23

23. DODATKY

23.1 Konstantní řezná rychlost

23.1.1 Popis konstantní řezné rychlosti

Konstantní řezná rychlost (KŘR) je způsob řízení otáček vřetene, kdy se ze zadané velikosti řezné rychlosti a okamžité polohy řídící souřadnice, neustále vypočítávají požadované otáčky vřetene.

KŘR není omezena na pohybové bloky a také ji možno plně řídit z PLC programu. Systém i PLC mají možnost omezit otáčky vřetene na zadanou hodnotu..

Pro obvodovou rychlost platí:

$$v = \omega.r = 2.\pi.n.x$$

x = poloha řídicí souřadnice (korigována vzhledem ke korekcím a posunutí)
v = obvodová rychlost (KŘR)
n = otáčky vřetene

Vypočtené napětí, které se zadává pro vřeteno (rozsah 0-7FFFh)

$$U_x = \frac{\frac{v}{2.\pi.x}.U.U_m}{P}$$

U_x = vysílané napětí na vřeteno(rozsah 0-7FFFh)

U = maximální napětí pro vřeteno pro daný převodový stupeň (poměr k max.napětí)
U_m= maximální napětí (7FFFh)
P = maximální otáčky dané převodové řady

23.1.2 Konstantní řezná rychlost – ovládání z NC programu

Systém používá pro KŘR dvě funkce:
G96 – Konstantní řezná rychlost s posuvem mm/ot
G97 – Konstantní řezná rychlost s posuvem mm/min

Řezná rychlost se zadává funkcí **„S“**, která má v tomto případě význam nikoli otáček, ale řezné rychlosti v metrech za minutu [m/min]. Například řezná rychlost `50m/min` by se zadala hodnotou „`S50.0`“. Kromě toho je možno při programování využít systémový parametr **„CCS“**. V tomto případě se naprogramuje: „`CCS=50.0`“.

Z NC programu možno kdykoli zadat omezení otáček pro KŘR pomocí systémového parametru **„SLIM“.** Systém omezuje otáčky vzhledem k menší hodnotě z maximálních otáček převodového stupně a ze zadaného omezení.

Příklad:

```
N10 G0 X300 Z100 M44 M3       "rychloposuv na průměr 300, otáčky 100ot/min
    SLIM=1000 S100            "omezení maximálních otáček na 1000ot/min
N20 G96 G1 F0.2 S90 X50       "zařadí KŘR 90m/min, sjetí na průměr 50, posuv
                              "F = 0.2mm/ot
```

Při programování rychloposuvu se mění otáčky rovněž podle průměru, pokud není funkce G96 odvolána programováním G94.

KŘR není omezena jen na pohybové bloky NC programu. Zařazení KŘR může proto být i v nepohybovém bloku. Také změna délkové korekce a změna posunutí počátku se uplatní okamžitě a nemusí se čekat na pohybový blok. Korekce a posunutí mají vliv na KŘR, protože KŘR se počítá na špičku nástroje a ne na suport. KŘR se může také uplatňovat v ručních pojezdech nebo v jiných druzích pohybu, například vlečení nebo pro pohyby řízené z PLC programu.

KŘR nikdy nepřepíná převodový stupeň, ale omezí otáčky na maximální otáčky daného převodového stupně (pokud není ještě jiné omezení otáček).

23.1.3 Konstantní řezná rychlost – ovládání z PLC programu

PLC program má k dispozici pole double-wordových hodnot pro sledování stavu KŘR. Na systému je možno jednotlivé hodnoty sledovat v prohlížení paměti systému , nebo v lad9c9m programu Wintechnol.

název	adresa	Popis
AKRR_ACT_V	A178h	Aktuální konstantní řezná rychlost [mm/min] (G96 nebo PLC)
AKRR_ACT_SMAX	A17Ch	Aktuální maximální otáčky [1/1000 ot/min] (SLIM nebo PLC)
AKRR_ACT_DIST	A180h	Aktuální korekce poloměru [1/8 µm] (jen PLC)
AKRR_ACT_VCORR	A184h	Aktuální korekce řezné rychlosti [mm/min] (jen PLC)
AKRR_ACT_R	A188h	Aktuální poloměr [1/8µm]
AKRR_ACT_P	A18Ch	Aktuální max.otáčky dle převodové řady [1/1000 ot/min]
AKRR_ACT_U	A190h	Aktuální rozsah napětí dle převodové řady [0-7FFFh]
AKRR_ACT_S	A194h	Aktuální vypočtené otáčky [1/1000 ot/min]
AKRR_ACT_OUT	A198h	Aktuální vypočtené napětí [0-7FFFh]

PLC program má možnost zadávat velikost KŘR, maximální otáčky, korekci poloměru pro výpočet KŘR a korekci řezné rychlosti. Také může KŘR aktivovat a případně modifikovat výpočet. Pro řízení PLC program používá bity umístěné v řídicím bajtu **AKRR_CNTR.**

Název bitu	Váha		popis
EXT_AKRR_V	0	0	Velikost KŘR se řídí z NC programu (G96,G97+ S)
		1	Velikost KŘR zadává PLC program v buňce **AKRR_V**
EXT_AKRR_SMAX	1	0	Maximální otáčky zadává NC program (G76)
		1	Maximální otáčky zadává PLC program v buňce **AKRR_SMAX**
EXT_AKRR_DIST	2	0	PLC program nezadává korekci poloměru
		1	PLC program zadává korekci poloměru v buňce **AKRR_DIST**
EXT_AKRR_VCORR	3	0	PLC program nezadává korekci řezné rychlosti
		1	PLC program zadává korekci řezné rychlosti v **AKRR_VCORR**
EXT_AKRR_REQ	4	0	Aktivace KŘR se řídí z NC programu (G96,G97)
		1	Aktivace KŘR z PLC programu
EXT_AKRR_SUPORT	5	0	Standardní výpočet KŘR
		1	Výpočet KŘR nezapočítává korekce a posunutí
EXT_AKRR_P	6	0	Standardní zadání max.otáček převodu v „ChannelConfig“
		1	Externí zadání max.otáček převodového stupně
EXT_AKRR_U	7	0	Standardní zadání max.rozsahu převodu v „ChannelConfig“
		1	Externí zadání maximálního rozsahu výstupu
EXT_AKRR_R		0	Standardní zadání ze systému
		1	Externí zadání poloměru pro KŘR z PLC

název	adresa	Popis
AKRR_V	A19Ch	Velikost KŘR zadávaná z PLC [mm/min]
AKRR_SMAX	A1A0h	Maximální otáčky zadávané z PLC [1/1000 ot/min]
AKRR_DIST	A1A4h	Korekce poloměru zadávaná z PLC [1/8 µm]
AKRR_VCORR	A1A8h	Korekce řezné rychlosti zadávaná z PLC [mm/min]
AKRR_P	A1ACh	Max.otáčky přev.stupně z PLC [1/1000 ot/min]
AKRR_U	A1B0h	Rozsah napětí dle převod.rady z PLC [0-7FFFh]
AKRR_R	A1B4h	Poloměr z PLC [1/8 um]

