselné identifikátory **COB-Id** (communication object identifier). Každý komunikační objekt je vlastněn jedním uzlem sběrnice, ostatní uzly mohou data tohoto objektu číst nebo do nich zapisovat podle typu objektu. COB-Id objektu se zpravidla odvozuje z typu objektu a identifikátoru uzlu (Node-Id), toto přiřazení ovšem může být změněno vrstvou DBT.

Jeden komunikační objekt umožňuje přenést maximálně 8 byte dat.

Komunikační objekty se dělí do skupin, jako například:

**SDO** (Service Data Object) - **objekty servisních dat** - slouží pro správu sběrnice a přenos stavových informací

**PDO** (Process Data Object) - **objekty procesních dat** - slouží k přenášení vlastních procesních (např. vstup/výstupních) dat po sběrnici.

## Standard CANopen

---

Samotná sběrnice CAN je typu peer-to-peer, tzn. že se na ní v principu nerozlišuje master a slave. V jednotlivých vrstvách aplikačních protokolů CANopen ovšem přesto z logiky věci vyplývá role mastera a slavea pro jednotlivé uzly sběrnice.

Na několika místech se u sběrnice CAN setkáme s termínem **Minimum Capability Device**. Toto je standardem specifikováno jako určitá minimální podmnožina všech vlastností definovaných standardem CANopen, kterou musí zařízení podporovat, aby bylo připojitelné ke sběrnici CAN, řízené dle standardu CANopen. Do této minimální

podmnožiny patří například alespoň omezená podpora vrstvy NMT (viz níže) a implicitní mapování objektů procesních dat.

Aplikační protokoly standardu CANopen jsou postaveny na vrstvě **CMS** (CAN Messaging System) standardu CAL. Lze je rozdělit opět do několika vrstev:

#### Vrstva správy sítě **NMT** (Network Management)

Povinná součást implementace sběrnice CAN. Slouží k zapojování uzlů do sběrnice, udržování a sdílení informací o stavu jednotlivých uzlů, detekci výpadků a ztráty spojení, apod. Právě jedno zařízení na sběrnici musí být masterem vrstvy NMT (**NMT-Master**), ostatní zařízení jsou vzhledem k vrstvě NMT slavey (**NMT-Slave**). NMT-Masterem je zpravidla jeden z řídicích systémů, připojených na sběrnici.

Vrstva NMT využívá několika objektů servisních dat (SDO).

Vrstva NMT může být implementována na dvou úrovních - může být úplná (**NMT\_Full**) nebo na úrovni tzv. Minimum Capability Device (**NMT\_MCD**). U připojovaného zařízení je třeba najít v dokumentaci, jakou úroveň vrstvy NMT podporuje, a tuto úroveň navolit při parametrizaci vrstvy NMT v PSP3 (parametr **MCD** funkčního modulu **CAN\_Node**). Pokud zařízení umožňuje zvolit činnost na obou úrovních, je z hlediska kvality správy sítě vždy vhodné zvolit úplnou variantu. Některá zařízení (např. ADC-CAN firmy AMiT a další zařízení jiných výrobců) jsou specifikována jako zařízení třídy Minimum Capability Device, a přesto podporují úplnou implementaci vrstvy NMT.

Na straně řídicího systému se NMT-vrstva parametrizuje pomocí modulu **CAN\_NMT\_M** (NMT-Master) nebo **CAN\_NMT\_S** (NMT-Slave) ve spojení s příslušným počtem instancí modulu **CAN\_Node**, kterými se vytváří seznam ostatních uzlů přítomných na sběrnici.

Jedním z důležitých úkolů vrstvy NMT je **Node-Guarding**. Spočívá v periodickém dotazování NMT-Slavea na jeho stav. Při inicializaci NMT-Slavea určí NMT-Master periodu, ve které toto dotazování bude probíhat (**Guard-Time**) a násobek této periody, po jehož uplynutí bez dotazu má NMT-Slave detekovat ztrátu spojení s NMT-Masterem (**LifeTime-Factor**). Při detekci ztráty spojení NMT-Slave přechází do tzv. bezpečného stavu, tj. např. nastavuje své výstupy do vhodně určeného bezpečného stavu, při kterém nehrozí poškození připojené technologie. Způsob určení tohoto bezpečného stavu závisí na konkrétním NMT-Slaveovi. Při inicializaci NMT-Slavea tento nejprve navrhne hodnoty Guard-Time a LifeTime-Factoru podle své parametrizace, načež mu NMT-Master tyto hodnoty určí, přičemž navrhované hodnoty může a nemusí brát v úvahu (NMT-Master realizovaný funkčním modulem **CAN\_NMT\_M** je v úvahu nebere). Rozhodující jsou hodnoty určené NMT-Masterem.

#### Vrstva objektů procesních dat **PDO** (Process Data Objects)

Tato vrstva bývá často také označována (mírně nepřesně) jako vrstva **CMS**.

Podle specifikace CANopen Minimum Capability Device má každý uzel přidělen čtyři (po jednom pro AI, AO, DI a DO) identifikátory objektů procesních dat (PDO), odvozené z čísla uzlu podle standardizovaného schématu.

Vlastníkem PDO je vždy vstupně/výstupní zařízení, jehož vstupům nebo výstupům daný PDO odpovídá. Uzel vlastníci nějaký PDO (zpravidla vstupně/výstupní zařízení) můžeme pro zjednodušení popisu nazvat termínem **PDO-Slave**, ostatní zařízení jsou vzhledem k tomuto PDO v pozici, označované jako **PDO-Master**.

