1

1. ÚVOD DO PROGRAMOVÁNÍ

1.1 Základní pojmy

U číslicově řízených strojů se k přenosu informace o obráběném polotovaru či součásti používá kódovaného zápisu zvaného **partprogram** (program součásti).

U systémů CNC872iTD resp. iTQ se sled těchto informací ukládá do zálohované paměti, kterou může tvořit flash disk s kapacitou řádově jednotky GByte nebo pevný disk s kapacitou řádově desítky až stovky GByte.

Kromě partprogramů jsou v zálohované paměti uloženy také soubory s konfiguračními daty systému, makrocykly, pevnými cykly atd.

Partprogram, vytvořený technologem a uložený na vhodném médiu (nejčastěji USB flash disk či disk nadřazeného počítače apod.), je pak periferním zařízením zaveden do zálohované paměti systému. Partprogram se dá vytvořit buď strojním zpracováním na počítači po zavedení základních geometrických a technologických údajů a parametrů obráběcího stroje (AUTOPROG, KOVOPROG, ALFACAM apod.) nebo ručně, výpočtem koncových bodů všech elementárních pohybů nástroje a doplněním příslušných technologických funkcí. Vytvořený partprogram se přenese do paměti systému (viz Návod k obsluze). Partprogram lze zapsat do paměti systému také přímo z ovládacího panelu (v editoru systému nebo s grafickou podporou tvorby partprogramů).

Kódový zápis geometrie a technologie součástky tvořící partprogram je tvořen sledem přípustných znaků (tzv. adres). Tento sled znaků musí jednoznačně popisovat obráběcí postup na konkrétním stroji, musí být jednoznačně identifikovatelný jako celek a ve formě výpisu na tiskárně nebo obrazovce displeje musí být snadno srozumitelný a přehledný.

Při tvoření partprogramu je třeba vycházet z těchto údajů:
a) geometrie stroje (souřadný systém, orientace os, nulové body)
b) geometrie polotovaru (možnost kolize obrobek - nástroj během obrábění, umístění obrobku v souřadné soustavě stroje)
c) geometrie nástroje (rozměry, tvar, korekce dráhy nástroje na tvar obrobku)
d) geometrie výsledného obrobku (dána výkresem součástí)
e) technologické a řezné podmínky (řezné rychlosti, síla třísky atd.)
f) ostatní podmínky důležité pro činnost obráběcího stroje (chlazení, velikost posuvů, otáček, čísla nástrojů, korekcí atd.)

Výstavba partprogramu musí pak vyhovovat předepsané syntaxi zápisu, aby byla zaručena jednoznačnost vyjádření.

Partprogram je sestaven z řady jednotlivých elementárních operací, tzv. **bloků**. Blok se skládá z dílčích údajů, tzv. **slov**. Každé slovo pak obsahuje (až na některé výjimky) **adresu**, udávající o jaký druh informace jde a **číselný údaj**, udávající rozměrovou hodnotu nebo kódové přiřazení k nějaké funkci či operaci. Jeden blok může být rozepsán na více řádků, další blok je určen pouze adresou N!

Partprogram může být napsán klasickým způsobem (vycházejícím z doporučení ISO), ale umožňuje i zápis podle pravidel a doporučení pro řídicího systému CNC872, které oproti klasickému zápisu umožňuje více možností a bude popsán v tomto návodu.

*Příklad jednoho **bloku** partprogramu:*

```
N20 G1 X10.355 Z625.50 F300 S150 T12 M0
```

Slova v tomto bloku jsou : `N20, G1, X10.355, Z625.50, F300, S150, T12, M0`
Adresami jsou .: `N, G, X, Z, F, S, T, M`
Číselné údaje jsou: `20, 1, 10.355, 625.50, 300, 150, 12, 0`

Příklad rozepsání téhož bloku do více řádků:

```
N20   G1 X10.355 Z625.50
      F300 S150
      T12 M0
N30   G54 G0 X0 Y0 Z0
```

Pozn.: Programování do jisté míry zachovává kompatibilitu se staršími systémy MEFI, hlavně pokud se nepoužívaly aritmetické operace a práce s parametry. Naopak práce s parametry a aritmetické operace jsou silnou stránkou systémů CNC872 - především zjednodušují a zpřehledňují programování složitých partprogramů. Na eventuelní rozdíly bude v textu upozorněno.

Příklad použití nových způsobů programování:

```
N20   G1 AxGX=PolohaX+10 AxGZ=625.50
      Feed=300 S150
      Tool(2,42) Delay(5)
N30   If(Poloha == 0) G0 X100 Y100 Z100 EndIf
```

1.2 Kód vstupních informací

Systém přijímá vstupní informace partprogramu v textovém tvaru t.j. bez parity. Případná diakritická znaménka češtiny či jiných jazyků se mohou vyskytovat pouze v komentáři partprogramu. Adresy, t.j. znaky, jež systém zpracovává, jsou uvedeny v tabulce v kapitole 2.4.

2

2. STAVBA PARTPROGRAMU

2.1 Syntaxe partprogramu

Syntaxe partprogramu umožňuje do jisté míry zpracování partprogramů, vytvořených pro starší typy řídicích systému MEFI. Rozšíření syntaxe o nové prvky se týká jak vlastního partprogramu, tak hlavně hlavičkových souborů partprogramu.
Jednotlivá klíčová slova lze uvádět kdekoliv v partprogramu, každé klíčové slovo musí být uvedeno samostatně na řádku a musí ho předcházet znak #, například:

```
#INL(FileName.ext)
```

Klíčové slovo INL (include, vložit) způsobí, že se do zdrojového textu partprogramu vloží obsah souboru specifikovaného v závorkách za klíčovým slovem. Text se bude vkládat bezprostředně za kulatou závorkou ukončující jméno souboru, který se má vložit. Syntaxe:

```
#INL(FileName.ext)
```

kde FileName.ext je jméno souboru, který se má vložit do zdrojového textu partprogramu. Soubor s uvedeným jménem se bude hledat nejprve v adresářích s uživatelskými hlavičkovými soubory (adresář **CNC User File\Include**), potom v adresáři, ve kterém budou umístěny systémové hlavičkové soubory (**CNC Machine File\Include, resp. WinCNC\Include**).

V partprogramu je možné používat také klíčových slov MAC a CYC. Tato klíčová slova specifikují buď nějaké soubory, nebo nějaký číselný parametr, který se použijí společně s partprogramem,. Syntaxe pro klíčová slova specifikující jméno souboru je:

```
XXX(FileName.ext)
```

kde XXX je klíčové slovo a FileName.ext je jméno souboru. Syntaxe pro klíčová slova specifikující číselný parametr je:

```
XXX n
```

kde XXX je klíčové slovo a n je celé číslo. Podrobnější popis jednotlivých klíčových slov shrnuje tabulka:

Klíčové slovo	**Popis**
INL	Pro vložení jména souboru, který se má vložit do zdrojového textu partprogramu
MAC	Volba souboru makrocyklů. Všechny soubory makrocyklů, které se mají načíst musí být specifikovány pomocí tohoto klíčového slova. Klíčové slovo může být uvedeno v hlavičce vícekrát.
CYC	Volba souboru pevných cyklů. Klíčové slovo může být uvedeno v hlavičce vícekrát (pevné cykly mohou být ve více souborech).

Definice informačních hlášení (technologických zpráv v partprogramu) je definována kdekoliv ve zdrojovém textu partprogramu, tedy i v libovolném souboru vloženém do partprogramu pomocí klíčového slova INL.
Syntaxe:

```
&n 'text hlášení'
```

kde n je číslo hlášení a text hlášení je uveden mezi apostrofy. Řetězec mezi apostrofy musí být uveden na

jednom řádku. Text hlášení může obsahovat tzv. escape sekvence, které umožňují vkládání speciálních znaků a parametrů do řetězce. Tyto sekvence začínají zpětným lomítkem, jejich přehled je uveden v tabulce:

Escape sekvence	Význam
\n	Nový řádek (LF).
\t	Tabulátor.
\'	Apostrof (apostrof nelze psát do řetězce přímo).
\\	Zpětné lomítko (zpětné lomítko nelze psát do řetězce přímo, má-li být ve výsledném řetězci zpětné lomítko je třeba ho v zápisu zdvojit).
\nnn	Znak s ASCII hodnotou nnn (zapsána dekadicky, vždy tři cifry).
\xhh	Znak s ASCII hodnotou hh (zapsána hexadecimálně, vždy dvě cifry).
\r	Reálná hodnota. Místo této sekvence se vloží zápis reálné hodnoty, která bude uvedena jako příslušný parametr funkce MSG.
\i	Celočíselná hodnota. Místo této sekvence se vloží zápis celočíselné hodnoty, která bude uvedena jako příslušný parametr funkce MSG.

Je zavedena možnost definice maker (neplést s makrocykly!). Makro může být definována kdekoliv ve zdrojovém textu partprogramu, tedy i v libovolném souboru vloženém do partprogramu pomocí klíčového slova INL (stejně jako definice informačního hlášení). Syntaxe:

```
$MacroName(par1, par2, …) macro text
```

kde **MacroName** je jméno makra, v kulatých závorkách jsou uvedeny formální parametry (nepovinné) makra a **macro text** představuje rozvoj makra. Jméno makra musí začínat písmenem a může obsahovat písmena, číslice a znak '_'. Makro může mít libovolný počet formálních parametrů, pokud nemá žádné parametry, mohou se vynechat i kulaté závorky. Text makra může obsahovat formální parametry, které se při rozvoji nahradí skutečnými parametry. V textu makra je možné použít znak '|', který má stejný význam jako mezera, ovšem při expanzi makra se vypustí (lze použít v případě, že v rozvoji makra má vzniknout jedno „slovo" složením několika parametrů). Definice makra musí být uvedena na jednom řádku, má-li být rozdělena na více řádků, musí se na konec rozděleného řádku vložit znak zpětné lomítko (kromě posledního řádku !):

```
$MacroName(Par1, Parr2, Par3)       \
                   macro text       \
                   macro text cont
```

Makra definovaná tímto způsobem je možno použít („volat") ve zdrojovém textu partprogramu za místem definice.
V programu ve tvaru ISO se připouští zápis Ra = výraz, přičemž Ra je platný parametr, výraz může obsahovat parametry, operátory +, -, * a /, konstanty a funkce, prioritu je možné upravovat závorkami.
V programu ve tvaru ISO se umožňuje přímé poloměrové programování kružnic (bez nutnosti úpravy preprocesorem).
Pro programování parametrů se zavádí zápis R@=… . Při překladu se zavináč nahradí číslem, přičemž první výskyt znaku @ v bloku se nahradí číslem dle konfigurace (???) a každý další vždy číslem o 1 větší.
Rychlost se programuje vždy v mm/min nebo mm/ot (nezávisle na tom, zda se číslo zadá s desetinnou čárkou či nikoliv!).

Direktivy preprocesoru:
Direktivami preprocesoru se rozumí řídící příkazy (klíčová slova), definice informačních hlášení a definice maker. Direktivy preprocesoru musí začínat na novém řádku, uvozené znakem # (klíčová slova), & (definice informačních hlášení), resp. $ (makra) a nesmí být rozdělené na více řádků. Pouze definici makra je možné rozložit na více řádků pomocí znaku zpětné lomítko. Preprocesor bude pracovat jednoprůchodově, to znamená, že v rozvinu makra nemůže být uvedeno další makro!

2.2 Úvod do stavby partprogramu

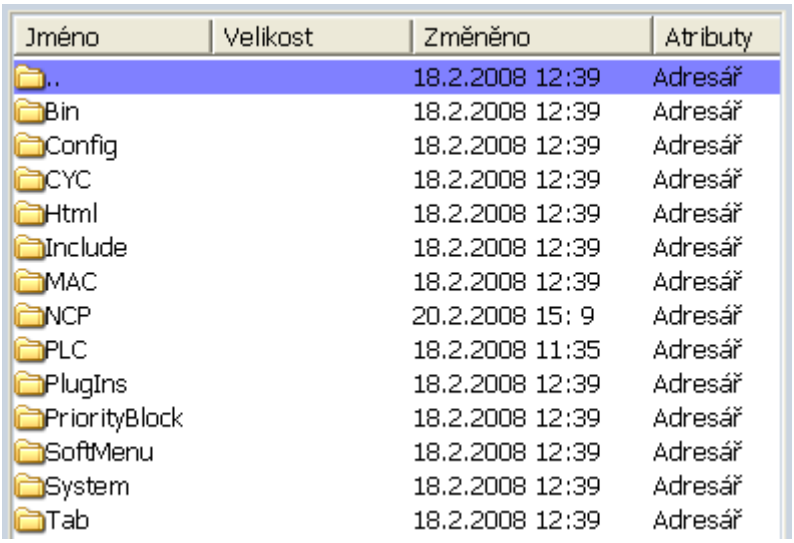
Partprogram sestává z bloků, které se postupně vykonávají po odstartování partprogramu. Blok obsahuje například zadání dráhy a typ interpolace, kterou se má systém přesunout do koncového bodu bloku, technologické funkce, které se mají vykonat buď na začátku nebo na konci bloku, aritmetické operace s parametry, deklaraci konstant, rozhodovací podmínky, volání podprogramů, makrocyklů či pevných cyklů apod. Vyjmenované akce mohou být v bloku samostatně nebo současně. Blok partprogramu může být i prázdný, tj. nemusí obsahovat nic. Jednotlivé bloky jsou odděleny adresou N, tj. v jednom bloku se vykoná vše, co je naprogramováno mezi oddělovačem bloků, což je adresa N. Z toho plyne, že jeden bloku partprogramu může být rozepsán na více řádek.

Kromě slov partprogramu, což je klasický zápis partprogramu podle ISO, lze v partprogramu používat i nové způsoby programování, především používání MAKER (pozor, neplést s makrocykly), práce a operace s parametry v matematickém zápise a s pojmenovanými parametry, používání systémových proměnných atd.

Protože v dalším textu se pojem MAKRO a MAKROCYKLUS často vyskytuje, poznamenejme, že MAKRO slouží ke zjednodušení a zpřehlednění programování tím, že preprocesor nahradí systémové nebo uživatelem naprogramované MAKRO za jeho rozvoj v partprogramu. MAKROCYKLUS je naopak samostatný soubor, který se vyvolává z partprogramu a po vykonání příkazů zapsaných v MAKROCYKLU se navrátí zpět do partprogramu. MAKROCYKLUS může obsahovat MAKRA.

2.3 Ukládání partprogramů v systému

Organizace adresářové struktury na řídicím systému je součástí návodu k obsluze, nicméně pro potřeby „Návodu k programování“ uvedeme několik důležitých poznámek. Na obrázku je uvedena podadresářová struktura adresáře **CNC User Files**, která je určena uživateli konkrétního stroje. **Při eventuelním upgrade systému nebo PLC programu zůstává beze změny**, tj. žádné soubory zde uvedené nebudou při instalaci změněny ani modifikovány.



Jméno	Velikost	Změněno	Atributy
..		18.2.2008 12:39	Adresář
Bin		18.2.2008 12:39	Adresář
Config		18.2.2008 12:39	Adresář
CYC		18.2.2008 12:39	Adresář
Html		18.2.2008 12:39	Adresář
Include		18.2.2008 12:39	Adresář
MAC		18.2.2008 12:39	Adresář
NCP		20.2.2008 15: 9	Adresář
PLC		18.2.2008 11:35	Adresář
PlugIns		18.2.2008 12:39	Adresář
PriorityBlock		18.2.2008 12:39	Adresář
SoftMenu		18.2.2008 12:39	Adresář
System		18.2.2008 12:39	Adresář
Tab		18.2.2008 12:39	Adresář

Z hlediska programátora jsou důležité především tyto adresáře:

- NCP
- MAC
- Include
- PriorityBlock
- Tab
- CYC

Důležité upozornění:
Pokud budou v tomto návodu uváděny pouze samotné názvy adresářů, vždy se bude jednat o adresáře umístěné v adresáři **CNC User Files,** neboť pouze tento adresář je určen uživateli k běžnému používání. Pokud se bude jednat o adresáře umístěné jinde, bude na to upozorněno, případně bude uvedena úplná cesta.

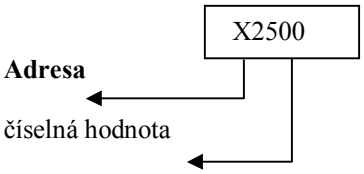
- **Adresář NCP** – zde se ukládají partprogramy, které vytvoří programátor buď přímo na systému, nebo se sem zkopírují z dostupného periferního připojení. Adresář NCP může mít další vnořené podadresáře s libovolnými názvy. Obecně mohou být partprogramy uloženy v libovolném jiném adresáři na disku, nicméně doporučuje se zachovávat toto uspořádání
- **Adresář MAC** – slouží jako úložiště MAKROCYKLU. Rovněž tento adresář může obsahovat další libovolné podadresáře. Při volání MAKROCYKLU se prohledává tento adresář.
- **Adresář Include** – zde se ukládají hlavičkové soubory, vytvořené obsluhou resp. programátorem pro konkrétní stroj. Hlavičkové soubory obsahují např. programátorem nadefinovaná MAKRA, které lze následně využívat v partprogramech či různé konstanty atd. Kromě MAKER, které si vytvoří a zde uloží uživatel systému, jsou k dispozici i hlavičkové soubory dodaná výrobcem systému nebo tvůrcem PLC pro konkrétní stroj. Ty se nacházejí ve stejnojmenných adresářích INCLUDE, které jsou ale součástí adresářů CNC Machine Files resp. WinCNC. Obsah těchto adresářů ale nesmí být uživatelem změněn.
- **Adresář PriorityBlock –** Obsahuje, resp. může obsahovat partprogram jednoho, tzv. prioritního bloku. Prioritní blok obsahuje funkce, které se nastaví při každé volbě programu, nebo např. při centrální anulaci. Typicky zde bývají např. funkce G40 (zrušení korekcí), G80 (zrušení volání pevných cyklů), G90 (absolutní programování) nebo systémová MAKRA – např. Translate(0,0,0) (zrušení programové transformace) apod. Obvykle je ale prioritní blok nastaven tvůrcem PLC (ve spolupráci s uživatelem stroje) a je tudíž v adresáři CNC Machine Files\PriorityBlock a po nastavení se již nemění.
- **Adresář TAB** – zde se ukládají tabulkové soubory, obvykle jsou ale spíše opět součástí adresáře CNC Machine Files\
- **Adresář CYC** – zde se ukládají soubory s pevnými cykly – pokud si je uživatel vytváří sám. Výrobce dodává pevné cykly (jsou součástí instalace a jsou v adresáři WinCNC\CYC. Uživatel si vlastní pevné cykly ukládá do adresáře CNC User files\CYC

2.4 Slovo partprogramu

2.4.1 Stavba slova

Elementárním stavebním prvkem partprogramu je tzv. slovo (instrukce programu) nebo makro. Každé slovo sestává z písmene adresy a jedno nebo víceciferného čísla obsahu adresy s případným znaménkem a (nepovinnou) desetinnou tečkou.

Příklad slova:



Makro je uživatelem nadefinované slovo, které preprocesor rozvine do kódu, který si uživatel (eventuelně výrobce systému) nadefinuje. Makro může být bez parametrů, případně parametrizované.

Příklad makra:

```
ENDPROGRAM  „ makro nadefinované výrobcem, provede rozvoj na funkci M2
LineAX(Angle,X)  „ makro provede výpočet přímky, zadané úhlem a jedním
koncovým bodem, v tomto případě souřadnicí X.
```

2.4.2 Psaní obsahu čísla

a) U zápisu hodnoty mohou být vynechány úvodní nuly před první významovou číslicí. Podle místních zvyklostí lze někdy psát úvodní nuly, především u funkcí (např. lze psát G00 i G0, M03 i M3 apod.), u ostatních adres se úvodní nuly, především kvůli přehlednosti, nepíší.
b) U zápisu hodnot souřadnic znamená údaj dráhu v mm, případně ve stupních.
   Např.:
   `X 3620 = 3620 mm, Z1.2 = 1.2mm apod., B48 = 48 stupňů apod.`
c) U zápisu hodnot souřadnic s desetinnou tečkou mohou být vynechány úvodní nuly před první platnou číslicí před desetinnou tečkou
   Např.:
   `X 0.12 = 0.12 mm`
   `X .12 = 0.12 mm`
d) Znaménko "+" může, ale nemusí být uváděno.
e) Nese-li slovo nulovou informaci, musí být zapsána minimálně jedna nula.
f) Mezi adresou a číselnou hodnotou slova, nebo mezi znaménkem a číselnou hodnotou slova může, ale nemusí být jedna, případně více mezer. Naopak **číselná hodnota nesmí být přerušena mezerou ani tabulátorem**. V případě parametrického programování je číselná hodnota nahrazena parametrem (viz kapitola o parametrickém programování).

Např.:

```
X36.12          správně
X  36.12        správně (mezera za adresou)
X36.  12        chybně (mezera uprostřed čísla)
X  -  36.12     správně (mezery mezi adresou a znaménkem  a mezi
                znaménkem a hodnotou)
```

2.4.3 Typy slov

Podle jakosti informace, kterými jsou jednotlivá slova partprogramu nositeli, rozdělujeme slova a jejich adresy do dvou základních skupin, a to na rozměrové a bezrozměrové. Rozměrové adresy mají hodnotu danou reálným číslem (typ REAL), bezrozměrové mají hodnotu danou celočíselným typem, integer (typ INT)

a) rozměrová slova - vyjadřují kromě hodnoty také rozměr. Mohou být kladná i záporná, s desetinnou tečkou nebo bez ní. Patří sem např. adresy : A,B,C,U,V,W,X,Y,Z,I,J,K,F,R .
b) bezrozměrová slova - vyjadřují pouze hodnotu. Patří sem např. adresy: N,G,M,T

Některá slova partprogramu se v systému pamatují trvale, tzn. že jejich platnost sahá do všech následujících bloků partprogramu, dokud nejsou přepsána jinou hodnotou téže adresy. Druhou skupinu slov tvoří slova, která mají platnost pouze v bloku, ve kterém byla programována.

2.4.4 Přehled programovatelných adres

V následujících tabulkách je uveden přehled všech programovatelných adres s uvedením jejich typu (reálné číslo nebo číslo typu integer) a přehled G a M funkcí

Seznam programovatelných adres

Pokud je adresa uvedena jako **konfigurovatelná,** může se podle typu stroje použít pro libovolné účely.
V komentáři je uvedeno obvyklé použití u standardních strojů. Poslední sloupec v tabulce udává u některých adres ekvivalentní zápis pomocí systémových parametrů nebo pomocí funkcí. Úplný přehled všech systémových parametrů a funkcí je v samostatných tabulkách.
Konfigurovatelné adresy A,B,C,U,V,W,X,Y,Z mohou obecně mít tyto ekvivalenty:
AXI0 – AXI8
AXAUX0 – AXAUX2
AXGX, AXGY, AXGZ
Tyto adresy jsou v tabulce uvedeny jako „Pozn1“

Adresa	Typ	Popis	Ekvivalent jako systémový parametr
/	-/INT	REZERVA (není implementováno - Podmíněné vynechání bloku)	
!	-	REZERVA (není implementováno)	
A	REAL	Konfigurovatelná adresa podle typu stroje (časté použití jako osa)	Pozn1
B	REAL	Konfigurovatelná adresa podle typu stroje (časté použití jako osa)	Pozn1
C	REAL	Konfigurovatelná adresa podle typu stroje (časté použití jako osa)	Pozn1
D	INT	Volba položky z tabulky korekcí	Za určitých předpokladů se může použít také ToolLenX,ToolLenY,ToolLenZ,ToolRadius (viz příklady dále v návodu)
E	INT	Pomocná funkce (pro PLC)	
F	REAL	• Rychlost posuvu • Časová prodleva pro G04	• Feed • DelayTime
G	INT	Přípravné funkce (viz seznam G-funkcí)	ProgrG(GNo)
H	REAL	• Pomocná funkce (pro PLC) • Úhel pro polární souřadnice	
I	REAL	Konfigurovatelná adresa, nejčastější použití jako interpolační parametr: • Vzdálenost středu kružnice ve směru geometrické osy X	CCX
J	REAL	Konfigurovatelná adresa, nejčastější použití jako interpolační parametr: • Vzdálenost středu kružnice ve směru geometrické osy Y	CCY
K	REAL	Konfigurovatelná adresa, nejčastější použití jako interpolační parametr: • Vzdálenost středu kružnice ve směru geometrické osy Z	CCZ
L	INT	• Číslo volaného podprogramu pro G71. • Číslo volaného makrocyklu pro G72 • Číslo zařazovaného pevného cyklu pro	

		G76. • Číslo cílového bloku skoku pro G73 • Číslo podprogramu, makrocyklu či pevného cyklu pro G79.	
M	INT	Pomocná funkce	ProgrM(MNo)
N	INT	Číslo bloku (číselná hodnota za znakem N nemusí být použita)	
O	REAL	Konfigurovatelná adresa • Pomocná funkce (pro PLC) • Orientace nástroje – prvek směrového vektoru	
P	REAL	Konfigurovatelná adresa • Pomocná funkce (pro PLC) • Orientace nástroje – prvek směrového vektoru	
Q	REAL	Konfigurovatelná adresa • Pomocná funkce (pro PLC) • Orientace nástroje – prvek směrového vektoru	
R	REAL	Poloměr kruhu pro G2 a G3.	CR
S	REAL	Otáčky vřetena	SPINDLESPEED
T	INT	Volba nástroje	Často to může být parametrizované PLC makro, např. Tool( )
U	REAL	Konfigurovatelná adresa, časté použití pro lineární pohyb rovnoběžný s osou X	Pozn1
V	REAL	Konfigurovatelná adresa, časté použití pro lineární pohyb rovnoběžný s osou Y	Pozn1
W	REAL	Konfigurovatelná adresa, časté použití pro lineární pohyb rovnoběžný s osou Z	Pozn1
X	REAL	Konfigurovatelná adresa, nejčastěji jako osa X	Pozn1
Y	REAL	Konfigurovatelná adresa, nejčastěji jako osa Y	Pozn1
Z	REAL	Konfigurovatelná adresa, nejčastěji jako osa Z	Pozn1

Přípravné G-funkce a pomocné M-funkce jsou dále rozděleny do skupin. Z hlediska programátora není číslo skupiny důležité, platí pouze, že přípravné G-funkce z jedné skupiny nesmí být programovány v témže bloku. Totéž platí pro pomocné M-funkce. Přehled G a M funkcí je uveden v následujících tabulkách:

Seznam G-funkcí

Skupina	Funkce	Popis	Ekvivalent jako systémový parametr nebo parametry, které se ke G-funkci mohou vázat
0	0	Najíždění rychloposuvem	
	1	Lineární interpolace	
	2	Kruhová interpolace ve směru hodinových ručiček (CW)	CR, CREV
	3	Kruhová interpolace proti směru hodinových ručiček (CCW)	CR, CREV
	10	Zkosení rohu se zadanou délkou zkosení CHLEN, nebo se zadanou vzdáleností začátku/konce zkosení od rohu CHLENT.	CHLEN, CHLENT
	11	Zaoblení rohu se zadaným poloměrem ROUNDR	ROUNDR

	12	Kruhová interpolace, tečné napojení na předchozí blok	
	13	Kruhová interpolace, tečné napojení na následující blok	
	33	Řezání závitů	
1	17	Volba pracovní roviny X-Y	
	18	Volba pracovní roviny Z - X	
	19	Volba pracovní roviny Y-Z	
2	5	Bez pětiosé transformace (zrušení pětiosé transformace)	
	6	Pětiosá transformace – zadání a interpolace přímo pomocí os které naklápějí nástroj	
	7	Pětiosá transformace - zadání a interpolace pomocí vektoru orientace nástroje	TOOLX, TOOLY, TOOLZ
	8	Pětiosá transformace - zadání úhlem úkosu, interpoluje se úhel úkosu a úhel naklopení v tečné rovině	BEVELANGLE
3	40	Bez poloměrové korekce (zrušení poloměrové korekce)	
	41	Poloměrová korekce vlevo	ToolRadius,RCAngle,RCMethod, RCChangeMethod,RCOptions,TNP
	42	Poloměrová korekce vpravo	ToolRadius,RCAngle,RCMethod, RCChangeMethod,RCOptions,TNP
4	23	Plynulá návaznost bloků (mezi bloky se nezastavuje – obálková rychlost)	
	24	Napojení bloků bez plynulé návaznosti (mezi bloky klesne rychlost na 0)	
5	50	Posunutí nulového bodu, číslo řádku v tabulce í udává adresa L	Translace(),ATranslate(),WTranslate(),WATranslate()
	53	Posunutí 0 nulového bodu	Dtto
	54	Posunutí 1 nulového bodu	Dtto
	55	Posunutí 2 nulového bodu	Dtto
	56	Posunutí 3 nulového bodu	Dtto
	57	Posunutí 4 nulového bodu	Dtto
	58	Posunutí 5 nulového bodu	Dtto
	59	Posunutí 6 nulového bodu	Dtto
6	94	Posuv v mm/min bez konstantní řezné rychlosti	
	95	Posuv v mm/ot bez konstantní řezné rychlosti	
	96	Posuv v mm/ot s konstantní řeznou rychlostí	SLIM,CCS
	97	Posuv v mm/min s konstantní řeznou rychlostí	SLIM,CCS
7	70	Konec podprogramu, makrocyklu nebo pevného cyklu	END
	71	Volání podprogramu, číslo podprogramu udává adresa L	Call(SubNo)
	72	Volání makrocyklu, číslo makrocyklu udává adresa L	CallMacro(MacNo)
	73	Skok na programový blok, jehož číslo udává adresa L	JMP(BlockNo)
	79	Začátek podprogramu, makrocyklu či pevného cyklu, číslo udává	BEGIN

		adresa L	
8	76	Zařazení volání pevného cyklu na konci každého bloku, číslo pevného cyklu udává adresa L	
	80	Zrušení opakovaného volání pevného cyklu na konci každého bloku	
	81	Zařazení volání pevného cyklu č. 81 na konci každého bloku	
	82	Zařazení volání pevného cyklu č. 82 na konci každého bloku	
	83	Zařazení volání pevného cyklu č. 83 na konci každého bloku	
	84	Zařazení volání pevného cyklu č. 84 na konci každého bloku	
	85	Zařazení volání pevného cyklu č. 85 na konci každého bloku	
	86	Zařazení volání pevného cyklu č. 86 na konci každého bloku	
	87	Zařazení volání pevného cyklu č. 87 na konci každého bloku	
	88	Zařazení volání pevného cyklu č. 88 na konci každého bloku	
	89	Zařazení volání pevného cyklu č. 89 na konci každého bloku	
9	90	Režim absolutního programování souřadnic	
	91	Režim přírůstkového programování souřadnic	
10	4	Časová prodleva, velikost v sekundách je zadána funkcí F	

Seznam M-funkcí

Pozn.:
M-funkce jsou ve většině případů plně v kompetenci PLC programu pro konkrétní stroj. V seznamu jsou M-funkce podle doporučení ISO. Seznam platných M-funkcí pro konkrétní stroj je součástí dokumentace ke stroji. Opět platí, že v jednom bloku nelze psát M-funkce z jedné skupiny. Pokud se ale M-funkce programují pomocí systémových parametrů, toto pravidlo neplatí – používá se to např. v rozhodovacích podmínkách. Podrobněji v dalších částech návodu. Ekvivalenty systémových parametrů v tabulce nejsou, pro příklad uveďme pouze standardní makro pro konec programu: ENDPROGRAM se nahradí funkcí M2.