23.2 Nelineární korekce

23.2.1 Popis a konfigurace pro nelineární korekce

Nelineární korekce umožňují korigovat nepřesnosti v odměřování a tím významně zvýšit přesnost obrábění. Nepřesnosti se většinou objeví při nepřímém odměřování. Nepřesnost může nastat například i vlastní vahou suportu při poloze ve větší vzdálenosti od stojanu (tzv. „padání“ osy). Všechny tyto nepřesnosti lze odstranit nelineárními korekcemi.

Každá rovinná nelineární korekce je vytvořena jedním párem tabulek pro kladný a záporný směr řídící souřadnice. Každá tabulka je funkčním vztahem, který určuje závislost řízené souřadnice na řídící souřadnici. Každá souřadnice může mít 2 nezávislé nelineární korekce (NK1 a NK2).

Například pro rovinnou korekci souřadnice X (řízena souřadnice je X) platí 4 funkční závislosti :

$\Delta X_1 = NK1_{PLUS}$ (1.řídicí osa) ;například 1.řídící souřadnice může být X - osa koriguje sama sebe
$\Delta X_1 = NK1_{MINUS}$ (1.řídicí osa)
$\Delta X_2 = NK2_{PLUS}$ (2.řídicí osa) ;například 2.řídící souřadnice může být Y- závislost jedné souř a -
$\Delta X_2 = NK2_{MINUS}$ (2.řídicí osa) ;dnice na druhé

ΔX_1…… korekce polohy řízené souřadnice od 1. nelineární korekce
ΔX_2…… korekce polohy řízené souřadnice od 2. nelineární korekce
$NK1_{PLUS}$ …. 1. nelineární korekce pro kladný směr první řídící souřadnice
$NK2_{MINUS}$ … 2. nelineární korekce pro záporný směr druhé řídící souřadnice

Systém si načítá nelineární korekce po zapnutí systému ze souborů. Názvy těchto souborů (pro jednotlivé osy a směry) jsou spolu s dalšími informacemi uvedené v konfiguračním souboru typu „`ChannelConfig`“ v XML tvaru. Soubory musí být umístěné v adresáři „`Config`“ a doporučuje se použít skupinu „`CNC User Files`“. Soubory musí být také přidány do skriptu „`PLC.NSI`“ pro tvorbu SETUPu PLC.

Příklad:

Příklad konfigurace pro nelineární korekce (přesný význam atributů je uveden dále).
Do elementu „`Servo`“ je přidáno:

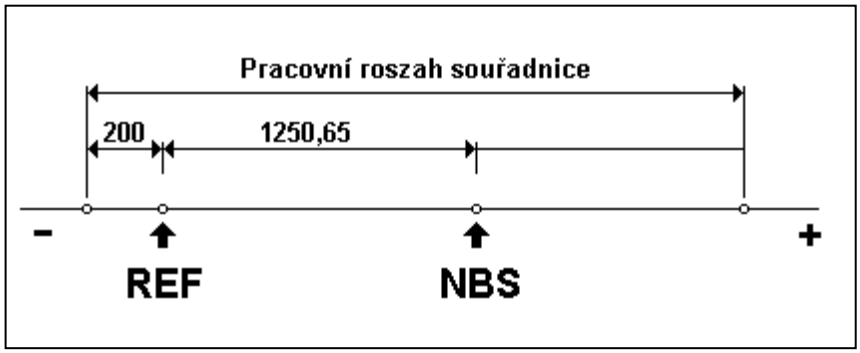
```
<NonLinearCompensation
      Active="1"
      Step="25.0"
      Begin="10.5"
      FileNamePos="Nk2plus.txt"
      FileNameNeg="Nk2minus.txt">
</NonLinearCompensation>
```

Soubory `Nk2plus.txt a Nk2minus.txt` jsou v adresáři „`CNC User Files\Config`“.

element **Servo**			**konfigurace servosmyčky**	
	element **NonLinearCompensation**		**nelineární korekce** (jedna servosmyčka může používat až 3)	
		atribut **Active**	**Je daná nelineární korekce použita?**	
			0	Nelineární korekce je zakázána (default)
			1	Nelineární korekce je povolena
		atribut **Type**	**Typ nelineární korekce**	
			0	základní (default)
			1	cyklická
			2	dynamická
		atribut **ControllingAxis**	**Řídící NC osa pro danou nelineární korekci**	
			0,1,..	řídicí NC osa – stejná jako číslo serva (default)
			xx	jiné přiřazení pro řídicí NC osu
		atribut **Step**	**Krok po kterém je korekce uvedena v tabulce [mm]**	
			1	krok tabulky 1 mm (default)
			xx	krok tabulky, maximálně 32 mm
		atribut **Begin**	**Umístění bodu, ke kterému se vztahuje první položka tabulky** (v mm vzhledem k nulovému bodu stroje)	
			0	nulový bod stroje a začátek tabulky jsou totožné (default)
			xx	míra v mm
		atribut **StepCount**	**Počet kroků tabulky**	
			xx	počet kroků, maximálně 1000 (default)
		atribut **CycleLen**	**Délka cyklu pro cyklickou nelineární korekci [mm]**	
			xx	Délka cyklu
		atribut **FileNamePos**	**Jméno souboru, ze kterého se má načíst tabulka pro kladný směr** Soubor je umístěn v adresáři „Config“	
			Abc..	Jméno souboru pro kladný směr
		atribut **FileNameNeg**	**Jméno souboru, ze kterého se má načíst tabulka pro záporný směr** Soubor je umístěn v adresáři „Config“	
			Abc..	Jméno souboru pro záporný směr

