AC servopohony Hitachi

Programovací software pro serii AD

Programovatelné funkce

Uživatelská příručka

Blahopřejeme Vám k pořízení HITACHI střídavého servopohonu..
Tato příručka popisuje užívání programovatelných funkcí střídavého servopohonu serie AD_3. Prosím prostudujte pozorně tuto příručku, spolu se standardní uživatelskou příručkou pohonu aby jste při porovozu efektivně využili všech předností a programových možností servopohonu HITACHI serie AD.

Tato příručka je určena pro koncového uživatele.

Tuto příručku uschovejte pro případné další nahlédnutí.

HITACHI

ADPr01601BX

■ DŮLEŽITÉ

Uložte tuto příručku tak, aby byla přístupná všem pracovníkům pracujícím, nebo jinak využívajícím servopohony HITACHI serie AD s programovatelnými funkcemi.

Před započetím práce s pohonem s programovatelnými funkcemi pečlivě prostudujte tuto příručku společně se základní uživatelskou příručkou pohonu serie AD. Používejte software v souladu se zásadami bezpečného provozu, možnostmi stroje, provozními předpisy a ostatními zásadami užití uvedenými v dokumentaci.

Vždy používejte software tak, aby jste nepřekračovali specifikovaná omezení uvedená v této příručce. Je také nezbytné provádět pravidelné prohlídky ostatní preventivní úkony aby nedocházelo k chybám.

■ POZNÁMKY

- Obsah této příručky může být změněn bez předchozího upozornění.
- Tuto příručku pečlivě uschovejte.
- Žádná část této příručky nesmí být reprodukována nebo přetisknuta bez svolení výrobce.
- Tato příručka byla připravována s největší pečlivostí a pozorností, pokud v ní naleznete chyby a opomenutí, nebo budete mít pochybnosti o obsahu některého tvrzení kontaktujte prosím Vašeho zástupce HITACHI.
- HITACHI neodpovídá za důsledky použití této příručky a produktu v ní popsaného.

Historie oprav

No. 1	Popis	Datume	číslo příručky
1	Počáteční vydání	2003/02/17	ADPr01601BX

Ve výšeuvedených vydáních byly opraveny typografické chyby, opomenutí a nesprávné výklady nebo přidána dodatečná vysvětlení.

BEZPEČNOSTNÍ PŘEDPOKLADY

Před použitím tohoto výrobku pečlivě prostudujte tuto příručku společně s příručkou pro uživatele servopohonu a řiďte se všemi bezpečnostními upozorněními.

Bezpečnostní předpoklady uvedené v této příručce jsou rozděleny do kategorií NEBEZPEČÍ a UPOZORNĚNÍ.

⚠ NEBEZPEČÍ : Tento symbol NEBEZPEČÍ říká, že situace může vyústit v případě nedodržení správného postupu k závažnému ohrožení zdraví a smrti nebo ke zničení zařízení.

⚠ CAUTION : Tento symbol UPOZORNĚNÍ říká, že situace může vyústit v případě nedodržení správného postupu k méně závažnému ohrožení nebo ke zničení zařízení.

Pokud nejsou respektována upozornění uvedená pod označením POZNÁMKA (Pozn.) může dojít k závažným důsledkům v závislosti na situaci.

Příkazy a upozornění uvedená ve všech třech kategoriích mají všechna svoji důležitost a je nutné je vždy respektovat.

Po přečtení je nutné tuto příručku uložit na přístupném místě tak, aby uživatel zařízení do ní mohl kdykoliv nahlédnout.

■ Předpoklady užití

⚠ NEBEZPEČÍ

1. Před použitím programu proveďte bezpečné vypnutí pohonu (bezpečnostní stop), protože servopohon se může vymknout kontrole a způsobit nebezpečné situace např. při nesprávném odlaďování programu.

⚠ CAUTION

1. Při ladění uživatelského programu je lépe napřed odladit program pouze se servomotorem a následně připojit poháněný stroj. Ujistíte se tak, že se zařízení bude chovat správně a nemůže se vymknout kontrole.