Skupina	Popis	
1	00	Programový stop
	01	Volitelný stop (viz možnosti jízdy)
	02	Konec partprogramu
	30	Konec partprogramu „ s převinutím“ – provede se automaticky opětovná volba právě odjetého partprogramu. Pozn: tuto vlastnost nutno povolit v registrech nastavením **EnableRewind** = 1 (nastavení zajistí návrhář PLC)
2	03	Start vřetena "CW"
	04	Start vřetena "CCW"
	05	Stop vřetena
	19	Stop vřetena v orientovaném bodě
3	40	Rozsah otáček vřetena je vypočten přímo z funkce S
	41	Otáčky vřetena rozsah 1
	42	Otáčky vřetena rozsah 2
	43	Otáčky vřetena rozsah 3
	44	Otáčky vřetena rozsah 4

4		REZERVA
5	7	Zapnutí chlazení 2
	8	Zapnutí chlazení 1
	9	Vypnutí chlazení 1a 2
	17	Zapnutí chlazení 1a 2
6	50	Zapnutí chlazení 3
	51	Zapnutí chlazení 4
	52	Vypnutí chlazení 3 a 4
	53	Zapnutí chlazení 3 a 4
7	10	Upnutí obrobku
	11	Uvolnění obrobku
8	49	Překlenutí ručního FEED OVERRIDE
	48	Zrušení překlenutí (zařazení) FEED OVERRIDE
9	06	Výměna nástroje
	60	Výměna obrobku
10	06	Volitelné M-funkce
11	06	Volitelné M-funkce
12	06	Volitelné M-funkce
13	06	Volitelné M-funkce
14	06	Volitelné M-funkce

Pozn.:
Jaké M- a G-funkce se uplatní v případně, že nejsou v partprogramu programované je dáno konfigurací tzv. prioritního bloku, jenž je popsán samostatné kapitole.

2.5 Číslovaní bloků, začátek a konec partprogramu

Jednotlivá slova a další prvky (např.MAKRA) se v partprogramu sestavují do programových bloků. Programový blok tvoří základní informační jednotku nesoucí údaj o geometrii, technologii obrábění, může obsahovat pomocné výpočty, rozhodovací podmínky apod. Formát bloku má proměnnou délku.

Každý blok je uveden adresou N a číslem bloku, které je nepovinné. Číslo bloku slouží pro orientaci v partprogramu a pro volbu bloku, proto se doporučuje je psát. Nezbytné je psát číslo bloku, pokud se na něj provádí skok. Číslo bloku může být i nula, což je to samé jako N bez čísla. V partprogramu se mohou vyskytovat jak číslované bloky, tak bloky N0 nebo N bez čísla. Pokud je blok číslovaný, musí být číslo bloku jedinečné, tj. v partprogramu se nesmí opakovat stejná čísla bloků. Maximální povolené číslo bloku je N2147483646 (maximální kladné číslo typu integer)

Partprogram začíná (resp. může začínat, není povinné) klíčovým slovem resp. makrem **PROGRAM**.
(Pozn.: Kvůli kompatibilitě se staršími systémy může začínat partprogram také znakem procenta a max. šestimístným číslem – zápis ale nemá žádný praktický význam).

Partprogram končí klíčovým slovem **ENDPROGRAM**

Klíčová slova **PROGRAM** a **ENDPROGRAM** jsou pomocná **MAKRA,** která jsou definovaná v hlavičkovém souboru StdHdr.nch a jsou nastavena výrobcem systému. Patří ke skupině maker, která zjednodušují a zpřehledňují programování. Rozvoj makra (stav k revizi 09 Rev.376) **PROGRAM** je prázdný, v rozvoji makra **ENDPROGRAM** je funkce M2., **ENDPROGRAMREWIND** je funkce M30. O programování pomocí maker je pojednáno v samostatné kapitole.

Doporučený formát partprogramu *Interpretace takto zapsaného partprogramu:*

```
N PROGRAM
N10 …                                   N10 …
N20 …                                   N20 …
N30 …                                   N30 …
……                                      ……
N ENDPROGRAM                            N M2
```

Makro ENDPROGRAM resp. ENDPROGRAMREWIND neznamená konec souboru, za tímto klíčovým slovem mohou následovat ještě podprogramy.

2.5.1 Další pokyny k sestavení programového bloku

V partprogramu se mohou vyskytovat komentáře. KOMENTÁŘ je řetězec libovolných znaků (kromě uvozovek) uzavřený mezi uvozovkami. Pokud je za komentářem uveden znak konce řádku (CR,LF), nemusí být koncové uvozovky uvedeny.

Příklad:

```
N10 X100 Y100     " Toto je komentář"
N20 X200 Y200     " Toto je komentář
" "
"
N30 X100 " Toto je komentář " Y200 Z300
N40 X100 " Toto je komentář   Y200 Z300
```

Pozn.:
V bloku N40 se do komentáře zahrne i Y200 Z300, neboť nejsou uvedeny druhé uvozovky, takže se za konec komentáře považuje konec řádky.

a) Každý programový blok musí začínat adresou N - číslo bloku. Blok končí před následujícím znakem N (číslo následujícího bloku) nebo koncem souboru. Jeden blok partprogramu může být tedy rozepsán na libovolný počet řádků. Pořadí slov v jednom bloku je libovolné – záleží pouze na zvyklostech programátora. Délka řádku v podstatě není omezena, nicméně je vhodné psát maximálně 50 znaků na jednu řádku, aby při chodu partprogramu byl vidět celý řádek, bez nutnosti v něm horizontálně rolovat.
b) Systém pracuje s proměnnou délkou bloku. To znamená, že v každém bloku může být libovolný počet slov. Každá adresa (slovo) může být zapsána v jednom bloku pouze jednou, s výjimkou skupinových funkcí (M,G), u kterých může být zapsána jedna hodnota z každé skupiny a parametrů, které mohou být v bloku použity vícekrát.
c) Místo hodnoty adresy je možné u všech adres kromě N zapisovat číslo parametru - viz kapitola o parametrickém programování.
d) Jako oddělovače mezi jednotlivými slovy bloku lze užít libovolný počet mezer. Mezery se mohou navíc užít i uvnitř slova, avšak pouze mezi adresou a číslem (nikdy ne uvnitř číselné hodnoty !).

2.6 Podprogramy, makrocykly a pevné cykly

Složitější partprogram součásti se může skládat nejen z vlastního partprogramu, ale může obsahovat i volání podprogramů, makrocyklů a tzv. pevných cyklů. Podprogramy se vztahují pouze k danému partprogramu, za jehož koncem musí být bezprostředně zapsány. Makrocykly a pevné cykly se nacházejí v samostatných souborech a jsou z těla partprogramu event. podprogramu vyvolány příkazem.

Podprogramy, makrocykly i pevné cykly lze volat třemi různými způsoby – klasickým zápisem G-funkce, zápisem MAKRA a voláním funkce. **Hned na úvod poznamenejme, že se doporučuje používat třetí způsob zápisu a to pomocí funkce.** Důvody rozebereme podrobně v následující kapitole.

2.6.1 Podprogramy

Podprogramem se rozumí určitá skupina programových bloků, které mají standardní úvodní a závěrečný blok. Podprogram logicky patří pouze k danému partprogramu a může být volán pouze z tohoto partprogramu – je zapsán bezprostředně za logickým koncem partprogramu (za klíčovým slovem ENDPROGRAM resp. ENDPROGRAMREWIND resp. za funkcí M2 nebo M30).
Podprogram se vyvolá zápisem makra SUB(SubNo), kde formální parametr SubNo je číslo podprogramu. Vlastní podprogram je uvozen klíčovým slovem BEGIN(SubNo) a ukončen klíčovým slovem END. V příkladu uvedená čísla bloků jsou nepovinná.

Volání a zápis vlastního podprogramu uvádí následující příklad:

```
N PROGRAM          " začátek partprogramu
N …
N100 SUB(10)       " vyvolání podprogramu číslo 10
N110 SUB(20)       " vyvolání podprogramu číslo 20
N …
N ENDPROGRAM       " konec partprogramu

N200 BEGIN(10)     " Začátek podprogramu číslo 10
N210 …
N220 END           " Konec podprogramu číslo 10

N300 BEGIN(20)     " Začátek podprogramu číslo 20
N310 …
N320 END           " Konec podprogramu číslo 20
```

V příkladu bylo použito systémových maker SUB(SubNo), BEGIN, END. Pro porovnání uvádíme zápis bez použití maker, který se u nových systému už nedoporučuje používat (snad jen při eventuelní použití partprogramů ze starších systémů MEFI). Nejsou použita ani makra začátku a konce partprogramu. Funkce je nicméně stejná.

```
%100               " začátek partprogramu
N …
N100 G71 L10       " vyvolání podprogramu číslo 10
N110 G71 L20       " vyvolání podprogramu číslo 20
N120 …
N200 M30           " konec partprogramu

N200 G79 L10       " Začátek podprogramu číslo 10
N210 …
N220 G70           " Konec podprogramu číslo 10
```

```
N300 G79  L20     " Začátek podprogramu číslo 20
N310 …
N320 G70          " Konec podprogramu číslo 20
```

Ani jeden z uvedených způsobů volání (SUB() resp. G71 Lxx) však nelze použít, pokud chceme volat podprogram podmíněně, tj. na základě vyhodnocení nějaké podmínky funkcí If (viz kapitola o funkcích). Bez dalšího vysvětlení uvedeme příklad takového bloku, kde se vyhodnotí podmínka, např. reálný parametr R1 a je-li jeho hodnota nenulová, volá se podprogram č. 1, jinak se volá podprogram č.2:

```
N10 If(I1) SUB(1) Else SUB(2) EndIf       " Nelze použít !!
N10 If(I1) G71 L1 Else G71 L2 EndIf       " Nelze použít !!
```

Ani jeden z těchto zápisů nelze použít, neboť v bloku N10 se vyskytují stejné funkce (G71) z jedné skupiny, jak je na první pohled zřejmé z druhého řádku a první řádek je jenom zjednodušený zápis, který se rozvine do stejného zápisu jako v druhém řádku. Programovat v jednom bloku stejné adresy nebo G, resp. M-funkce z jedné skupiny systém zakazuje.

Pro vyvolání podprogramu se proto doporučuje používat funkci **CALL(SubNo)**, kde SubNo je číslo podprogramu. Vidíme, že zápis je velmi podobný jako zápis pomocí makra, pouze místo SUB(SubNo) se použije CALL(SubNo). Zápis vlastních podprogramů je stejný. Pokud se programuje volání pomocí funkce **CALL(SubNo),** tak můžeme využít i volání různých podprogramů v podmínce

```
N10 If(I1) CALL(1) Else CALL(2) EndIf          " Povoleno
```

Příklad volání podprogramů funkcí CALL( ):

```
$VYMENA     20
N PROGRAM         " začátek partprogramu
N …
N100 CALL(10)     " vyvolání podprogramu číslo 10
N110 CALL(VYMENA) " vyvolání podprogramu číslo 20 (rozvoj makra VYMENA)
N ENDPROGRAM
"
N200 BEGIN(10)    " Začátek podprogramu číslo 10
N210 …
N220 END          " Konec podprogramu číslo 10

N300 BEGIN(VYMENA)" Začátek podprogramu číslo 20
N310 …
N320 END          " Konec podprogramu číslo 20
```

Druhý podprogram má místo čísla uvedeno VYMENA, což ale není nic jiného než rozvoj makra, které je nadefinované před začátkem partprogramu a jeho rozvojem je číslo 20. Tento způsob lze používat např. pro snadnější zapamatování názvů podprogramů.

Po skončení podprogramu je obvyklý požadavek, aby hodnoty technologických a pomocných funkcí zůstaly ve stavech tak, jak je nastavil podprogram. Tato vlastnost se v případě potřeby dá změnit funkcí **SubOpt(Option,Set)** – podrobněji popsáno v kapitole o funkcích.

Podprogram je možno editovat v témže rozsahu jako vlastní partprogram. V programovém bloku partprogramu, ze kterého je volán podprogram, je přípustné programovat i ostatní posuvové a technologické funkce. Vlastní odskok na zvolený podprogram se provede až v závěru bloku, t.j. po vykonání ostatních programovaných operací. Po provedení podprogramu se řízení vrátí na následující blok partprogramu, ze kterého byl odskok proveden.

2.6.2 Makrocykly

Makrocyklem se rozumí určitá skupina programových bloků, které tvoří partprogram pro typickou součást nebo část součásti vyráběnou na konkrétním stroji. Makrocyklus je obecně vztažen ke všem uloženým partprogramům, ze kterých může být volán (může být volán z vlastních partprogramů i jejich podprogramů). Časté je použití makrocyklů pro činnosti, které jsou společné všem partprogramům, např. nájezd do polohy pro výměnu nástroje a výměna nástroje. Je vhodné využít i parametrického programování.

Volání makrocyklu je možno provést z kteréhokoliv bloku libovolného partprogramu. Volání má podobný tvar jako u podprogramů, místo klíčového slova Call(SubNo) se použije **CallMacro(MacNo)**, kde MacNo je číslo makrocyklu, resp. místo klíčového slova SUB(SubNo) se použije **MAC(MacNo)**, kde MacNo je číslo makrocyklu, Protože makrocykly nejsou součástí partprogramu, musí být povinně na začátku partprogramu uvedena jejich deklarace, která znamená příkaz pro načtení deklarovaného souboru do paměti. Makrocykly se standardně ukládají do adresáře MAC. Pokud by bylo nutné makrocykly nějakým způsobem dále třídit, je možné v adresáři MAC založit další podadresáře.

Doporučený způsob používání makrocyklů je následující. Uživatel si vytvoří soubor, nazvaný například MCWHN13.NCP (makrocykly pro stroj WHN13) a do něj postupně zapisuje makrocykly, tak jak je během používání stroje vytváří.

Deklarace makrocyklů (tj. načtení do paměti) v partprogramu je následující :

```
#MAC(MCWHN13.NCP)       "Soubor MCWHN13.NCP je umístěn v adresáři MAC
#MAC(.\DESK\TOOL10.NCP) "Soubor TOOL10.NCP je umístěn v adresáři MAC\DESK
```

Nebo-li pokud je soubor s makrocykly umístěn přímo v adresáři MAC, uvede se jen jeho název. Pokud je umístěn v nějakém podadresáři adresáře MAC, musí se uvést relativní cesta k tomuto souboru vzhledem k adresáři MAC (tečka před lomítkem)

Doporučené volání makrocyklů z partprogramu uvádí následující příklad. Poznamenejme, že s omezeními, uvedenými u podprogramů, lze makrocykly také volat zápisem MAC. Příklad už není uveden.

```
#MAC(MCWHN13.NCP)
N PROGRAM          " začátek partprogramu
N …
N …
N100 CallMacro(1) " vyvolání makrocyklu číslo 1
N105
N110 CAllMacro(2) " vyvolání makrocyklu číslo 2
N …
N ENDPROGRAM       " konec partprogramu
```

Požadovaný makrocyklus může být rovněž volán i z kteréhokoliv bloku jiného makrocyklu (tzv. vnořování makrocyklů). Po skončení makrocyklu zůstávají (obvykle) hodnoty technologických a pomocných funkcí ve stavech tak, jak je nastavil makrocyklus. Opět je možné tuto vlastnost změnit nastavením **SubOpt(Option,Set)** – viz kapitola o funkcích.

Příklad souboru MCWHN13.NCP s makrocykly:

```
N BEGIN(1)         " začátek makrocyklu číslo 1
N …                " tělo makrocyklu např. pro výměnu nástroje
N …
N END              " konec makrocyklu číslo 1
N BEGIN(2)         " začátek makrocyklu číslo 2
N …                " tělo makrocyklu např. otáčení stolu
N …
N END              " konec makrocyklu číslo 2
N BEGIN(3)         " začátek makrocyklu číslo 3
N …                " tělo makrocyklu např. pro proměřováni obrobku
N …
N END              " konec makrocyklu číslo 3
```

2.6.3 Skoky v partprogramu

Skoky jsou povoleny pouze na programové bloky v rámci programového celku, t.j. v rámci partprogramu, nebo podprogramu či makrocyklu ve směru vzad i vpřed (tj. ve směru k začátku nebo konci partprogramu, podprogramu nebo makrocyklu). Nelze tedy použít např. skoku z partprogramu do podprogramu.

Pro zápis skoku se použije funkce **JMP(BlockNo),** kde BlockNo je číslo bloku, na který se provede skok. Skok směrem „dozadu“, tj. k začátku partprogramu bez další podmínky, která určí, kdy cyklus opustit, nemá praktický význam, protože by došlo by k „nekonečnému“ zacyklení partprogramu (vyhlásí se chyba „Program obsahuje příliš mnoho bloků“).

Skok směrem „dopředu“, ke konci partprogramu, se použije pokud chceme např. ve fázi odlaďování partprogramu přeskočit určitý úsek. Většinou se ale funkce skoku používá ve spojení s rozhodovacími podmínkami – viz kapitola o funkcích.

Pro ilustraci použití je uveden příklad s podmínkou (funkce If), viz kapitola o funkcích). V bloku N100 se porovnají parametry 7 a 8 a v případě jejich rovnosti se skočí zpět na blok N20. Za blokem N20 je úsek partprogramu, kde se parametry I7 a I8 nastaví na základě nějaké podmínky, která se v bloku N100 vyhodnotí. Poznamenejme, že někdy musí vyjít nerovnost, aby se program nezacyklil a pokračoval dál na blok N110 a dále.

```
N PROGRAM          " začátek partprogramu
N10 …
N20 …
…                  " výpočty parametrů I7,I8
…
N100 If(I7 == I8) JMP(20) EndIf
N110
N120
N …
N ENDPROGRAM
```

Často je potřeba programovat skok zpět několikrát po sobě. Můžeme sice využít rozhodovací podmínky If a nadefinovat si parametry, které budeme inkrementovat a porovnávat, jak bylo popsáno v předešlém příkladu, ale můžeme použít také předdefinované makro (od revize systému 822), které provede n-krát skok zpět na zvolený blok, resp. přesněji řečeno provede n-krát zvolený úsek partprogramu (první průchod a skok zpět n-1 krát, t.j. celkem n průchodů). Zápis je mnohem jednodušší. Uvedeme rozvoje tohoto makra, které má tři parametry:

```
$Loop(BlokNo, Count, Counter)       \
      Counter = Counter + 1         \
      If(Counter < Count)     \
            Jmp(BlokNo)             \
      Else                          \
            Counter = 0             \
      EndIf
```

Jednotlivé parametry mají tento význam:
BlockNo: Číslo bloku, kterým smyčka začíná (musí být před blokem, kde je programováno Loop)
Count: Počet průchodů
Counter: Čítač počtu průchodů (celočíselný parametr)

Příklad:
Úsek partprogramu mezi bloky N110 a N130 se provede 5x

```
N PROGRAM
...
```

```
N90 I1=0                „ Případná inicializace čítače průchodů
....
N100 X0 G0 G90          „ Najede absolutně na X0
N110 G91 X1             „ Jede přírůstkově v ose X o 1mm
N120 ...
N130 Loop(110,5,I1)     „ Skok na blok N110 se 4x zopakuje
N140 ...
```

Druhý parametr udává celkový počet průchodů (5), nikoli počet skoků (ten je o jednu menší)! Úsek partprogramu mezi bloky N110 a N130 se tedy provede 5x, v bloku N140 bude míra X5. Pomocný čítač počtu průchodů je v příkladu v parametru I1. Pokud se použitý parametr v partprogramu nepoužije, nemusí se inicializovat (t.j. nulovat). Pokud by se použil v části partprogramu před použitím makra Loop, musí se předem inicializovat, t.j. vynulovat (blok N90). Inicializaci můžeme pro jistotu provést vždy.

Makra Loop můžeme do sebe vnořovat, pro každé makro ale musíme použít jiný parametr pro čítač průchodů, zde použit I1 a I2. Koncová míra souřadnic v příkladu bude X5 a Y25

```
N PROGRAM
N10  G90 X0 Y0 G00
N20  X1 G91 G00         „ přírůstkově X1
N30  Y1                 „ přírůstkově Y1
N40  Loop(30,5,I1)      „ skok na blok N30
N50
N60  Loop(20,5,I2)      „ skok na blok N20
N70
N ENDPROGRAM
```

2.6.4 Makra

V kapitole 2.1 byla zmíněna možnost používat v partprogramech tzv. MAKRA a uvedena jejich definice. Zde uvedeme příklady používání maker v partprogramech.
Makra si může uživatel vytvářet sám. Kromě toho jsou k dispozici některá předdefinovaná (systémová) makra, umístěná obvykle v hlavičkových souborech (t.j. v adresářích C:\Program Files\MEFI\WinCNC\Include nebo C:\Machine Fines\Include, soubory s příponou *.NCH).
Uživatel si svoje makra rovněž může uložit do hlavičkových souborů (v adresáři D:\CNC User Fines\Include) v případě, že je chce používat v různých partprogramech, nebo může makro zapsat přímo do partprogramu (obvykle se MAKRA píší na začátek), pokud se jedná o jednorázové použití.

Jako příklad, kde jsou uvedená systémová makra, můžeme uvést systémový hlavičkový soubor StdHdr.nch, který je umístěný v adresáři:
C:\Program Files\MEFI\WinCNC\Include
V něm jsou uvedená např. tato makra (konstanty)

```
$IMPERIAL    1                                  " Palcové zadávání - inch, inch/min, ...
$METRIC      0                                  " Metrické zadávání – mm, mm/min
$PI          3.14159265358979323846264338327795 " Ludolfovo číslo
```

V následujícím příkladu partprogramu jsou uvedená makra použita, kromě nich jsou vytvořena ještě dvě makra uživatele:

```
$Obvod      RPARAM
$Polomer    RPARAM

N PROGRAM
N X0 G00 G90
N LENGTHUNIT = IMPERIAL       " Programování délek v palcích
N X1 G0                       " Pojede na míru 25,4mm (= 1 palec)
```

```
N X0
N LENGTHUNIT = METRIC         " Programování délek v mm
N X1 G0                       " Pojede na míru 1 mm
N X0
N Polomer = 10
N Obvod = 2 * PI * Polomer
N MsgShow(1,'Obvod kruhu o poloměru \r = \r',Polomer,Obvod)
N ENDPROGRAM
```

Textový řetězec IMPERIAL je v preprocesoru nahražen 1, textový řetězec METRIC je nahražen 0. Stejný výsledek bychom tedy dostali, pokud bychom použili tato čísla. Pojmenované konstanty jsou ovšem lépe zapamatovatelné. Makra Obvod a Polomer jsou nahražena automaticky vygenerovanými reálnými parametry.

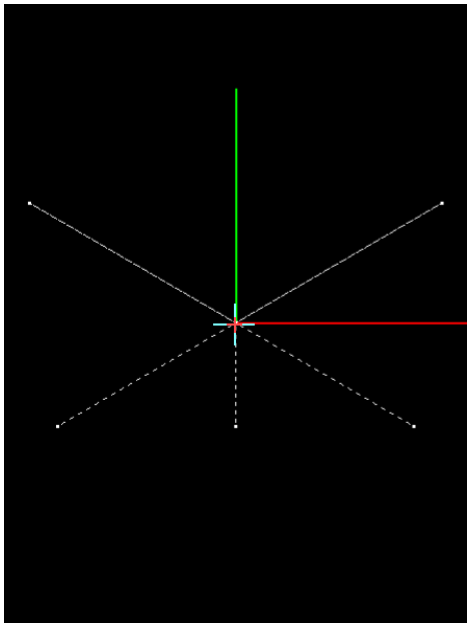
Další příklad uvádí rozvoj složitějšího makra. Jedná se o makro, které generuje přímku, resp. koncové body přímky na základě úlu a souřadnice X (makro LineAX), resp úhlu a souřadnice Y (makro LineAY). Makro má dva parametry, první parametr je úhel, druhý parametr je souřadnice X (resp. souřadnice Y). V makru jsou dále ošetřeny stavy při zadání úhlu, kdy tangens není definován (hlásí se chyba). Pozn: kvůli přehlednosti makra nejsou ošetřeny stavy pro úhly větší než 360 stupňů, ve skutečnosti by i tento stav měl být ošetřen.

```
$LineAX(Angle,ProgrX)                                                    \
if(OR(OR(EQ(Angle,90),EQ(Angle,270)),OR(EQ(Angle,-90),EQ(Angle,-270))))  \
  Err('Pro LineAX nejsou úhly 90, -90, 270 a -270 stupňů povoleny!')     \
else                                                                     \
  AxGY=AxGY+(ProgrX-AxGX)*Tan(Angle)                                     \
  AxGX=ProgrX                                                            \
endif

$LineAY(Angle,ProgrY)                                                    \
if(OR(OR(EQ(Angle,0),EQ(Angle,180)),OR(EQ(Angle,0),EQ(Angle,-180))))     \
  Err('Pro LineAY nejsou úhly 0, 180 a -180 stupňů povoleny!')           \
else                                                                     \
  if(OR(OR(EQ(Angle,90),EQ(Angle,270)),OR(EQ(Angle,-90),EQ(Angle,-270)))) \
    AxGY=ProgrY                                                          \
  else                                                                   \
    AxGX=AxGX+(ProgrY-AxGY)/Tan(Angle)                                   \
    AxGY=ProgrY                                                          \
  endif                                                                  \
endif
```

Příklad partprogramu, který používá výše uvedená makra

```
N PROGRAM
N G00 X0 Y0
N LineAX(30,100)  " Úhel 30 stupňů, X100
N G00 X0 Y0
N LineAX(150,-100)" Úhel 150 stupňů X-100
N G00 X0 Y0
N LineAY(-30,-50) " Úhel -30 stupňů Y-50
N G00 X0 Y0
N LineAY(210,-50) " Úhel 210 stupňů Y-50
N G00 X0 Y0
N LineAY(90,-50)  " Úhel 90 stupňů  Y-50
N G00 X0 Y0
N ENDPROGRAM
```



3

3. Funkce

3.1 Význam funkcí

Funkce jsou systémová předdefinovaná makra, která usnadňují programování, zpřehledňují tvorbu partprogramu Přehled všech dostupných systémových funkcí je uveden v následující tabulce.