Nastavení začátku tabulky (atribut „Begin“)

Začátek tabulky od nulového bodu stroje (NBS) řídící souřadnice se zadává v atributu „Begin“. Pokud není nula stroje (NBS) na kraji pracovního prostoru souřadnice (v záporném směru), zadá se jako parametr „Begin“ vzdálenost NBS od záporné krajní polohy pracovního rozsahu souřadnice. Tato vzdálenost je totožná se začátkem působení nelineárních korekcí. Vzdálenost NBS a reference je uvedena pro jednotlivé osy v konfiguraci pro NC osy (element „NCAxis“ atribut „MachineNullPoint“). Pokud souřadnice „jezdí“ i za referenční bod , přičte se ještě tato vzdálenost a součet se zadá jako parametr „Begin“. Pro uvedený příklad na obrázku pro osu X by bylo nastaveno MachineNullPoint="-1250.65".
K této hodnotě se přičte -200.
Atribut „Begin“ by měl hodnotu: Begin="-1450.65".



23.2.2 Soubory s tabulkou korekcí

V souboru s korekcemi jsou uvedeny v textovém tvaru naměřené hodnoty korekcí pro příslušnou osu a směr. Každá hodnota korekce je uvedena na samostatném řádku. Řádek začíná pořadovým číslem kroku s dvojtečkou (velikost kroku je uvedena v konfiguraci v atributu „`Step`"), za kterým následuje hodnota korekce.

Velikost kroku nelineárních korekcí může být v rozsahu 1 až 32.767. V rámci zadaného kroku systém určuje aktuální korekci pomocí lineární interpolace.
V tabulce nelineárních korekcí nemusí být obsaženy všechny kroky. Nezadané údaje systém opět dopočítá lineární interpolací z nejbližších zadaných hodnot. Této vlastnosti se využívá tehdy, když potřebujeme mít větší rozestup měřených hodnot.

Příklad:
Zadané hodnoty korekce mají rozestup 100mm (100000 μm). Velikost kroku může být maximálně 32.767 μm, proto si zvolíme krok například 10.000 mm a do tabulky korekcí zadáme jen každou desátou hodnotu:

```
000: 00
010: 08           „řádek odpovídá 10x10000 = 100000 μm
020: 02
... atd.
```

Systém si vnitřně automaticky doplní nezadané kroky pomocí lineární interpolace:

například systém vnitřně dopočítá úsek mezi kroky 10 a 20 takto:

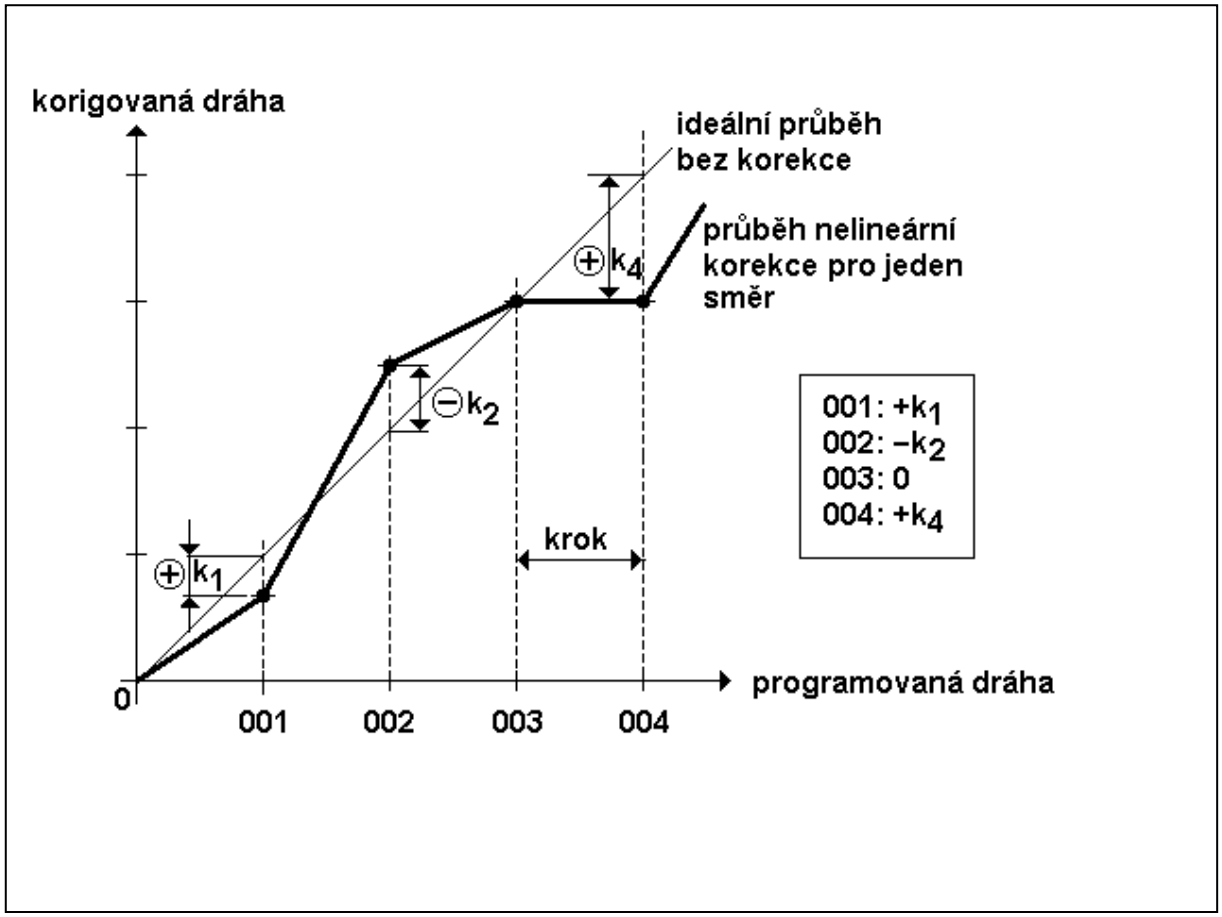
```
( 010: 08
  011: 07
  012: 06
  013: 06
  014: 05
  015: 05
  016: 04
  017: 03
  018: 03
  019: 02
  020: 02 )
```