OBSAH

1. Úvod
   - 1.1 Typové označení produktu 1-2
   - 1.2 Instalace software AHF 1-3
   - 1.3 Programovatelné provozní funkce 1-4

2. Vytvoření a provádění programu
   - 2.1 Specifikace jazyka 2-2
   - 2.2 Přehled konfigurace 2-3
   - 2.3 Vytváření programu 2-4
   - 2.4 Prověření syntaxe 2-5
   - 2.5 Provádění programu 2-8

3. Syntaxe
   - 3.1 Kódová forma 3-2
     - 3.1.1 Řádek 3-2
     - 3.1.2 Značka 3-2
     - 3.1.3 Instrukční slovo 3-2
     - 3.1.4 Parametry 3-2
     - 3.1.5 Poznámka 3-2
   - 3.2 Okno pro zadávání dat 3-3
     - 3.2.1 Název proměnné 3-3
     - 3.2.2 Definice 3-3
     - 3.2.3 Výsledek výpočtu 3-3
     - 3.2.4 Poznámka 3-3
   - 3.3 Číselné hodnoty 3-4
   - 3.4 Operátory 3-4
   - 3.5 Podmínky 3-5

4. Základy programování
   - 4.1 Základní polohování 4-2
   - 4.2 Volba polohy (jednobodový vstup) 4-4
   - 4.3 Volba polohy (binární vstup) 4-8
   - 4.4 Volba polohy (zadání stavem vstupů) 4-12

5. Editování programu
5.1 Zahájení a ukončení 5-2
5.2 Postup vypracování programu 5-4
5.3 Práce se soubory 5-5
5.4 Kódové okno - postupy zadávání 5-7
5.5 Editování programu 5-14
5.6 Datové okno – postupy zadávání 5-19
5.7 Sestavení programu 5-22
5.8 Nahrání programu do servopohonu 5-23
5.9 Dávkové zpracování programu - sestavení / nahrání / chod 5-24
5.10 Načtení programu ze servopohonu 5-25
5.11 Ladění / chod programu 5-27
5.12 Vytištění programu 5-35
5.13 Zobrazení pomoci 5-36
5.14 Informace o verzi 5-36

6. Instrukční slova – popis, použití
6.1 Vstupní a výstupní svorky 6-2
6.2 Rezervované proměnné 6-3
6.3 Seznam instrukčních slov 6-6
6.4 Řídící programové instrukce 6-10
6.5 Pohybové instrukce 6-25
6.6 Instrukce vstupů a výstupů 6-56

7. Nesnáze, chyby a jejích příčiny
7.1 Seznam ochran v programovém provozu 7-2
7.2 Vhodný zásah pří chybě 7-3
7.3 Obsah chyb a zásahy při sestavování programu 7-5

POZNÁMKY

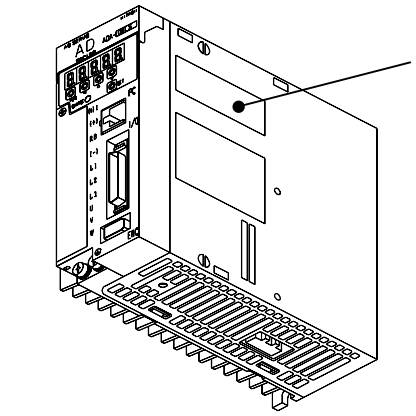
KAPITOLA 1 ÚVOD

Tato kapitola popisuje typ produktu a vysvětluje nastavení pro využití programovatelných funkcí.

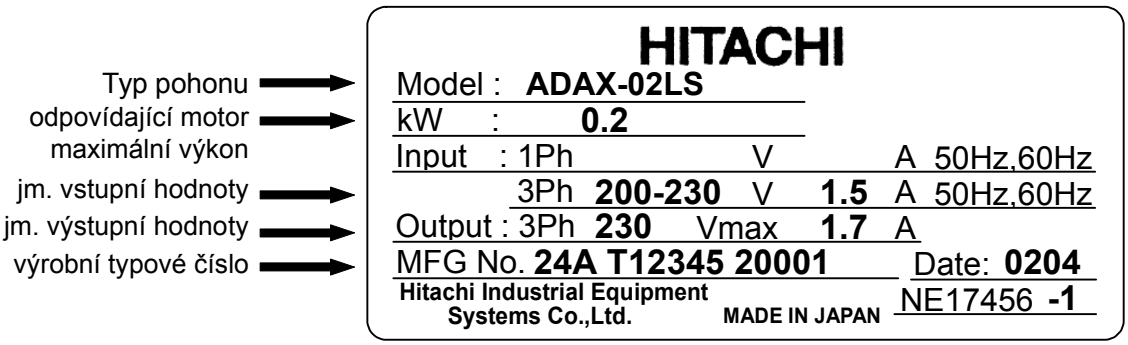
1.1 Typové označení produktu.....1-2
1.2 Instalace software AHF.....1-3
1.3 Programovatelné provozní funkce 1-4

1.1 Typové označení produktu

Na štítku přístroje zjistíte jeho provozní hodnoty a zda se jedná o typ podporující programovatelné provozní funkce.



