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

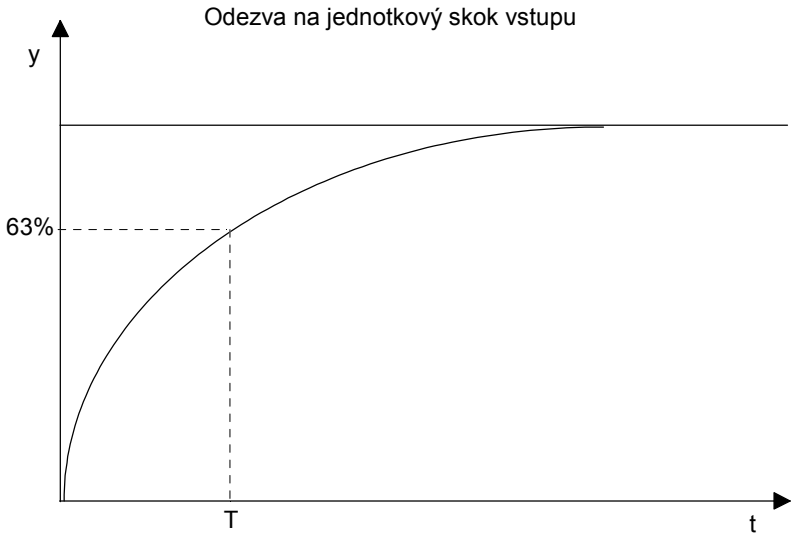
Příklad

```
                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.

Filtr1R	Modul realizuje filtr 1. řádu (setrvačný článek)

Popis



Parametry

Vstup	IN	F MF	Vstupní proměnná.
Výstup	OUT	F MF	Výstupní proměnná.
T	IN	Konst F MF	Časová konstanta filtru [s]. Je to doba, za kterou výstup dosáhne 63% hodnoty vstupu při odezvě na skok vstupu.

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.

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 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

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	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	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.

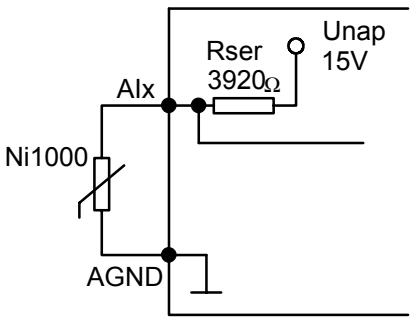
Příklad

Příklady byly uvedeny v popisu příkazu .

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 $\upsilon\,[°C]$ probíhá podle vztahu:
$\upsilon = C_1 \cdot \Delta R + C_2 \cdot \Delta^2 R + C_3 \cdot \Delta^3 R + C_4 \cdot \Delta^4 R$, kde $\Delta R = R - R_0$, $R_0 = 1000\Omega$.

Výše uvedený vztah je přibližnou inverzní náhradou vztahu:
$R = R_0 \cdot (1 + A \cdot \upsilon + B \cdot \upsilon^2 + C \cdot \upsilon^3 + D \cdot \upsilon^4)$,
převzatého z materiálů výrobce snímačů - firmy Sensit.
Největší chyba inverzní náhrady je $\Delta_{\max}\,[°C]$, viz níže uvedenou tabulku.

Výše uvedený přepočetní 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 $\upsilon\,[°C]$ podle lineárního vztahu:
$\upsilon = \frac{R - R_0}{R_0 \cdot \text{Citlivost} \cdot 10^{-6}}$.