3.1.1 Přehled funkcí

UMinus(Val)	Unární minus
Plus(Val1, Val2)	Sčítání
Minus(Val1, Val2)	Odčítání
Multiply(Val1, Val2)	Násobení
Divide(Val1, Val2)	Dělení
Mod(Val1, Val2)	Zbytek po dělení
Eq(Val1, Val2)	Rovno
NEq(Val1, Val2)	Nerovno
Less(Val1, Val2)	Menší
LE(Val1, Val2)	Menší nebo rovno
Greater(Val1, Val2)	Větší
GE(Val1, Val2)	Větší nebo rovno
Not(Val1)	Logický not (Val1 i návratová hodnota je integer)
And(Val1, Val2)	Logický and (Val1, Val2 i návratová hodnota je integer)
Or(Val1, Val2)	Logický or (Val1, Val2 i návratová hodnota je integer)
Xor(Val1, Val2)	Logický xor (Val1, Val2 i návratová hodnota je int eger)
Cond(Cond, TrueVal, FalseVal)	Podmíněný výraz, je-li podmínka Cond splněna, vrací hodnotu TrueVal, jinak FalseVal
If(Cond)	Podmínka se považuje za splněnou, je-li Cond nenulová hodnota. Operace programované mezi If a ElseIf/Else/EndIf se provedou jen v případě, že podmínka je splněna.
ElseIf(Cond)	Podmínka se považuje za splněnou, je-li Cond nenulová hodnota. Operace programované mezi ElseIf a následujícím ElseIf/Else/EndIf se provedou jen v případě, že podmínka je splněna a dosud nebyla splněna podmínka pro předchozí If/ElseIf.
Else	Operace programované mezi Else a EndIf se provedou jen v případě, že nebyla splněna podmínka pro žádný z odpovídajících If/ElseIf.
EndIf	Konec podmínky
Jmp(BlockNo)	Nepodmíněný skok v rámci programového celku
Call(SubNo)	Volání podprogramu
CallMacro(MacNo)	Volání makrocyklu
CallCycle(CycNo)	Volání pevného cyklu
SubOpt(Option, Set)	Vlastnosti podprogramu/makrocyklu/pevného cyklu

PreserveR(From, To)	Úchova zadaných reálných parametrů v podprogramu/makrocyklu/pevném cyklu
PreserveI(From, To)	Úchova zadaných celočíselných parametrů v podprogramu/makrocyklu/pevném cyklu
BAnd(Val1, Val2)	Bitový and (Val1, Val2 i návratová hodnota je integer)
BOr(Val1, Val2)	Bitový or (Val1, Val2 i návratová hodnota je integer)
BXor(Val1, Val2)	Bitový xor (Val1, Val2 i návratová hodnota je integer)
BNot(Val1)	Bitový not (Val1 i návratová hodnota je integer)
ShL(Val, BitCnt)	Binární posun doleva (shift left) (Val i návratová hodnota je integer)
ShR(Val, BitCnt)	Binární posun doprava (shift right) (Val i návratová hodnota je integer)
Int(Val)	Převod reálného čísla na celé
Real(Val)	Převod celého čísla na reálné
Round(Val)	Zaokrouhlení reálného čísla
Trunc(Val)	Celá část reálného čísla
Tenths(Val)	Desetinná část reálného čísla
Abs(Val)	Absolutní hodnota
Sgn(Val),	Znaménko (Val > 0: 1, Val == 0: 0, Val < 0: -1)
Even(Val)	Sudý
Odd(Val)	Lichý
Sqr(Val)	Druhá mocnina
Sqrt(Val)	Druhá odmocnina
Exp(Val)	Exponenciela (e^x)
Exp2(Val)	2^x
Exp10(Val)	10^x
Log(Val)	Přirozený logaritmus
Log2(Val)	Dvojkový logaritmus
Log10(Val)	Desítkový logaritmus
Pow(Val1,Val2)	x^y (Val1, Val2 a návratová hodnota jsou reálná čísla)
Sin(Val)	Sinus
Cos(Val)	Cosinus
Tan(Val)	Tangents
ASin(Val)	Arkus sinus
ACos(Val)	Arkus kosinus
ATan(Val)	Arkus tangents
SinH(Val)	Sinus
CosH(Val)	Cosinus
TanH(Val)	Tangents
ASinH(Val)	Arkus sinus
ACosH(Val)	Arkus kosinus
ATanH(Val)	Arkus tangents hyperbolický
ProgrG(GNo)	Programování G funkce
ProgrM(MNo)	Programování M funkce
SetGAx(XNo, YNo, ZNo)	Nastavení geometrických os (zadají se čísla interních os, např. (0,1,2) pro první tři interní osy)

Funkce pro práci s programovou transformací (absolutní - ruší předchozí programovanou transformaci)

Translate(X, Y, Z)	Posunutí
Rotate(X, Y, Z, Angle)	Rotace okolo zadaného vektoru (jednotky úhlu viz AngleUnit)
RotateX(Angle)	Rotace okolo osy X (jednotky úhlu viz AngleUnit)
RotateY(Angle)	Rotace okolo osy Y (jednotky úhlu viz AngleUnit)
RotateZ(Angle)	Rotace okolo osy Z (jednotky úhlu viz AngleUnit)
Scale(Scale)	Změna měřítka (stejné měřítko pro všechny osy
ScaleA(X, Y, Z)	Změna měřítka (různé měřítko pro jednotlivé osy)
MirrorX	Zrcadlení osy X (změna znaménka v X, projeví se v Pos1)
MirrorY	Zrcadlení osy Y (změna znaménka v Y, projeví se v Pos1)
MirrorZ	Zrcadlení osy Z (změna znaménka v Z, projeví se v Pos1)
Mirror(X, Y, Z)	Zrcadlení ve směru zadaného vektoru

Funkce pro práci s programovou transformací (aditivní)

ATranslate(X, Y, Z)	Posunutí
ARotate(X, Y, Z, Angle)	Rotace okolo zadaného vektoru (jednotky úhlu viz AngleUnit)
ARotateX(Angle)	Rotace okolo osy X (jednotky úhlu viz AngleUnit)
ARotateY(Angle)	Rotace okolo osy Y (jednotky úhlu viz AngleUnit)
ARotateZ(Angle)	Rotace okolo osy Z (jednotky úhlu viz AngleUnit)
AScale(Scale)	Změna měřítka (stejné měřítko pro všechny osy)
AScaleA(X, Y, Z)	Změna měřítka (různé měřítko pro jednotlivé osy)
AMirrorX	Zrcadlení ve směru osy X (změna směru v X)
AMirrorY	Zrcadlení ve směru osy Y (změna směru v Y)
AMirrorZ	Zrcadlení ve směru osy Z (změna směru v Z)
AMirror(X, Y, Z)	Zrcadlení ve směru zadaného vektoru

Funkce pro práci s transformací polotovaru (absolutní - ruší předchozí programovanou transformaci)

WTranslate(X, Y, Z)	Posunutí
WRotate(X, Y, Z, Angle)	Rotace okolo zadaného vektoru (jednotky úhlu viz AngleUnit)
WRotateX(Angle)	Rotace okolo osy X (jednotky úhlu viz AngleUnit)
WRotateY(Angle)	Rotace okolo osy Y (jednotky úhlu viz AngleUnit)
WRotateZ(Angle)	Rotace okolo osy Z (jednotky úhlu viz AngleUnit)
WScale(Scale)	Změna měřítka (stejné měřítko pro všechny osy
WScaleA(X, Y, Z)	Změna měřítka (různé měřítko pro jednotlivé osy)
WMirrorX	Zrcadlení ve směru osy X (změna směru v X)
WMirrorY	Zrcadlení ve směru osy Y (změna směru v Y)
WMirrorZ	Zrcadlení ve směru osy Z (změna směru v Z)
WMirror(X, Y, Z)	Zrcadlení ve směru zadaného vektoru

Funkce pro práci s transformací polotovaru (aditivní)

WATranslate(X, Y, Z)	Posunutí
WARotate(X, Y, Z, Angle)	Rotace okolo zadaného vektoru (jednotky úhlu viz AngleUnit)
WARotateX(Angle)	Rotace okolo osy X (jednotky úhlu viz AngleUnit)
WARotateY(Angle)	Rotace okolo osy Y (jednotky úhlu viz AngleUnit)
WARotateZ(Angle)	Rotace okolo osy Z (jednotky úhlu viz AngleUnit)
WAScale(Scale)	Změna měřítka (stejné měřítko pro všechny osy)
WAScaleA(X, Y, Z)	Změna měřítka (různé měřítko pro jednotlivé osy)
WAMirrorX	Zrcadlení ve směru osy X (změna směru v X)
WAMirrorY	Zrcadlení ve směru osy Y (změna směru v Y)
WAMirrorZ	Zrcadlení ve směru osy Z (změna směru v Z)
WAMirror(X, Y, Z)	Zrcadlení ve směru zadaného vektoru

Funkce pro nastavování parametrů pro dynamické řízení rychlosti

SelDynControlSet(SetNo)	Volba sady parametrů pro dynamické řízení rychlosti

Funkce pro oznamování chyb, upozornění a informací

MsgShow(ShowNo, MsgNo,...)	Zobrazení informačního hlášení
MsgHide(ShowNo, ...)	Skrytí informačního hlášení
Err(MsgNo, ...)	Výpis chybového hlášení do okna výstupu, další průběh stejný jako po chybě
Wrn1(MsgNo, ...)	Výpis důležitého upozornění do okna výstupu
Wrn2(MsgNo, ...)	Výpis upozornění do okna výstupu
Wrn3(MsgNo, ...)	Výpis málo důležitého upozornění do okna výstupu
Info(MsgNo, ...)	Výpis informace do okna výstupu

Funkce pro práci se soubory

OpenFile(FileName)	Otevření souboru. Vrací číslo souboru (Int).
CloseFile(FileNo)	Uzavření souboru.
WriteFile(FileNo, Text, ...)	Zápis do souboru.

Funkce pro práci s tabulkou korekcí

CompTLinesCount	Čtení počtu řádků tabulky korekcí.
CompTSetLinesCount	Nastavení počtu řádků tabulky korekcí.
CompTGetToolRadius(Line)	Čtení poloměrové korekce z daného řádku tabulky korekcí.
CompTGetToolLenX(Line)	Čtení délkové korekce ve směru osy X z daného řádku tabulky korekcí.
CompTGetToolLenY(Line)	Čtení délkové korekce ve směru osy Y z daného řádku tabulky korekcí.
CompTGetToolLenZ(Line)	Čtení délkové korekce ve směru osy Z z daného řádku tabulky korekcí.
CompTSetToolRadius(Line, Val)	Nastavení poloměrové korekce pro daný řádek tabulky korekcí.
CompTSetToolLenX(Line, Val)	Nastavení délkové korekce ve směru osy X pro daný řádek tabulky korekcí.
CompTSetToolLenY(Line, Val)	Nastavení délkové korekce ve směru osy Y pro daný řádek tabulky korekcí.
CompTSetToolLenZ(Line, Val)	Nastavení délkové korekce ve směru osy Z pro daný řádek tabulky korekcí.

Funkce pro práci s tabulkou posunutí

OffsTLinesCount	Čtení počtu řádků tabulky posunutí.
OffsTSetLinesCount	Nastavení počtu řádků tabulky posunutí.
OffsTGetOffsX(Line)	Čtení posunutí ve směru osy X z daného řádku tabulky posunutí.
OffsTGetOffsY(Line)	Čtení posunutí ve směru osy Y z daného řádku tabulky posunutí.
OffsTGetOffsZ(Line)	Čtení posunutí ve směru osy Z z daného řádku tabulky posunutí.
OffsTSetOffsX(Line, Val)	Nastavení posunutí ve směru osy X pro daný řádek tabulky posunutí.
OffsTSetOffsY(Line, Val)	Nastavení posunutí ve směru osy Y pro daný řádek tabulky posunutí.
OffsTSetOffsZ(Line, Val)	Nastavení posunutí ve směru osy Z pro daný řádek tabulky posunutí.

Funkce pro práci s PLC konstantami

GetPlcConst(CnstID)	Zjišťování hodnoty zadané PLC konstanty

Funkce pro práci s PLC tabulkami

PlcTCount	Čtení počtu PLC tabulek
PlcTLinesCount(TableNo)	Čtení počtu řádků PLC tabulky
PlcTSetLinesCount(TableNo, LineCnt)	Nastavení počtu řádků PLC tabulky
PlcTColsCount(TableNo)	Čtení počtu sloupců PLC tabulky
PlcTGetSelLine(TableNo)	Čtení indexu zvoleného řádku PLC tabulky
PlcTSetSelLine(TableNo, LineNo)	Nastavení indexu zvoleného řádku PLC tabulky
PlcTColIndex(TableNo, ColID)	Zjišťování indexu sloupce PLC tabulky podle zadaného ID
PlcTColType(TableNo, ColNo)	Zjišťování typu dat pro zadaný sloupec PLC tabulky
PlcTGetInt(TableNo, LineNo, ColNo)	Čtení celočíselné hodnoty z PLC tabulky
PlcTGetReal(TableNo, LineNo, ColNo)	Čtení reálné hodnoty z PLC tabulky
PlcTSetInt(TableNo, LineNo, ColNo, NewVal)	Nastavení celočíselné hodnoty v PLC tabulce
PlcTSetReal(TableNo, LineNo, ColNo, NewVal)	Nastavení reálné hodnoty v PLC tabulce
PlcTSetString(TableNo, LineNo, ColNo, Str, ...)	Nastavení textové hodnoty v PLC tabulce

Funkce pro práci se sdílenou pamětí

PlcSACount	Zjišťování počtu oblastí sdílené paměti
PlcSASize(SANo)	Zjišťování velikosti zadadné oblasti sdílené paměti
PlcSAReadInt(SANo, Offset)	Čtení celočíselné hodnoty ze sdílené paměti bez čekání na odjetí předchozích bloků
PlcSAReadReal(SANo, Offset)	Čtení reálné hodnoty ze sdílené paměti bez čekání na odjetí předchozích bloků
PlcSAWriteInt(SANo, Offset, Val)	Zápis celočíselné hodnoty do sdílené paměti bez čekání na odjetí předchozích bloků
PlcSAWriteReal(SANo, Offset, Val)	Zápis reálné hodnoty do sdílené paměti bez čekání na odjetí předchozích bloků
PlcSAReadIntW(SANo, Offset, Val)	Čtení celočíselné hodnoty ze sdílené paměti s čekáním na odjetí předchozích bloků
PlcSAReadRealW(SANo, Offset, Val)	Čtení reálné hodnoty ze sdílené paměti s čekáním na odjetí předchozích bloků

PlcSAWriteIntW(SANo, Offset, Val)	Zápis celočíselné hodnoty do sdílené paměti s čekáním na odjetí předchozích bloků
PlcSAWriteRealW(SANo, Offset, Val)	Zápis reálné hodnoty do sdílené paměti s čekáním na odjetí předchozích bloků

Funkce pro práci s přímo sdílenými proměnnými

ReadPlcVarInt(VarName)	Čtení hodnoty přímo sdílené celočíselné proměnné PLC bez čekání na odjetí předchozích bloků.
ReadPlcVarReal(VarName)	Čtení hodnoty přímo sdílené reálné proměnné PLC bez čekání na odjetí předchozích bloků.
ReadPlcVarIntW(VarName)	Čtení hodnoty přímo sdílené celočíselné proměnné PLC s čekáním na odjetí předchozích bloků.
ReadPlcVarRealW(VarName)	Čtení hodnoty přímo sdílené reálné proměnné PLC s čekáním na odjetí předchozích bloků.
ReadRtmVarInt(VarName)	Čtení hodnoty přímo sdílené celočíselné proměnné RTM bez čekání na odjetí předchozích bloků.
ReadRtmVarReal(VarName)	Čtení hodnoty přímo sdílené reálné proměnné RTM bez čekání na odjetí předchozích bloků.
ReadRtmVarIntW(VarName)	Čtení hodnoty přímo sdílené celočíselné proměnné RTM s čekáním na odjetí předchozích bloků.
ReadRtmVarRealW(VarName)	Čtení hodnoty přímo sdílené reálné proměnné RTM s čekáním na odjetí předchozích bloků.

Funkce pro řízení záznamu, měřicí sonda

RecWait	Funkce pozastaví „záznam“ partprogramu do paměti (viz příklady)
RecNoWait	Funkce povolí „záznam“ partprogramu do paměti v případě, kde by byl za normální situace pozastaven
TPMeas	Zastaví pohyb a načte polohu v případě, že dojde k sepnutí měřicí sondy v bloku, kde je programovaná tato funkce (viz příklady)

Funkce pro vyvolání dialogu z partprogramu

AskUserRealNo(DefVal,MinVal,MaxVal,'Text')	Funkce zobrazí dialogové okno pro zadání reálného čísla. Nabídne číslo, zadané v parametru DefVal, které je možné přepsat hodnotou v rozsahu MinVal až MaxVal. Při zadání čísla mimo rozsah se hlásí chyba. Funkce vrátí zadanou hodnotu. Obvykle se používá současně s funkcí RecWait. V dialogovém okně se vypíše parametr 'Text', např. „Zadejte velikost korekce“. (viz příklady).
AskUserIntNo(DefVal,MinVal,MaxVal,'Text')	Funkce zobrazí dialogové okno pro zadání reálného čísla. Nabídne číslo, zadané v parametru DefVal, které je možné přepsat hodnotou v rozsahu MinVal až MaxVal. Při zadání čísla mimo rozsah se hlásí chyba. Funkce vrátí zadanou hodnotu. Obvykle se používá současně s funkcí RecWait. V dialogovém okně se vypíše parametr 'Text', např. „Zadejte číslo nástroje“. (viz příklady).
AskUserYesNo('Text')	Funkce zobrazí dialogové okno s nápisem „Text“ a tlačítky ANO a NE. Po stisku ANO vrací funkce hodnotu 1 (TRUE), po stisku NE vrací funkce hodnotu 0 (FALSE). (viz příklady)

3.2 Příklady použití funkcí v partprogramu

3.2.1 Matematické a logické funkce

Matematických a logických funkcí je větší počet, princip používání uvedeme na několika příkladech, které použijeme např. v podmínkových funkcích (If, ElseIf, Else, Endif). Pro zápis matematických funkcí se používá tzv. prefixová (také nazývaná polská) notace, t.j. zápis operátorů předchází operandy.

Funkce:	**Plus(Val1, Val2)** **Minus(Val1, Val2)** **Multiply(Val1, Val2)** **Divide(Val1, Val2)** **UMinus(Val1)**
Parametry:	**Val1** – první operand **Val2** – druhý operand

Funkce pro základní matematické operace sčítání, odčítání, násobení a dělení sečte (odečte, vynásobí, vydělí) dva operandy, výsledek je návratová hodnota funkce, t.j. musí se uložit do nějaké proměnné, jinak se hlásí chyba. K těmto základním matematickým funkcím patří ještě Unární minus (funkce má jen jeden parametr), která mění znaménko operandu na opačné.

Příklad:
Provedeme tyto operace se dvěma reálnými čísly, které jsou uložené v proměnných (t.j. v parametrech) CISLO1 a CISLO2, výsledek uložíme do proměnné VYSLEDEK. Při deklaraci proměnných použijeme automatické přidělení čísla reálného parametru pomocí RPARAM (podrobněji viz kapitola Programování parametrů) :

```
$CISLO1     RPARAM
$CISLO2     RPARAM
$VYSLEDEK   RPARAM
....
N  CISLO1 = 14.56
   CISLO2 = 28.30
N  VYSLEDEK = Plus(CISLO1,CISLO2)
N  VYSLEDEK = Minus(CISLO1,CISLO2)
N  VYSLEDEK = Multiply(CISLO1,CISLO2)
N  VYSLEDEK = Divide(CISLO1,CISLO2)
N  VYSLEDEK = UMinus(CISLO1)


....
```

V proměnné VYSLEDEK se postupně objeví hodnoty: 42.86, -13.74, 421.048, 0.514488 a -14.56

Pozn:
Tyto základní matematické operace se mohou v partprogramu psát jednoduše i klasickým zápisem (v tzv. infixové notaci), t.j. :

```
N  CISLO1 = 14.56
   CISLO2 = 28.30
N  VYSLEDEK = CISLO1 + CISLO2
N  VYSLEDEK = CISLO1 - CISLO2
N  VYSLEDEK = CISLO1 * CISLO2
N  VYSLEDEK = CISLO1 / CISLO2
N  VYSLEDEK = -CISLO1
```

Funkce:	**If(Cond)** **ElseIf(Cond)**
Parametry:	**Cond –** podmínka
Funkce:	**Else** Pokud není podmínka splněna, pokračuje se příkazy Else
Funkce:	**EndIf** Konec podmínky

Podmínka je splněna, je-li Val nenulová hodnota. V tom případě se pokračuje příkazy, uvedenými za. uzavírací závorkou podmínky Val. Podmínka Val může být složená z dalších funkcí. Funkce If se často využívá se pro větvení programu nebo skoky. Není-li podmínka splněná, pokračuje se příkazy za funkcí Else, která ovšem nemusí být uvedena – pak se pokračuje za funkcí Endif.

Příklad:
Provedeme vyhodnocení jednoduchého matematického zápisu, ve kterém pracujeme s reálnými parametry R1 a R2, které budou nastaveny na hodnoty 10.2 a 20.3. Pokud bude podmínka splněna, což v našem zadání ano, provede se volání podprogramu číslo 1. Nebyla-li by podmínka splněna, provedlo by se volání podprogramu číslo 2. V příkladu jsou použity funkce UMinus( ), Plus( ), CALL( )
-(R1 + R2) = - 30.5

```
N PROGRAM
N10  R1=10.2  R2=20.3
N20  If(EQ(UMinus(Plus(R1,R2)),-30.5)) CALL(1) Else CALL(2) Endif
N30 …
N40
N ENDPROGRAM

N BEGIN(1)  " podprogram číslo 1
N …
N END
N BEGIN(2)  " podprogram číslo 2
N …
N END
```

Příklad:
Provedeme vyhodnocení zápisu, ve kterém se porovnávají hodnoty pomocí funkcí **GE( ), LE ( ), AND( ) a NOT( ),** ve kterém pracujeme s celočíselnými parametry I1, I2, I3, I4 které budou nastaveny na hodnoty 10, 20, 30, 40. Podle výsledků se nastaví parametr R5 na hodnotu 100.0 nebo 200.0 a následujícím bloku odjede osa X míru, nastavenou v tomto parametru (v našem zadání se nastaví R5=100.0)
Matematicky by podmínka zapsala:
If ((NOT (R1 >= R2)) AND (R3 <= R4)) ….

```
N PROGRAM
N10  I1=10 I2=20 I3=30 I4=40 R5=0
N20  If(AND(NOT(GE(R1,R2)),LE(R3,R4))) R5=100.0 Else R5=200.0 Endif
N30  XR5 G01
N40
N ENDPROGRAM
```

Funkce:	**And(Val1,Val2)** **BAnd(Val1,Val2)**
Parametry:	**Val1** – první operand **Val2** – druhý operand

Na těchto funkcích si ukážeme rozdíl mezi logickým AND a bitovým AND. Operandy musí být celočíselné (integer), proto je deklarujeme jako IPARAM.
Příklad: provedeme logický AND a bitový AND mezi čísly 5 a 4.
Logický AND pracuje s celým „slovem“, např. 16-ti nebo 32-bitovým a rozlišuje pouze, zda je hodnota (celého slova) nulová nebo nenulová. Pokud jsou oba operandy nenulové, výsledek je 1, ve všech ostatních případech je výsledek 0

```
„Nenulové číslo“  AND   „Nenulové číslo“  = 1
„Nula“            AND   „Nenulové číslo“  = 0
„Nenulové číslo“  AND   „Nula“            = 0
„Nula“            AND   „Nula“            = 0

$CISLO1     IPARAM
$CISLO2     IPARAM
$VYSLEDEK   IPARAM
....
N  CISLO1 = 5
   CISLO2 = 4
N  VYSLEDEK = And(CISLO1,CISLO2)
```

V proměnné výsledek bude 1
Bitový AND pracuje s jednotlivými bity, proto je názornější si čísla 5 a 4 převést do binární formy (např. jako osmibitové slovo):

```
5 = 00000101
4 = 00000100
```

Bitový AND pak provádí logický součin mezi jednotlivými bity stejné váhy a výsledek je 1 pouze na „pozici“, kde jsou oba bity v jedničce. V našem případě jsou obě jedničky pouze na druhém bitu (počítá se od nuly zprava)

```
5 = 00000101
4 = 00000100
```

Výsledek bude:

```
4 = 00000100
```

Podobně pracují podle pravidel logických operací i funkce Or, Xor, BOr,BXor,Not a BNot

Funkce:	**ShL(Val, BitCnt)** **ShR(Val, BitCnt)**
Parametry:	**Val** – operand **BitCnt** – o kolik bitů posunout (doprava nebo doleva)

Funkce pro posun bitů doprava nebo doleva. Funkci lze provést pouze s celočíselnou proměnnou. Pro praktické využití je nutná znalost psaní binárních čísel, je totiž vhodné si číslo napsat v binárním tvaru. Možné využití demonstruje následující, spíše jen školní, příklad. Prakticky lze tento příklad řešit i několika jinými způsoby. Je třeba obrobit řadu 40 otvorů, se stejnou roztečí. Otvor se obrobí v podprogramu číslo 1. V každém čtvrtém otvoru je ale třeba provést nějakou doplňující operaci, která se provede podprogramem číslo 2. Kdy volat podprogram číslo 2 vyhodnotíme pomocí funkce posunu doprava. V příkladu jsou pro názornost informační výpisy a podprogramy vyjíždí v ose Y, takže na grafice je názorně vidět, že každý čtvrtý průchod provede operaci navíc (vyjede v ose Y výš).

```
" Program pro demonstraci binárního posunu doprava
$Cislo      IPARAM
$Citac      IPARAM
$StavBitu0  IPARAM
$Bit0       1

N PROGRAM
N10 Cislo  = 8          " 8 je binárně 1000, 1 se bude posouvat doprava
    Citac  = 10         " Citac do 10 (4x10=40 otvorů)
N20 AxGX = 0 G0 G90
"
" Cyklus
```

```
"
N30 Call(1)                    " Obrobí otvor
N40 StavBitu0 = BAnd(Cislo,Bit0)  " Binárním AND získá stav na nultém bitu
    MsgShow(2,'Stav bitu 0 = \i',StavBitu0)
N50 if(StavBitu0)              " Je-li stav nultého bitu 1, zavolá podprog. 2
      Cislo = 8                " Inicializace 8 = 1000 binárně
      Citac = Citac - 1        " Dekrementace čítače (po 4 otvorech)
      Call(2)                  " Volání podprogramu č. 2
    else
      Cislo = Shr(Cislo,1)     " Posun doprava bin.: 1000, 0100, 0010, 0001
    endif
N60 MsgShow(3,'Cislo = \i',Cislo)   " Zobrazí posun 8, 4, 2, 1
N70 AxGX = AxGX + 50           " Posun o 50mm v ose X
N80 if(EQ(Citac,0))            " Všechny otvory obrobeny ?
      Jmp(100)                 " ANO - skok na blok N100
    else
      Jmp(30)                  " NE - skok na blok N30
    endif
N100
N ENDPROGRAM
"
" Podprogramy
"
N Begin(1)
N Y50 G0
  MsgShow(1,'Základní operace')
N Y0
N MsgHide(1)
N End
"
N Begin(2)
N Y100 G0
  MsgShow(1,'Přídavná operace')
N Y0
N MsgHide(1)
N End
```

Funkce:	**Int(Val)** **Real(Val)** **Round(Val)** **Trunc(Val)** **Tenths(Val)** **Sgn(Val)** **Even(Val)** **Odd(Val)** **Pow(Val)** **Cond(Cond1, Trueval,FalseVal)**
Parametry:	**Val** – operand **Cond1 –** podmínka **Trueval –** vrácená hodnota, je-li Cond1 TRUE **FalseVal –** vrácená hodnota, je-li Cond1 FALSE

Funkce pro převod integer čísel na reálná a naopak, zaokrouhlování, získání celé a desetinné části reálného čísla, test sudého a lichého čísla, test znaménka, podmíněný výraz. Použití je patrné z následujícího příkladu partprogramu, který uvedené funkce používá. Výsledky se vypisují.

```
$RCISLO    RPARAM
$RVYSLEDEK RPARAM
$ICISLO    IPARAM
$IVYSLEDEK IPARAM
```

```
$Cond1     IPARAM
$TrueVal   IPARAM
$FalseVal  IPARAM
$TrueOrFalse IPARAM

N PROGRAM
N RCISLO = -13.752
N IVYSLEDEK = Int(RCISLO)                " prevod Real na Int
N MsgShow(1,'IVYSLEDEK =\i',IVYSLEDEK)   " IVYSLEDEK = -13
N RVYSLEDEK = Real(IVYSLEDEK)            " prevod Int na Real
N MsgShow(1,'RVYSLEDEK =\r',RVYSLEDEK)   " RVYSLEDEK = -13.000000
"
N RVYSLEDEK = Round(RCISLO)              " Zaokrouhlení reálného čísla
N MsgShow(1,'RVYSLEDEK =\r',RVYSLEDEK)   " RVYSLEDEK = -14.000000
"
N RVYSLEDEK = Trunc(RCISLO)              " Celá část reálného čísla
N MsgShow(1,'RVYSLEDEK =\r',RVYSLEDEK)   " RVYSLEDEK = -13.000000
"
N RVYSLEDEK = Tenths(RCISLO)             " Desetinná část reálného čísla
N MsgShow(1,'RVYSLEDEK =\r',RVYSLEDEK)   " RVYSLEDEK = -0.752000
"
N IVYSLEDEK = Sgn(RCISLO)                " Znaménko reálného čísla
N MsgShow(1,'IVYSLEDEK =\i',IVYSLEDEK)   " IVYSLEDEK = 1/0 = záporné/kladné
"
N ICISLO = 4
N if(Even(ICISLO))                       " Je cislo sudé?
    MsgShow(1,'Sudé číslo, ICISLO = \i', ICISLO)     " ANO - tento výpis
  else
    MsgShow(1,'Liché číslo, ICISLO = \i', ICISLO)    " NE
  endif
"
N ICISLO = 4
N if(Odd(ICISLO))                        " Je cislo liché?
    MsgShow(1,'Liché číslo, ICISLO = \i', ICISLO)    " ANO
  else
    MsgShow(1,'Sudé číslo, ICISLO = \i', ICISLO)     " NE - tento výpis
  endif
"
N RCISLO = 3
N RVYSLEDEK = Pow(RCISLO,RCISLO)        " 3 ^ 3
N MsgShow(1,'RVYSLEDEK =\r',RVYSLEDEK)  " RVYSLEDEK = 27.000000

N Cond1 = 0
N TrueVal  = 10
N FalseVal = 20
N TrueOrFalse = Cond(EQ(Cond1,1), TrueVal, FalseVal)
N MsgShow(1,'TrueOrFalse = \i',TrueOrFalse)
N ENDPROGRAM
```