Poloha typového štítku přístroje
(1.5kW a méně)

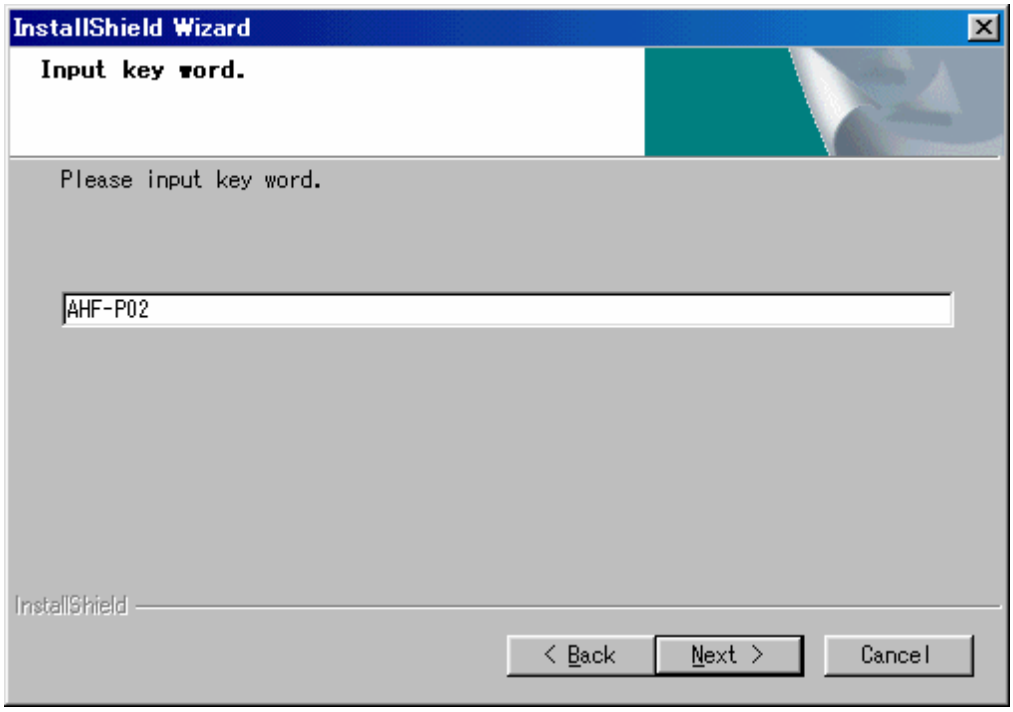


Obsah typového štítku

1.2 Instalace software AHF

Pro instalaci software AHF s možností editace je nutné klíčové slovo.(Model No. AHF-P02). Při zadávání klíčového slova se prosím řiďte následujícími pokyny:

AHF - P02
- Číslo "0"
- Znaménko iminus "-"



Postup instalace a nastavení software AHF je popsán v příslušné příručce. (Zadání klíčového slova je vyžadováno v kroku 11)

Podmínky pro zadání klíčového slova

1) Vložte jednobytový alfanumerický znak.
2) Pokud vložíte nesprávné klíčové slovo, budete požádáni o opakované zadání.
3) Pokud máte již nainstalován software AHF-P01, je nutné jej napřed odinstalovat a pak teprve instalovat software AHF-P02.
4) Pokud instalujete software AHF-P02 po odinstalovaní AHF-P01, nemusí být potřeba použít všech instalačních disků (No. 1 to 4).

1.3 Programovatelné provozní funkce

1.3.1 Nastavení programovatelných provozních funkcí

Před použitím programovatelných provozních funkcí je potřeba změnit nastavení servopohonu, jak je naznačeno níže. Tovární nastavení pohonu je prováděno v průběhu výroby a odpovídá standardnímu servopohonu.

