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
ComWEmpty	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
ErrSig32	Obsluha chyby v systému (umožňuje zpracovat až 32 signálů)	PSE
Exit	Ukončení běhu podprogramu	PSE
FastBoot	Zrychlení startu systému bez čekání na I/O	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
GetIPConf	Získání údajů z IP-konfigurace stanice	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
IRCIn	Čtení hodnoty z hardwareového čítačového vstupu / vstupu pro inkrementální čidla polohy	PSE
IRCMode	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

Modul	Popis	Knihovna
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
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
SetIPConf	Zápis údajů do IP-konfigurace stanice	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 zpožďování digitálních signálů	PSE
TimerOff	Zpoždění sestupné hrany	PSE
TimerOn	Zpoždění náběžné hrany	PSE
TimerPuls	Generování pulzu od náběžné hrany	PSE

Modul	Popis	Knihovna
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
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_NumAI	Čtení numerické hodnoty analogového vstupu na sběrnici ARION	ARION
ARN_NumAO	Zápis numerické hodnoty do an. výstupu na sběrnici ARION	ARION
ARN_SfAO	Definice bezpečného stavu AO v síti ARION	ARION
ARN_SfDO	Definice bezpečného stavu DO v síti ARION	ARION
ARN_TRX25	Připojení VF-modulu bezdrátové sítě Jablotron TRX-25 na ARION	ARION
DM_RFC	Připojení VF-modulu bezdrátové sítě DM-RFC na sběrnici ARION	ARION
TRX_Termo	Čtení z termostatu v bezdátové síti Jablotron	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

Modul	Popis	Knihovna
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
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
LCDAMiN2D	Interpretr dat z LCDSHELLu pro paralelní terminál 4x20 znaků s miniklávesnicí (AMiNi2D)	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

Modul	Popis	Knihovna
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 přepočet šestnáctice impulzních vstupů	MATRIX
MImpIn	Ošetření šestnácti impulzních vstupů	MATRIX
MtxAdd	Součet dvou matic podle pravidel maticového počtu	MATRIX
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 používvají 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 prvek matice. 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 celou matici (zadává se pouze jméno) nebo o řádek

Typ	Význam
	matice (zadává se jméno a číslo řádku) nebo o sloupec matice (zadává se jméno a číslo sloupce).
ML	Jméno maticové databázové proměnné typu `ML`. Běžně budeme tímto symbolem rozumět prvek matice. 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 celou matici (zadává se pouze jméno) nebo o řádek matice (zadává se jméno a číslo řádku) nebo o sloupec matice (zadává se jméno a číslo sloupce).
MF	Jméno maticové databázové proměnné typu `F`. Běžně budeme tímto symbolem rozumět prvek matice. 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 celou matici (zadává se pouze jméno) nebo o řádek matice (zadává se jméno a číslo řádku) nebo o sloupec matice (zadává se jméno a číslo sloupce).
Bit	Bit databázové proměnné . Může být zadán dvěma způsoby: - jméno proměnné typu `I` 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 `bit`. 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 `int`. 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 `word`. 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 `long`. 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 `dword`. 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 `float`. Tento typ parametru může být pouze v LA-modulech, v normálních modulech není.

Typ	Význam
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 zadávájí se jako číslo nebo výběr z hodnot anebo logický výběr ANO/NE. Přiklad 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>	
		...	
		...	

```
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

...
```

BinIn	Čtení jednoho binárního signálu

Popis

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

Parametry

Signál	IN	DI	Signál logického kanálu DI z nějž bude modul číst.
Negace	PAR	Konst	Příznak negace vstupního signálu.
Výstup	OUT	Bit	Bit proměnné, do nějž 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.

BinOut	Zápis jednoho binárního signálu

Popis

Modul zapíše hodnotu zvoleného bitu databázové proměnné typu `I` na výstup logického kanálu DO. Zapisovaný kanál lze negovat.

Parametry

Proměnná	IN	Bit	Bit proměnné, který se zapíše do výstupního kanálu.
Negace	PAR	Výběr	ANO=Negovat výstupní signál.
Kanál	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.

Case	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 **Switch**, pro niž je tato větev příkazu **Case** aktivní.