Funkce:	**PreserveR(From,To)** **PreserveI(From,To)**
Parametry:	**From** – od jakého parametru (včetně) uschovat **To** – do kterého parametru (včetně) uschovat

Funkce pro úschovu reálných (PreserveR) nebo celočíselných (PreserveI) parametrů. Pokud se využívá parametrické programování, doporučuje se v každém podprogramu, případně makrocyklu nebo pevném cyklu použít výše uvedené funkce pro úschovu parametrů. Pokud bychom tyto funkce nepoužili a např. v podprogramu změnily parametry, které si nastavil hlavní program, může se změnit chování hlavního programu Používá se v případě, že se využívá parametrické programování. Následující příklad demostruje úschovu parametrů při volání podprogramů. V prvním podprogramu se parametry neuchovají, v druhém ano. Informační výpisy ukazují

změny parametrů. Obnova parametrů se děje v případě použití těchto funkcí automaticky, t.j. není žádná funkce pro obnovu.

```
" Program pro demonstraci uschovy parametrů

N PROGRAM
N10  R1=11 R2=22 R3=33 R4=44
N20  MsgShow(1,'R1=\r R2=\r R3=\r R4=\r',R1,R2,R3,R4)
N30  Call(1)
N40  MsgShow(1,'R1=\r R2=\r R3=\r R4=\r',R1,R2,R3,R4)
N50  R1=11 R2=22 R3=33 R4=44
N60  MsgShow(1,'R1=\r R2=\r R3=\r R4=\r',R1,R2,R3,R4)
N70  Call(2)
N80  MsgShow(1,'R1=\r R2=\r R3=\r R4=\r',R1,R2,R3,R4)
N90  " Parametry R1 – R3 byly obnoveny, parametr R4 zůstal změněný
N100 ENDPROGRAM
"
" Podprogramy
"
N Begin(1)  " Změní parametry bez předešlé úschovy
N1000  R1=55 R2=66 R3=77 R4=88
N End
"
N Begin(2)  " Změní parametry, ale nejprve je uschová
N2000  PreserveR(1,3)             " Uschová reálné parametry 1 až 3
N2100  R1=55 R2=66 R3=77 R4=88    " Změní parametry 1 až 4
N2200  MsgShow(1,'R1=\r R2=\r R3=\r R4=\r',R1,R2,R3,R4)
N End
```

3.2.2 Programování G a M funkcí

Funkce:	**ProgrG(GNo)**
Parametry:	**GNo –** číslo G-funkce
Funkce:	**ProgrM(MNo)**
Parametry:	**MNo –** číslo M-funkce

Obě funkce jsou ekvivalentní přímému zápisu podle ISO, tj. například ProgrM(41) je totožné, jako zápis M41. Zápis pomocí funkcí se používá např. v podmínkových blocích a v partprogramech resp. různých makrocyklech, které se řeší obecně. Často se využívá systémových parametrů. Jako příklad uvedeme volbu otáček při reverzaci vřetene.
Např. v proměnné SmerOt bude nastavené, zda chceme závitovat závitovaní hlavičkou pravý nebo levý závit. V bloku pak bude programováno:

```
N10  If(SmerOt)  ProgrM(3) Else ProgrM(4) EndIf
```

Nelze napsat místo ProgrM(3) přímo M3 a místo ProgrM(4) přímo M4, neboť by byly programovány dvě funkce jedné skupiny v jednom bloku, což je zakázané.

Rozdíl v použití klasického zápisu M funkce (případně G funkce) podle ISO a zápisu pomocí volání funkce ProgrM( ), resp ProgrG( ) ilustruje i následující příklad části partprogramu, ve kterém podle nastavení proměnné „Priznak“ voláme makrocyklus např. pro nějaké měření. Je-li Priznak = 1, zavolá se makrocyklus Mereni a po návratu chceme vykonávání partprogramu pozastavit programovým stopem M00. Je-li Priznak = 0, makrocyklus se nezavolá a proto nechceme ani zastavit vykonávání partprogramu.
Volání makrocyklu i programový stop je v podmínkovém bloku „ if “ . Správný zápis je uveden v příkladu. Pokud bychom totiž v bloku „ if “ uvedli místo ProgrM(0) pouze klasický zápis M0, docházelo by v bloku N200 k zastavení **vždy**, t.j. bez ohledu na to zda se volá makrocyklus Mereni v závislosti na nastavení proměnné „Priznak“.

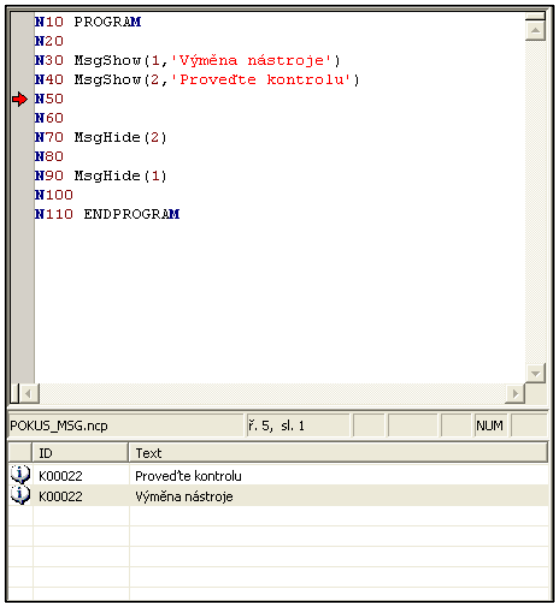
```
$Priznak IPARAM
....
N ....
N100  Priznak = 1
N ....
N ....
N200  if(EQ(Priznak,1))       " Je-li Priznak = 1 ...
        CallMacro(Mereni)     " ... tak volej makrocyklus Mereni ...
        ProgrM(0)             " ... a po návratu programový stop
      endif
N ....
....
```

3.2.3 Funkce pro oznamování chyb, upozornění a informací

Funkce:	**MsgShow(ShowNo, MsgNo)**
Parametry:	**ShowNo –** Pořadí zprávy, má význam při nutnosti zobrazovat více zpráv najednou. **MsgNo** - text text zprávy mezi ‘apostrofy‘ nebo identifikační číslo zprávy, uvedené v

v deklaraci zprávy (deklarace může být kdekoli v partprogramu před jejím vyvoláním nebo libovolném souboru, vloženém do partprogramu pomocí klíčového slova INL (include) .

Funkce:	**MsgHide(ShowNo)**
Parametry:	**ShowNo –** pořadí zprávy, která se má smazat. Pokud se uvede 0, smažou se všechny zprávy.



Uvedené funkce slouží pro výpis zpráv, které technolog zapíše do partprogramu.. Zprávy se vypisují při chodu partprogramu v okně „Trvalé chyby a hlášení“.

Po vykonání bloku N40 jsou zobrazeny obě zprávy (viz obr.). V bloku N30 i N40 by mohla být místo textů v apostrofech uvedena čísla zpráv. Obě variant jsou možné. Zprávy se automaticky řadí tak, že naposledy vyslaná zpráva je na prvním řádku.

Po vykonání bloku N70 zmizí zpráva, vyslaná jako druhá v pořadí, po vykonání bloku N90 zmizí zpráva vyslaná jako první v pořadí. Pokud se kdekoli uvede MsgHide(0), zmizí všechny zprávy.

Pokud by se texty deklarovaly např. na začátku partprogramu nebo v libovolném souboru, který by se připojil pomocí klíčového slova INL, byly by texty deklarovány např. takto:

```
&1  ’…’
&2  ’…’
&3  ’ Výměna nástroje’
&4  ’ Proveďte kontrolu’
&5  ’…’
```

a v partprogramu by bylo uvedeno:

```
N30  MsgShow(1,3)
N40  MsgShow(2,4)
```

Výpis proměnných v textu zprávy

Kromě statického výpisu zpráv, jak bylo uvedeno v předešlém příkladu, je možné vypisovat v textu i dynamicky měnící se proměnné, což se může s výhodou používat např. pro různé čítače, počítadla kusů apod. Výpis proměnných umožňují tzv. escape sekvence, které umožňují vkládání specielních znaků a parametrů do řetězce. Tyto sekvence začínají zpětným lomítkem:

\r Místo této sekvence se vloží reálná hodnota, která je uvedena jako parametr funkce MsgShow
\i Místo této sekvence se vloží celočíselná hodnota, která je uvedena jako parametr funkce MsgShow

Kromě proměnných se mohou vkládat i další specielní znaky:
\n Místo této sekvence se vloží nový řádek
\nnn Místo této sekvence se vloží znak s ASCII hodnotou nnn (zapsané dekadicky a povinně tři cifry)
\xhh Místo této sekvence se vloží znak s ASCII hodnotou zapsanou hexadecimálně (vždy dvě cifry)
\' Místo této sekvence se vloží apostrof (apostrof nelze psát do řetězce přímo!)
\\ Místo této sekvence se vloží zpětné lomítko. To nelze psát přímo, musí se zdvojit, máli být

Následující příklad uvádí možnost výpisu zprávy, kde chceme zobrazit úhel ß a odchylku v mikronech. Úhel bude např. vypočítán a uložen do reálné proměnné UHEL, podobně odchylka bude tentokrát v celočíselné proměnné ODCHYLKA.

Příklad:

```
$UHEL                   R100  " Přiřazení proměnné reálnému parametru
$ODCHYLKA               I100  " Přiřazení proměnné celočíselnému parametru

N PROGRAM
N UHEL = 45.5     ODCHYLKA = 10
N MsgShow(1,'úhel \223 = \r, povolená odchylka = \i \181m',UHEL,ODCHYLKA)
N ...
N MsgHide(0)
N ENDPROGRAM
```

V okně pro výpis zpráv a chybových hlášení bude vypsáno:
`'úhel ß = 45.50000, povolená odchylka = 10µm`

Escape sekvence \223 byla nahrazena znakem ß
Escape sekvence \r byla nahrazena hodnotou reálné proměnné UHEL
Escape sekvence \i byla nahrazena hodnotou celočíselné proměnné ODCHYLKA
Escape sekvence \181 byla nahrazena znakem µ

Pozn.: kódy znaků lze získat v libovolné ASCII tabulce.

Příklad výpisu čítače cyklů. Čítač „Cit“ se nastaví na začátku programu na hodnotu 10. V cyklu se dekrementuje. Vypisuje se zpráva, kolik cyklů ještě zbývá. Na konci programu v bloku N50 je použit rozhodovací blok, zda se čítač nerovná nule. Pokud ne, jde se znovu do cyklu. pokud ano, programse ukončí.

```
$Cit  I1
N PROGRAM
N10 Cit = 10 X0 G0 G90
N20 MsgShow(1,'Zbývá cyklů: \i',Cit)
N30 AxGx = AxGx + 10  G01 F1000
N40 Cit=Cit-1
N50 if(NEq(Cit,0)) Jmp(20) endif  "Není-li čítač nula, skok na N20
N99 ENDPROGRAM
```

Funkce:	**Err(MsgNo)**
Parametry:	**MsgNo –** číslo chybového hlášení.

Funkce je podobná předešlé funkci pro výpis zpráv. Rozdíl je pouze v zobrazení. Chyba, jejíž text je deklarován stejným způsobem jako zprávy, se zobrazí v chybovém okně uprostřed obrazovky (viz obr.) po **přípravě bloku** , což je ve většině případů (není to ale podmínka) už po volbě partprogramu. Chyba se musí kvitovat tlačítkem OK. Tímto způsobem může technolog pomocí načtení systémových parametrů např. zkontrolovat otáčky a eventuelně vyhlásit chybu nebo zkontrolovat parametry kružnice např. na rozsah apod. V partprogramu je zápis možný dvěma způsoby. Buď uvést jako parametr funkce Err číslo chyby (její deklarace je kdekoli v partprogramu, obvykle na začátku uvedena jako &1 a text v apostrofech) a nebo jako parametr uvést přímo text mezi apostofy.

Příklad:

```
&1 'Chybně zadaný parametr kružnice'
....
N320  if(Greater(Prumer,MaxPrum)
            Err(1)                                    " možno zapsat takto
            Err('Chybně zadaný parametr kružnice 1')  " nebo takto
      endif
....
```

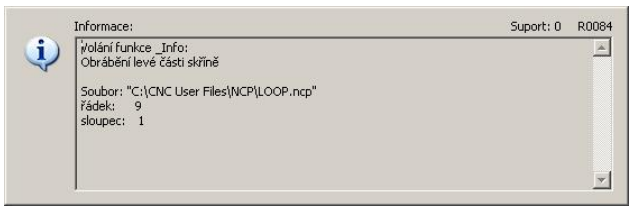
Funkce:	**Wrn1(MsgNo)** **Wrn2(MsgNo)** **Wrn3(MsgNo)**
Parametry:	**MsgNo –** číslo zprávy upozornění.

Tyto tři funkce jsou podobné předešlé funkci Err pro výpis chyby. Zápis je stejný. Používá se na upozornění obsluhy a lze je programovat ve třech stupních důležitosti.

Wrn1 je Důležité upozornění
Wrn2 je Upozornění
Wrn3 je Málo důležité upozornění.

Ve všech případech se zobrazuje okno (viz obr.), liší se pouze nápisem nad rámečkem, kde je buď text „Důležité upozornění“ nebo „Upozornění“ nebo „Málo důležité upozornění“. Upozornění se musí kvitovat tlačítkem OK. Tímto způsobem může technolog provádět libovolné kontroly a případně hlásit tato upozornění.

Funkce:	**Info(MsgNo)**
Parametry:	**MsgNo –** číslo informačního hlášení.



Funkce podobná jako funkce Err – používá se stejný způsob zápisu, vypisuje se jako informační hlášení. Informační hlášení se musí kvitovat tlačítkem OK

3.2.4 Programování geometrických a synchronních os

Řídicí systém pracuje, tj. provádí veškeré výpočty týkající se prostorové rychlosti, v geometrickém prostoru X,Y,Z. Ve skutečnosti (prakticky se to týká strojů, které mají více než tři osy) mohou být oním tříosým geometrickým prostorem libovolné osy, které jsou na konkrétním stroji. Typický příklad jsou např. horizontky, kde osa vřetene je označena jako Z a pinola, která se pohybuje stejným směrem je označena jako W.
Systém má možnost přepínat libovolné osy do geometrického prostoru funkcí SetGAx(XNo, YNo, ZNo)

Geometrické osy je možné podle potřeby přepínat v partprogramu. Pokud je potřeba u stroje přepínat geometrické osy, obvykle je v prioritním bloku nastavena nejčastější kombinace, např. pro první tři interní osy se naprogramuje v prioritním bloku:

```
N SetGAx(0,1,2)
```

Přepnutí v partprogramu pro jiné osy se provede podobně.
Např. u horizontek se v některých případech vrtá osou Z, v některých případech pinolou, t.j. osou W (která je kupříkladu nakonfigurovaná jako 4 interní osa).

Přepnutí se provede naprogramováním

```
N SetGAx(0,1,3)   „ vrtá se pinolou W
```
nebo
```
N SetGAx(0,1,2)   „ vrtá se osou Z
```

Pokud jsou programovány více než tři osy, provede se výpočet prostorové rychlosti pro nastavené geometrické osy. Ostatní osy, programované současně s geometrickými osami, jedou tak, aby dojeli ve stejný okamžik jako geometrické osy. Jsou to tzv. **synchronní osy**. Pokud je synchronní (tj. negeometrická osa) osa programována v bloku samostatně, jede rychlostí, která se programuje systémovým parametrem. Pokud by nebyla rychlost systémovým parametrem zadána, jede rychlostí, která je přednastavena v souborech typu *.ChannelConfig

Pozn.: mohou existovat ještě tzv. asynchronní osy. Jedná se o různé pomocné osy, které má plně režii PLC program a obvykle jsou programovány např. M-funkcemi. Návod na jejich ovládání musí být součástí návodu ke konkrétnímu stroji.

Příklad:
Přepínání geometrických os si ukážeme na následujícím příkladu. Prakticky se tento případ doporučuje spíše udělat způsobem, který je popsán v poznámce za příkladem. Pro ilustraci to ale postačí. Předpokládejme stroj, který má 4 osy X,Y,Z, a čtvrté ose přiřadíme adresu W. Osy Z a W budeme přepínat na geometrické podle toho, kterou osou budeme vrtat, zda vřetene (osa Z) nebo pinolou (osa W). Budeme programovat současný pohyb tří os, který by se využil např. při nájezdu do určité polohy.

```
N PROGRAM
N10 SetGAX(0,1,2)       " Zvolí geometrické osy X, Y, Z
N20 X0 Y0 Z0 G01 F1000  " Výchozí poloha 0,0,0
N30 X100 Y100 Z100      " Najede do polohy 100,100,100
N40 X0 Y0 Z0            " Vrátí se do polohy 0,0,0
"
N50 SetGAX(0,1,3)       " Přepne na geometrické osy X, Y, W
N60 X0 Y0 W0            " Výchozí poloha 0,0,0
N70 X10 Y10 W10         " Najede do polohy 100,100,100
N80 X0 Y0 W0            " Vrátí se do polohy 0,0,0
N90 SetGAX(0,1,2)       " Přepne zpět na geometrické osy X, Y, Z
N ENDPROGRAM
```

V bloku N10 se zvolí geometrické osy X,Y,Z ,kterými se najede v dalších blocích do bodu 100,100,100 a pak zpět do nuly. V bloku N50 přepneme geometrické osy na X,Y,W, resp. 0,1,3 (čísla os se počítají od nuly).

V blocích N60, N70 a N80 se může programovat buď W, jak je v příkladu výše, ale stejný účinek by nastal, kdybychom programovali Z:

```
N60 X0 Y0 Z0            " Výchozí poloha 0,0,0
N70 X10 Y10 Z10         " Najede do polohy 100,100,100
N80 X0 Y0 Z0            " Vrátí se do polohy 0,0,0
```

V obou případech by se pohybovala pouze 4. osa. Jestliže je konfigurací dáno, že geometrické osy mají přiřazeny adresy X,Y,Z pak při přepnutí na jinou interní osu (v našem příkladu na 3) jede fyzicky tato osa i když je programovaná, lépe řečeno napsaná v programu, osa Z.

Pozn.:
Obecně se dá říci, že přepínání geometrických os v tomto případě není nejvhodnější. Nastaví-li partprogram geometrické osy, zůstanou nastaveny trvale, dokud se nezmění (např. prioritním bloku může být defaultně X,Y,Z). třeba po centrální anulaci. V každém případě to s sebou nese nutnost o tomto stavu (tj. jaké osy jsou geometrické) informovat trvale obsluhu, což je v případě přepínáním pomocí SetGAX( ) obtížné. Pokud by se vyslal v režimu RUP blok s jízdou v ose Z, musí obsluha vědět, zda pojede Z nebo W.

U tohoto typu strojů se spíše doporučuje varianta nepřepínat geometrické osy a ponechat stále X,Y,Z. Pokud potřebujeme jet osou W, zařídí to PLC program pomocí M-funkce, kterou přepne pouze servosmyčku do požadované osy.

Toho lze využít pro zjednodušení programování. Programátor píše program pouze pro osy X,Y,Z.

3.2.5 Vlastnosti volání podprogramů – funkce SubOpt( )

Při volání podprogramu, makrocyklu či pevného cyklu můžeme pomocí funkce SubOpt(Option,Set) ovlivňovat některé stavy, v jakých se má program nacházet po návratu z uvedených programových celků. Typickou situací jsou např. stavy M-funkcí po návratu z pevného cyklu. Obvykle je požadováno, aby M-funkce byly po návratu z pevného (např. vrtacího) cyklu ve stavu, v jakém byly před voláním vrtacího cyklu. Naopak po návratu z podprogramu, který je součástí partprogramu, se tato vlastnost nepožaduje, t.j. M-funkce mají zůstat ve stavu, v jakém je nastaví podprogram.

Například pokud by nebyla u vrtacího cyklu nastavena vlastnost, že se mají M-funkce vrátit do stavu před voláním, tak po návratu ze závitovacího cyklu G84 (ve kterém je reverzace vřetena z M3 na M4) by se vřeteno točilo opačným směrem, t.j. M4 a museli bychom v partprogramu po návratu programovat M3. Pokud by se programovalo více děr se závitem za sebou, musela by v každém bloku být M3, což je nepraktické.

Hodnoty SubOpt jsou nastaveny defaultně tak, že po volání pevných cyklů a makrocyklů se všechny níže uvedené vlastnosti vrací na hodnoty před voláním pevných cyklů a makrocyklů, zatímco po volání podprogramů ne. Obvykle tento stav vyhovuje. Pokud je nutné ho měnit, použijí se níže uvedené konstanty při volání funkce SubOpt(Option,Set). Volat tuto funkci je povoleno pouze v pevném cyklu, makrocyklu nebo podprogramu, t.j. nesmí být volána z hlavního partprogramu (hlásila by se chyba)

Funkce:	**SubOpt (Option,Set)**
Parametry:	**Option** = Konstanta podle uvedeného seznamu (lze uvést i jen číslo – ale nedoporučuje se)
	Set = 0/1 = Nenastavit / Nastavit vlastnost podle parametru Option

Příklad:

```
SubOpt (SUBOPT_RESTOREM,1)
SubOpt (11,1)
```

Obě výše uvedená volání funkce SubOpt ( ) jsou totožná a možná a obě nastavují vlastnost obnovit po návratu z volání hodnoty M funkcí do stavu před voláním. Tento stav je defaultní pro pevné cykly a makrocykly. Pro podprogramy je defaultně:

```
SubOpt (SUBOPT_RESTOREM,0)
SubOpt (11,0)
```

Pokud defaultní stav vyhovuje, funkce se nemusí volat.

Příklad použití funkce SubOpt( ). Před voláním podprogramů se vřeteno točí M3. V obou podprogramech se nastaví funkce M4. Při návratu z podprogramu č.1 se vrátí M3, při návratu z č.2 zůstane navolená M4.

```
N PROGRAM
N
N M3              " Roztočí vřeteno M3
N CALL(1)         " Volá podprogram č.1
N                 " Trvá funkce M3
N CALL(2)         " Volá podprogram č.2
N                 " Zůstane M4
N M5              " Stop vřetene
N ENDPROGRAM
"
N BEGIN(1)
N SubOpt(SUBOPT_RESTOREM,1)   " Po návratu vrátí M funkce (zde vrátí M3)
N M4              " Roztočí vřeteno M4
N END
"
N BEGIN(2)
N SubOpt(SUBOPT_RESTOREM,0) " Po návratu nevrátí M funkce (t.j. zůstane M4)
N M4              " Roztočí vřeteno M4
N END
```

Následují seznam uvádí konstanty, kterými se nastavují vlastnosti. Za pojmenovanou konstantou je uvedena její číselná hodnota.

Odpověď ANO na níže uvedené otázky - Parametrem Set = 1 (nastavit vlastnost)
Odpověď NE na níže uvedené otázky - Parametrem Set = 0 (nenastavit vlastnost)

SUBOPT_RESTOREINTERPOLATION	0	Obnovit po návratu z volání typ interpolace do stavu před voláním?
SUBOPT_RESTORERADIUSCOMP	1	Obnovit po návratu z volání typ poloměrové korekce do stavu před voláním?
SUBOPT_RESTORERADIUSCOMPPLANE	2	Obnovit po návratu z volání korekční rovinu do stavu před voláním?
SUBOPT_RESTORECONTINUOUSMODE	3	Obnovit po návratu z volání způsob navazování bloků do stavu před voláním?
SUBOPT_RESTORESPEEDMODE	4	Obnovit po návratu z volání způsob zadávání rychlosti do stavu před voláním?
SUBOPT_RESTOREINCREMENTALMODE	5	Obnovit po návratu z volání způsob zadávání rozměrů do stavu před voláním?
SUBOPT_RESTOREDIAMETERPROGR	6	Obnovit po návratu z volání průměrové/poloměrové programování pro režim absolutního programování?
SUBOPT_RESTOREDIAMETERINCPROGR	7	Obnovit po návratu z volání průměrové/poloměrové programování pro režim přírůstkového programování?
SUBOPT_RESTORELENGTHUNIT	8	Obnovit po návratu z volání používané délkové jednotky?
SUBOPT_RESTOREFEEDUNIT	9	Obnovit po návratu z volání používané jednotky rychlosti?
SUBOPT_RESTOREANGLEUNIT	10	Obnovit po návratu z volání používané úhlové

		jednotky?
SUBOPT_RESTOREM	11	Obnovit po návratu z volání hodnoty M funkcí podle stavu před voláním?
SUBOPT_RESTOREFEED	12	Obnovit po návratu z volání programovanou rychlost podle stavu před voláním?
SUBOPT_RESTOREREVFEED	13	Obnovit po návratu z volání programovanou rychlost podle stavu před voláním?
SUBOPT_RESTORESPINDLESPEED	14	Obnovit po návratu z volání programované otáčky vřetene podle stavu před voláním?
SUBOPT_RESTOREFEEDOVR	15	Obnovit po návratu z volání programovaný override rychlosti podle stavu před voláním?
SUBOPT_RESTORESPINDLESPEEDOVR	16	Obnovit po návratu z volání programovaný override otáček vřetene podle stavu před voláním?
SUBOPT_RESTORESPINDLESPEEDLIMIT	17	Obnovit po návratu z volání programovaný limit otáček vřetene podle stavu před voláním?
SUBOPT_RESTORECONSTCUTTINGSPEED	18	Obnovit po návratu z volání programovanou konstantní řeznou rychlost podle stavu před voláním?
SUBOPT_RESTOREPTRANSFORM	19	Obnovit po návratu z volání parametry programové transformace do stavu před voláním?
SUBOPT_RESTOREWTRANSFORM	20	Obnovit po návratu z volání parametry transformace polotovaru do stavu před voláním?
SUBOPT_RESTORELENCOMP	21	Obnovit po návratu z volání parametry délkové korekce do stavu před voláním?
SUBOPT_RESTOREOFFSET1	22	Obnovit po návratu z volání programovaný vektor posunutí 0 do stavu před voláním?
SUBOPT_RESTOREOFFSET2	23	Obnovit po návratu z volání programovaný vektor posunutí 1 do stavu před voláním?
SUBOPT_RESTOREDYNAMICCONTROL	24	Obnovit po návratu z volání parametry pro dynamické řízení rychlosti do stavu před voláním?

3.2.6 Měřicí sonda

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=10   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

Poloha os v okamžiku sepnutí/uvolnění sondy při měření v prostoru pos2:
TPMEASAXXPOS2
TPMEASAXYPOS2
TPMEASAXZPOS2

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
TPMEASAXI3POS0
TPMEASAXI4POS0
TPMEASAXI5POS0
TPMEASAXI6POS0
TPMEASAXI7POS0
TPMEASAXI8POS0
atd. pro další prostory.

3.2.7 Funkce RecWait a RecNoWait

Funkce RecWait a RecNoWait ovlivňují „záznam“ partprogramu do paměti.

- RecWait – pozastaví záznam partprogramu do paměti
- RecNoWait - povolí „záznam“ partprogramu do paměti v případě, kde by byl za normální situace pozastaven

Typické použití funkcí je nejlépe patrné z příkladů. V následující kapitole je příklad na použití funkce RecWait, kdy je potřeba pozastavit záznam, protože se v partprogramu použije vyvolání dialogu, který rozhodne o dalším pokračování partprogramu (na základě odpovědi ANO/NE na zvolenou otázku). Teprve po odpovědi program „ví“, jak bude pokračovat a proto může pokračovat i záznam do paměti.