Syntaktická pravidla pro rovinné korekce:

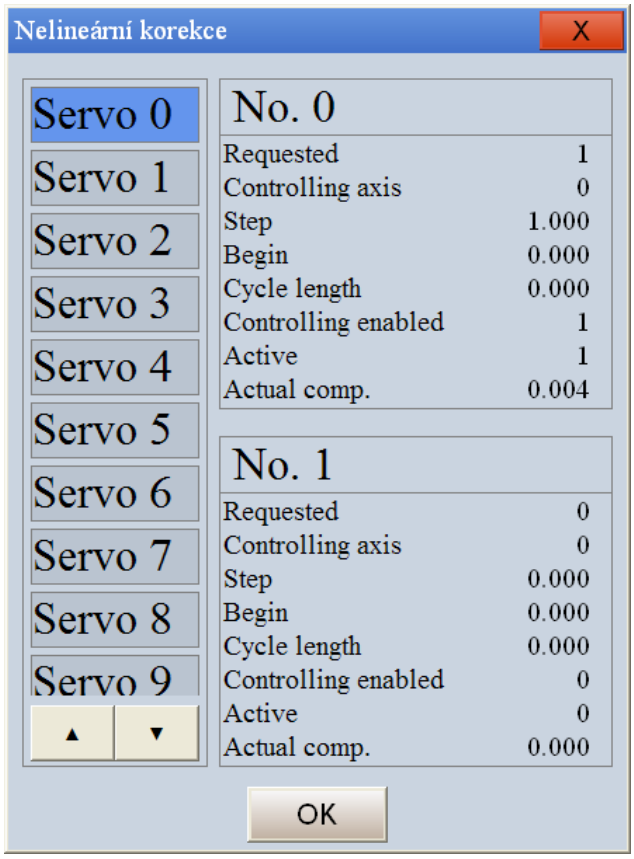
Soubor může obsahovat komentáře, uvozené uvozovkami.
Číslo kroku je trojmístné číslo počínaje 001(s eventuelními úvodními nulami), následované povinně dvojtečkou.
Za dvojtečkou následuje kladné nebo záporné číslo (znaménko plus se neuvádí), udávající v mikronech korekční hodnotu. Maximální povolená hodnota čísla je +/-32000 (mikronů).
V řádcích může být libovolný počet mezer a tabulátorů.
Pořadové číslo kroku musí být vzestupné, ale nemusí být obsaženy všechny kroky. Nezadané údaje systém dopočítá lineární interpolací z nejbližších zadaných hodnot.

23.2.3 Znaménka rovinných korekcí

Na obrázku je uvedeno zadávání znamének v korekčním souboru. Znaménko plus udává, že je nutné korekci přičíst, abychom se dostali na ideální průběh, znaménko mínus udává, že je nutné korekci odečíst, abychom se dostali na ideální průběh.



Pro kontrolu účinku korekcí je možno použít diagnostiku nelineárních korekcí:



Název	Popis	Odpovídá
Requested	Nelineární korekce je požadována	Element `NonLinearCompensation` Atribut `Active`
Controlling axis	Řídicí NC osa pro danou korekci (0,1,2,..)	Element `NonLinearCompensation` Atribut `ControllingAxis`
Step	Krok korekce v tabulce [mm]	Element `NonLinearCompensation` Atribut `Step`
Begin	Umístění bodu, ke kterému se vztahuje první položka tabulky	Element `NonLinearCompensation` Atribut `Begin`
Cycle length	Délka cyklu pro cyklickou nelineární korekci [mm]	Element `NonLinearCompensation` Atribut `CycleLen`
Controlling enabled	Aktivace nelineární korekce od řídící osy (např. osa byla zreferována)	`NlcAxisEnable` Jednotlivé bity povolují nelineární korekci pro jednotlivé osy (NELKOR_FLAG)
Active	Nelineární korekce je aktivní	Všechny podmínky pro aktivaci nelin.korekce (`NlcAxisEnable, NlcServoEnable, NlcSysServoEnable,…`)
Actual comp.	Aktuální hodnota kompenzace [mm]	Právě vybraná a interpolovaná hodnota korekce z tabulky

23.3 Sériové porty pro PLC

23.3.1 Konfigurace pro sériové porty PLC

Konfigurace pro sériové porty pro PLC se provede v konfiguračním souboru „`ChannelConfig`".

element **`ComChannel`**		**nastavení pro jeden sériový kanál pro PLC**	
	atribut **`No`**	**Logické číslo COM kanálu (0, ...). Pod tímto číslem s ním pracuje PLC**	
		0,1,..	logické číslo COM kanálu
	atribut **`Active`**	**Je daný sériový kanál použit?**	
		0	Sériový kanál je zakázán (default)
		1	Sériový kanál je povolen
	atribut **`ServiceMethod`**	**Způsob obsluhy sériového kanálu**	
		Buffered	komunikace je pomocí bufferu předávána primárnímu procesoru, data nejsou zpracovávána v reálném čase (default)
		RT	sériový port obsluhuje přímo daný procesor, komunikace je obsluhována v reálném čase (připravuje se)
	atribut **`PhysicalComNo`**	**Číslo sériového portu podle Windows (COM1, ...)**	
		1,2,..	číslo sériového portu podle Windows
	atribut **`ComSpeed`**	**Přenosová rychlost sériového portu (bit/s)**	
		115200	rychlost 11520 bit/s (default)
		xx	zadaná přenosová rychlost
	atribut **`ComDataBits`**	**Počet datových bitů**	
		8	počet datových bitů 8 (default)
		7	počet datových bitů 7
	atribut **`ComStopBits`**	**Počet stop bitů**	
		2	2 stop bity (default)
		1	1 stop bit
	atribut **`ComParity`**	**Parita**	
		None	bez parity (default)
		Odd	lichá parita
		Even	sudá parita
		Mark	paritní bit log.1
		Space	paritní bit log.0

Příklad:

```
<ComChannel No="0"
      Active="1"
      PhysicalComNo="2"
      ComDataBits="8"
      ComSpeed="19200">
</ComChannel>
```

23.3.2 Instrukce pro sériové porty PLC

Funkce pro obsluhu sériového portu pro PLC program jsou implementovány jako 4 speciální instrukce.