- ♦ Vstupní PDO (pro analogové a digitální vstupy) je vlastněn vstupním zařízením (uzlem), ostatní uzly mohou tento PDO přijímat buď tak, že je vstupní uzel sám vyšle na sběrnici (např. při změně hodnoty signálu), nebo si mohou jeho zaslání vyžádat vysláním tzv. remote frame.

PDO-Master se na straně řídicího systému parametrizuje pomocí modulů **CAN\_AI**, **CAN\_DI**, popř. **CAN\_PDO** v režimu "příjem".

Má-li řídicí systém vystupovat jako PDO-Slave (což se používá pro předávání s jiným ř.s.), parametrizuje se pomocí modulů **CAN\_AI\_S**, **CAN\_DI\_S**, popř. **CAN\_PDO** v režimu

“odpovídač”.

- ♦ Výstupní PDO (analogové a digitální výstupy) je vlastněn výstupním zařízením (uzlem), jiný uzel může do tohoto PDO po sběrnici zapisovat. PDO-Master se na straně řídicího systému parametrizuje pomocí modulů **CAN\_AO**, **CAN\_DO**, popř. **CAN\_PDO** v režimu “vysílání”. Má-li řídicí systém vystupovat jako PDO-Slave (což se používá pro předávání s jiným ř.s.), parametrizuje se pomocí modulů **CAN\_AO\_S**, **CAN\_DO\_S**, popř. **CAN\_PDO** v režimu “příjem”.

Je třeba mít na paměti, že pozice PDO-Master/PDO-Slave není nijak svázána s pozicí NMT-Master/NMT-Slave. Vstupně/výstupní zařízení je zpravidla NMT-Slave a zároveň PDO-Slave všech svých PDO. Řídicí systém ovšem může být PDO-Master vzdálených PDO i přesto, že je NMT-Slave. Řídicí systém může být současně PDO-Masterem některých PDO a PDO-Slavem jiných PDO (to se používá např. pro komunikaci mezi dvěma ř.s.), to vše bez ohledu na to, jestli je NMT-Masterem nebo NMT-Slavem.

Vrstva PDO je svázána s vrstvou NMT, protože je třeba zajistit, aby se moduly vrstvy PDO nepokoušely přenášet objekty, jejichž vlastníci (PDO-Slave) nejsou na sběrnici v daném okamžiku připojeni. Tato vazba se zajišťuje pomocí čísla uzlu PDO-Slave, které je parametrem každého funkčního modulu vrstvy PDO. Tyto moduly před vydáním každého komunikačního požadavku zjišťují u vrstvy NMT stav příslušného PDO-Slavea a vydání komunikačního požadavku podmiňují zapojením PDO-Slavea do sběrnice.

#### Další vrstvy

Existuje specifikace dalších vrstev aplikačních protokolů pro speciální účely, např. **DBT** (distributor), **LMT** (Layer Management). Toto jsou nepovinné součásti implementace CANopen a knihovna **CAN** je nepodporuje.

### Specifické vlastnosti implementace CANopen na zařízeních firmy AMiT

#### Obsluha jednoho PDO více funkčními moduly

Jeden objekt procesních dat (PDO) může obsahovat až 8 byte dat, přičemž jejich struktura může záviset na konkrétním PDO-Slaveovi. Řídicí systémy AMiT přitom pracují s databázovými proměnnými o velikosti maximálně 4 byte. Aby bylo možno přijímat či vysílat PDO o větší velikosti, je možno buďto použít maticovou proměnnou (ze které se vezme potřebný počet sloupců v jednom řádku podle velikosti PDO) nebo zpracování PDO rozložit do několika volání příslušného funkčního modulu vrstvy PDO. Tímto způsobem lze rovněž zpracovat nepravidelnou strukturu dat některého PDO.

Jestliže má větší počet modulů vrstvy PDO v parametrech uveden stejný COB-Id (ten je u modulů **CAN\_xy**, a **CAN\_xy\_S** jednoznačně odvozen z typu modulu a z Node-Id), pracují ve skutečnosti s tímtež PDO. Mohou nezávisle na sobě pracovat s různými oblastmi dat tohoto PDO.

Příklad1: PDO-Slave s Node-Id 1 zasílá v PDO digitálních vstupů 32 datových bitů (4 byte) a potom jeden stavový byte. Níže uvedený příklad načte datové bity do proměnných `MyIntVar1` a `MyIntVar2`, stavový byte do proměnné `MyIntVar3`:

```
CAN_DI :17001, 1, 1, NONE.0, 2, 0, MyIntVar1[0,0]
CAN_DI :17001, 1, 0, NONE.0, 2, 2, MyIntVar2[0,0]
CAN_DI :17001, 1, 0, NONE.0, 1, 4, MyIntVar3[0,0]
```

Jelikož všechny moduly jsou typu DI a mají stejný Node-Id, mají také stejný COB-Id a tedy pracují se stejným objektem. První modul zpracovává nultý a první byte dat, druhý modul druhý a třetí byte, třetí modul čtvrtý byte. Na pořadí modulů nezáleží.