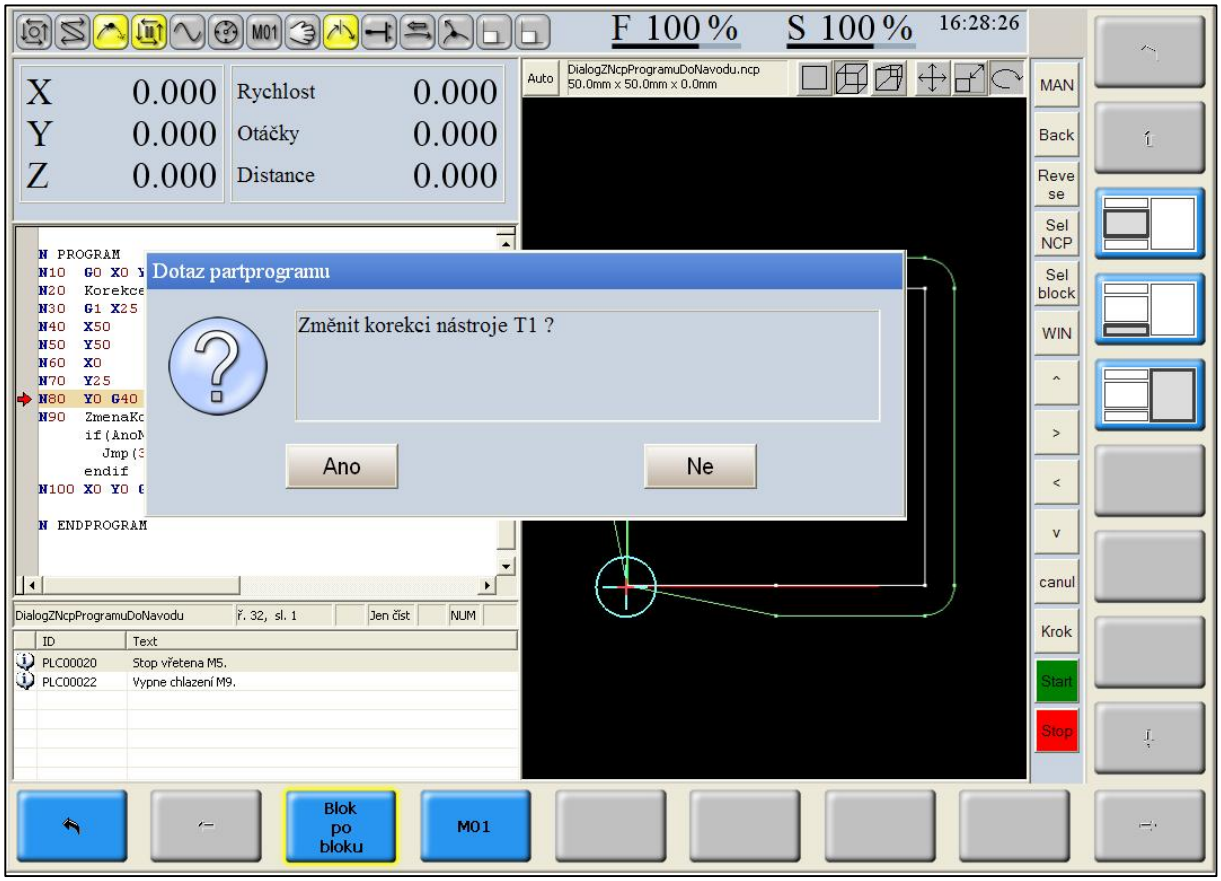
Naopak se mohou vyskytnout v partprogramu situace, kdy se automaticky zastaví záznam do paměti, protože nelze zjistit polohu pro výpočet následujících bloků. Typickým příkladem může být např. výměna nástroje, kterou provádí PLC program tím způsobem, že pomocí polohovacích jednotek (tj. systém neví o poloze) odjíždí do místa výměny nástroje. Neboli systém musí počkat na dokončení výměny, aby se dozvěděl skutečnou polohu a pak může pokračovat v záznamu do paměti. Pokud ale PLC program zajistí, že se po výměně nástroje pomocí polohovacích jednotek vrátí zpět do místa, odkud nájezd na výměnu začal, může se použít funkce RecNoWat, tj. systém může provést kompletní záznam do paměti, protože má k dispozici všechny polohy, potřebné pro výpočet.

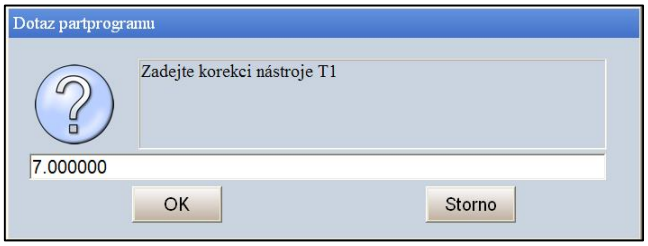
3.2.8 Vyvolání dialogu z partprogramu

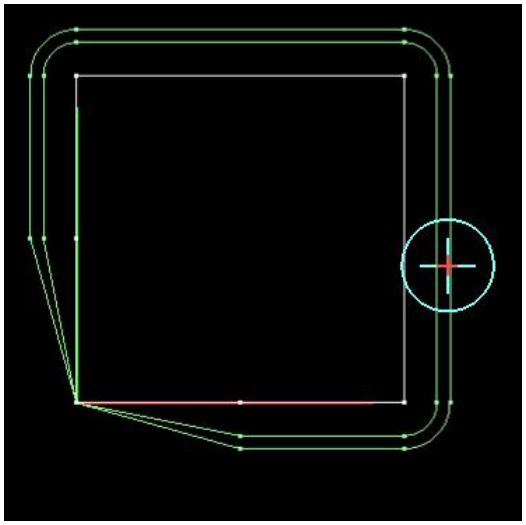
Pomocí funkcí AskUserRealNo( ) a AskUserIntNo( ) můžeme vyvolat dialogové okno a zadat reálnou nebo celočíselnou hodnotu, kterou funkce vrátí do proměnné, se kterou můžeme v partprogramu pracovat. Další funkcí pro vyvolání dialogu je AskUserYesNo( ), která zobrazí textový dotaz a vrací hodnotu 1 (TRUE) nebo 0 (FALSE) podle toho, zda na dotaz odpovíme (tj. stiskneme tlačítko) ANO nebo NE.

Uvedené funkce použijeme v příkladu partprogramu, který objíždí čtverec s poloměrovou korekcí G42, jejíž hodnota je zadaná v proměnné „KOREKCE“. Na začátku je korekce nastavená na hodnotu 5.0. Po objetí čtverce se objeví dotaz „Změnit korekci nástroje T1?“ (viz obr.) Pokud odpovíme ANO, zobrazí se další dialogové okno pro zadání poloměrové korekce. Okno je „předplněné“ původní hodnotou 5.0, kterou změníme na 7.0. Po potvrzení zadání program jede čtverec se změněnou korekcí.

V příkladu jsou výše uvedené funkce použité v makru „ZměnaKorekce“. Toto makro má jako vstupní parametr číslo nástroje, které se zobrazí v dotazu. Povšimněme si též použití funkce RecWait, která pozastaví záznam do paměti. Pokud by nebyla použita, dialogové okno by se objevilo ihned po volbě partprogramu (tj. záznam programu by proběhl až do konce, tj. nebyl by pozastaven funkcí RecWait). Použití funkce RecWait může způsobit i nevykreslení zbylé části partprogramu v grafickém znázornění. To se dokončí až po vykonání funkce RecWait.







Na obrázku je vidět další průchod se změněnou korekcí.

```
" Příklad programu na použití dialogu z programu
" Program objíždí čtverec X - Y o velikosti hrany 50mm
$Korekce        RPARAM
$ActToolRadius  RPARAM
$AnoNe          IPARAM

$ZmenaKorekce(ToolNo)                                                    \
  AnoNe = AskUserYesNo('Změnit korekci nástroje T\i ?',ToolNo)           \
  if(AnoNe)                                                              \
   Korekce = AskUserRealNo(Korekce,0.0,50.0,'Zadejte korekci T\i',ToolNo) \
   RecWait                                                               \
  endif

N PROGRAM
N10  G0 X0 Y0 G23
N20  Korekce = 5.0
N30  G1 X25  F1000  G42 ToolRadius = Korekce   " Korekce G42
N40  X50
N50  Y50
N60  X0
N70  Y25
N80  Y0 G40
N90  ZmenaKorekce(1)             " Dotaz na změnu korekce
     if(AnoNe)                   " Byla změna korekce ?
       Jmp(30)                   " ANO - tak skok zpět s novou korekcí
     endif
N100 X0 Y0 G0                    " NE - pokračuje dál
N ENDPROGRAM
```

3.3 Programování parametrů

V systému lze používat tři skupiny parametrů:

- Systémové
- Reálné (REAL)
- Celočíselné (INT)

3.4 Systémové parametry

Jedná se o předdefinovanou skupinu parametrů, jimž je vnitřně přiřazena adresa S a pořadové číslo. Pro programování se však využívají tyto parametry pojmenované podle použití. Seznam systémových parametrů je uveden v následující tabulce:

3.4.1 Přehled systémových parametrů

AUTONUMR	Automatické číslo pro reálné parametry
AUTONUMI	Automatické číslo pro celočíselné parametry
AUTONUMS	Automatické číslo pro systémové parametry

DIMENSIONENTRYTOLERANCE	Dvojnásobek maximální povolené chyby při zadávání souřadnic (nejmenší hodnota programovatelná s daným počtem desetinných míst). Odvozuje se od počtu desetinných míst, které se používají při zadávání. Pokud se například zadávají míry na 3 desetinná místa, pak se vlivem zaokrouhlení mohou lišit od správné hodnoty max. o +-0.0005, čemuž odpovídá hodnota této konstanty 0.001.
DIMENSIONRESOLUTION	Rozlišení při počítání s mírami (body které leží u sebe blíže než udává tato hodnota budou považovány za totožné). Používá se pouze pro míry, které již "nejsou dotčeny" chybou zadání.
DIMENSIONMAX	Max. přípustná míra., viz CRecCreatorImpl::m_rDimensionMax

Programované polohy interních os s geometrickým významem:

AXI0	Interní osa 0
AXI1	Interní osa 1
AXI2	Interní osa 2
AXI3	Interní osa 3
AXI4	Interní osa 4
AXI5	Interní osa 5
AXI6	Interní osa 6
AXI7	Interní osa 7
AXI8	Interní osa 8

Programované polohy interních os bez geometrického významu:

AXAUX0	Pomocná osa 0
AXAUX1	Pomocná osa 1
AXAUX2	Pomocná osa 2

Interpolační parametry:

CCX	Interpolační parametr 1 (podle ISO obvykle I)
CCY	Interpolační parametr 2 (podle ISO obvykle J)
CCZ	Interpolační parametr 3 (podle ISO obvykle K)

Orientace nástroje:

TOOLX	X-ová složka vektoru

TOOLY	Y-ová složka vektoru
TOOLZ	Z-ová složka vektoru

Programované polohy geometrických os:

AXGX	Geometrická osa X
AXGY	Geometrická osa Y
AXGZ	Geometrická osa Z

Vektor délkové korekce:

TOOLLENX	Vektor délkové korekce, složka X
TOOLLENY	Vektor délkové korekce, složka Y
TOOLLENZ	Vektor délkové korekce, složka Z

Poloměrová korekce/ekvidistanta:

TOOLRADIUS	Poloměr nástroje
EQDOFFS	Ofset pro jízdu po ekvidistantě -
RCANGLE	Mezní úhel, od kterého se budou vkládat oblouky
RCMETHOD	Způsob řešení poloměrové korekce pokud se parametry korekce nemění. 0 - jízda na kolmici (RCMETHOD_NORM), 1 - jízda na průsečík/vložení oblouku (RCMETHOD_KONT)
RCCHANGEMETHOD	Způsob řešení poloměrové korekce při řazení/rušení/změně. Možnosti jako u RCMETHOD.
RCOPTIONS	Další příznaky pro práci s poloměrovými korekcemi: • bit 0: 1 - pro první a poslední blok s G00 za resp. před blokem pracovním posuvem se nastaví příznak změny směru poloměrové korekce • bit 1: 1 - zakázat vkládání oblouků v případě, že úhel je menší než limit, ale průsečík ekvidistant neexistuje
TNP	Tool Nose Position – orientace špičky nástroje u soustruhů a karuselů (0 – 9)

Interpolační parametry:

CR	Poloměr kružnice (Circle Radius)
CREV	Počet otáček kružnice (0 ... nejkratší dosažení cílového bodu kružnice, >0 ... přidá se zadaný počet otáček
CHLEN	Délka zkosení (Chamfer Length)
CHLENT	Délka zkosení ve směru předchozího/následujícího bloku (Chamfer Length Tangential)
ROUNDR	Poloměr zaoblení (Round Radius)

Rychlost, otáčky vřetene:

FEED	Programovaná rychlost v mm/min
REVFEED	Programovaná rychlost v mm/ot (otáčkový posuv)
SPINDLESPEED	Programované otáčky vřetene v ot/min
FEEDOVR	Programovaný override rychlosti v %
SPINDLESPEEDOVR	Programovaný override otáček vřetene v %
SLIM	Limit otáček vřetene pro konstantní řeznou rychlost
CCS	Rychlost pro režim konstantní řezné rychlosti, t.j. konstantní řezná rychlost v m/min

Dynamické řízení rychlosti:

ACCELERATIONTYPE	Způsob rampování (0 - lineární, 1 - parabolické)
PARABACCEL	Parabolické zrychlení (pro parabolický úsek rampy v m s^{-3})
LINEARACCEL	Lineární zrychlení (pro lineární úsek rampy v m/s^2)
REQACCURACY	Kritérium přesnosti (požadovaná "přesnost" v mm při řízení rychlosti)
MAXACCELERATION	Kritérium přetížení (maximální zrychlení m/s^2)
MAXCENTRIFUGALACCELERATION	Kritérium odstředivého zrychlení pro kruhovou interpolaci (maximální zrychlení pro kruhovou interpolaci m/s^2) (R411)
TRANSITIONFEEDCONTROL	Způsob řízení kriteriální rychlosti: 0 - automaticky na základě zadaných kritérií, 1 - zadaná rychlost (TRANSITIONFEED),

	2 - programovaná rychlost (F)
TRANSITIONFEED	Rychlost na přechodu mezi bloky, pokud je TRANSITIONFEEDCONTROL 1.

Průměrové programování, jednotky

DIAMETERPROGR	1 - průměrové programování, 0 - poloměrové programování (pro soustruhy a karusely, při zapnutém **absolutním** programování)
DIAMETERINCPROGR	1 - průměrové programování, 0 - poloměrové programování (pro soustruhy a karusely, při zapnutém **přírůstkovém** programování)
LENGTHUNIT	Délkové jednotky pro zadávání rozměrů (METRIC / IMPERIAL)
FEEDUNIT	Jednotky pro zadávání rychlosti (METRIC / IMPERIAL)
ANGLEUNIT	Úhlové jednotky pro zadávání (RADIANS / DEGREES / GRADS). Doporučuje se nastavit v prioritním bloku!

Vektor posunutí 1 nulového bodu:

OFFS1AXI0	Vektor posunutí 1, složka 0
OFFS1AXI1	Vektor posunutí 1, složka 1
OFFS1AXI2	Vektor posunutí 1, složka 2
OFFS1AXI3	Vektor posunutí 1, složka 3
OFFS1AXI4	Vektor posunutí 1, složka 4
OFFS1AXI5	Vektor posunutí 1, složka 5
OFFS1AXI6	Vektor posunutí 1, složka 6
OFFS1AXI7	Vektor posunutí 1, složka 7
OFFS1AXI8	Vektor posunutí 1, složka 8

Vektor posunutí 2 nulového bodu:

OFFS2AXI0	Vektor posunutí 2, složka 0
OFFS2AXI1	Vektor posunutí 2, složka 1
OFFS2AXI2	Vektor posunutí 2, složka 2
OFFS2AXI3	Vektor posunutí 2, složka 3
OFFS2AXI4	Vektor posunutí 2, složka 4
OFFS2AXI5	Vektor posunutí 2, složka 5
OFFS2AXI6	Vektor posunutí 2, složka 6
OFFS2AXI7	Vektor posunutí 2, složka 7
OFFS2AXI8	Vektor posunutí 2, složka 8

Matice programové transformace (Program Transformation Matrix):

PTMATRIX00	Matice programové transformace, složka 0,0
PTMATRIX10	Matice programové transformace, složka 1,0
PTMATRIX20	Matice programové transformace, složka 2,0
PTMATRIX30	Matice programové transformace, složka 3,0
PTMATRIX01	Matice programové transformace, složka 0,1
PTMATRIX11	Matice programové transformace, složka 1,1
PTMATRIX21	Matice programové transformace, složka 2,1
PTMATRIX31	Matice programové transformace, složka 3,1
PTMATRIX02	Matice programové transformace, složka 0,2
PTMATRIX12	Matice programové transformace, složka 1,2
PTMATRIX22	Matice programové transformace, složka 2,2
PTMATRIX32	Matice programové transformace, složka 3,2
PTMATRIX03	Matice programové transformace, složka 0,3
PTMATRIX13	Matice programové transformace, složka 1,3
PTMATRIX23	Matice programové transformace, složka 2,3
PTMATRIX33	Matice programové transformace, složka 3,3

Matice transformace polotovaru (Workpiece Transformation Matrix):

WTMATRIX00	Matice transformace polotovaru, složka 0,0
WTMATRIX10	Matice transformace polotovaru, složka 1,0
WTMATRIX20	Matice transformace polotovaru, složka 2,0
WTMATRIX30	Matice transformace polotovaru, složka 3,0
WTMATRIX01	Matice transformace polotovaru, složka 0,1
WTMATRIX11	Matice transformace polotovaru, složka 1,1
WTMATRIX21	Matice transformace polotovaru, složka 2,1
WTMATRIX31	Matice transformace polotovaru, složka 3,1
WTMATRIX02	Matice transformace polotovaru, složka 0,2
WTMATRIX12	Matice transformace polotovaru, složka 1,2
WTMATRIX22	Matice transformace polotovaru, složka 2,2
WTMATRIX32	Matice transformace polotovaru, složka 3,2
WTMATRIX03	Matice transformace polotovaru, složka 0,3
WTMATRIX13	Matice transformace polotovaru, složka 1,3
WTMATRIX23	Matice transformace polotovaru, složka 2,3
WTMATRIX33	Matice transformace polotovaru, složka 3,3

Aktuální pozice os před provedením jakékoliv transformace (tato poloha se programuje, pouze pro čtení):

AX0POS0	
AX1POS0	
AX2POS0	
AX3POS0	
AX4POS0	
AX5POS0	
AX6POS0	
AX7POS0	
AX8POS0	
AX9POS0	
AX10POS0	
AX11POS0	
AX12POS0	
AX13POS0	
AX14POS0	
AX15POS0	

Aktuální pozice os po provedení programové transformaci (pouze pro čtení):

AX0POS1	
AX1POS1	
AX2POS1	
AX3POS1	
AX4POS1	
AX5POS1	
AX6POS1	
AX7POS1	
AX8POS1	
AX9POS1	
AX10POS1	
AX11POS1	
AX12POS1	
AX13POS1	
AX14POS1	
AX15POS1	

Aktuální pozice os po provedení transformaci polotovaru (pouze pro čtení):

AX0POS2	
AX1POS2	
AX2POS2	
AX3POS2	

AX4POS2	
AX5POS2	
AX6POS2	
AX7POS2	
AX8POS2	
AX9POS2	
AX10POS2	
AX11POS2	
AX12POS2	
AX13POS2	
AX14POS2	
AX15POS2	

Aktuální pozice os po provedení posunutí 1 (pouze pro čtení):

AX0POS3	
AX1POS3	
AX2POS3	
AX3POS3	
AX4POS3	
AX5POS3	
AX6POS3	
AX7POS3	
AX8POS3	
AX9POS3	
AX10POS3	
AX11POS3	
AX12POS3	
AX13POS3	
AX14POS3	
AX15POS3	

Aktuální pozice os po provedení posunutí 2 (pouze pro čtení):

AX0POS4	
AX1POS4	
AX2POS4	
AX3POS4	
AX4POS4	
AX5POS4	
AX6POS4	
AX7POS4	
AX8POS4	
AX9POS4	
AX10POS4	
AX11POS4	
AX12POS4	
AX13POS4	
AX14POS4	
AX15POS4	

Aktuální pozice os po provedení pětiosé transformace (pouze pro čtení):

AX0POS5	
AX1POS5	
AX2POS5	
AX3POS5	
AX4POS5	
AX5POS5	
AX6POS5	
AX7POS5	

AX8POS5	
AX9POS5	
AX10POS5	
AX11POS5	
AX12POS5	
AX13POS5	
AX14POS5	
AX15POS5	

Aktuálni pozice os po provedení transformace stroje (vstup do servosmyčky, pouze pro čtení):

AX0POS6	
AX1POS6	
AX2POS6	
AX3POS6	
AX4POS6	
AX5POS6	
AX6POS6	
AX7POS6	
AX8POS6	
AX9POS6	
AX10POS6	
AX11POS6	
AX12POS6	
AX13POS6	
AX14POS6	
AX15POS6	

Celočíselné parametry pro PLC:

IPLC0	
IPLC1	
IPLC2	
IPLC3	
IPLC4	

Reálné parametry pro PLC:

RPLC0	
RPLC1	
RPLC2	
RPLC3	
RPLC4	

Konstanty:

TRUE	1	
FALSE	0	
RCMETHOD_NORM	0	Pro řízení poloměrových korekcí
RCMETHOD_KONT	1	Pro řízení poloměrových korekcí
METRIC	0	Metrické zadávání – mm, mm/min, ...
IMPERIAL	1	Palcové zadávání – inch, inch/min ...
RADIANS	0	Radiány
DEGREES	1	Stupně
GRADS	2	Grady
NORM	0	Pro MAKRA
KONT	1	Pro MAKRA
PI	3.14159...	Ludolfovo číslo

Systémové parametry se programují jako běžné parametry, tj. za jejich symbolickým názvem se uvede rovnítko a hodnota. Jako hodnoty se mohou použít předdefinované konstanty, např. Ludolfovo číslo PI.

Příklad:

```
ToolRadius = 25          „ Hodnota poloměrové korekce
LENGTHUNIT = 1           „ Programování délek v palcích
LENGTHUNIT = IMPERIAL    „ Programování délek v palcích (předdef.konstantou)
```

3.4.2 Reálné a celočíselné parametry

Parametry, které lze v systému používat pro parametrické programování adres a pro různé výpočty, jsou podle hodnoty, se kterou pracují, rozděleny na reálné a celočíselné.
Pozn.: kromě reálných a celočíselných parametrů existují ještě systémové parametry, označené písmene S. Ty jsou ale vyhrazeny pro systém a uživatel je používá jako pojmenované parametry – např. rychlost FEED je systémový parametr S287

Reálné parametry (typ reálné číslo, REAL) se programují adresou R.
Celočíselné parametry (typ integer, INT) se programují adresou I

Příklad:

```
N10    R12=125.450      I26=320
```

Parametru R12 je přiřazeno reálné číslo 125.450
Parametru I26 je přiřazena celočíselná hodnota 320

Používání parametrů tak, jak je uvedeno v předešlém příkladu je, zvláště pokud by se použilo více parametrů, značně nepřehledné, je nutné si ke každému parametru přidat komentář, aby bylo jasné k čemu je parametr použit.

Proto se doporučuje používat parametry pojmenované (t.j využívat makra). Jméno se parametru přiřadí definicí makra, jak je uvedeno v následujícím příkladu deklarace maker:

```
$SIRKA_PATKY            R100
$DELKA_PATKY            R101
$RYCHLOST_HRUBOVACI     R102
$POCET_PRUCHODU         I100
```

Pozn: uvedeno pro úplnost, doporučený způsob přiřazení parametrů viz. následující kapitola

Pojmenované proměnné SIRKA_PATKY se přiřadí reálný parametr R100, proměnné POCET_PRUCHODU se přiřadí celočíselný parametr I100 atd. V partprogramu se pak s parametry pracuje např. takto:

```
N10  SIRKA_PATKY=500   RYCHLOST_HRUBOVACI=200
N20  G01 AXGX=(SIRKA_PATKY/2)   FEED=RYCHLOST_HRUBOVACI
N30  POCET_PRUCHODU=POCET_PRUCHODU + 1
N40 ….
```

V bloku N10 se např. naplní parametry hodnotami, v bloku N20 jede souřadnice X (je programována pomocí systémového parametru AxGX, tj. jako geometrická osa X) na hodnotu 250 (polovina šířky patky) rychlost 200mm/min, v bloku N30 se inkrementuje čítač průchodů o 1 apod.

Parametry mohou být deklarované např. na začátku partprogramu nebo v hlavičkovém souboru, který se deklaruje klíčovým slovem #INL, například:

```
#INL  (HLAVICKA.NCH)
```

Pokud se neuvede úplná cesta, je tento soubor uložen standardně v adresáři D:\CNC User Files\INCLUDE.

3.4.3 Automatické přidělení parametrů

Aby nedošlo k překřížení parametrů, t.j. aby se zabránilo duplicitnímu použití stejného parametru na různé proměnné, doporučuje se používat **automatické přidělování parametrů** tak, že místo konkrétního parametru se uvede pouze RPARAM pro reálné parametry a IPARAM pro celočíselné parametry. Příklad uvedený v předešlé kapitole bychom pak zapsali vhodněji takto:

```
$SIRKA_PATKY            RPARAM
$DELKA_PATKY            RPARAM
$RYCHLOST_HRUBOVACI     RPARAM
$POCET_PRUCHODU         IPARAM
```

Systém přiřadí konkrétní parametry sám – z hlediska uživatele je přidělené číslo nepodstatné.

4

4. DRUHY POHYBU

4.1 Geometrické, synchronní, interní osy a NC osy

Před popisem druhů pohybu a interpolací je třeba definovat pojmy geometrické osy, synchronní osy, interní osy a NC osy. Poznámky ke geometrickým osám jsou uvedeny také v kapitole 3. Z hlediska programátora jsou důležité hlavně geometrické a synchronní osy.

Geometrické osy – Geometrické osy jsou tři a v daném okamžiku definují prostor – pravotočivý souřadný systémem X-Y-Z. Pouze v těchto osách provádí systém interpolace a v těchto osách se definuje rychlost posuvu. Pokud se má provádět interpolace osy, která není geometrická, což může přicházet v úvahu u složitějších strojů, musí být přepnuta (např. z partprogramu) jako osa geometrická (viz funkce SetGAx(XNo, YNo, ZNo) v kapitole 3). Jednoduché stroje mají pevné přiřazení os v konfiguraci

Synchronní osy – Pokud jsou programované společně s geometrickými osami, znamená to v podstatě pouze to, že jejich pohyb začne současně s geometrickými osami a skončí pohyb rovněž současně s geometrickými osami. Pokud jsou programovány samostatně, jedou rychlostí, která se zadá systémovým parametrem (??? Název !!!), tj. neplatí pro ně rychlost programovaná adresou F nebo systémovým parametrem Feed! Pokud není rychlost systémovým parametrem zadaná, jedou rychlostí přednastavenou konfiguračních souborech typu *.ChannelConfig.

Asynchronní osy - obvykle jsou plně v režii PLC programu, většinou se jedná o různé pomocné osy, často se programují pouze M-funkcemi. Jejich použití musí být v návodu pro konkrétní stroj. Pohyb se startuje na začátku bloku, jedou „svojí“ rychlostí, t.j. mohou skončit pohyb dříve nebo později než eventuelně současně programované geometrické osy

Interní osy - používají se při přípravě partprogramu, je to pole hodnot, které lze nastavovat a číst z partprogramu. Interních os může být 9 a programují se AXI0 – AXI8. Pro běžné programování se většinou tento zápis nepoužívá. Využití má především v obecných programech (makrocyklech, pevných cyklech), které se píší bez ohledu na to, jaké osy budou skutečně na stroji. Tři z interních os jsou vždy označeny jako geometrické – buď v konfiguraci, nebo přepnutím z partprogramu

NC osy – osy, se kterými pracuje modul reálného času. Mohou být připojeny k interním osám – z toho se odvodí jejich chování (geometrické/synchronní/asynchronní osy). Mohou být řízeny z PLC

4.2 Stavění souřadnic - funkce G00

Stavěním souřadnic se rozumí přemístění nástroje do koncového (programovaného) bodu rychloposuvem. Koncová poloha se programuje v absolutních nebo inkrementálních mírách (platí obecně pro všechny druhy pohybů). Toto přemístění nástroje je zadáno v bloku funkcí **G00**, která je současně nositelem informace pro provádění pohybu rychloposuvem. Velikost rychloposuvu (pro každou osu) je v systému zadána pevně jako strojní konstanta (atribut FeedRT v souboru Channel0.ChannelConfig) a v bloku se neprogramuje. Při rychloposuvu je zaručen plynulý rozjezd a dojezd na začátku a konci pohybu. V jednom bloku je možno programovat stavění i více souřadnic najednou.