Instrukce pro sdílený přístup	Popis
COM_SEND	Zařazení znaku do fronty pro vysílání na COM port
COM_SEND_STR	Zařazení řetězce do fronty pro vysílání na COM port
COM_RECV	Čtení znaku z příjmové fronty z COM portu
COM_CONTROL	Řízení COM portu z PLC

instrukce	**COM_SEND**

funkce **Zařazení znaku do fronty pro vyslání na COM port**

syntax **COM_SEND com**
COM_SEND com [, val]

1.parametr **„com"** **číslo logického COM kanálu**
2.parametr **„val"** **hodnota pro vyslání**

Instrukce slouží pro zařazení znaku do výstupní fronty, které jsou vysílány přes COM kanál na periferii. Znak se očekává v DR registru nebo ve 2. parametru instrukce. Pokud při řazení nevznikla chyba, například je plný buffer, instrukce vrátí v RLO registru hodnotu log.1. Pokud instrukce proběhla pořádku, tak RLO registr bude nastaven na hodnotu log.0.
Význam parametrů:

Parametr	Hodnota (příklad)	Význam
com	1	Logický COM kanál (ne fzyický)
val	xx	Znak pro vyslání (bajt)

Příklad:

```
LOD         SendChar          ;Znak pro vyslání
COM_SEND    1                 ;Zařazení znaku do fronty na log.kanál 1
JL1         Error             ;nezařazeno = plná fronta
                              ;zařazeno
```

instrukce	**COM_SEND_STR**

funkce **Zařazení řetězce do fronty pro vyslání na COM port**

syntax **COM_SEND com, poin, len**

1.parametr **„com"** **číslo logického COM kanálu**
2.parametr **„poin"** **pointr nebo název řetězce pro vyslání**
3.parametr **„len"** **počet znaků které se mají vyslat**

Instrukce slouží pro zařazení řetězce do výstupní fronty, která je vyslána přes COM kanál na periferii. Pointr na řetězec je definován názvem datové proměnné nebo návěštím u instrukci „str". Dále je nutno zadat délku dat pro vyslání.
Rozdíl oproti vícenásobnému použití instrukce COM_SEND je v tom, že systém zabezpečí vyslání řetězce najednou bez možné časové prodlevy mezi vysíláními znaky.
Pokud při řazení nevznikla chyba, například je plný buffer, instrukce vrátí v RLO registru hodnotu log.1. Pokud instrukce proběhla v pořádku, tak RLO registr bude nastaven na hodnotu log.0.

Význam parametrů:

Parametr	Hodnota (příklad)	Význam
com	1	Logický COM kanál (ne fzyický)
poin	xx	Název datové proměnné nebo pole výstupního bufferu. Parametr se předává instrukci odkazem (ne hodnotou). Může být také návěstí u instrukci „str".
len	xx	Počet znaků pro vyslání

Příklad:

```
BUF_COM:    str   20,'Pokus vysilani'
PACK_1:     DS    20


  COM_SEND_STR  1,BUF_COM,14  ;Zařazení řetězce do fronty na log.kanál 1
  JL1           Error         ;nezařazeno = plná fronta
                              ;zařazeno

  COM_SEND_STR  1, PACK_1,14  ;Zařazení bufferu z pole PACK_1
  JL1           Error         ;nezařazeno = plná fronta
                              ;zařazeno
```

instrukce	**COM_RECV**

funkce **Čtení znaku z příjmové fronty**

syntax **COM_RECV com**
COM_RECV com [,val]

1.parametr **„com"** **číslo logického COM kanálu**
2.parametr **„val"** **buňka pro příjem**

Instrukce slouží pro příjem znaku z fronty, které byly přijaty z periferii přes COM kanál. Instrukce po vykonání vrátí v datovém registru načtený znak (nebo bude naplněn 2. parametr instrukce). Pokud byla prázdná fronta a žádný znak se nepřijal, instrukce vrátí v RLO registru hodnotu log.1.
Význam parametrů:

Parametr	**Hodnota (příklad)**	**Význam**
com	1	Logický COM kanál (ne fzyický)
val	xx	Buňka pro příjem znaku (bajt)

Příklad:

```
COM_RECV    1                   ;Příjem znaku z COM kanalu
JL1         Neprijato           ;Skok, když se nic nepřijalo
STO         RecvChar            ;Přijmutí znak
```

instrukce	**COM_CONTROL**

funkce **Řízení COM kanálu**

syntax **COM_CONTROL** **com, mode**

1.parametr **„com"** **číslo logického COM kanálu**
2.parametr **„mode"** **řízení**

Instrukce slouží pro některé operace s COM kanálem.

Význam parametrů:

Parametr	**Hodnota (příklad)**	**Význam**
com	1	Logický COM kanál (ne fzyický)
mode	CLEAR	Vynulování příjmové fronty
	RECEIVED	Počet znaků v příjmové frontě
	STATE	Načtení statusu COM kanálu

Význam hodnot pro dekódování stavu COM kanálu:

název	**hodnota**	**význam**
PLC_COM_ERROR	1	COM kanál je v chybě
PLC_COM_REQ	2	COM kanál není požadován a není nakonfigurován
PLC_COM_SEND_OVER	4	přetečení vysílacího bufferu
PLC_COM_RECV_OVER	8	přetečení příjmového bufferu

23.4 Instrukce pro sdílený přístup CAN kanálu

Sdílené použití CAN kanálu v PLC programu znamená, že PLC program používá CAN-BUS kanál společně se systémem, nebo alespoň pro jeho řízení používá systémové prostředky. Inicializaci CAN kontroléru a řízení vysílání a příjmu paketů také provádí systém. PLC program jen žádá a zařazení svých paketů do fronty pro vyslání a čte frontu přijmutích paketů.