Příklad2: PDO-Slave s Node-Id 3 očekává v PDO analogových výstupů 24 bitů hodnoty pro D/A převodník a potom jednobyteový příkaz. Níže uvedený příklad seskládá PDO dle těchto požadavků z proměnné `MyFltVar` (analogová hodnota) a `MyIntVar` (příkaz).

```
CAN_AO :17001, 3, 0, NONE.0, 0x0018, 0, MyFltVar[0,0], 5, 0.0, 5.0, 0.0, 100.0
CAN_PDO :17001, 3, 771, Vysílání, 1, NONE.0, 1, 3, MyIntVar[0,0]
```

Jelikož oba moduly pracují s PDO s COB-Idem 771 (u modulu CAN\_AO to nepřímo vyplývá z Node-Idu), pracují se stejným objektem. První modul vyplňuje nultý až druhý byte dat, druhý modul třetí byte. Na pořadí modulů nezáleží.

Chceme-li se vyhnout použití nekomplikovanějšího (protože nejobecnějšího) modulu **CAN\_PDO**, můžeme výše uvedený příklad přepsat s použitím modulu **CAN\_AO** (nemůžeme použít **CAN\_DO**, protože ten pracuje s úplně jiným komunikačním objektem):

```
CAN_AO :17001, 3, 0, NONE.0, 0x0018, 0, MyFltVar[0,0], 5, 0.0, 5.0, 0.0, 100.0
CAN_AO :17001, 3, 0, NONE.0, 0x0008, 3, MyIntVar[0,0], 256, 0.0, 256, 0.0, 256
```

#### Virtuální uzly

Existují případy (např. zařízení ADC\_CAN s více než osmi moduly DO nebo více než osmi moduly DI), kdy jedno zařízení využívá více než jeden PDO jenoho typu (např. typu DO). Toto je mírně nestandardní rozšíření specifikace Minimum Capability Device dle standardu CANopen. Takové zařízení se pak z hlediska vrstvy PDO chová, jako by se jednalo o dvě zařízení s různými Node-Idy (zpravidla za sebou následujícími), ovšem z hlediska vrstvy NMT se stále jedná jen o jedno zařízení. Druhý, rozšiřující uzel není možno ve vrstvě NMT parametrizovat jako běžný uzel sběrnice, protože by se vrstvě NMT nikdy nepodařilo tento uzel inicializovat a komunikace s ním by tedy nebyla možná. Je tedy třeba vrstvě NMT oznámit, že se na sběrnici nachází tzv. virtuální uzel, jehož stav je určen stavem jiného uzlu. K tomu slouží parametr *Virtuální* funkčního modulu **CAN\_Node**. U běžných uzlů sběrnice se tento parametr ponechává na implicitní nulové hodnotě (zobrazí se jako *NeníVirt*), u virtuálních uzlů se zadá Node-Id běžného uzlu, se kterým je virtuální uzel svázán.

Příklad: Na sběrnici je uzel typu ADC\_CAN se 16ti moduly DO s Node-Idem 5 (to odpovídá stavu DIP přepínačů OFF ON OFF OFF OFF ...). Pro devátý až šestnáctý modul DO se na tomto uzlu automaticky vytváří virtuální uzel s Node-Idem 6. V PSP3 se tento stav naparametrizuje následovně:

```
17001 CNC_C167      125kbit/s
17002 CAN_NMT_M      :17001
      CAN_Node       :17002, 5, 500, 4, NONE.0, NeníVirt, NMT_Full
      CAN_Node       :17002, 6, 500, 4, NONE.0, 5, NMT_Full
```

#### Nenakonfigurované uzly

Knihovna **CAN** umožňuje komunikovat i s uzly, které vůbec nejsou uvedeny v seznamu funkčních modulů **CAN\_Node** příslušných k použitému NMT-modulu - nenakonfigurované uzly. Vrstva NMT stav nenakonfigurovaných uzlů stále vyhodnocuje jako "úspěšně připojen ke sběrnici", a proto nebrání přenosům PDO do a z těchto uzlů, pokud samotný řídicí systém je ke sběrnici úspěšně připojen.

To se používá ve dvou případech:

- Připojujeme-li se k zařízení, které neodpovídá standardu CANopen, resp. jeho vrstva NMT neodpovídá ani specifikaci NMT\_Full ani NMT\_MCD. Zařízení musí být připraveno ke komunikaci bez požadavku na inicializaci NMT-Masterem. Pokud číslování PDO daného zařízení odpovídá schématu dle specifikace CANopen Minimum Capability Device, používáme ke komunikaci s takovýmto zařízením funkční moduly **CAN\_xy**, přičemž použitý Node-Id neuvedeme v žádném modulu **CAN\_Node**. Pokud číslování PDO daného zařízení neodpovídá schématu dle specifikace CANopen Minimum Capability Device, používáme ke komunikaci s takovýmto zařízením funkční modul **CAN\_PDO** se smyšlenou hodnotou parametru Node-Id, která se nevyskytuje v žádném modulu **CAN\_Node**.
- Je-li PDO-Slavem některého PDO řídicí systém, který je současně NMT-Masterem sběrnice CAN, nemá tento NMT-Master přidělen žádný Node-Id. Pro funkční moduly **CAN\_xx\_S** na NMT-Masterovi použijeme libovolný nepoužitý Node-Id a na NMT-Slaveovi stejný Node-Id uvedeme v korespondujícím funkčním modulu **CAN\_xx**. Tento Node-Id

ovšem neuvádíme v žádném modulu **CAN\_Node** ani na jednom ř.s. NMT-Master nepotřebuje inicializovat ani testovat sám sebe, NMT-Slave nepotřebuje testovat, že je NMT-Master připojen ke sběrnici. Pokud je NMT-Slave sám zinicializován, je to dostatečným důkazem, že NMT-Master je funkční a připojený.