4.3 Lineární interpolace - funkce G01

Lineární interpolace se volí funkcí G01. V jednom bloku je možno naprogramovat interpolaci mezi jednou až n-souřadnicemi (n je maximální počet souřadnic podle konfigurace) naprogramováním souřadnic koncového bodu v příslušných osách. Je tedy možná i vzájemná kombinace lineárních a rotačních souřadnic. Pro lineární interpolaci je nutné, aby byla zadaná posuvová rychlost (pod adresou F nebo parametrem Feed). Rychlost F nemusí být uvedena v bloku s G01, ale v kterémkoli předcházejícím bloku. (u specielních strojů může být rychlost převzatá z technologické nebo jiné tabulky a v partprogramu nemusí být vůbec programovaná). Rychlost je vypočtena pro geometrický prostor X-Y-Z. Pokud je programován větší počet souřadnic, platí rychlost pro tento prostor, ostatní osy interpolují tak, aby dojely do koncového bodu současně s geometrickými osami X,Y,Z

Příklad

Absolutní programování

```
N10 G01 G90 X90 Y50
```

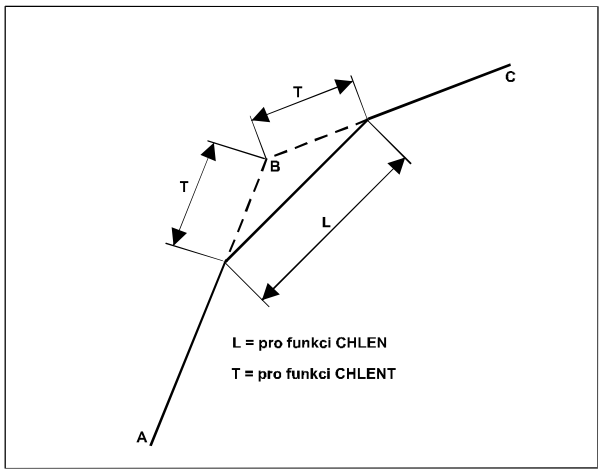
přírůstkové programování

```
N10 G01 G91 X40 Y30
```

Je-li výchozím bodem interpolace bod A o souřadnicích X=50, Y=20 a koncový bod B o souřadnicích X=90 Y=50, potom uvedené bloky vykonají stejnou dráhu.

4.3.1 Zkosení rohu – G10

Pomocí funkcí G10 lze programovat zkosení rohu se zadanou délku zkosení (funkce CHLEN) nebo se zadanou vzdáleností začátku/konce zkosení od rohu (funkce CHLENT). Použití vyplývá z uvedených příkladů:



Pokud se programují se dvě navazující úsečky A-B a B-C (viz obr.), je možné mezi ně vložit zkosení zadané buď jeho délkou L nebo vzdáleností začátku zkosení k průsečíku B.

Zadání zkosení délkou L:

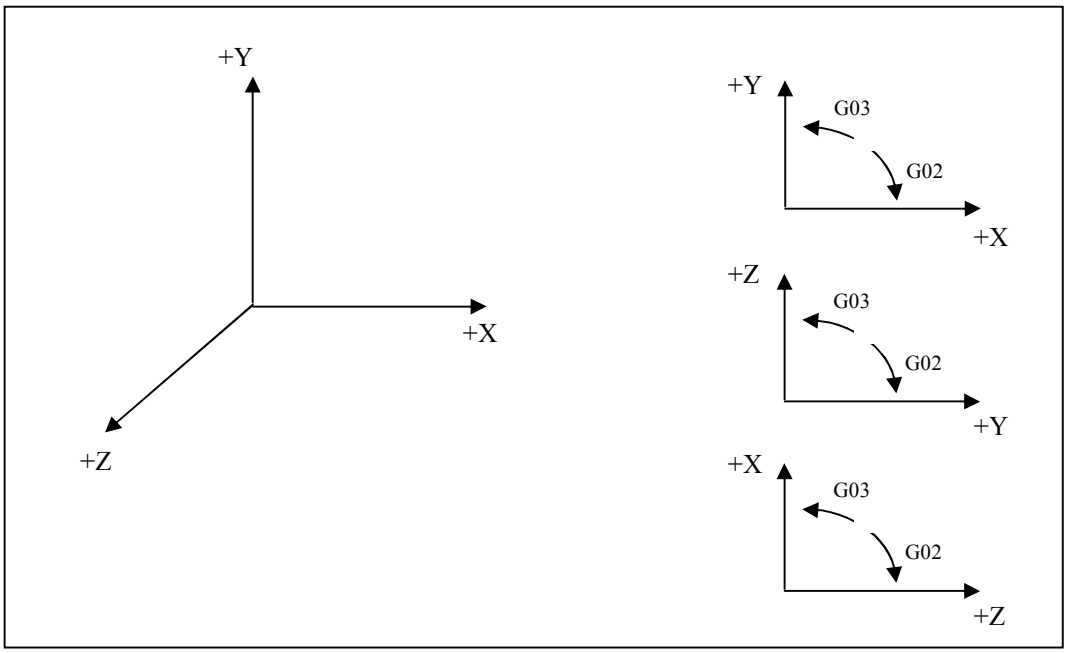
```
N10 G90 G0 X0 Y0
N20 G91 G1  F100 X50 Y150
N30 G10 CHLEN=30   " Délka L
N40 G1   X150 Y50
```

Zadání zkosení vzdáleností T:

```
N10 G90 G0 X0 Y0
N20 G91 G1  F100 X50 Y150
N30 G10 CHLENT=20   " Délka T
N40 G1   X150 Y50
```

4.4 Kruhová interpolace - funkce G02, G03

Kruhová interpolace se volí funkcí G02 (CW - pohyb ve směru hodinových ručiček) nebo G03 (CCW - pohyb proti směru hodinových ručiček). Vyjádření směru kruhové interpolace (G02 nebo G03) v libovolných rovinách pravotočivé souřadné soustavy pro geometrické osy X,Y,Z systém se určuje při pohledu na rovinu kruhové dráhy (obr.).



Kruhovou interpolaci je možné programovat pouze v rovině. Za rovinu interpolace můžeme zvolit rovinu určenou dvěma souřadnicemi.
Interpolační rovina je určena těmito funkcemi:

G17 – rovina X – Y

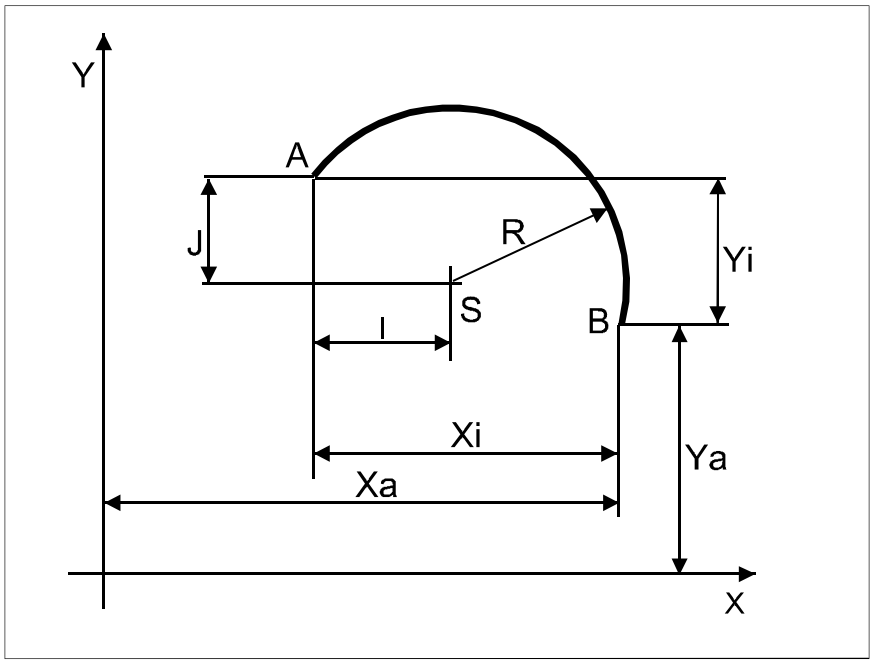
G18 – rovina Z - X

G19 – rovina Y - Z

Interpolační rovina se při programování kružnice nemusí uvádět za předpokladu, že je koncový bod kružnice zadán oběma souřadnicemi, čímž je rovina interpolace určena. Pokud by se uvedla pouze jedna souřadnice koncového bodu kružnice a druhá by zůstala beze změny (např. oproti předešlému bloku), musela by se interpolační rovina uvést. Přestože je to povoleno, doporučuje se spíše programovat obě souřadnice koncového bodu kružnice. Kromě interpolační roviny existuje ještě pracovní rovina (někdy též nazývána korekční rovina), která má G-funkce totožné jako interpolační rovina. Pracovní rovina se musí vždy uvést, pokud jsou použity poloměrové korekce. Obvykle je v prioritním bloku uvedena rovina X-Y, t.j. G17. Podrobněji je o korekčních rovinách pojednáno v kapitole o poloměrových korekcích.
Pozn:
Pozor na rovinu G18. Z hlediska programování G2 resp. G3 se na ní musí pohlížet jako na rovinu Z – X, tj. jakoby první osa byla Z a druhá osa X

4.4.1 Možnosti zadání kruhové interpolace

Kružnici je možné zadat několika způsoby. Zadáním koncového bodu a středu, zadáním koncového bodu a poloměru. Je možné zadat i opakování kružnice, což se využívá pro programování spirály. V dalším textu budou uvedeny všechny možnosti.



Obrázek platí pro rovinu G17

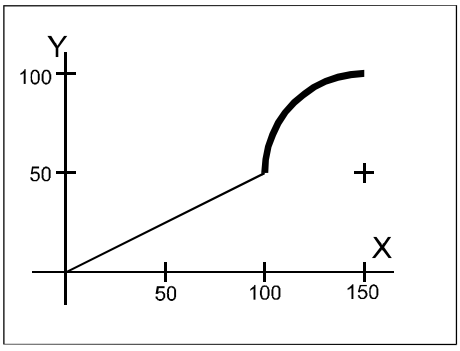
A – počáteční bod kružnice
B – koncový bod kružnice
S – střed kružnice
I – vzdálenost počátku a středu kružnice ve směru X
J - vzdálenost počátku a středu kružnice ve směru Y
Xa – X-ová souřadnice koncového bodu kružnice v abslotním programování
Xi - X-ová souřadnice koncového bodu kružnice v inkrement. programování
Ya – Y-ová souřadnice koncového bodu kružnice v abslotním programování
Yi - Y-ová souřadnice koncového bodu kružnice v inkrement. programování

4.4.2 Zadání kružnice koncovým bodem a středem

Programuje se koncový bod kružnice (absolutně nebo přírůstkově) a souřadnice středu (vždy jen přírůstkově od počátku kružnice) a eventuelně interpolační rovina. Souřadnice středu (interpolační parametry) se programují podle ISO adresami I, J, K nebo systémovými parametry CCX, CCY, CCZ.
I, J, K, resp. interpolační parametry jsou přiřazeny osám takto:

I resp. CCX - vzdálenost středu od počátku ve směru osy X (s ohledem na znaménko)
J resp. CCY - vzdálenost středu od počátku ve směru osy Y (s ohledem na znaménko)
K resp. CCZ - vzdálenost středu od počátku ve směru osy Z (s ohledem na znaménko)

Kružnice v rovině G17 (X-Y):



Příklad programovaní kružnice v rovině X-Y s počátečním bodem [100,50], koncovým bodem [150,100] a poloměrem 50

Absolutně:
```
N  G90 G2 X150 Y100 I50 J0
```
Nebo zápisem pomocí systémových parametrů:
```
N  G90 G2 AxGX=150 AxGY=100 CCX=50 CCY=0
```

Přírůstkově:
```
N  G91 G2 X50 Y50 I50 J0
```
Nebo zápisem pomocí systémových parametrů:
```
N  G91 G2 AxGX=50 AxGY=50  CCX=50 CCY=0
```

Souřadnice středu se programují vždy přírůstkově od počátku kružnice, s ohledem na znaménko. Pokud bychom v uvedeném příkladu jeli kružnici z bodu [150,100] do bodu [100,50], vypadal by blok následovně:

Absolutně:
```
N G90 G3 X100 Y50 I0 J-50
```
Nebo zápisem pomocí systémových parametrů:
```
N G90 G3 AxGX=100 AxGY=50 CCX=0 CCY=-50
```

Přírůstkově:
```
N G91 G3 X-50 Y-50 I0 J-50
```
Nebo zápisem pomocí systémových parametrů:
```
N G91 G3 AxGX=-50 AxGY=-50 CCX=0 CCY=-50
```

Kružnice v rovině G18 (Z-X):

Ta samá kružnice jako na obrázku, místo osy X je osa Z a místo osy Y je osa X:

Absolutně:
```
N  G90 G2 Z150  X100  K50  I0
N  G90 G2 AxGZ=150 AxGX=100  CCZ=50  CCX=0
```

Přírůstkově:
```
N  G91 G2 Z50 X50 K50 I0
N  G91 G2 AxGZ=50 AxGX=50  CCZ=50 CCX=0
```

Kružnice v rovině G19 (Y-Z):

Ta samá kružnice jako na obrázku, místo osy X je osa Y a místo osy Y je osa Z:

Absolutně:
```
N  G90 G2 Y150 Z100 J50 K0
N  G90 G2 AxGY=150 AxGZ=100 CCY=50 CCZ=0
```

Přírůstkově:
```
N  G91 G2 Y50 Z50 J50 K0
N  G91 G2 AxGY=50 AxGZ=50  CCY=50 CCZ=0
```

Zápis pouze jednoho koncového bodu

Pokud zůstává jeden z koncových bodů kružnice ve stejné poloze jako v předešlém bloku, nemusí se programovat, ale musí být správně definovaná interpolační rovina jednou z funkcí G17, G18, G19. Vzhledem k tomu, že vždy je nějaká interpolační rovina zadaná (přinejmenším obvykle G17 v prioritním bloku), mohla by se vykonat, bez toho, aby se vyhlásila chyba, jiná kružnice, než jsme zamýšleli:

V následujícím příkladu je správně programována půlkružnice v rovině Y-Z.

```
N G90  G1  Y100 Z50
N      G2  Y200 Z50 J50 K0
```

Vzhledem k tomu, že koncový bod v ose Z je stejný jako v předešlém bloku, nemusel by se programovat:

```
N G90  G1  Y100 Z50
N      G2  Y200     J50 K0
```

Pokud by ale nebyla kdekoli před blokem s půlkružnicí definována interpolační rovina G19, provedla by se kružnice v rovině G17, která je obvykle zvolena prioritně!

Správně musí být tedy napsáno:

```
N G90  G1  Y100 Z50
N G19  G2  Y200     J50 K0
```

Z těchto důvodů je vhodnější psát i v těch případech, kdy to není nezbytně nutné, vždy oba koncové body.

Programování celé kružnice

Pokud se programuje celá kružnice, tj. počáteční a koncový bod jsou totožné, nemusí se koncové body programovat. Opět za předpokladu, že je správně definovaná interpolační rovina!

V následujícím příkladu je naprogramovaná celá kružnice s počátkem v bodu Y100 Z50.

```
N G90  G1  Y100 Z50
N G19  G2          J50 K0
```

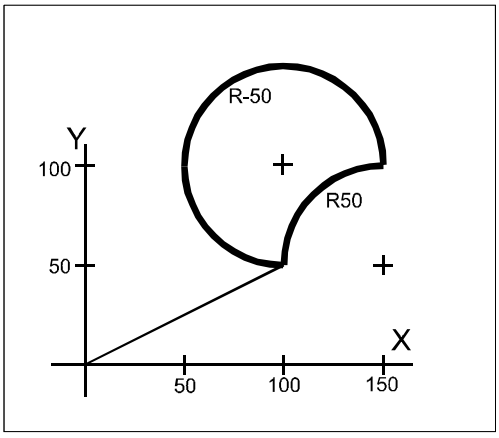
I v tomto případě je však vhodnější kvůli přehlednosti programovat koncové body, i když jsou totožné. Nemusí pak být uvedena ani interpolační rovina.

```
N G90  G1  Y100 Z50
N      G2  Y100 Z50 J50 K0
```

4.4.3 Zadání kružnice koncovým bodem a poloměrem

Ve většině případů je jednodušší zadávat kružnici nikoli souřadnicemi středu, ale poloměrem. Programují se opět koncové body kružnice a poloměr – buď adresou R nebo systémovým parametrem CR (Circle Radius)

Uvedeme pouze příklad pro rovinu G17



Absolutně:

```
N  G90 G2 X150 Y100 R50
```

Nebo zápisem pomocí systémových parametrů:

```
N  G90 G2 AxGX=150 AxGY=100 CR=50
```

Přírůstkově:

```
N  G91 G2 X50 Y50 R50
```

Nebo zápisem pomocí systémových parametrů:

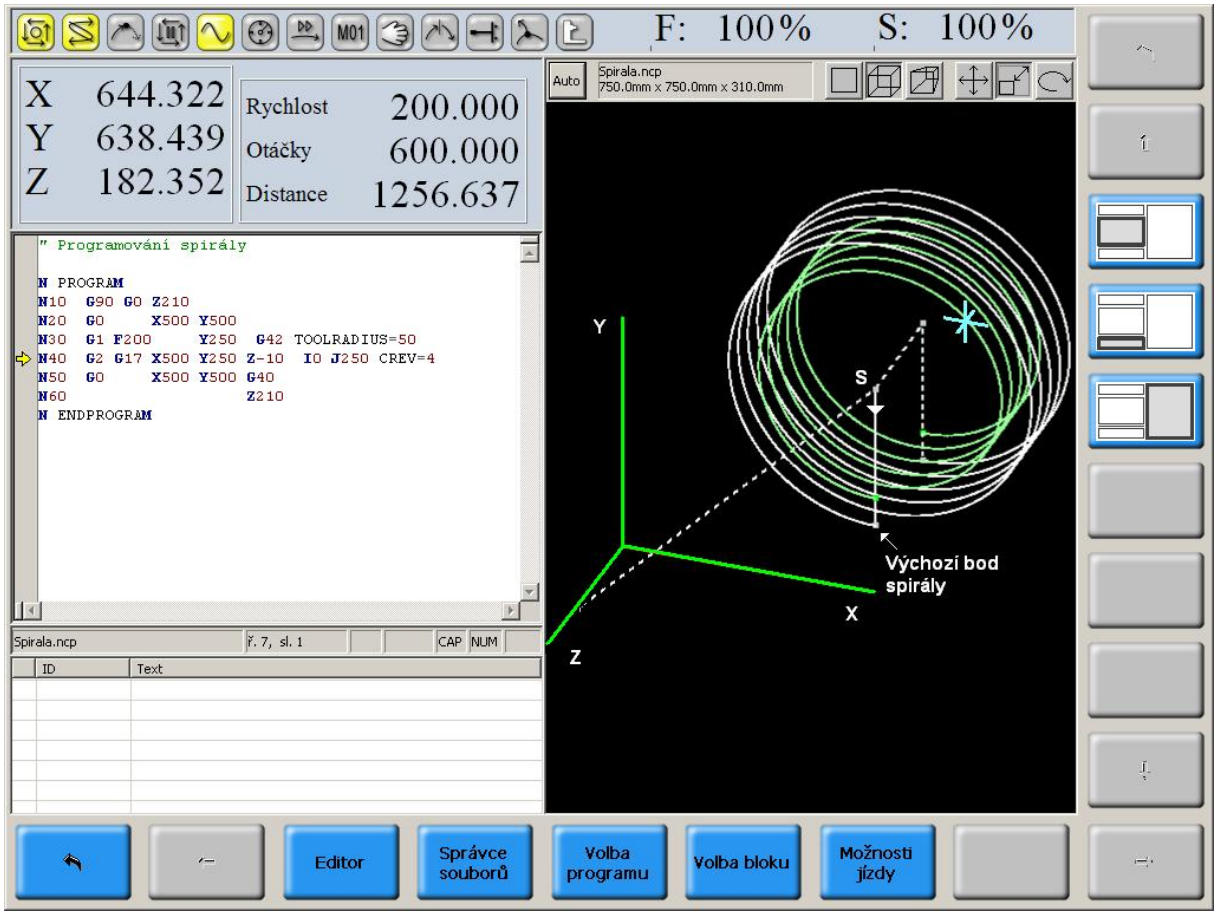
```
N  G91 G2 AxGX=50 AxGY=50  CR=50
```

Protože při zadání kružnice poloměrem vycházejí dvě řešení, bere se při programování kladného R kružnice s výsečí menší než 180 stupňů, při programování záporného R se vezme kružnice větší než 180 stupňů. (viz obr.)

Kruhová interpolace není omezena na jeden kvadrant. Programovaná kružnice může procházet i více kvadranty.

4.4.4 Programování šroubovice

Programování šroubovice je specielním případem programování kružnice. Šroubovice se programuje jako kružnice v příslušné interpolační rovině a k tomu třetí osa, kterou se programuje délka šroubovice. Dále se funkcí CREV programuje počet opakování, resp. počet otáček kružnice. Pozor, CREV určuje počet otáček, které se připočtou k zadané první kružnici. Chceme-li tedy šroubovici např. s deseti závity, bude CREV=9.

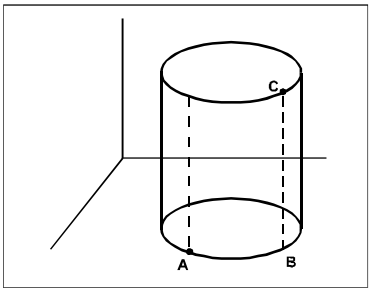


Na obrázku je uveden příklad šroubovice v interpolační rovině G17 o délce 220 mm (výchozí bod v ose Z na hodnotě 210 absolutně, koncový bod v ose Z na hodnotě -10 absolutně) s celkem 5 závity: Do výchozího bodu šroubovice se v tomto příkladu najíždí ze středu kružnice S.
Pozn.: V příkladu je uvedena šroubovice, projetá s poloměrovou korekcí 50mm. Poloměrová korekce se zařadí při jízdě ze středu kružnice S do výchozího bodu šroubovice a odvolá se opět při pohybu z konce šroubovice do středu kružnice. Programování poloměrových korekcí je v samostatné kapitole.

Příklad:

```
N PROGRAM
N10  G90 G0 Z210
N20  G0     X500 Y500
N30  G1 F2000    Y250  G42 TOOLRADIUS=50
N40  G2 G17 X500 Y250 Z-10  I0 J250 CREV=4
N50  G0     X500 Y500 G40
N60                   Z210
N ENDPROGRAM
```

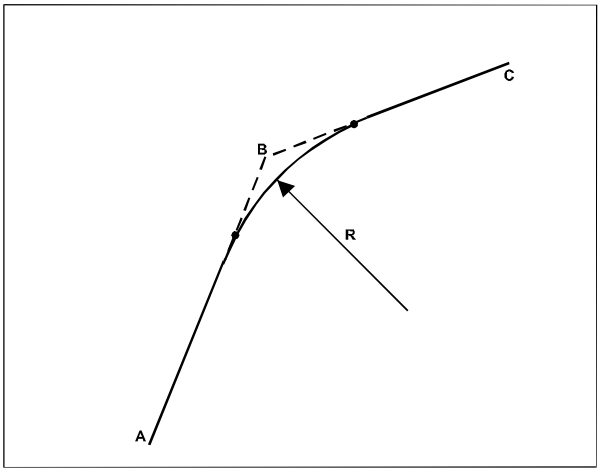
Pozor, u šroubovice je nutné vždy programovat **interpolační rovinu**, aby bylo jasné, kterých dvou os ze tří programovaných se týká kružnice. Ve třetí ose, která není v interpolační rovině se provádí lineární interpolace



Je možné programovat i neúplnou šroubovici, resp. programovaná kružnice nemusí být celý kruh, ale jen jeho část, naprogramovaná podle pravidel programování kružnic. Chceme-li šroubovici z bodu A do bodu C, naprogramujeme kružnici z bodu A do bodu B a do parametru CREV zadáme počet otáček kružnice. Počet otáček se bude počítat od bodu B (viz obr.).

4.4.5 Zaoblení rohu se zadaným poloměrem - G11

Podobně jako u zkosení lze mezi dvě programované úsečky A-B a B-C (viz obr.) vložit zaoblení se zadaným poloměrem. Programuje se funkcí G11 a systémovým parametrem ROUNDR. Blok se vloží mezi programované úsečky.



Zaoblení zadané poloměrem R:

```
N10 G90 G0 X0 Y0
N20 G91 G1  F100 X50 Y150
N30 G11 ROUNDR=30    " R=30mm
N40 G1   X150 Y50
```

V následujícím bloku za G10 musí být programována lineární interpolace, t.j funkce G1 (eventuelně G0, což je ale technologicky nelogické).

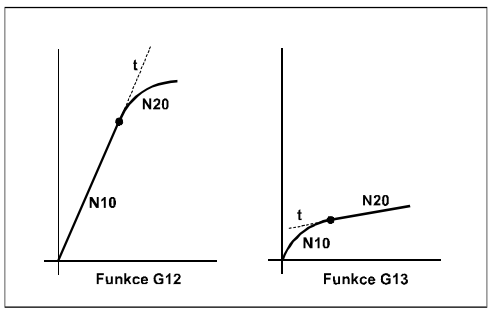
4.4.6 Kruhová interpolace s tečným napojením

Pro usnadnění programování kruhové interpolace jsou k dispozici dvě funkce

G12 – kruhová interpolace s tečným napojením na předchozí blok

G13 - kruhová interpolace s tečným napojením na následující blok

Programuje se pouze koncový bod kružnice, ostatní parametry dopočítá systém z podmínky, že kružnice je napojena z předchozího bloku tečně (G12), resp. následující blok je napojen tečně (G13).



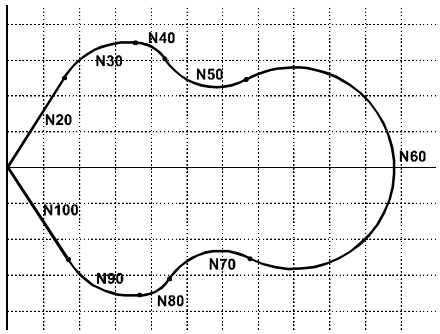
Příklad (G12):

```
N5  G90 G0 X0 Y0
N10 G1 F100 X50 Y150
N20 G12 X130 Y180
```

Příklad (G13):

```
N5 G90 G0 X0 Y0
N10 G13 F100 X40 Y60
N20 G1  X200 Y100
```

Funkce G12 (a G13) se s výhodou požívají při navazování kružnic, jak ukazuje následující příklad.




```
N PROGRAM
N10  G90 G0    X0   Y0
N20  G1  F200  X80  Y125
N30  G12       X180 Y175
N40  G12       X220 Y150
N50  G12       X335 Y125
"
N60  G12       X335 Y-125
N70  G12       X220 Y-150
N80  G12       X180 Y-175
N90  G12       X80  Y-125
N100 G01       X0   Y0
N ENDPROGRAM
```

4.5 Závitování

Závitování nožem (bez použití závitovacích cyklů) se programuje funkcí G33 a dále popsanými systémovými parametry. Ty jsou deklarovány v hlavičkovém souboru StdHdr.nch, který se standardně připojuje ke každému partprogramu. Uvedené systémové parametry je možné používat v libovolném partprogramu (viz příklady)

Pozn.:
Mnemotechnickému názvu odpovídá číslo systémového parametru (např. S306 je systémový parametr pro TPX – stoupání závitu v ose X). Při programování se ale tento způsob nepoužívá a proto z tohoto důvodu toto přiřazení neuvádíme.

Při programování řezání závitu nožem funkcí G33 sváže systém pohyb v souřadnici, pro níž bylo zadáno stoupaní, s pohybem vřetene. Pohyb ostatních interpolačních souřadnic je prováděn tak, aby byl výsledný pohyb v zadaných souřadnicích proveden po zadané trajektorii.

Seznam systémových parametrů pro závitování

TPX	Stoupání závitu ve směru osy X (Thread Pitch in X)
TPY	Stoupání závitu ve směru osy Y (Thread Pitch in Y)
TPZ	Stoupání závitu ve směru osy Z (Thread Pitch in Z)
TSPO	Úhlové posunutí začátku závitu pro vícechodé závity (Thread Starting Point Offset)
TRI	Délka nájezdu závitu (Thread Run-In length)
TRO	Délka výjezdu závitu (Thread Run-Out length)
TRIA	Úhel nájezdu závitu (Thread Run-In Angle)
TROA	Úhel výjezdu závitu (Thread Run-Out Angle)

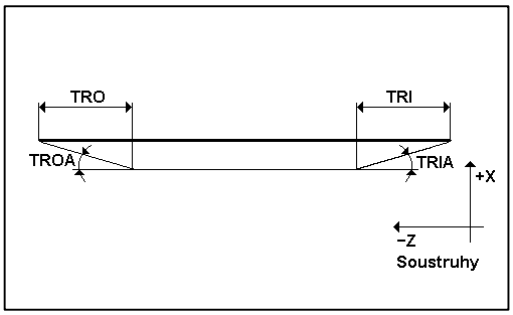
Systémové parametry pro závitování jsou obecné a proto jsou uvedeny pro všechny tři geometrické osy.
V následujících příkladech jsou uvedeny nejběžnější případy pro závitování na soustruhu, kde se stoupání udává ve směru osy vřetene, tj. v ose Z.

Příklad bloku se závitem bez výjezdu (předpokládají se otáčky):

```
N100  G33 TPZ=1.5  Z-50       „stoupání 1.5 mm
```

Funkce G33 (podobně jako G95 - otáčkový posuv mm/ot) sváže pohyb podle programovaných otáček. Oproti funkci G95 zajistí funkce G33 navíc „čekání“ na nulový pulz, kterým se závitování odstartuje. Tím je zajištěn definovaný začátek závitování, nutný pro opakovaný průchod závitem.