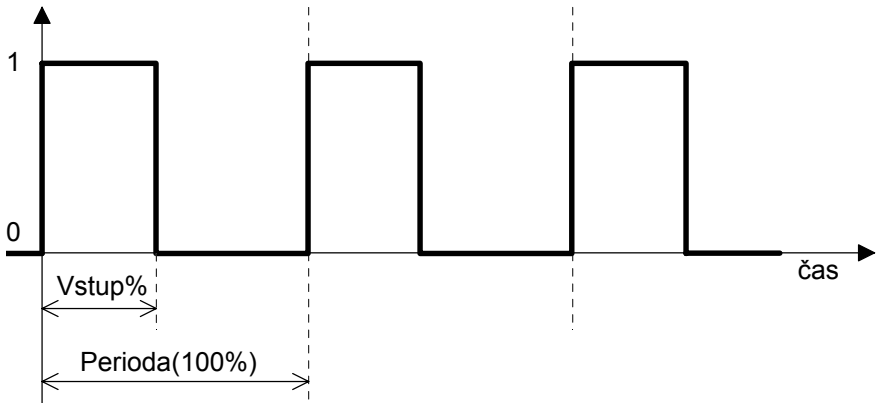
Tabulka koeficientů snímačů:

Koeficient	Typ snímače	
	Ni1000/5000ppm	**Ni1000/6180ppm**
$R_0\,[\Omega]$	1000	1000
Citlivost $[ppm]$	5000	6180
$C_1\,[°C \cdot \Omega^{-1}]$	$2.259 \cdot 10^{-1}$	$1.82447 \cdot 10^{-1}$
$C_2\,[°C \cdot \Omega^{-2}]$	$-5.957 \cdot 10^{-5}$	$-4.077 \cdot 10^{-5}$
$C_3\,[°C \cdot \Omega^{-3}]$	$1.700 \cdot 10^{-8}$	$1.628 \cdot 10^{-8}$
$C_4\,[°C \cdot \Omega^{-4}]$	$-2.890 \cdot 10^{-12}$	$-6.720 \cdot 10^{-12}$
$A\,[°C^{-1}]$	$4.427 \cdot 10^{-3}$	$5.485 \cdot 10^{-3}$
$B\,[°C^{-2}]$	$5.172 \cdot 10^{-6}$	$6.650 \cdot 10^{-6}$
$C\,[°C^{-3}]$	$5.585 \cdot 10^{-6}$	0
$D\,[°C^{-4}]$	0	$0.02805 \cdot 10^{-9}$
$\Delta_{\max}\,[°C]$	0.00579	0.0195

PWM	Pulzně šířková modulace

Popis

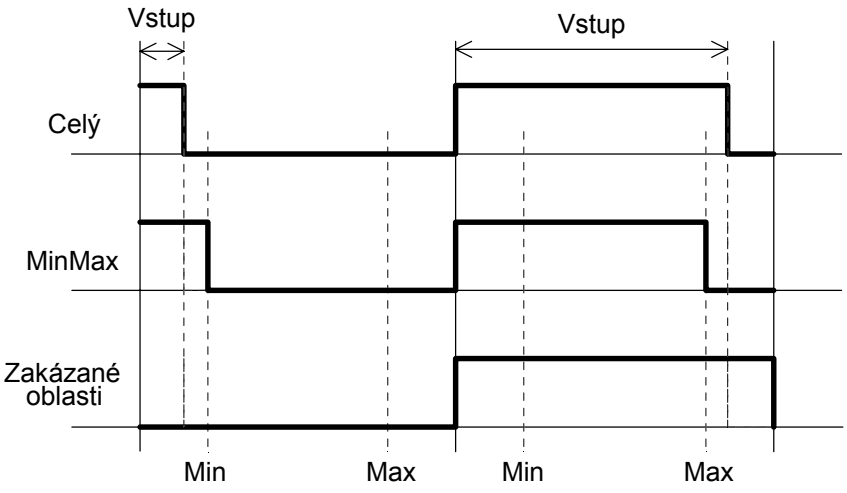
Modul převede proměnnou `Vstup` v rozsahu 0..100% na pulzně šířkovou modulaci, která má periodu `Perioda`. Rozlišení velikosti střídy je dané periodou procesu, ve kterém se modul nachází.



Parametry

Režim	PAR	Výběr	Režim činnosti

Interval	Konst	Rozsah a omezení pulzu. 0 **MinMax** Velikost pulzu (doba, ve které je výstup v "1") je modulem omezena na interval <`Min`,`Max`>. Pokud by měl být pulz kratší než je velikost `Min`, tak je omezen na velikost `Min`. Pokud by měl být pulz delší než je velikost `Max`, tak je omezen na velikost `Max`. 1 **Celý** Pulz může mít délku v celém rozsahu PWM tj. v intervalu <`0`,`Perioda`>. Délka 0 se projeví tak, že výstup je trvale v "0". Jestliže má délka hodnotu `Perioda`, projeví se to tak, že výstup je trvale v "1". 2 **Zakázané oblasti** Pulz nesmí být kratší než `Min` a zároveň nesmí být delší než `Max`. Pokud by měl být pulz kratší než je velikost `Min`, tak je na výstupu trvale "0". Pokud by měl být pulz delší než je velikost `Max`, tak je na výstupu trvale "1".