# Dodatek: Obsluha sítě vzdálených vstup/výstupních zařízení ARION

ARION je protokol rychlé sériové sítě vstup/výstupních zařízení. Reprezentanty periférií, které je možno na tuto síť připojit, jsou DINDO24, DINA024U, DINA024I, DINDI24A a DINAI24.

Výše uvedená zařízení jsou vybavena komunikačním rozhraním RS485, protokol ARION (a moduly knihovny **ARION**) však umožňuje komunikovat i po komunikačních rozhraních RS232 nebo RS422.

## Základní vlastnosti protokolu ARION

Protokol ARION umožňuje připojení:

- až 63 vstup/výstupních zařízení (uzlů) na komunikační rozhraní RS485 v režimech HalfDuplex nebo Autonomous, nebo
- až 63 výstupních zařízení (uzlů) na komunikační rozhraní RS232/RS422 v režimu Simplex, nebo
- jedno vstupní zařízení (uzel) na komunikační rozhraní RS232/RS422 v režimech Duplex, HalfDuplex nebo Autonomous, nebo
- jedno výstupní zařízení (uzel) na komunikační rozhraní RS232/RS422 v režimech Duplex nebo HalfDuplex

Protokol ARION dále obsahuje mechanismus kontroly uzlu. Ten slouží k tomu, aby podřízený uzel po uplynutí předem stanovené doby od posledního přijatého rámce detekoval, že došlo ke ztrátě spojení s řídicím systémem. Této detekce mohou zejména výstupní uzly využít k uvedení svých výstupů do předem stanoveného bezpečného stavu, ve kterém nehrozí riziko poškození ovládané technologie.

Podrobný popis protokolu ARION je k dispozici v samostatné dokumentaci.

## Volba komunikačního režimu

Komunikační režim se volí podle použitého komunikačního rozhraní a podle požadavků na rychlost přenosu dat, případně indikaci stavu přenosu.

Některé režimy mají dvě varianty, zakončené čísly 2 a 4. Toto rozlišení slouží pro podřízená zařízení vybavená více komunikačními rozhraními, aby se jim dalo najevo, které komunikační zařízení mají pro komunikaci s řídicím systémem používat. "2" znamená RS232/RS422, "4" znamená RS485.

Režim	Použitelný na rozhraních	Popis
Simplex2 Simplex4	RS232/RS422 RS485	Režim Simplex: Řídicí systém vysílá data do výstupních zařízení, zařízení neposílají odpověď. Režim není použitelný pro vstupní zařízení. Poskytuje nejrychlejší možnou obsluhu výstupních zařízení, ale neposkytuje žádnou zpětnou vazbu o provozuschopnosti zařízení. Dále prakticky znemožňuje využít mechanismus kontroly uzlu - je sice teoreticky možné stanovit podřízeným uzlům čas kontroly, po kterém přejdou do bezpečného stavu, ale řídicí systém, který nemá informaci o ztrátě spojení, neprovede reinicializaci takového uzlu, takže by vznikla neřešitelná situace. Tento režim jako jediný umožňuje připojit více (výstupních) zařízení na jednu linku RS232/RS422.

Režim	Použitelný na rozhraních	Popis
HalfDuplex2 HalfDuplex4	RS232/RS422 RS485	Režim HalfDuplex: Řídicí systém vysílá data do výstupních zařízení, zařízení posílají odpověď. Vstupních zařízení se řídicí systém dotazuje na data. Odpovědi se posílají s malou časovou prodlevou, aby se zabránilo kolizi přijímaných a vysílaných dat na poloduplexním komunikačním rozhraní, jako je RS485. Z hlediska rychlosti se jedná o kompromis mezi režimy Simplex a Autonomous, zvláště výhodný ve smíšených sítích vstupních a výstupních zařízení. Plná zpětná vazba o provozuschopnosti zařízení, plná využitelnost mechanismu kontroly uzlu.
Duplex	RS232/RS422	Režim Duplex: Podobný mechanismus jako v režimu HalfDuplex, ale využívá výhody plně duplexního komunikačního rozhraní, umožňujícího souběžné vysílání odpovědí a dalších dotazů. Nejrychlejší režim pro připojení jediného zařízení k řídicímu systému. Plná zpětná vazba o provozuschopnosti zařízení, plná využitelnost mechanismu kontroly uzlu.
Autonom2 Autonom4	RS232/RS422 RS485	Režim Autonomous: Výstupní zařízení vysílají data z vlastní iniciativy v časových oknech, určených řídicím systémem, s minimálními prodlevami. Odpadá komunikační režie dotazů. Je-li síť v režimu Autonomous, tak výstupní zařízení pracují v režimu HalfDuplex. Nejrychlejší režim pro připojení více než jednoho vstupního zařízení, oproti režimu HalfDuplex má menší datovou průchodnost pro výstupní zařízení. Vhodný pro síť složené jen ze vstupních zařízení nebo takové sítě, na nichž vstupním zařízením dáváme vyšší prioritu než výstupním. Plná zpětná vazba o provozuschopnosti zařízení, plná využitelnost mechanismu kontroly uzlu.