- TPX, TPY, TPZ = Stoupání závitu ve směru příslušné osy, udává dráhu osy na jednu otáčku. Rychlost posuvu je tedy závislá na otáčkách vřetena.
- TSPO = Úhlové posunutí začátku závitu. Pro jednochodý závit se nemusí uvádět. Pro dvouchodý závit je TSPO = 0 a TSPO = 180, tj.na jedné poloze osy X (u soustruhů) se provede nejprve závit s TSPO=0 a po návratu do Z-tového začátku závitu ve stejné poloze osy X se provede závit s TSPO=180. Podobně by se třikrát na stejné poloze X provedl závit tříchodý (TSPO = 0, TSPO = 120, TSPO = 240). Např. TSPO=240 znamená, že po nulpulzu se vřeteno otočí ještě o 240°, než se posuv rozjede.

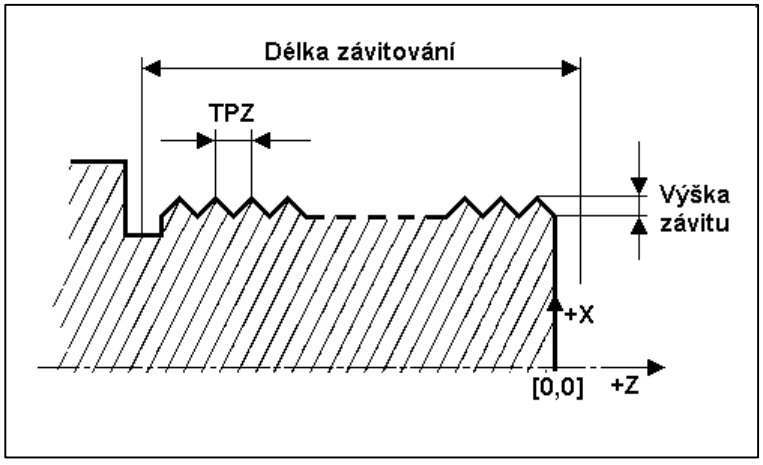


- TRO = délka výjezdu ze závitu – udává délku v příslušné ose (nejčastěji Z – soustruhy), kdy se závituje, ale současně se v druhé ose (X u soustruhů) „vyjíždí ze závitu pod úhlem, který je programován v systémovém parametru TROA. **Programovaná koncová poloha závitu (délka závitu) je v tomto případě včetně délky výjezdu!**
- TROA = úhel výjezdu ze závitu. Pod tímto úhlem se vyjíždí ze závitu po délce , která je programovaná v systémovém parametru TRO. Parametry TRO a TROA by měly být vzájemně smysluplné.
- TRI, TRIA = délka a úhel vjezdu do závitu – mají podobný význam jako u výjezdu, jejich použití ale není tak časté.

Jednochodý závit M10x1.5 bez výjezdu (do zápichu):

V následujícím příkladu je rozepsaný závit M10 se stoupáním 1.5mm. Délka závitu je programována tak, aby závit začínal před materiálem (např. 2mm) a končil v zápichu, ze kterého vyjede v ose X. Pro jednochodý závit bez výběhu je prakticky nutné ze systémových parametrů nastavit pouze stoupání, tj. parametr TPZ.

Pro závit M10x1.5 je výška závitu podle tabulek 0.92mm. Tato výška závitu se obrobí na několik průchodů. Na konečné hloubce se obvykle provedou ještě např. dva průchody kvůli začištění.



Ťloušťka třísky je proměnlivá tak, aby průřez odebíraného materiálu byl stejný (koeficient zmenšení, tj. kolikrát je následující tříska menší, je obvykle 0.8).

Pokud se nepoužije závitovaní cyklus, musíme X-ové souřadnice podle počtu třísek spočítat „ručně“

```
" Závit M10x1.5 – průměrové programování
```

```
" Výška závitu je 0.92
N1 PROGRAM
N10 X0 Z100 G00
N30 X12 Z2 M3 S300      " Nájezd do výchozí polohy, otáčky 300ot/min
"
N100 X9.459             " X-ová poloha 1. třísky
N110 G33 TPZ=1.5 Z-52   " Závitování 1. třísky
N120 X12 G00            " Výjezd v X do polohy pro přejezd zpět
N130 Z2                 " Návrat v Z do výchozí polohy
"
N200 X9.016             " X-ová poloha 2 třísky
N210 G33 TPZ=1.5 Z-52   " Závitování 2. třísky
N220 X12 G00            " Výjezd v X do polohy pro přejezd zpět
N230 Z2                 " Návrat v Z do výchozí polohy
"
N300 X8.664             " X-ová poloha 3. třísky
N310 G33 TPZ=1.5 Z-52   " Závitování 3. třísky
N320 X12 G00            " Výjezd v X do polohy pro přejezd zpět
N330 Z2                 " Návrat v Z do výchozí polohy
"
" ..... Další třísky
"
N400 Z100 G00 M5        " Odjezd do bezpečné polohy
N500 ENDPROGRAM
```

Jednochodý závit M10x1.5 s výjezdem:

Ten samý závit s výjezdem o délce 3.75mm a úhlem výjezdu 14° by se programoval podobně, pouze se přidají dva parametry definující délku a úhel výjezdu. Pro ilustraci je uveden pouze jeden průchod:

```
N1 PROGRAM
N10 X0 Z100 G00
N30 X12 Z2 M3 S300                          " Nájezd do výchozí polohy
"
N100 X9.459                                 " X-ová poloha 1. třísky
N110 G33 TPZ=1.5 TRO=3.75 TROA=14 Z-52      " Závitování 1. třísky s výběhem
N120 X12 G00                                " Výjezd v X
N130 Z2                                     " Návrat v Z
…
…
```

Dvouchodý závit M10x1.5 bez výjezdu:

```
N1 PROGRAM
N10 X0 Z100 G00
N30 X12 Z2 M3 S300            " Nájezd do výchozí polohy,
"
N100 X9.459                   " X-ová poloha 1. třísky
N110 G33 TSPO=0 TPZ=1.5 Z-52  " Závitování 1. třísky – první chod
N120 X12 G00                  " Výjezd v X do polohy pro přejezd zpět
N130 Z2                       " Návrat v Z do výchozí polohy
"
N101 X9.459                   " X-ová poloha 1. třísky
N111 G33 TSPO=180 TPZ=1.5 Z-52 " Závitování 1. třísky – druhý chod
N121 X12 G00                  " Výjezd v X do polohy pro přejezd zpět
N131 Z2                       " Návrat v Z do výchozí polohy
"
N200 X9.016                   " X-ová poloha 2 třísky
```

```
N210 G33 TSPO=0 TPZ=1.5 Z-52  " Závitování 2. třísky – první chod
N220 X12 G00                  " Výjezd v X do polohy pro přejezd zpět
N230 Z2                       " Návrat v Z do výchozí polohy
"
N201 X9.016                   " X-ová poloha 2 třísky
N211 G33 TSPO=180 TPZ=1.5 Z-52 " Závitování 2. třísky – druhý chod
N221 X12 G00                  " Výjezd v X do polohy pro přejezd zpět
N231 Z2                       " Návrat v Z do výchozí polohy
"
…… atd.
```

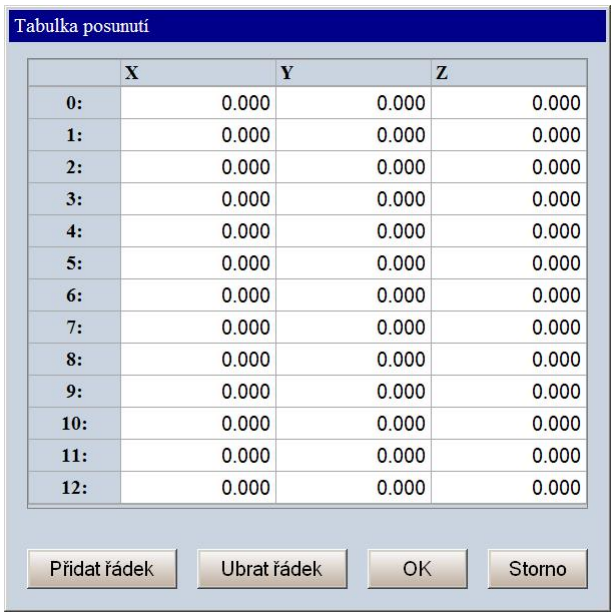
5

5. Posunutí počátku

Posunutí počátku – nulové body programu – umožňují programovat přenesení souřadného systému programu do libovolného bodu podle potřeb programátora. Při přesunutí souřadného systému (nulového bodu) se vztahují všechny další absolutní míry na nový nulový bod. Při inkrementálním programování nemá posunutí nulového bodu vliv na koncový bod programované dráhy. Při absolutním programování je programovaná koncová poloha dráhy automaticky přepočítána na novou koncovou polohu vztaženou k příslušnému nulovému bodu. Indikace absolutní polohy (okamžité polohy) je vztažena vždy vůči některému bodu stroje nebo programu (nulovému bodu programu). Vůči kterému bodu je indikace vztažena, je určeno G funkcí z páté skupiny.

5.1 Tabulka posunutí

Pokud se používá posunutí podle ISO, programované funkcemi G53 – G59 (toto programování je zachováno kvůli kompatibilitě se staršími systémy) nebo doporučeným novým způsob, který není omezen počtem posunutí (viz dále), je potřeba nejprve naplnit tabulku posunutí, která je k dispozici (obvykle) v menu TECHNOLOGIE a dále TABULKA POSUNUTÍ.



Tabulka je uvedena na obr. Je číslovaná od nuly, položky se do tabulky přidávají tlačítkem „Přidat řádek“ resp. „Ubrat řádek“ – pro vymazání položky. Pokud se programuje podle ISO funkcemi G53 – G59, je přiřazení následující:
G53 = 0
G54 = 1
G55 = 2
G56 = 3
G57 = 4
G58 = 5
G59 = 6

Doporučený způsob, který umožní využívat v podstatě libovolný počet posunutí (omezeno pouze konfigurací, resp. vymezenou pamětí) programuje se funkce:

G50 Lxxx,
kde xxx je číslo v tabulce posunutí.

Příklad:
```
N G50 L22   " Posunutí počátku, hodnota je v tabulce posunutí pod číslem 22

N G50 L0    " Výsledek stejný jako při programování G53 (v tabulce číslo 0)
N G50 L1    " Výsledek stejný jako při programování G54 (v tabulce číslo 1)
```
atd.

5.2 Posunutí pomocí systémových funkcí

Posunutí počátku lze programovat přímým zápisem pomocí funkcí pro programovou transformaci
TRANSLATE(X,Y,Z)
nebo funkcí pro aditivní posunutí
ATRANSLATE(X,Y,Z)

Je-li v tabulce posunutí např. pro osu X v řádce 12 hodnota 50.0, a pro osu Y hodnota 20.0, může se programovat:
```
N  G50 L12
```
Stejný výsledek dosáhneme pomocí funkcí pro programovou transformaci takto:
```
N  TRANSLATE (50,20,0)
```
Zrušení posunutí dosáhneme programováním nulových hodnot
```
N  TRANSLATE (0,0,0)
```

Stejným způsobem lze používat i programové transformace pro polotovar:
```
WTRANSLATE(X,Y,Z)
```
nebo funkcí pro aditivní posunutí polotovaru
```
WATRANSLATE(X,Y,Z)
```

6

6. ZADÁNÍ POSUVU

Pracovní posuv po obráběné křivce se programuje podle ISO pod adresou F nebo systémovým parametrem FEED (pro milimetrový posuv) resp REVFEED (pro otáčkový posuv). Posuv je zadáván způsobem podle G-funkce skupiny 6, nebo-li tato funkce určuje rozměr adresy F.

Posuv		G - funkce	Rozměr rychlosti F nebo FEED/REVFEED
Milimetrový	mm/min	G94	mm_min
	mm/min s KŘR	G97	mm/min
Otáčkový	mm/ot	G95	mm/ot
	mm/ot s KŘR	G96	mm/ot

Pozn.: KŘR = konstantní řezná rychlost
Příklady:

```
" Milimetrové posuvy:
" Rychlost programována podle ISO adresou F:
N  G94  F1000          "rychlost posuvu je  1000 mm/min.
N  G94  F15000         "rychlost posuvu je  15000 mm/min.
N  G94  F1.5           "rychlost posuvu je  1.5 mm/min.
N  G94  F0.1           "rychlost posuvu je  0.1 mm/min.

" Rychlost programována systémovým parametrem FEED:
N  G94  FEED=1000      "rychlost posuvu je  1000 mm/min.
N  G94  FEED=15000     "rychlost posuvu je  15000 mm/min.
N  G94  FEED=1.5       "rychlost posuvu je  1.5 mm/min.
N  G94  FEED=0.1       "rychlost posuvu je  0.1 mm/min.

" Otáčkové posuvy:
" Rychlost programována podle ISO adresou F:
N  G95  F10            "rychlost posuvu je  10 mm/ot.
N  G95  F1.5           "rychlost posuvu je  1.5 mm/ot.
N  G95  F0.1           "rychlost posuvu je  0.1 mm/ot.
N  G95  F0.05          "rychlost posuvu je  0.05 mm/ot.

" Rychlost programována systémovým parametrem REVFEED:
N  G95  REVFEED=10     "rychlost posuvu je  10 mm/ot.
N  G95  REVFEED=1.5    "rychlost posuvu je  1.5 mm/ot.
N  G95  REVFEED=0.1    "rychlost posuvu je  0.1 mm/ot.
N  G95  REVFEED=0.05   "rychlost posuvu je  0.05 mm/ot.
```

Pozn.:
Stejné rozměry jsou i pro milimetrový nebo otáčkový posuv s konstantní řeznou rychlostí

6.1 Konstantní řezná rychlost

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 v [mm] (korigována vzhledem ke korekcím a posunutí)
v = obvodová rychlost v [m/min] (KŘR)
n = otáčky vřetene [ot/min]

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

6.2 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 CCS=90 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).

7

7. KOREKCE NÁSTROJE

Korekce nástroje umožňují vytvořit obecný partprogram, který je použitelný pro různé průměry a délky nástrojů (fréz, soustružnických nožů apod.). Při výměně nástroje s jinými rozměry se upraví pouze příslušná korekce a partprogram zůstane beze změny.

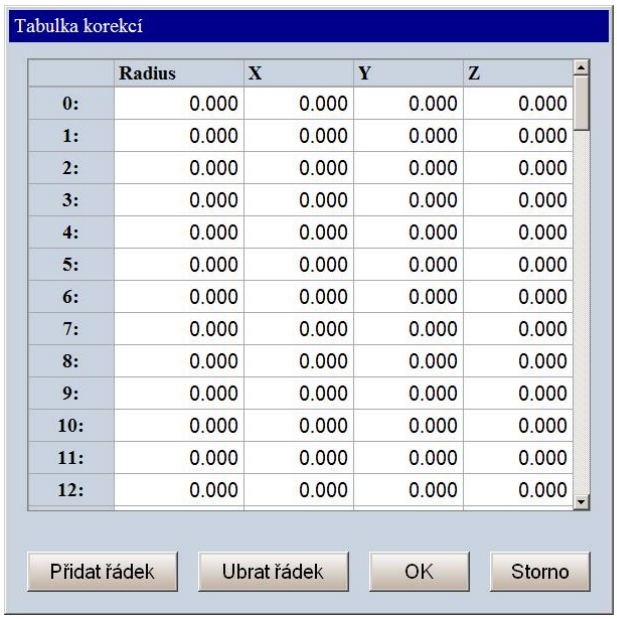
Korekce rozeznáváme dvojího druhu:

Korekce na poloměr nástroje - je určena G-funkcí ze 3. skupiny (G41,G42 a G40)
Korekce délkové

7.1 Tabulka korekcí

Jedna z možností, jak zadat hodnoty korekcí (poloměrové i délkových) je zápis do tabulky korekcí. Tabulka korekcí je v systému dostupná (obvykle) přes tlačítko TECHNOLOGIE a zde pak TABULKA KOREKCÍ.

Uvádíme příklad takovéto tabulky (s nulovými hodnotami – viz obr.), neboť obecně může mít jiný vzhled.



Tabulka obsahuje hodnoty poloměrové korekce (sloupec „Radius“) a hodnoty délkových korekcí pro geometrické osy X, Y, a Z.

Tlačítkem „Přidat řádek“ se přidá další položka na konec tabulky, tlačítkem „Ubrat řádek“ se odebere jedna položka, tj. jeden řádek od konce tabulky.

Počet řádek tabulky není omezen, resp. je omezen pouze pamětí systému. Tabulka začíná řádkem 0

Pokud je v programu odkaz na číslo řádky, resp. korekce, která v tabulce neexistuje, systém ohlásí chybu „Neplatné číslo řádku“

Způsob vyvolání korekce je uveden dále.

7.2 Poloměrové korekce s ekvidistantou

Poloměrové korekce umožňují programovat obrys obrobku neboli tzv. výkresové hodnoty bez ohledu na to, jaký nástroj bude ve svém výsledku k obrobení použit. Korigovaná dráha (dráha středu nástroje u frézek, dráha

středu poloměru břitu u soustruhů) leží na ekvidistantě. Při nespojitosti dvou po sobě následujících bloků záleží na konfiguraci, jakým způsobem systém tuto záležitost vyřeší. Systém může např. dojíždět na průsečík ekvidistant nebo vloží oblouk. Může záležet i na úhlu tečen, zda se pojede na průsečík ekvidistant nebo se vloží oblouk.

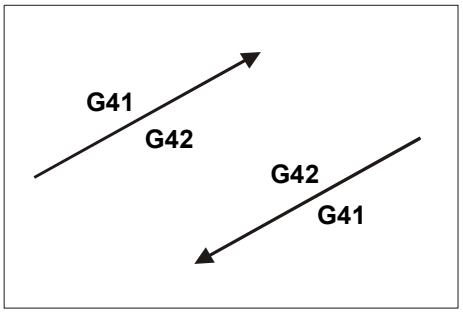
7.2.1 Pravidla pro programování poloměrové korekce:

Pro řízení poloměrových korekcí je nutné stanovit korekční rovinu. Ta je shodná s interpolačními rovinami, které byly uvedeny v kapitole 4 a programuje se G- funkcí ze skupiny .

G17	Rovina X – Y
G18	Rovina Z – X
G19	Rovina Y – Z

Pokud se používají poloměrové korekce pouze v rovině X – Y, je korekční (a interpolační) rovina G17 obvykle standardně zapsána v prioritním bloku a nemusí se proto programovat. Pouze pokud se využívají i jiné roviny, je třeba na zvolení správné roviny při zápisu partprogramu pamatovat

Poloměrové korekce se programují G-funkcemi ze skupiny 3:
- G41 - poloměrová korekce vlevo
- G42 - poloměrová korekce vpravo
- G40 - zrušení poloměrové korekce



G41 je korekce vlevo, tj. střed nástroje se pohybuje (pokud je poloměrová korekce kladné číslo) vlevo od programované dráhy ve směru pohybu.

G42 je korekce vpravo, tj. střed nástroje se pohybuje (pokud je poloměrová korekce kladné číslo) vpravo od programované dráhy ve směru pohybu. (viz obr.)

Korekce G41/G42 je trvalá funkce a platí až do odvolání funkcí G40. Korekce je platná pro zvolenou korekční rovinu. Pokud se korekční rovina explicitně nezvolí, platí prioritní korekční rovina, tj rovina, která je zadaná v prioritním bloku (obvykle G17). Eventuelní změna korekční roviny nesmí být provedena při zařazené poloměrové korekci.

Korekce může trvat i v blocích bez pohybu v korekční rovině, neboli systém překlene blok bez pohybu v korekční rovině a pohyb poté pokračuje, jakoby nepohybové bloky nebo pohyb v jiné než korekční rovině nebyly programované.

Pokud by hodnota korekce bylo záporné číslo, prohodily by se strany korekcí, tj. G41 se zápornou korekcí je to samé jako G42 s kladnou korekcí.

Poloměrová korekce je účinná (tj. zařadí se) na konci bloku, ve kterém byla vyvolána funkcí G41 nebo G42, t.j. začne platit v následujícím bloku. Vyvolání (zařazení funkcí G41 nebo G42) a odvolání (vyřazení funkcí G40) poloměrové korekce je povoleno pouze v **pohybovém bloku,** tj. v bloku, kde je **programován pohyb,** přičemž ve skutečnosti nemusí nutně pohyb bez korekce vůbec nastat - například jsou zopakovány ty samé souřadnice jako v předešlém bloku při absolutním programování, nebo je programována přírůstkově nula (např. blok N G91 X0 Y0)**.**

Nelze zařazovat a vyřazovat poloměrovou korekci na kružnici. Pokud by to bylo z nějakých důvodů nutné, musí se předřadit blok s lineární interpolací, kde je programovaná alespoň jedna osa, ale s nulovým pohybem.

7.2.2 Systémové parametry pro poloměrové korekce

Pro řízení poloměrových korekcí se používají tyto systémové parametry:

TOOLRADIUS

EQDOFFS
RCANGLE
RCMETHOD
RCCHANGEMETHOD
RCOPTIONS

TOOLRADIUS

Systémový parametr (typ REAL) pro zadání hodnoty poloměrové korekce. Je to alternativní možnost zadat hodnotu poloměrové korekce z partprogramu.

Příklad:
```
N  X200 Y100 G41 TOOLRADIUS=50
```

V bloku se zařadí korekce vlevo s poloměrem nástroje 50mm

EQDOFFS

Systémový parametr – offset pro jízdu po ekvidistantě. Druhá složka poloměrové korekce, která se přičte k zadané poloměrové korekci (TOOLRADIUS). Jedna z možností využití tohoto parametru je například použít ho jako „přídavek na čisto“. Pro hrubování i pro frézování načisto se programují stejné výkresové hodnoty. Nejprve se dráha projede s korekcí a nastavenou hodnotou EQDOFFS na hodnotu velikosti přídavku. Pro obrábění načisto se pouze EQDOFFS vynuluje (a např. se zmenší posuv apod.)

Příklad:
```
N PROGRAM
N G90 G0  X0  Y0  G40
N G1 F200 X50 Y50 G41 TOOLRADIUS=30 EQDOFFS=1  " Hrubování (přídavek 1mm)
N         X200
N         X250
N ...
N ...
N ...
N G90 G0  X0  Y0  G40
N G1 F100 X50 Y50 G41 TOOLRADIUS=30 EQDOFFS=0  " Obrábění načisto
N         X200
N         X250
N ...
N ...
N ...
N ENDPROGRAM
```

Pro obrobení načisto by se mohl samozřejmě použít i nástroj s jiným poloměrem (TOOLRADIUS), ale EQDOFFS by se také nastavil na nulu. Výsledek by byl stejný.

RCANGLE

Systémový parametr (typ REAL) pro zadání mezního úhlu, od kterého se budou vkládat oblouky. Je-li úhel tečen mezi následujícími bloky **větší nebo roven** meznímu úhlu, vloží se oblouk. V opačném případě se jede na průsečíky ekvidistant.

Mezní úhel se zadává buď v **radiánech** nebo ve **stupních –** záleží na tom, jak je nastaven systémový parametr ANGLEUNIT.
ANGLEUNIT = RADIANS …… Mezní úhel se zadává v radiánech
ANGLEUNIT = DEGREES …… Mezní úhel se zadává ve stupních

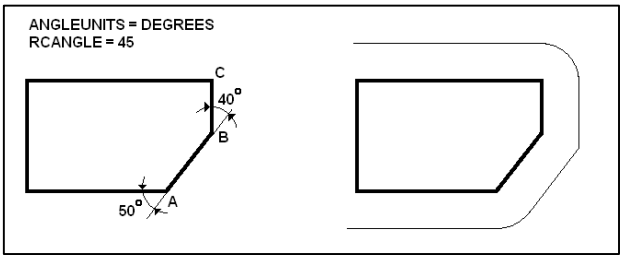
Systémový parametr ANGLEUNIT je obvykle nastaven v prioritním bloku, pokud není zadán, je defaultně přednastaven na stupně.

Pozn: Pro zadání radiánů se může použít systémová konstanta PI, jejíž hodnota je 3.141… a případně i aritmetické operace. Následující tabulka udává příklad několik hodnot mezních úhlů v radiánech:

Úhel [°]	*RCANGLE [rad]*
10°	*0.175*
30°	*0.524*
45°	*PI / 4*
60°	*1.047*
90°	*PI / 2*

Pokud je systémový parametr RCANGLE nastaven v **prioritním bloku**, např. na hodnotu PI/2, t.j. 90°, znamená to, že se budou oblouky vkládat při pravoúhlé dráze a větších úhlech a v partprogramu se nemusí nic programovat.

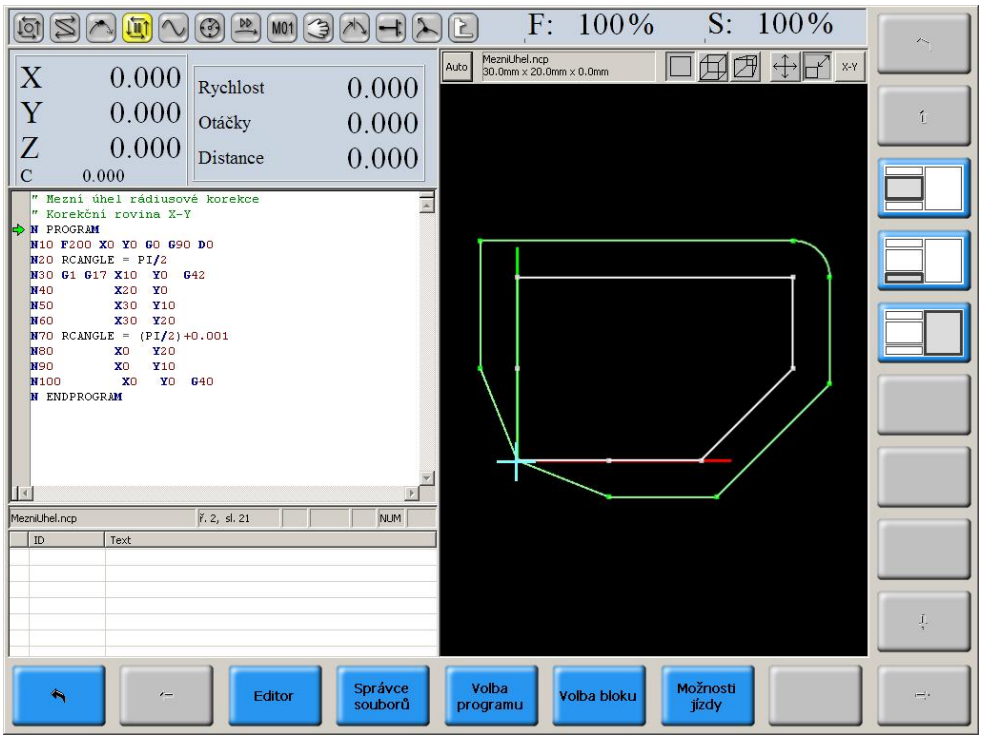
Příklad:



Pokud je nastaven RCANGLE na hodnotu 45^o , vloží se oblouk v bodě A, kde je úhel tečen 50^o a v bodě C, kde je úhel tečen 90^o. V bodě B je úhel 40^o t.j je menší než mezní úhel a tudíž se zde oblouk nevloží.

Pokud by v určitých případech bylo nutné mezní úhel měnit na jiné hodnoty než je v prioritním bloku, může se změna provést přímo v partprogramu naprogramováním jiné hodnoty RCANGLE. Změna bude platit pouze do té doby, dokud se nevykoná prioritní blok, což je např. při nové volbě programu nebo centrální anulaci, nebo se znovu nenastaví RCANGLE.

Na následujícím obrázku je jednoduchý partprogram, kde je použito nastavení mezního úhlu. V bloku N20 je nastaven mezní úhel na hodnotu PI/2, tj. 90°. Mezi bloky N60 a N80 je pravý úhel, mezní úhel je rovněž 90°, tj. vloží se oblouk. V bloku N70 je sice změna RCANGLE, ale výpočet, zda vložit oblouk za blokem N60 se provedl ještě s nastavením RCANGLE na PI/2. Změna RCANGLE = (PI/2)+0.001, se uplatní až v následujícím rohu, tj. za blokem N80. Ačkoli se RCANGLE změnil pouze o 0.001 radiánu, což jsou řádově setiny stupně, už je již mezní úhel větší než 90° a oblouk se tudíž nevloží.



RCMETHOD a RCCHANGEMETHOD

Systémové parametry RCMETHOD a RCCHANGEMETHOD určují způsoby řešení poloměrové korekce. RCMETHOD se použije pro nastavení řešení, pokud se parametry korekce nemění. RCCHANGEMETHOD se použije pro nastavení řešení při řazení, rušení a změně korekce.

Pro nastavení se používají tyto systémové konstanty, nebo ekvivalentně hodnoty 0 nebo 1:
0 = RCMETHOD_NORM – jízda na kolmici
1 = RCMETHOD_KONT – jízda na průsečík nebo vložení oblouku

Jízda na kolmici (RCMETHOD= RCMETHOD_NORM) znamená, že každý blok se zařazenou korekcí pojede na kolmici k začátku bloku. Při běžných způsobech práce se tento způsob nepoužívá. Za normální situace se používá nastavení RCMETHOD= RCMETHOD_KONT, t.j. systém jede na průsečíky ekvidistant, případně vkládá oblouky podle nastavení mezního úhlu.