Plynulá	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.

Začíná_1	Výběr		ANO=perioda PWM začíná log. "1" a končí log. "0", NE=perioda PWM začíná log. "0" a končí log. "1".

Vstup	IN	F MF	Vstupní proměnná s rozsahem 0..100%

Perioda	IN	Konst F MF	Perioda PWM [s].

Min	IN	Konst F MF	Minimální délka pulzu [s].

Max	IN	Konst F MF	Maximální délka pulzu [s].

Výstup	OUT	Bit DO	Výstupní bit.

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, T1action, 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).

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 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	Řídicí 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.

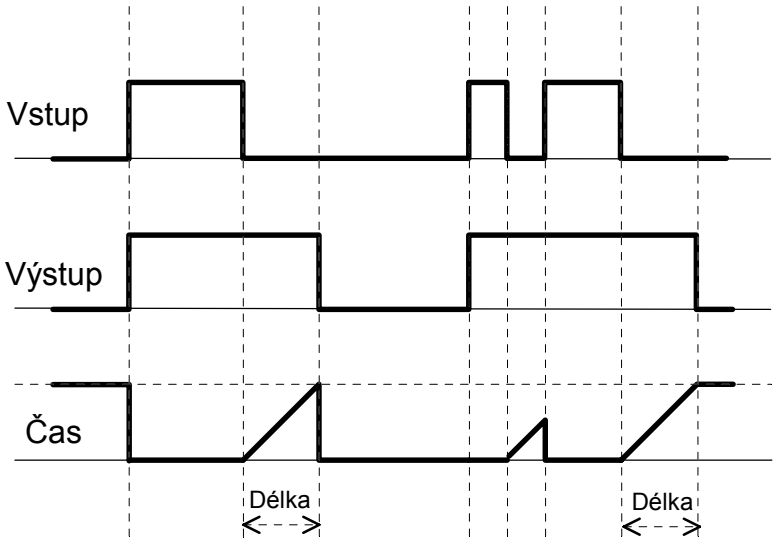
Příklad

```
      Switch Test, :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
```

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

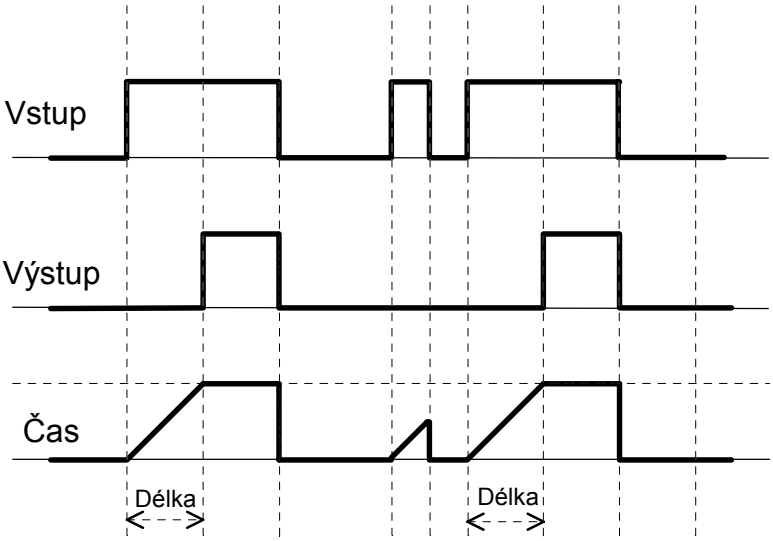
Vstup	IN	Bit	Vstupní signál.
Délka	IN	Konst L ML	Délka zpoždění [ms].
Výstup	OUT	Bit	Výstupní signál.
Čas	OUT	L NONE	Průběžný čas od poslední sestupné hrany vstupního signálu. Hodnota se pohybuje v intervalu 0 až `Délka`.

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.

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 kdekoliv 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 impulz / 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 µs. 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 µs, generuje trvalý stav OFF (pro střídu meší 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**.