#### Konstrukce aplikace využívající protokol ARION

Základem obsluhy protokolu ARION je funkční modul **ARION**. Zprostředkovává vazbu na konkrétní komunikační rozhraní, zajišťuje správu sítě a funkci mechanismu kontroly uzlu. Nezbytným doplňkem modulu **ARION** je modul **ARN\_NODE**. Vložením patřičného počtu modulů **ARN\_NODE** se modulu **ARION** vytváří seznam uzlů sítě, které má obsluhovat. Vlastní přenos dat zajišťují funkční modul **ARN\_AI**, **ARN\_AO**, **ARN\_DI** a **ARN\_DO**. Pro definici bezpečných hodnot, které mají být vystaveny na výstupy výstupních zařízení v případě ztráty spojení s řídicím systémem, slouží moduly **ARN\_SfAO** a **ARN\_SfDO**. Podrobněji viz popis výše uvedených modulů.

Způsob konstrukce aplikace využívající protokol ARION ukážeme na příkladu. Jedná se o obsluhu jednoho uzlu typu DINDO24 včetně definice bezpečných stavů a kontroly uzlu:

```
ProcInit:
17001 ARION          1, 19200, HalfDupl4
17002 ARN_NODE       :17001, 1, 100, @StavDO, DO, 24, 0x0000
                    ARN_SfDO   :17002, 0, @InitDO1, 16, 0, SafeVal1[0,0]

Proc00:
                    ARN_DO      :17002, 0, @StavDO1, 16, 0, Val1[0,0]
```

Podrobnější popis viz příklady v popisech jednotlivých použitých modulů.

# Dodatek: Použití digitálních výstupů jako frekvenčních nebo impulzních výstupů

---

Vybrané digitální výstupy řídicích systémů mohou být použity pro generování požadované frekvence od jednotek hertzů řádově do kilohertzů. Je nutno též vzít v úvahu frekvenční rozsah a zpoždění hran výstupního obvodu daného řídicího systému (hardwareové otázky nejsou předmětem tohoto dokumentu, tyto informace hledejte v technické příručce konkrétního řídicího systému).

Kromě požadované frekvence je možno generovat také série požadovaného množství impulzů požadovaného tvaru.

Rovněž je možné tytéž výstupy použít pro generování pulzní šířkové modulace (PWM).

## Použitelné digitální výstupy

Pro frekvenční/impulzní výstupy se používá hardwareová podpora procesoru C166/C167 (capture/compare jednotka), která zajišťuje vysokou přesnost časování jednotlivých hran signálu. Tato hardwareová podpora je vázána na konkrétní signály procesoru, proto lze využít pouze ty digitální výstupy, které jsou připojeny na tyto konkrétní signály. Které výstupy to jsou na kterém typu řídicího systému, najdete v přehledu níže v tomto dodatku. Sdílení capture/compare jednotky pro více výstupů, které obecně mohou mít v jednom okamžiku různé frekvence, vyžaduje při jejich využití přístup, při kterém je nutná softwareová obsluha při každé hraně výstupního signálu. Tato obsluha je realizována knihovnou PSE, uživatel se jí nemusí zabývat. Přesto je to třeba mít na paměti při rozvaze použitých výstupních frekvencí, protože frekvenční výstupy v tomto režimu spotřebovávají výkon procesoru, a to tím více, čím vyšší frekvence, resp. kratší impulzy, se generují.

## Kompatibilita s jinými funkcemi řídicího systému

**POZOR:** Frekvenční/impulzní výstupy nelze používat na systémech komunikujících po Ethernetu v rámci sítě DB-Net/IP, a to ani při přímém připojení systému na Ethernet, ani při použití převodníku 232TOETH.

## Funkční moduly pro frekvenční a impulzní výstupy

- ♦ Pro generování proměnné frekvence použijte modul **FreqOut** s proměnnou hodnotou parametru *Frekvence*, zpravidla s pevnou hodnotou parametru *Střída*.
- ♦ Pro generování impulzů (sérií impulzů) požadovaného tvaru použijte modul **PulseOut**.
- ♦ Pro generování pulzní šířkové modulace (PWM) použijte modul **FreqOut** s proměnnou hodnotou parametru *Střída*, zpravidla s pevnou hodnotou parametru *Frekvence*.

## Vztah frekvenčních / impulzních výstupů a standardní obsluhy digitálních výstupů

Signál použitý kdekoli v aplikaci v modulu **FreqOut** nebo **PulseOut** už nemůže být obsluhován pomocí modulů **DigOut**, **BinOut** apod. Modul **BinOut** nemá na tento signál žádný efekt, modul **DigOut** zapisuje jen do signálů daného kanálu, které nejsou použity v žádném modulu **FreqOut** ani **PulseOut**.

## Rozlišovací schopnost a nutnost předběžného určení maximální délky impulzů

Aby bylo možno správně inicializovat časovač použitý pro generování frekvencí, je třeba už ve fázi návrhu aplikace určit mezní (nejdelší) periodu, která může být na všech frekvenčních / impulzních výstupech v celé aplikaci použita. K tomu slouží parametr

MinFrekv modulu **FreqOut** (použije se jeho převrácená hodnota) a parametr MaxDélka modulu **PulseOut**. Tyto parametry zároveň přímo ovlivňují dosaženou rozlišovací schopnost. Chceme-li dosáhnout co nejlepší přesnosti generovaných frekvencí, resp. délek impulzů, měli bychom se předem důkladně zamyslet nad tím, jako nejnižší frekvenci, resp. jaký nejdelší impuls / mezeru mezi impulzy, budeme potřebovat, a nenastavovat parametr MinFrekv na zbytečně nízkou hodnotu, resp. parametr MaxDélka na zbytečně vysokou hodnotu.