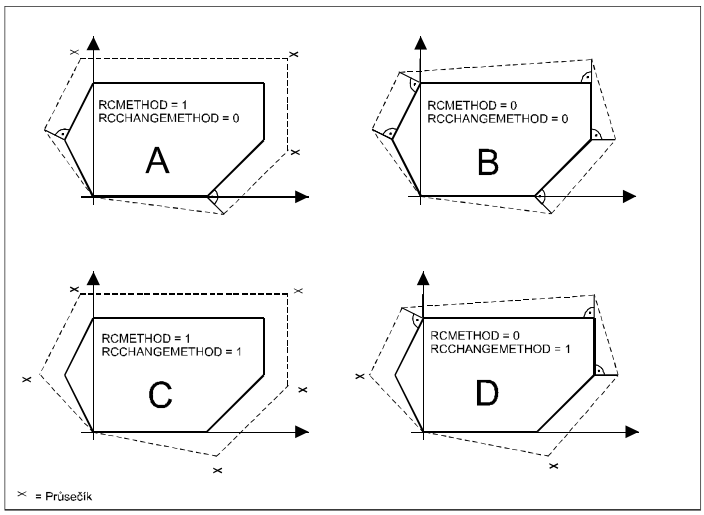
Jiná je situace při řazení, rušení a změně korekce. Tam je naopak ve většině případů požadován stav, aby se korekce zařadila na kolmici, t.j RCCHANGEMETHOD= RCMETHOD_NORM. Pokud by se nastavilo RCCHANGEMETHOD= RCMETHOD_KONT, zařadila by se korekce na průsečíku ekvidistant.

Jak se tato nastavení projeví, ilustrují následující obrázky, pro které byl použit partprogram, ve kterém se postupně nastavovaly parametry RCMETHOD a RCCHANGEMETHOD:

```
N PROGRAM
N10 RCANGLE=PI
N20 RCMETHOD=1          " RCMETHOD_KONT = 1
N30 RCCHANGEMETHOD=0    " RCMETHOD_NORM = 0
"
N40 G0 G90 X0 Y0     F200
N50 G1 G17 X20   Y0  G42 TOOLRADIUS=4
N60          X30   Y10
N70          X30   Y20
N80          X0    Y20
N90          X-5   Y10
N100         X0    Y0  G40
```

```
N ENDPROGRAM
```

Ve velké většině praktického použití se parametry nastavují podle obr. A, tj. zařazování a vyřazování, eventuelně změna korekce jezdí na kolmici (blok N50 a N100) , ale pokud se parametry korekce nemění, tj. korekce je trvale zařazena, jezdí se na průsečík ekvidistant, eventuelně se vkládají oblouky (ostatní bloky s korekcí). Ostatní případy jsou speciální. Vzhledem k tomu, že nastavení těchto parametrů není nutné měnit, jsou obvykle přednastaveny v prioritním bloku Křížkem je na obrázku označen stav jízdy na průsečík ekvidistant.

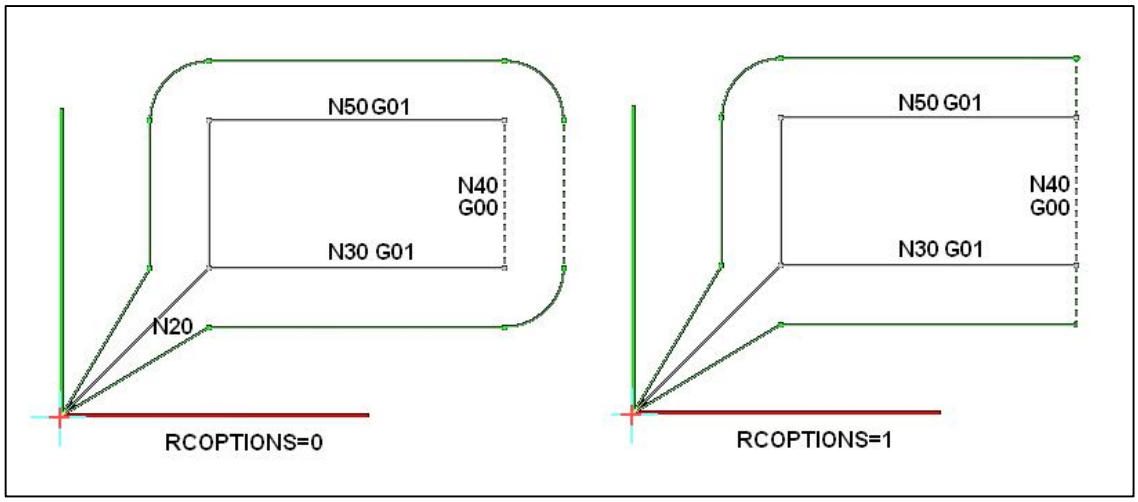


RCOPTIONS

Pomocné příznaky pro práci s poloměrovými korekcemi:

Bit	Hodnota[0,1]	Popis
0	1	Pro první a poslední blok rychloposuvem za resp. před blokem pracovním posuvem se nastaví příznak změny parametrů poloměrové korekce
1	1	Zakázat vkládání oblouků v případě, že úhel je menší než limit, ale průsečík ekvidistant neexistuje

Jak se projeví nastavení RCOPTIONS (Bit 0) vyplývá z následujícího obrázku. Obdélník se projíždí se zařazenou korekcí. Blok N30 a N50 jede pracovním posuvem, blok N40 rychloposuvem. Je-li RCOPTIONS=1 (v bitu 0), uplatní se pravidlo, že poslední blok před rychloposuvem nastaví příznak změny parametrů poloměrové korekce, t.j. v tomto případě jakoby v bloku N30 bylo vyřazení korekce a tudíž se dojede na kolmici A v následujícím bloku za rychloposuvem se jakoby korekce znovu zařadí, t.j. začne se na kolmici.
Je-li RCOPTIONS=0, je korekce trvale zařazena i v bloku s rychloposuvem.



Pozn.:
Parametr RCOPTIONS = 0 se používá u frézek

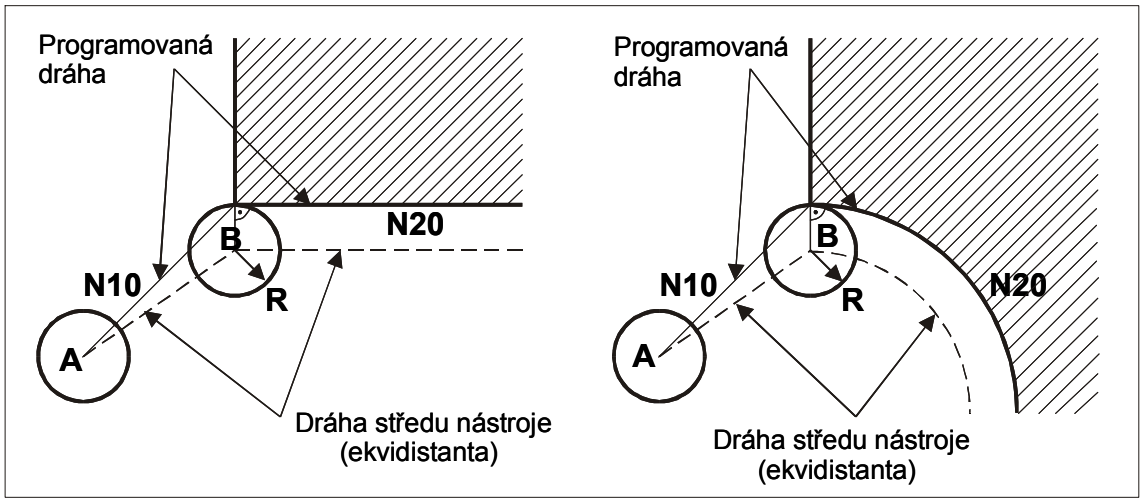
Parametr RCOPTIONS = 1 se používá převážně u řezacích strojů a jeho účelem je najet nebo dojet na kolmici, jako by se korekce zařazovala, resp. vyřazovala před blokem s rychloposuvem. Je to z toho důvodu, že se často programují dráhy, které se následně rychloposuvem vracejí zpět. Pokud by korekce jela na průsečík ekvidistant, mohlo by v určitém případě dojít k nevhodnému vypnutí technologie.

Bit 1 se ponechává (u frézek a soustruhů) většinou nastavený na 0, tj. povoleno vložení oblouků při neexistujícím průsečíku ekvidistant.

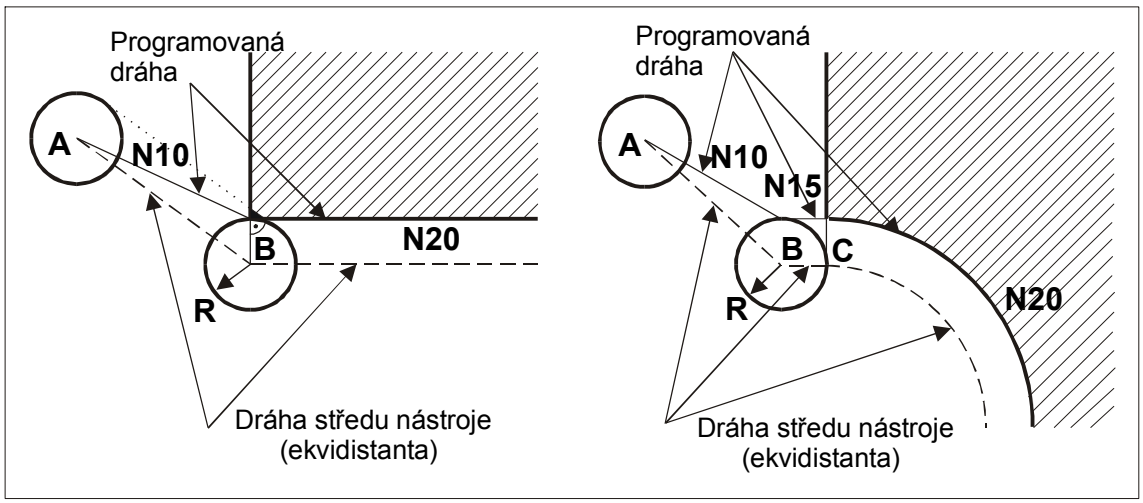
7.2.3 Řazení poloměrové korekce a průběh korekce

Následující pravidla při zařazování poloměrové korekce platí pro nastavení:
RCMETHOD = 1
RCCHANGEMETHOD = 0

Korekci lze zařadit pouze při lineární interpolaci v bloku, kde je programován pohyb (skutečný pohyb nemusí nastat). Dráha středu nástroje se pohybuje z počátečního bodu bloku N na kolmici k dráze bloku N+1, t.j. blok N+1 se celý provede již se zařazenou korekcí. Na obrázku je v bloku N10 zařazena poloměrová korekce G42, korigovaná dráha je znázorněna čárkovaně z bodu A do bodu B. Bod B, na který najíždí, leží na kolmici k dráze bloku N20. Stejným způsobem se určí bod pro nasazení korekce v případě, že následuje kružnice, jak je uvedeno na tomtéž obrázku vpravo. Bod B rovněž leží na kolmici k tečně k programované kružnici.



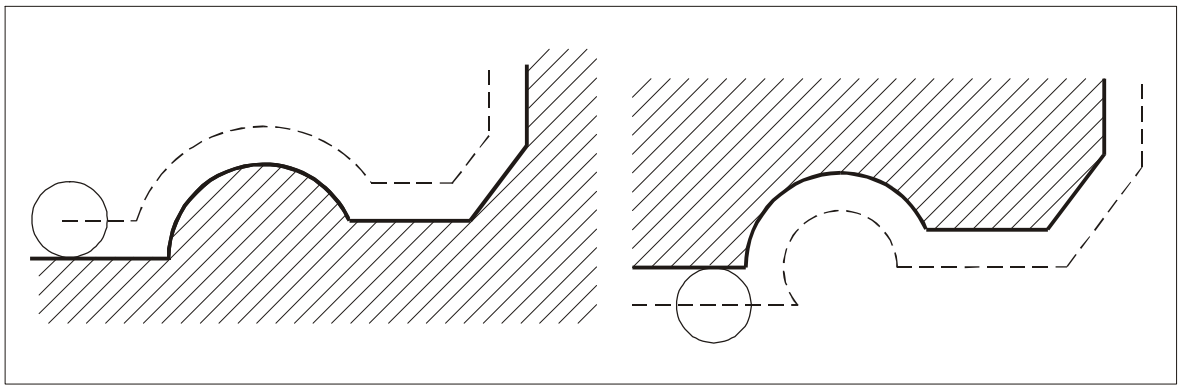
Při zařazování korekce je třeba zvolit správný směr nájezdu. Na následujícím obrázku vlevo je uveden chybný úhel nájezdu. Při tomto směru nájezdu na korekci by došlo k chybnému obrobení rohu, neboť obrys nástroje protne obrobek předtím, než najede na kolmici k dalšímu bodu.



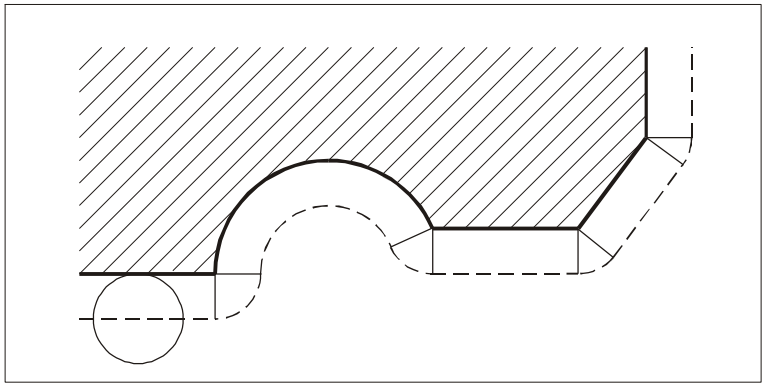
Vhodnější je tedy nasadit korekci ještě před materiálem, jak je uvedeno na obrázku vpravo. Pokud se korekce nasadí ještě před obrobkem, což je obvyklý případ, tak na úhlu nájezdu nezáleží. Nasazení korekce skončí

v bodu B, do bodu C (blok N15) již program jede se zařazenou korekcí.

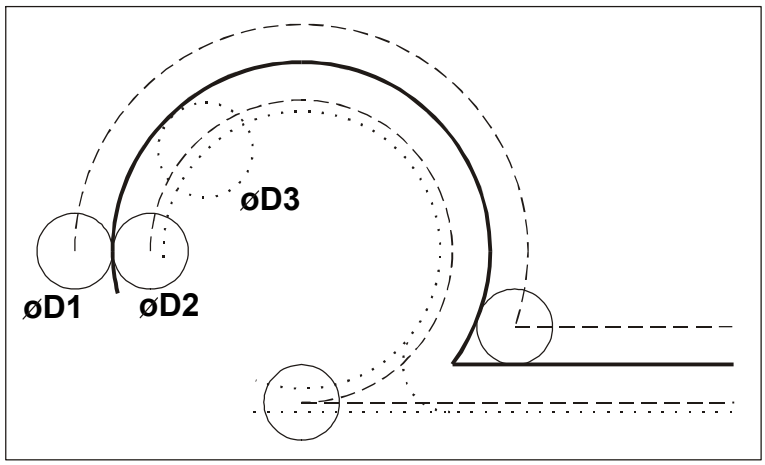
Je-li korekce zařazena, pohybuje se střed nástroje po ekvidistantách (označeny čárkovaně) vzdálených o poloměr nástroje od programované dráhy. Na následujících obrázcích jsou uvedeny příklady korigované dráhy pro korekci vlevo i vpravo.



Oblouky se vloží podle nastavení parametru RCANGLE, na následujícím obrázku je RCANGLE=PI/4, tj. oblouky se vkládají, pokud je úhel 45 stupňů a větší.



Pokud jezdí střed nástroje na průsečíky ekvidistant, musí se při programování dbát, aby vždy existoval průsečík ekvidistant. Standardně ale bývá nastaveno RCOPTIONS bit1=0, tj. neprotnou-li se ekvidistanty dvou po sobě následujících bloků, vloží se oblouk. Pouze pokud bychom chtěli jezdit zásadně vždy na průsečík ekvidistant, nastavilo by RCOPTIONS bit1=1. Pokud by v tomto případě neexistoval průsečík ekvidistant, systém by zahlásil chybu „průsečík ekvidistant neexistuje“. Tento případ může nastat při nevhodné kombinaci velikosti korekce a úhlu tečen v bodě, kde se protíná dráha dvou po sobě následujících bloků. Na obr. je uveden příklad při programování kružnice a přímky.

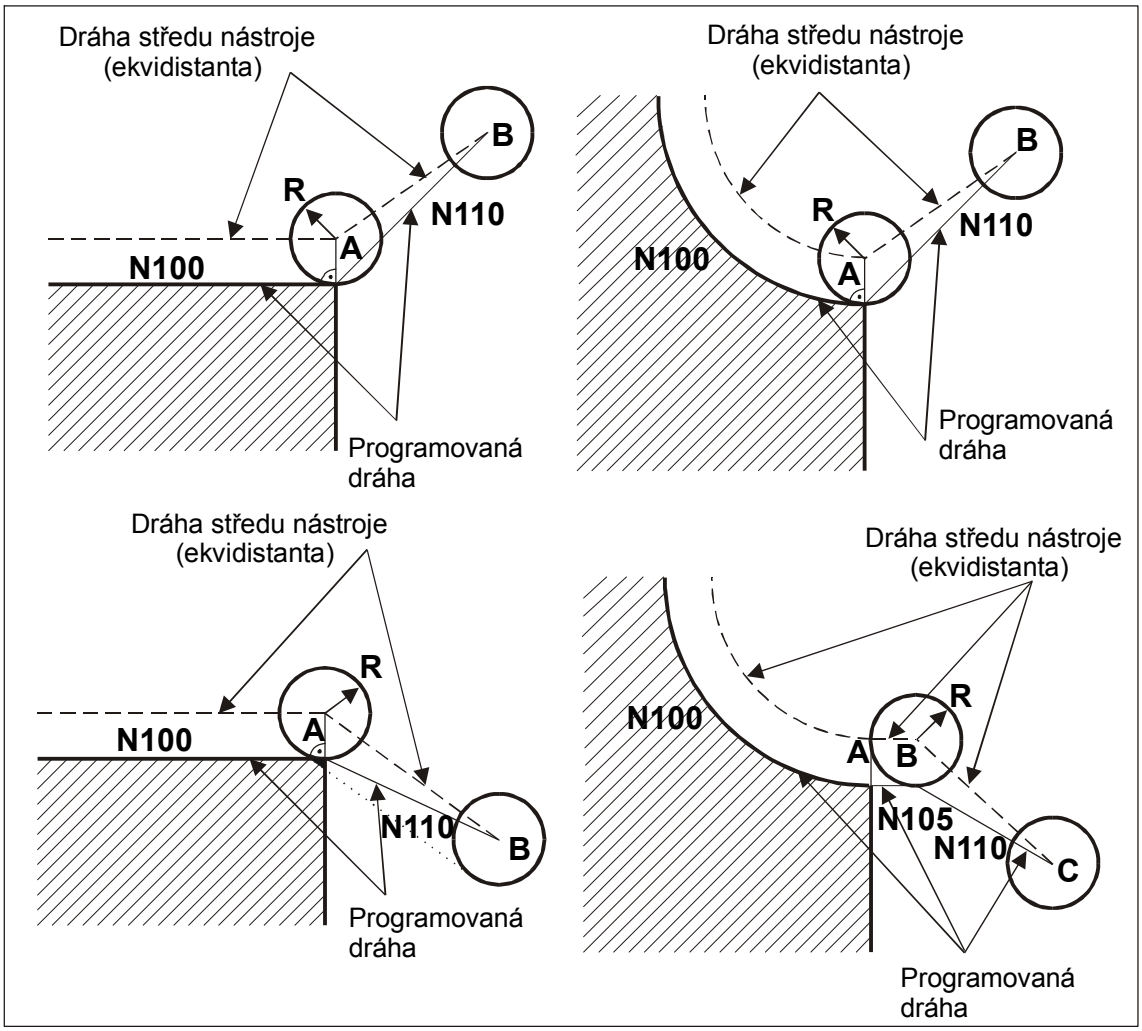


Při korekci vlevo G41 – nástroj o průměru D1 - nenastanou žádné problémy, ekvidistanty se vždy protnou. Při korekci vpravo G42 a nástroji o průměru D2 se ekvidistanty rovněž protnou i když nástroj již zajíždí relativně daleko. Při korekci vpravo G42 a průměru nástroje D3 se ovšem ekvidistanty již neprotnou (dráha na obr. je vyznačena tečkovaně) a systém hlásí uvedenou chybu, pokud by bylo RCOPTIONS nastaveno, jak bylo uvedeno výše, jinak se vloží oblouk.
Pozn.:
V některých případech (pokud se nevkládají oblouky) je sice nalezen průsečík ekvidistant, ale leží daleko. Obvykle se jedná o technologicky nevhodné případy, které se v praxi nevyskytují. Pokud se vyskytnou, je třeba zvolit vhodnější způsob programování

.

7.2.4 Vyřazení poloměrové korekce

Odvolání poloměrové korekce se provede funkcí G40 a je možné, stejně jako řazení G41,G42, pouze při lineární interpolaci, resp. v bloku, kde je programován pohyb, i když opět fyzicky nemusí (při jízdě bez korekce), nastat. Koncový korigovaný bod posledního bloku před G40 leží na kolmici k tečně, která prochází koncovým programovaným. Odvolání korekce je uvedeno na obr. Stejně jako u zařazování korekce se musí dbát na směr bloku, ve kterém je programována funkce G40. Neplatí to pro případ, že korekci odvoláváme již mimo obrobek.



Na obr. je odvolání korekce G40 programováno v pohybovém bloku (tj. je programován pohyb alespoň v jedné z os korekční roviny).

7.3 Délkové korekce

7.3.1 Korekce na délku nástroje

Hodnoty délkové korekce mohou být zadané v tabulce, nebo je možné je programovat v partprogramu přímým zápisem. Tabulka pro délkové korekce je stejná jako pro poloměrové korekce a je popsána v kapitole 7.1,

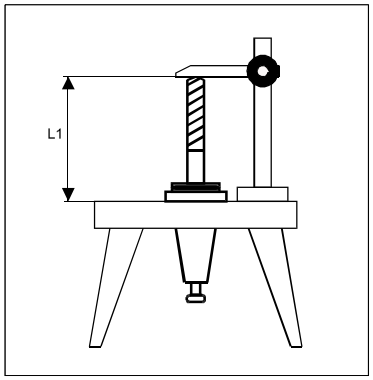
Přímé programování délkové korekce se provádí systémovými parametry:

TOOLLENX – vektor délkové korekce pro osu X
TOOLLENY – vektor délkové korekce pro osu Y
TOOLLENZ – vektor délkové korekce pro osu Z

Příklad:

```
N10 TOOLLENZ = 145              " zařadí délkovou korekci v ose Z 145 mm
```

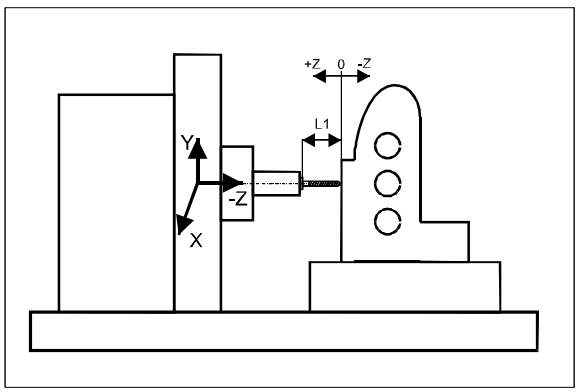
Jednou z možností využívat délkové korekce z tabulky uvedeme na následujícím příkladu. Na některých typech strojů se často používá způsob, že se stanoví „nuly“ na obrobku přímo na stroji, např. „škrtnutím“ nástroje o obrobek nebo pomocí různých trnů či měrek.



Předpokládejme tři různé nástroje, např. dva různé vrtáky a frézu. V měřicím zařízení změříme délky nástrojů (viz obr.).

První vrták má délku L1 - např. 180 mm
Druhý vrták má délku L2 – např. 250 mm
Fréza má délku L3 – např. 125 mm

Změřené délky nástrojů zadáme do tabulky korekcí pro osu Z, počínaje korekcí číslo 1 (korekce číslo 0 se ponechává celá nulová), do které zadáme hodnotu 180, do korekce číslo 2 pro osu Z zapíšeme 250 a do korekce číslo 3 zapíšeme 125 (viz obr. v kapitole 7.1)



Nyní se provede nastavení nuly v ose Z na stroji (viz obr.) . Nastavení se provede s vypnutou délkovou korekcí v ose Z, např. v režimu RUP do políčka TECHNOLOGIE zapíšeme D0 a stiskneme STRAT. Nulová délková korekce může být rovněž zadána v prioritním bloku, pak stačí odstartovat CENTRÁLNÍ ANULACI.

V ručním režimu se najede vrtákem na dotyk do místa, kde chceme mít nulu v ose Z. V menu RUČNÍ se nachází tlačítko (pokud je nakonfigurováno) NASTAVENÍ SOUŘADNIC (Nastavení Polohy). Zaškrtneme osu Z a do políčka napíšeme 180, což je délka L1 nástroje číslo jedna a potvrdíme tlačítkem OK. Na indikaci se v ose Z zobrazí hodnota 180. Stále máme vyřazenou, resp. nulovou délkovou korekci v ose Z.
Pokud nyní, např. v režimu RUP, zvolíme korekci D1, změní se indikace na nulu, což je stav, kterého jsme chtěli docílit.

Podobně pokud bychom „škrtali nulu“ frézou (nástroj číslo 3), zapsali bychom do políčka hodnotu 125 a po potvrzení by se indikovala v ose Z poloha 125. Po zvolení korekce D3 by se indikovala 0

Nastavení nuly je uložené v souboru, tj. je zachováno i po vypnutí stroje.

8

8. TECHNOLOGICKÉ FUNKCE

Význam některých funkcí uvedených v této kapitole může být modifikován tvůrcem interfejsu pro daný stroj. Přesný význam a seznam použitých M-funkcí musí dodat tvůrce interfejsu stroje.

8.1 Časová prodleva

Časovou prodlevu je možné programovat systémovou funkcí DELAY(t), kde t je čas v sekundách nebo podle ISO funkcí G04 ze skupiny 10 a adresou F, která v tomto případě neznamená rychlost, ale čas v sekundách!

Příklad:

```
N  DELAY(1)            " časová prodleva 1 sec.
N  DELAY(3.5)          " časová prodleva 3,5 sec.
N  DELAY(0.2)          " časová prodleva 0.2 sec.
N  DELAY(20)           " časová prodleva 20 sec.
```

Nebo ty samé hodnoty programované podle ISO

```
N  G04 F1          " časová prodleva 1 sec.
N  G04 F3.5        " časová prodleva 3,5 sec.
N  G04 F0.2        " časová prodleva 0.2 sec.
N  G04 F20         " časová prodleva 20 sec.
```

8.2 Otáčky vřetene

Otáčky vřetene se programují adresou S. Velikost otáček je reálné číslo. Rozsah je daný konfigurací stroje.

Příklad :

```
N10   S55.8        " 55,8 ot/min
N20   S0.5         " 0,5  ot/min
```

Zařazení otáčkového rozsahu převodového stupně je podle doporučení programováno funkcemi M41, M42, M43, M44.

Start otáčení ve směru CW je zadán funkcí M3, start ve směru CCW je zadán funkcí M4. Po odstartování pohybu těmito funkcemi trvá otáčení tak dlouho, dokud není programována funkce M5 (STOP vřetene) nebo M19 (STOP vřetene v orientovaném bodě).

8.3 Chlazení nástroje

Pro ovládání chlazení jsou v systému vyčleněny 2 skupiny M funkcí 5 a 6, které mohou ovládat 4 samostatné chladicí okruhy. Čísla M funkcí jsou vedena v souhrnném přehledu adres návrháře PLC.

8.4 Přerušení a konec partprogramu

V systému lze programovat čtyři M-funkce (M00,M01,M02,M30), kterými lze ukončit nebo přerušit zpracovávaný partprogram:

M0	Nepodmíněný stop Zpracovávaný partprogram program se přeruší po vykonání všech operací v bloku. Jaké funkce se provedou musí být součástí návodu PLC
M1	Podmíněný stop Systém se zachová shodně jako při M00 v případě, že pracuje v modifikaci M01 (možnosti jízdy)
M2	Konec partprogramu
M30	Konec partprogramu s „převinutím“. Odjetý partrprogram se automaticky zvolí znovu a stiskem START je možné jej ihned spustit. *Pozn: tuto vlastnost nutno povolit v registrech nastavením **EnableRewind** = 1(nastavení zajistí návrhář PLC)*

Pozn.:
Tvůrce interfejsu pro konkrétní stroj může chování těchto funkcí do jisté míry modifikovat.

8.5 Upnutí a uvolnění obrobku

Funkce jsou v režii PLC.

8.6 Výměna nástroje a obrobku

Funkce jsou v režii PLC.

OBSAH