Masky bitů pro vyhodnocování		
Symbol	**Maska**	**Význam**
CAN_ERR_OK	00h	Vše v pořádku
CAN_ERR_XMTFULL	01h	Plná fronta pro vysílání
CAN_ERR_RCVEMPTY	02h	Nepřijala se nová zpráva

23.4.1 Instrukce pro práci s CAN-BUSem se sdíleným přístupem

Funkce pro sdílenou obsluhu CAN-BUSu jsou do PLC programu implementovány jako 2 speciální instrukce.

Instrukce pro sdílený přístup	**Popis**
CAN_SEND	Zařazení paketu do fronty pro vysílání na periferii
CAN_RECV	Čtení paketu z příjmové fronty

instrukce	**CAN_SEND**

funkce	**Zařazení paketu do výstupní fronty**	
syntax	**CAN_SEND**	**can, pmsg**
1.parametr	**„can"**	**číslo CAN kanálu**
2.parametr	**„pmsg"**	**pointer na paket**

Instrukce slouží pro zařazení paketu do výstupní fronty paketů, které jsou vysílány přes CAN kanál na periferii. Instrukce po vykonání vrátí v datovém registru výsledek, který je složen z masek chyb (popsaných dříve). Pokud při řazení nevznikla chyba, vrácená hodnota bude CAN_ERR_OK (00h). pokud se paket do fronty nepodařilo zařadit (fronta je přeplněna), bude vrácená hodnota CAN_ERR_XMTFULL (01h)

Význam parametrů:

Parametr	**Hodnota (příklad)**	**Význam**
can	1, 2	Logický CAN kanál (CAN1, CAN2)
pmsg	CANBUFF_OUT	Pointer, který ukazuje na vyhrazené místo 12 bajtů. Návěští u instrukce STR, kde je místo pro paket

Příklad:

```
CANBUFF_OUT:      STR   12               ;výstupní paket

      CAN_SEND    2, CANBUFF_OUT         ;Zařazení paketu
      EQ          CAN_ERR_OK             ;Bylo zařazeno ?
      JL1         OK                     ;zařazeno
                                         ;nezařazeno
```

Podrobnější příklad bude uveden v závěru této kapitoly

instrukce	**CAN_RECV**

funkce **Čtení paketu z příjmové fronty (Nedoporučuje se používat)**

syntax **CAN_RECV can, pmsg**

1.parametr **„can"** **číslo CAN kanálu**
2.parametr **„pmsg"** **pointer na paket**

Instrukce je uvedena jen pro kompatibilitu a nedoporučuje se používat. Místo ní je zavedena instrukce **CAN_RECV_REQ**, které popis je dále.

Instrukce slouží pro čtení paketu z fronty paketů, které byly přijaty z periferii přes CAN kanál. Instrukce po vykonání vrátí v datovém registru výsledek, který je složen z masek chyb (popsaných dříve). Pokud při čtení nevznikla chyba, vrácená hodnota bude CAN_ERR_OK (00h). pokud se žádný paket pro PLC nepřijal, bude vrácená hodnota CAN_ERR_RCVEMPTY (02h)

Význam parametrů:

Parametr	**Hodnota (příklad)**	**Význam**
can	2	Logický CAN kanál (CAN2)
pmsg	CANBUFF_IN	Pointer, který ukazuje na vyhrazené místo 12 bajtů. Návěští u instrukce STR, kde je místo pro paket

Příklad:

```
CANBUFF_IN:       STR   12               ;vstupní paket

      CAN_RECV    2, CANBUFF_IN          ;Příjem paketu
      EQ          CAN_ERR_OK             ;Bylo přijato ?
      JL1         OK                     ;přijato
                                         ;nic se nepřijalo
```

Podrobnější příklad bude uveden v závěru této kapitoly

instrukce	**CAN_RECV_REQ**

funkce **Požadavek pro volání událostní procedury po přijmu paketu**

syntax **CAN_RECV_REQ can, objl, objh, idl, idh, proc, pmsg**

1.parametr	**„can"**	**číslo CAN kanálu**
2.parametr	**„objl"**	**minimální ID objektu zprávy**
3.parametr	**„objh"**	**maximální ID objektu zprávy**
4.parametr	**„idl"**	**minimální NODE-ID**
5.parametr	**„idh"**	**maximální NODE-ID**
6.parametr	**„proc"**	**název událostní procedury**
7.parametr	**„pmsg"**	**pointer na paket**

Instrukce má návratové hodnoty:

RLO=1	... operace dokončena bez chyb
RLO=0	... operace dokončena, ale požadavek nebyl akceptován
DR (DWRD)	... handle žádosti o volání událostní procedury

Instrukce slouží pro nastavení žádosti o volání událostní procedury v PLC po příjmu určitých „objednaných" paketů z CAN kanálu (nastavení „CALLBACK" volání). Když instrukce proběhne bez chyb (RLO=1) , tak si PLC program může uchovat „handle žádosti" do vlastní proměnné typu DWORD. Tuto proměnnou může PLC program využít v případě, že by později chtěl žádost o volání procedury zrušit.
Instrukce může být použita v modulu MODULE_INIT.

Význam parametrů:

Parametr	Hodnota (příklad)	Význam
can	1, 2	Logický CAN kanál (CAN1, CAN2)
objl	například 580h	Minimální ID objektu zprávy, které příjem se očekává. Jedná se o horní 4 bity z 11-bitové adresy COB-ID (podle normy CanOpen)
objh	-	Maximální ID objektu zprávy, které příjem se očekává. Parametr je nepovinný. Musí se uvést, když je požadován příjem intervalu zpráv. Pokud parametr nebude uveden (bude nahrazen znakem „-") tak se bude očekávat příjem zpráv jednoho objektu.
idl	například 21h	Minimální NODE-ID objektu zprávy, které příjem se očekává. Jedná se o 7 dolních bitů z 11-bitové adresy COB-ID.
idh	-	Maximální NODE-ID objektu zprávy, které příjem se očekává. Parametr je nepovinný. Musí se uvést, když je požadován příjem intervalu zpráv. Pokud parametr nebude uveden (bude nahrazen znakem „-") tak se bude očekávat příjem zpráv jednoho NODE-ID.
proc	například P_ON_CANRECV	Název událostní procedury, která se spustí po příjmu požadovaného paketu. Jedná se o standardní proceduru napsanou v PLC programu. Procedura po zavolání bude mít v DR registru nastaveno číslo logického CAN kanálu (1, 2).
pmsg	například sCANRECV	Pointer, který ukazuje na vyhrazené místo 12 bajtů. Návěští u instrukce STR (kde je místo pro paket) do kterého se naplní celý přijatý paket, který odpovídá požadovanému COB-ID.