Hodnota parametrů MinFrekv, resp. MaxDélka určuje zároveň rozlišovací schopnost použitého časovače. Ta bude nejvýše rovna maximální periodě dělené 32768, nejméně však 400 ns.

Maximální periodou se rozumí maximum ze všech převrácených hodnot parametrů MinFrekv ve všech použitých modulech **FreqOut**, jakož i ze všech hodnot parametrů MaxDélka ve všech použitých modulech **PulseOut**.

Tímto způsobem je délka generovaných pulzů omezena shora. Délka jednoho trvání stavu ON nebo OFF dále nesmí přesáhnout 1.66777 s. Celá perioda signálu tedy nemůže přesáhnout 3.35544 s, takže minimální generovatelná frekvence je 0.2980233 Hz.

Omezení délky generovaných pulzů zdola je dáno napevno kódem příslušných modulů, a to tak, že délka jednoho trvání stavu ON nebo OFF musí být nejméně 10  $\mu$ s. Pro modul **FreqOut** to znamená omezení generované frekvence na 50 kHz při střídě 50 %, při jiné střídě se v okamžiku, kdy by se délka generovaných pulzů zkrátila pod 10  $\mu$ s, generuje trvalý stav OFF (pro střídu menší než 50 %), resp. trvalý stav ON (pro střídu větší nebo rovnou 50 %).

---

### Procesní stanice typu ADiS

Tento typ řídicího systému není vybaven žádnými digitálními výstupy použitelnými jako frekvenční / impulzní výstupy.

---

### Procesní stanice typu ADiS-F

Tento typ řídicího systému není vybaven žádnými digitálními výstupy použitelnými jako frekvenční / impulzní výstupy.

---

### Procesní stanice typu ADiS167

Tento typ řídicího systému není vybaven žádnými digitálními výstupy použitelnými jako frekvenční / impulzní výstupy.

---

### Procesní stanice typu AMAP98

Jako frekvenční nebo impulzní výstupy je možno použít signály 2 a 3 logického kanálu digitálních výstupů číslo 4.

---

### Procesní stanice typu AMAP99

Jako frekvenční nebo impulzní výstupy je možno použít všechny 4 signály logického kanálu digitálních výstupů číslo 4.

---

### Procesní stanice typu AMiRiS-CA a AMiRiS-X

Jako frekvenční nebo impulzní výstupy je možno použít prvních 5 signálů (signály 0 až 4) logického kanálu digitálních výstupů číslo 0.

*Pozn.: V případě, že je jako výstupní jednotka použito zařízení AREL7S2P-X, tak použití tohoto zařízení pro frekvenční / impulzní výstupy obecně nelze doporučit.*

---

### Procesní stanice typu AMiRiS99

Jako frekvenční nebo impulzní výstupy je možno použít:

- ♦ nejvyšší 4 signály (signály 4 až 7) logického kanálu digitálních výstupů číslo 0.
- ♦ všech 8 signálů logického kanálu digitálních výstupů číslo 1.

---

### Procesní stanice typu APT2100

Tento typ řídicího systému není vybaven žádnými digitálními výstupy použitelnými jako frekvenční / impulzní výstupy.

### **Procesní stanice typu ART267, ART267A**

---

Jako frekvenční nebo impulzní výstupy je možno použít všech 8 signálů logického kanálu digitálních výstupů číslo 0.

### **Procesní stanice typu ART4000, ART4000F a ART4000M**

---

Jako frekvenční nebo impulzní výstupy je možno použít všech 8 signálů logického kanálu digitálních výstupů číslo 0.

*Pozn.: U řídicího systému ART4000M je teoreticky možno použít i všech 8 signálů logického kanálu digitálních výstupů číslo 1, ovšem v případě, že je jako výstupní jednotka použito zařízení AREL7S2P-X, tak použití tohoto zařízení pro frekvenční / impulzní výstupy obecně nelze doporučit (na žádném kanále).*

### **Procesní stanice typu AMiNi, AMiNi-E a AMiNi2D**

---

Jako frekvenční nebo impulzní výstupy je možno použít všech 8 signálů logického kanálu digitálních výstupů číslo 0.

### **Procesní stanice typu AMiNi-T a AMiNi-TE**

---

Tento typ řídicího systému není vybaven žádnými digitálními výstupy použitelnými jako frekvenční / impulzní výstupy.

### **Procesní stanice typu MEST100**

---

Tento typ řídicího systému není vybaven žádnými digitálními výstupy použitelnými jako frekvenční / impulzní výstupy.

### **Procesní stanice typu ADOS100/200**

---

Tento typ řídicího systému není vybaven žádnými digitálními výstupy použitelnými jako frekvenční / impulzní výstupy.

### **Procesní stanice typu APT3000**

---

Tento typ řídicího systému není vybaven žádnými digitálními výstupy použitelnými jako frekvenční / impulzní výstupy.