Návěští	PAR	Návěští	Návěští následujícího příkazu **EndCase** (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
```

DigIn	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

Kanál	IN	DI16	Číslo logického kanálu DI, z nějž bude modul číst.

Proměnná	OUT	I	Jméno proměnné, do níž budou načtené (příp. ještě negované) signály uloženy.

Negace	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.

DigOut	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

Proměnná	IN	I	Jméno proměnné, která obsahuje signály, které modul zapíše do výstupního kanálu (příp. negaci).

Kanál	OUT	DO16	Číslo logického kanálu DO, na nějž bude modul psát.

Negace	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.

Else	Pokračovací část podmíněného příkazu **If**

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 **EndIf.** Toto návěští je automatické, lokální, generuje se tedy automaticky.

Příklad

```
          If    Test.0, :00000    Je-li digitální signál
                                  (bit) nastaven,
           . . .                  proveď tuto sekvenci
                                  příkazů/modulů,
:00000    Else  :00001            a není-li nastaven,
           . . .                  proveď tuto sekvenci
:00001    EndIf                  Následující
                                  příkazy/moduly prováděj
                                  vždy
```

EndCase 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
```

EndIf	Ukončení podmíněného příkazu **If**

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,
            . . .                   proveď tuto sekvenci
                                    příkazů/modulů,
:00000  Else  00001                 a není-li nastaven,
            . . .                   proveď tuto sekvenci
:00001  EndIf                       Následující
                                    příkazy/moduly prováděj
                                    vždy
```

EndSwitch Ukončení přepínače **Switch**

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
```

Exit	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.

If	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

Podmínka	IN	Bit	Podmínka provedení příkazů/modulů mezi **If** a následujícím **Else** nebo **EndIf**.

Návěští	PAR	Návěští	Návěští následujícího příkazu **Else** nebo **EndIf** - automatické, lokální, je tedy generované automaticky.

Příklad

```
         If      Test.0, :00000  Je-li digitální signál
                                 (bit) nastaven,
        . . .                    proveď tuto sekvenci
                                 příkazů/modulů,
:00000  Else  :00001             a není-li nastaven,
           . . .                 proveď tuto sekvenci
:00001  EndIf
                                 Následující
                                 příkazy/moduly
                                 prováděj vždy
```

Let	Vyhodnocení aritmetického, relačního nebo logického výrazu

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 možností odkazu na bit je odkaz pomocí alias jména. Uveďme 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. Uveďme opět několik příkladů:

`13.6`	reálné číslo
`15`	celé číslo
`0x1234`	šestnáctkové celé číslo
`0o12345`	osmičkové celé číslo
`0b1011101000101`	dvojkové celé číslo
`Var + 13.6`	hodnota proměnné zvětšená o číslo
`Var1 / Var2`	podíl dvou proměnných
`Matice[1, Index]`	prvek maticové proměnné, řádek 1, sloupec daný obsahem jednoduché proměnné
`Var1+Var2*(Var3 – Var4[1,Index]/Var5)`	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	celočíselné 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 několik 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.

Funkce je definovaná pouze pro hodnoty x větší nebo rovny nule. Pro záporné hodnoty vrací funkce nulu.

Avg(výraz1, výraz2 [, výraz3 [, výraz4 ...]]])
Funkce vrací aritmetický průměr hodnot všech výrazů, které jsou argumenty funkce. Funkce má proměnný počet argumentů.

Min(výraz1, výraz2 [, výraz3 [, výraz4 ...]]])
Funkce vrací minimální hodnotu z hodnot všech výrazů, které jsou argumenty funkce. Funkce má proměnný počet argumentů.

Max(výraz1, výraz2 [, výraz3 [, výraz4 ...]]])
Funkce vrací maximální hodnotu z hodnot všech výrazů, které jsou argumenty funkce. Funkce má proměnný počet argumentů.

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

Uveďme 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. Stejné 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 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. Naskýtá se tedy otázka, jak tyto situace zpracuje příkaz **Let**.
Zaveďme 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	Bool, Int, Long, Float	Bool
	>=	větší nebo rovno	Bool, Int, Long, Float	Bool
	==	rovno	Bool, Int, Long, Float	Bool
	!=	nerovno	Bool, Int, Long, Float	Bool
	<=	menší nebo rovno	Bool, Int, Long, Float	Bool
	<	menší	Bool, Int, Long, Float	Bool
Volný výsledný typ	*	násobení	Int, Long, Float	*)
	+	součet	Int, Long, Float	*)
	-	rozdíl	Int, Long, Float	*)
	div	celočíselné 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. Uveďme 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.

Záverečné 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 využívejte 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 bežně 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 cislice

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

Výraz	PAR	Řetězec	Jediný parametr příkazu **Let**.

Syntaxe byla popsána výše.

Příklad

Příklady byly uvedeny v popisu příkazu.

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 řídící proměnnou. V této oblasti jsou definovány jednotlivé větve přepínače pomocí dvojice příkazů **Case-EndCase**. Každá dvojice specifikuje jednu hodnotu řídící 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 **EndSwitch** - automatické a lokální, je tedy generováno automaticky.