Příklad bude uveden později.

Struktura COB-ID (**C**ommunication **Ob**ject **Id**entifier) podle normy CANopen :

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	x	x	x	x	x	**Function (0-0Fh)**				**NODE-ID (0-7Fh)**						

instrukce	**CAN_RECV_DEL**

funkce **Zrušení požadavku pro volání událostní procedury přijmu paketu**

syntax **CAN_RECV_END handle**

1.parametr **„handle"** **handle žádosti o volání událostní procedury**

Instrukce má návratové hodnoty:

RLO=1 ... operace dokončena bez chyb
RLO=0 ... operace dokončena, ale požadavek nebyl akceptován

Instrukce slouží pro zrušení žádosti volání událostní procedury v PLC po příjmu paketů z CAN kanálu. Jako parametr instrukce se musí zadat název proměnné typu DWORD, kde je uloženo „handle žádosti", které vrátila instrukce CAN_RECV_REQ.

Příklad:

Žádost o zachytávání odpovědi na SDO pakety (objekt 580h) pro jednotku INOUT08 s NODE-ID=21h.

```
;Buňka DWORD pro uložení Handle
dwHANDLE:   ds    4

;Místo pro přijatý paket:
sCANRECV:   STR   12


;Událostní procedura pro zpracování odpovědi na SDO paket
PROC_BEGIN  P_ON_CANRECV
      LOD   word.sCANRECV.CAN_DATA+1      ;Index zprávy
      EQ    cnst.1008h
      ;....
      ;....
PROC_END    P_ON_CANRECV


;Žádost o volání procedury P_ON_CANRECV
      CAN_RECV_REQ  2, 580h, -, 21h, -, P_ON_CANRECV, sCANRECV
      STO   dwHANDLE


;Zrušení žádosti o volání procedury
      CAN_RECV_END  dwHANDLE
```

23.4.2 Příklad pro vstupy a výstupy na nesystémovou CAN periferii se sdíleným přístupem

Příklad:
ID adresa periferie je nastavena na 10h.

V deklaraci dat:

```
CANBUFF_IN:       STR 12      ;misto na prijem paketu
CANBUFF_OUT:      STR 12      ;misto pro vysilani paketu
```

V modulu MODULE_MAIN:

```
      CAN_RECV 2,CANBUFF_IN                    ;Cteni paketu
      EQ          CAN_ERR_OK                   ;Je novy paket ?
      JL0         NIC_NEPRIJATO

;Test na prijem paketu ID=10h+1, delka=2, Byte1=Opc=1, Byte2=Input
      LOD         WORD.CANBUFF_IN.CAN_ID       ;Test ID
      ANDB        CNST.7FFh                    ;11 bitove ID
      EQ          CNST.10h+1                   ;Odpoved od ID=10h ?
      JL0         CAN_OTH
      LOD         BYTE.CANBUFF_IN.CAN_LEN      ;Delka
      EQ          CNST.2                       ;Je delka = 2 ?
      JL0         CAN_OTH
      LOD         BYTE.CANBUFF_IN.CAN_DATA+0   ;Opcode
      EQ          CNST.1                       ;Opcode 1 pro vstupy ?
      JL0         CAN_OTH
      LOD         BYTE.CANBUFF_IN.CAN_DATA+1   ;Sejmuti vstupnich dat !
      STO         IP_CAN                       ;Místo pro nova data
CAN_OTH:                                       ;Jiny paket, opakujeme


;**** Obsluha CAN-BUSu ... vysilani vystupu ****
      LOD         CNST.10h                     ;Vyslani na ID=10h
      STO         WORD.CANBUFF_OUT.CAN_ID
      LOD         CNST.0h                      ;Neni "remote request"
      STO         BYTE.CANBUFF_OUT.CAN_RTR
      LOD         CNST.2                       ;delka paketu
      STO         BYTE.CANBUFF_OUT.CAN_LEN
      LOD         CNST.2                       ;Opcode = 2
      STO         BYTE.CANBUFF_OUT.CAN_DATA+0
      LOD         IP0                          ;data pro vyslani
      STO         BYTE.CANBUFF_OUT.CAN_DATA+1

      CAN_SEND  2,CANBUFF_OUT                  ;Zarazeni paketu do fronty
```

23.5 Dotyková sonda

23.5.1 Konfigurace pro dotykovou sondu

Konfigurace pro dotykovou sondu v souboru typu „ChannelConfig“.

```
<TouchProbe No="0"
      Type="-2">
</TouchProbe>
```

Konfigurace pro typ sondy:

Typ sondy	popis
0	Sonda není použita
1	Sonda je zapojena na jednotku souřadnic SU05 nebo SU06
-1	Sonda je řešena v rámci PLC, aktuální poloha se získá s velkosti diferenčního čítače. Používá se vstupní signál v rastru milisekundy.
-2	Sonda je řešena v rámci PLC, aktuální poloha se získá z pole **SERVO_POSITION[x]** (inkrementy pohonu nebo odměřování). Možnost rychlého vstupu například použitím jednotky EtherCAT EL1252 s časovou značkou v rastru nanosekund

23.5.2 Programování sondy v NCP programu

Pro sejmutí polohy měřicí sondy se používá funkce **TPMeas**. Pokud je tato funkce programována v bloku, kde je pohyb, dojde při sepnutí sondy k zastavení pohybu. Poloha všech os se uloží do paměti, odkud můžeme polohu požadovaných os načíst např. do parametrů k dalšímu zpracování.