### **Procesní stanice typu ADiR**

---

Jako frekvenční nebo impulzní výstupy je možno použít signály IO2, IO3 a IO4 (logický kanál DO0, signály DO0.2 až DO0.4). Pro tento účel musí být použitý signál nakonfigurován jako výstupní použitím funkčního modulu **ChanMode**.

# Dodatek: Zabezpečení systému hlídacím časovačem (watchdogem)

---

Hlídací časovač (watchdog) je zařízení určené k ochraně systému proti hardwareovému a do jisté míry i softwareovému selhání, které naruší normální běh programu. Chrání systém tím způsobem, že pokud software do určité předem stanovené limitní doby nevykoná předem stanovenou akci (tzv. obsluha hlídacího časovače), hlídací časovač (nezávislý na software) restartuje systém. To dává šanci detekovat chybové chování systému, případně učinit opatření k nápravě.

Všechny řídicí systémy firmy AMIT jsou vybaveny hlídacím časovačem (je přímo součástí použitých procesorů).

## Problémy při využívání hlídacích časovačů

Využití hlídacích časovačů ve volně programovatelných řídicích systémech obecně přináší několik **problémů**:

- a) Maximální doba, na kterou lze nastavit limit pro periodickou obsluhu hlídacího časovače, bývá pro některé účely příliš krátká. To často programátory vede ke vkládání kódu obsluhy na mnoho míst v programu, čímž se podstatně redukuje míra spolehlivosti, kterou použití hlídacího časovače přinese.
- b) Jeden společný časovač je obtížné správně použít pro hlídání bezchybné funkce většího množství programových vláken, popř. přerušovacích rutin obsluhujících různé externí události.
- c) V případě událostí, které mají opakovaně nastávat jen za určitého stavu systému, není obvykle možné hlídací časovač krátkodobě deaktivovat v ostatních stavech.
- d) U volně programovatelných řídicích systémů se vyskytují situace, že na úrovni operačního systému nelze určit časový limit pro provedení určité části kódu - tento limit může určit jedině autor aplikace.

## Mechanismus logických hlídacích časovačů

Jako odpověď výše zmíněné problémy byl v operačním systému NOS od V3.25 implementován mechanismus logických hlídacích časovačů. V rámci tohoto mechanismu rozeznáváme:

- ♦ **Fyzický** hlídací časovač **FHČ** (nezávislý na software, v celém systému jen jeden). Limitní doba jeho obsluhy se řídí omezeními použitého hardware. Není přímo obsluhován ani uživatelem, ani operačním systémem s výjimkou speciálního servisního procesu.
- ♦ **Logické** hlídací časovače **LHČ**. Je jich více, jeden pro každou hlídanou programovou smyčku nebo pro každou obslužnou rutinu periodické události. Jsou navzájem odlišeny číslem kanálu. Jsou využívány operačním systémem, nebo aplikací. Každý z nich má vlastní limit, do kdy musí proběhnout jejich obsluha. Tento limit se neřídí hardwareovými omezeními. Jednotlivé LHČ je možno i deaktivovat, tedy uvést do stavu, kdy jejich další obsluha není vyžadována. Reaktivují se první obsluhou.
- ♦ **Uživatelské** hlídací časovače **UHČ**. Jsou zvláštním případem logických hlídacích časovačů. Jsou zpřístupněny k použití autorovi aplikace. Jsou rovněž navzájem rozlišeny číslem kanálu, operační systém zajišťuje, že nedojde ke konfliktu mezi čísly UHČ a LHČ, takže uživatel může bez omezení používat libovolná čísla UHČ ve stanoveném rozsahu.

Mechanismus funguje tak, že FHČ je periodicky obsluhován speciálním servisním procesem. Tato obsluha je ovšem podmíněna tím, že u žádného LHČ (včetně UHČ) nedošlo k vypršení časového limitu pro provedení obsluhy.

Dojde-li k vypršení časového limitu kteréhokoliv LHČ (včetně UHČ) nebo k selhání funkce speciálního servisního procesu, provede FHČ nezávisle na software restart systému.

LHČ jsou využívány operačním systémem ke kontrole správné funkce hardwareové i softwareové obsluhy všech periodických událostí a těch událostí, u nichž lze předpokládat, že do předem známé doby musejí nastat.

UHC obsluhuje uživatel prostřednictvím funkčního modulu **Watchdog**.

#### Klasifikace selhání funkce systému

Normální funkce systému může být narušena některým z následujících **typů selhání**, detekovatelných prostřednictvím hlídacích časovačů:

- a) Některá nebo všechny obslužné rutiny přerušení (včetně uživatelských procesů `QUICK`, `HighSpeed`, `ITR`,...) přestanou být vykonávány. K tomu může dojít vlivem selhání hardware nebo proto, že selhavší software zakázal vyvolávání přerušení.
- b) Některá obslužná rutina přerušení se neukončí. K tomu může dojít vlivem selhání hardware nebo proto, že obslužná rutina špatně zpracuje určitou kombinaci vstupních dat, což ji uvede do nekonečné smyčky.
- c) Kód vykonávaný na pozadí (včetně uživatelských procesů 0 až 15 a `IDLE`) přestane vykonávat svou funkci. Příčiny mohou být stejné jako u typu b).