Příklad

```
Switch Test, :NONE   Přepínač podle hodnoty
                     proměnné Test
   Case 0, :NONE  Větev pro hodnotu
                  proměnné = 0
      . . .          Příkazy/moduly větve
   EndCase           Konec větve
   Case 1, :NONE  Větev pro hodnotu
                  proměnné = 1
    . . .
   EndCase
   Case 2, :NONE  Větev pro hodnotu
                  proměnné = 2
      . . .
   EndCase
                     Implicitní větev pro všechny
                     ostatní hodnoty proměnné
      . . .
EndSwitch            Konec přepínače
```

Konkrétní příklad:

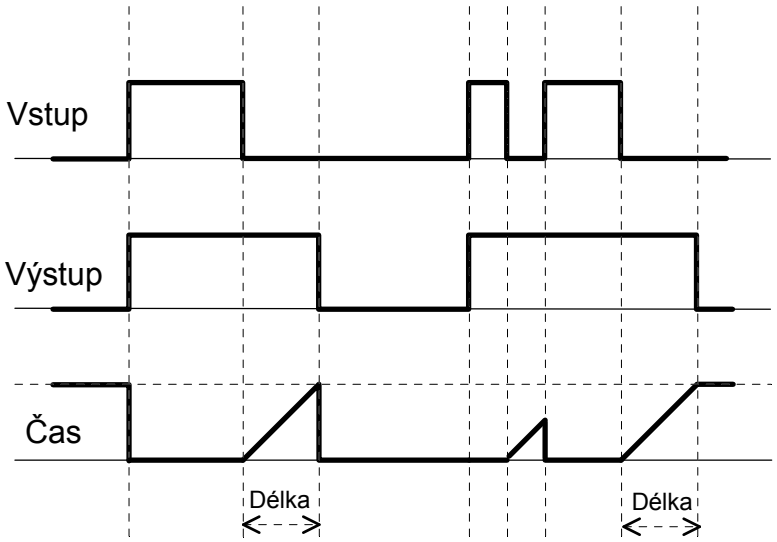
```
Switch     mode, :NONE
  Case     1, :NONE
    Let    res = 10
  EndCase
  Case     2, :NONE
    Let    res = 20
  EndCase
  Let      res = 30
EndSwitch
```

Příkaz `Switch` se řídí hodnotou proměnné `mode`. Je-li hodnota `mode` = 1, přiřadí se do proměnné `res` hodnota 10. Je-li hodnota `mode` = 2, přiřadí se do proměnné `res` hodnota 20. Pro všechny ostatní hodnoty proměnné `mode` se do proměnné `res` přiřadí hodnota 30.

TimerOff	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

Vstup	IN	Bit	Vstupní signál.

Délka	IN	Konst L ML	Délka zpoždění [ms].

Výstup	OUT	Bit	Výstupní signál.

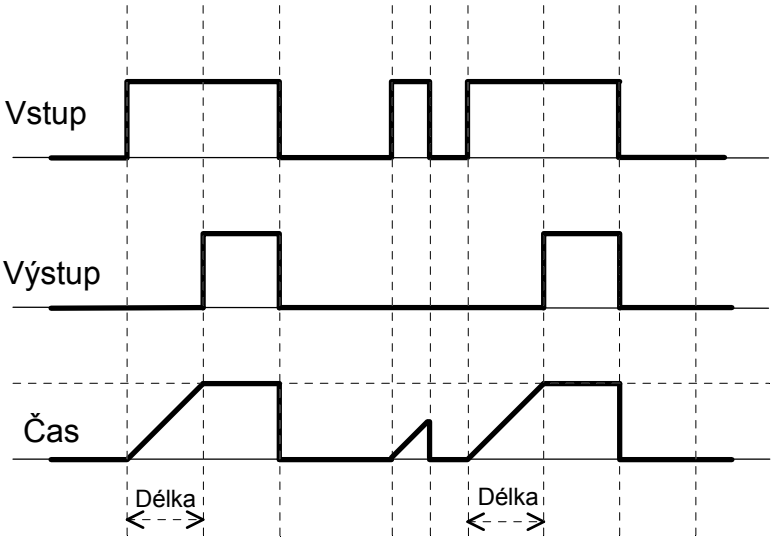
Čas	OUT	L NONE	Průběžný čas od poslední sestupné hrany vstupního signálu. Hodnota se pohybuje v intervalu 0 až `Délka`.

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”.

TimerOn	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

Vstup	IN	Bit	Vstupní signál.

Délka	IN	Konst L ML	Délka zpoždění [ms].

Výstup	OUT	Bit	Výstupní signál.

Čas	OUT	L NONE	Průběžný čas od poslední náběžné hrany vstupního signálu. Hodnota se pohybuje v intervalu 0 až `Délka`.

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.