Příklad:

```
N10 TPMeas G01 G90 Feed=100 X10
N20 If (TPMEASRESULT)              "Test, zda došlo k sepnutí sondy
      R1 = TPMEASAXXPOS0
    Else
      Error(...)
    EndIf
```

V bloku, který následuje po měření je nutné otestovat , zda skutečně došlo k sepnutí sondy. To se provede otestováním systémové proměnné **TPMEASRESULT:**
TPMEASRESULT = 1 …. Došlo k sepnutí
TPMEASRESULT = 0 …. Nedošlo k sepnutí

Funkce TPMeas může mít 3 parametry:
1.parametr reakce na náběžnou nebo sestupnou hranu, 0= náběžná hrana (default), 1=sestupná hrana
2.parametr Konec bloku, 0= po příchodu signálu se zastaví (default), 1= pokračuje se do konce bloku
3.paremetr index dotykové sondy (0 = default)

Polohy **geometrických** se ukládají do těchto proměnných:

Poloha os v okamžiku sepnutí/uvolnění sondy při měření v prostoru **POS0**:
TPMEASAXXPOS0
TPMEASAXYPOS0
TPMEASAXZPOS0

Poloha os v okamžiku sepnutí/uvolnění sondy při měření v prostoru **POS1**:
TPMEASAXXPOS1
TPMEASAXYPOS1
TPMEASAXZPOS1

atd. podobně pro měření v prostoru **POS3**, **POS4**, **POS5**, **POS6**

Lze získat i polohu **interních** os pro měření v prostorech Pos0 – Pos6 v těchto proměnných:

Poloha interních os v okamžiku sepnutí/uvolnění dotykové sondy při měření v prostoru pos0.
TPMEASAXI0POS0
TPMEASAXI1POS0
TPMEASAXI2POS0
…
atd. pro další osy a prostory.

23.5.3 Rozhraní pro PLC program

Pro ovládání sondy z PLC programu jsou definovány bajtová pole **TPS0_IN**, **TPS1_IN** atd. a v nich bitové signály:

Bit	Popis
flTouchProbe0	Signál z PLC, že došlo k dotyku sondy. PLC tento signál nastavuje v rastru 1ms. Po zpracování dat musí PLC tento signál vynulovat.
flTouchProbe0Err	Dotyková sonda je v chybě

Při použití rychlého vstupu s časovou značkou, musí PLC program nastavit poměr v rámci 1 milisekundy, kde ke kontaktu došlo. Poměr se nastavuje jako reálné číslo do proměnné: **rTouchTimeRatio**.

```
( StartTimeCyclicOperation - LatchPos +  TimeDelay ) / MasterCycleTime
```

PLC program má možnost číst naměřené hodnoty v reálných polích:

rTouchPos0[]
rTouchPos1[]
rTouchPos2[]
rTouchPos3[]
rTouchPos4[]
rTouchPos5[]
rTouchPos6[]

Vždy 16 reálných hodnot pro každou osu a prostor v milimetrech.

Příklad:

```
MODULE_BLOCK_INIT

      ;Snimani rychleho vstupu pro dotykovou sondu
      LDR         BLK[0].TouchProbeMeas
      JL0         LBL_TPSE
      MECH_INIT   mTouchProbe
      FL          1,mTouchProbe
   LBL_TPSE:

;Snimani rychleho vstupu sondy v MODULE_FAST

MECH_BEGIN  mTouchProbe
      FL    0,flTouchProbe0
      LDR   inInFastIn1             ;ceka na vstup s casovou znackou
      EX1
      LDR   inInFastIn1             ;nastupni hrana
      EX0
      LOD   QWRD.qwTimeCyclicOp1    ;cas následujícího cyklu [ns]
      SU    QWRD.qwInFastLatchPos1  ;cas prichodu signalu
      REAL  QWRD
      DIVB  CNST.1000000.0          ;Cyklus EtherCAT mastra [ns]
      STO   REAL.rTouchTimeRatio
      FL    1,flTouchProbe0         ;Signal pro RTM
      TIM   -,100
      FL    0,flTouchProbe0
MECH_END    mTouchProbe
```

Pro typ sondy „-2“ se aktuální poloha se získá z dwordového pole SERVO_POSITION[]. Ve většině případů se systém automaticky zabezpečí aktualizování tohoto pole, například pro EtherCAT pohony „EDRV“ (viz návod „Servopohony“). V případě, že PLC má vlastní implementaci servosmyček, tak musí také zabezpečit plnění pole SERVO_POSITION.

23.5.4 Použití rychlého vstupu

Možnost použití rychlého vstupu, například použitím jednotky EtherCAT Beckhoff EL1252 s časovou značkou v rastru nanosekund.

Pro jednotku EL1252 je nutno definovat logické spoje mezi PLC a EtherCAT:

```
<!-- Rychle vstupy 1 EL1252 -->
<Connection Source="ECAT.IN_FAST.Latch.LatchPos1"
            Destination="PLC.Input.InFast.LatchPos1"
            Connected="1"></Connection>

<Connection Source="ECAT.IN_FAST.Channel 1.Input"
            Destination="PLC.Input.InFast.In1"
            Connected="1"></Connection>

<Connection Source="ECAT.LXM32M.Mapped to PI memory.Start Time Cyclic
Operation"  Destination="PLC.Input.TimeCyclicOp1"
            Connected="1"></Connection>
```

Upozornění:
Pro výpočet přesného času je nutno namapovat proměnnou „Start Time Cyclic Operation“ přímo z paměti komunikačního obvodu na adrese 0x990. Proměnná obsahuje čas začátku příštího cyklu výměny dat na EtherCATu, který je odvozen z „Distribuovaného clocku“. Nesmí být použit například nabízený prvek jednotky EL1252 „SysTime“, protože ten obsahuje čas okamžiku průchodu EtherCAT paketu jednotkou. Tento čas je zatížen chybou rozptylu periody paketů „Jiter“.
Proměnná „Start Time Cyclic Operation“ může být získána pro celou síť jen jednou a to například z EtherCAT pohonů (v některých verzích EtherCAT Mastra se bude muset zrušit LRW support).

Definice logických prvků pro jednotku EL1252 v PLC programu

```
;Rychle vstupy
DEF_IN inInFastIn1,       'InFast.In1',        TYPE_BIT,    - ,FAST
DEF_IN qwInFastLatchPos1, 'InFast.LatchPos1',  TYPE_UNS_64, - ,FAST
DEF_IN qwInFastLatchNeg1, 'InFast.LatchNeg1',  TYPE_UNS_64, - ,FAST
DEF_IN qwTimeCyclicOp1,   'TimeCyclicOp1',     TYPE_UNS_64, - ,FAST
```