#### Automatické zabezpečení operačním systémem

Operační systém bez účasti autora aplikace zabezpečuje systém proti selhání typů a) a b) s výjimkou procesů `ITR` a ostatních externích událostí, jejichž okamžik vyvolání není operačnímu systému předem znám. Proti selhání typu c) je systém automaticky zabezpečen jen v případě, že je zároveň doprovázeno selháním typu a) nebo b), což platí v případě drtivé většiny selhání vyvolaných chybou hardware.

I v případě, že autor aplikace nepoužije nikde v aplikaci funkční modul **Watchdog**, bude systém zabezpečen proti drtivé většině hardwareových selhání.

#### Role autora aplikace v zabezpečení proti selhání

Bez spolupráce autora aplikace není možné systém zabezpečit proti softwareovým selháním typu c) ani proti hardwareovým selháním typu c), která nenaruší činnost obslužných rutin periodických přerušení. V případě kódu vykonávaného na pozadí má totiž jen autor aplikace k dispozici informaci, jaký je za předpokladu normální funkce maximální interval mezi okamžiky, kdy aplikace vykoná nějakou konkrétní část kódu. Tento interval může být silně ovlivněn případným použitím smyček **While** apod. Zpravidla lze určit nějaký limitní interval, ve kterém by měl aplikační kód zpracovat nějaké určité místo programu.

- ♦ Obvykle lze určit limit ve smyslu "nejméně jednou za X milisekund musí aplikace vstoupit do procesu `IDLE`". Konkrétní hodnota X (která může ležet kdekoli mezi 1 a nekonečnem, to v případě že proces `IDLE` například obsahuje záměrně vytvořenou nekonečnou smyčku) je ovšem známa jen autorovi aplikace.
- ♦ Nejsnadnější (ovšem bohužel nikoliv univerzální) řešení bývá využití nastavené periody některého z procesů 0 až 15. Má-li proces 0 nastavenou periodu např. 1 s, pak by první funkční modul vložený do tohoto procesu měl být zbruhá každou sekundu vyvolán. Nestane-li se tak do nějakého násobku této doby (doporučuje se nejméně dvoj- až trojnásobek, aby se nechal prostor pro okamžiky krátkodobého přetížení procesoru), indikuje to selhání typu c). Tato logika ovšem může být narušena použitím např. smyček **While** ve kterémkoliv z procesů 0 až 15, `IDLE`, pokud doba trvání smyčky závisí na externí události.
- ♦ Nelze-li kvůli výskytu částí kódu (zpravidla smyček) s předem neurčitelnou dobou normálního provádění použít žádnou z výše uvedených metod, je možno alespoň určit maximální dobu trvání jednoho průchodu každou z takovýchto smyček. Indikátorem selhání typu c) je potom několikanásobné překročení této doby, UHC je třeba obsluhovat uvnitř každé takové smyčky.

Bez spolupráce autora aplikace není možné systém zabezpečit ani proti selháním typů a) a b) v případě procesů `ITR`. Jen autor aplikace může stanovit kritéria typu "aplikace spustila motor, jestliže tedy do X milisekund nebude vyvolán proces `ITR0`, zpracovávající pulzy od otáčkoměru, indikuje to selhání."

Spolupráce autora aplikace na zabezpečení proti selhání spočívá v použití funkčního modulu **Watchdog** (viz jeho popis). Tento modul obsluhuje UHČ a stanovuje, že do uživatelem určeného časového limitu musí být tento UHČ obsloužen znovu. UHČ jsou rozlišeny číslem tzv. kanálu. Jeden kanál (tedy jedna instance UHČ) může být obsluhován z různých míst aplikačního programu několika voláními modulu **Watchdog** se stejnou hodnotou parametru Kanál.

#### Příklady aplikačního zabezpečení

Výše popsané nejsnadnější řešení selhání typu c) (je-li použitelné) tedy spočívá ve vložení jediného funkčního modulu do procesu 0:

```
Proc00 (perioda 1 sekunda):
    Watchdog 0, 3000
```

Tímto jediným modulem jsou zabezpečeny proti všem selháním typu c) všechny procesy 0 až 15, IDLE, neboť všechny tyto procesy sdílejí společné programové vlákno - tzn. selže-li jeden, selžou všechny.

Zabezpečení procesů ITR proti selháním typu a) a b) by ve výše naznačeném případě mohlo vypadat asi takto:

```
ProcITR00:
    Watchdog 1, 5000          Obsluha UHČ
    RPM1      ...

Proc00:
    If @MotorChod
        If @MotorBezi
            Else
                BinOut    #Motor, 1
                Let      @MotorBezi = 1
                Watchdog 1, 5000          Aktivace UHČ
            EndIf
        Else
            If @MotorBezi
                BinOut    #Motor, 0
                Let      @MotorBezi = 0
                Watchdog 1, -1          Deaktivace UHČ
            EndIf
        EndIf
```

Oba přístupy je samozřejmě možno v jedné aplikaci kombinovat. Proto je autorovi aplikace dána možnost použít více nezávislých UHČ pro zabezpečení různých druhů událostí, a proto i v příkladech byly použity dva různé UHČ: číslo 0 v prvním příkladě a číslo 1 ve druhém příkladě.

#### Aplikační reakce na detekované selhání

Chce-li autor vytvořit (po provedení restartu) nějakou vlastní programovou reakci na detekované selhání, má možnost využít (typicky v procesu INIT) funkční modul **StartType** (viz jeho popis a příklad v popisu).