Název parametru	Označení parametru	Nastavení	Tovární hodnota
Volba povelu polohy	FA-22	Pro	PLS

1.3.2 Kontaktní vstupní a výstupní svorky a metoda řízení

(1) Kontaktní vstupní a výstupní svorky

Je-li parametr FA-22 nastaven na hodnotu "Pro" jak bylo ukázáno výše, slouží kontaktní vstupní a výstupní svorky jako obecné vstupy a výstupy. Stav těchto obecných vstupů a výstupů je ovlivnitelný příkazy X(00) až X(11) a Y(00) až Y(07).

Bližší vysvětlení naleznete v kapitole 6.1 Vstupní a výstupní svorky.

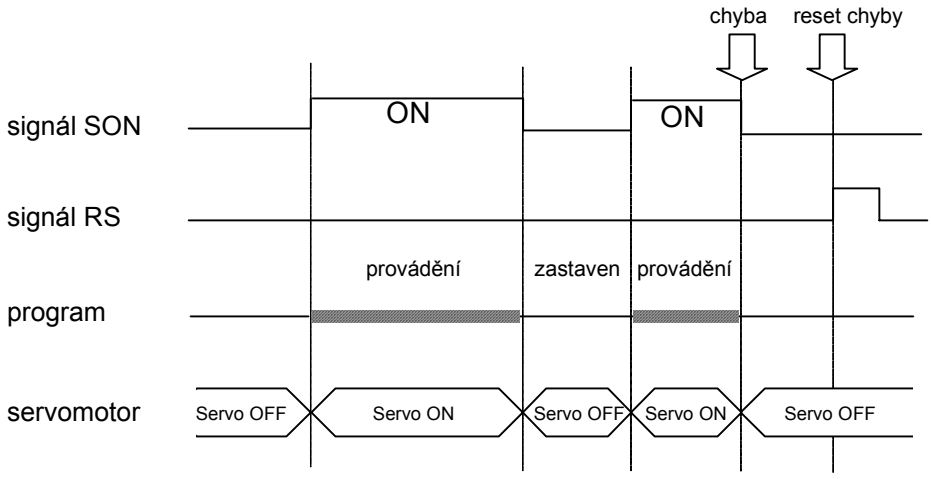
Polarita povelů X(00) až X(11) a Y(00) až Y(07) je "0" v rozpojeném stavu a "1" v uzavřeném stavu. Protože změna polarity svorek není možná, je potřeba všechny potřebné změny provést v uživatelském programu. Vyjímku tvoří signály SON a RS.

(2) Metoda řízení

Přednastavenou metodou řízení při použití programovatelných provozních funkcí je polohová regulace. Volba rychlostní, momentové a jiné regulace se děje povelem. Provádění programu je spuštěno přechodem signálu SON do stavu ON (parametr FA-22 má hodnotu Pro. Při provádění programu je servomotor zapnut a pohon je ve stavu „servo ON“. Dojde-li k přechodu signálu SON do stavu OFF, provádění programu se zastaví a servomotor je vypnut (stav servo OFF)

Signál SON lze použít pro realizaci bezpečnostního zastavení nebo jiného okamžitého zastavení.

Dojde-li k chybě, je provádění programu okamžitě ukončeno, a servomotor je zastaven. V tomto případě je nutné provést reset krátkodobou aktivací signálu RS při vypnutém signálu SON.



(3) Proměnné, výstupní svorky a ostatní počáteční hodnoty

Tabulka níže ukazuje proměnné, výstupní svorky a počáteční hodnotu druhé výchozí polohy pro polohování:

Název	v době od zapnutí sítě a prvního zapnutí SON	druhé zapnutí SON
P(),N(),T(),U() ACC(),DEC()	Okno nastavení dat	předchozí hodnoty (data v datovém okně jsou sestavena při zápisu programu)
regulační konstanty (J, KFC, etc.)	Je použito nastavení parametrů Fd-xx .	předchozí hodnoty (předchozí hodnoty v programu)
výstupní svorky Y()	Všechny svorky jsou 0 (OFF).	
druhá výchozí poloha	0	

POZNÁMKY

KAPITOLA 2 VYTVOŘENÍ A PROVÁDĚNÍ PROGRAMU

Tato kapitola pojednává o postupu vytváření a provádění programu.

2.1 Specifikace jazyka 2-2
2.2 Přehled konfigurace 2-3
2.3 Vytváření programu 2-4
2.4 Prověření syntaxe 2-5
2.5 Provádění programu 2-8

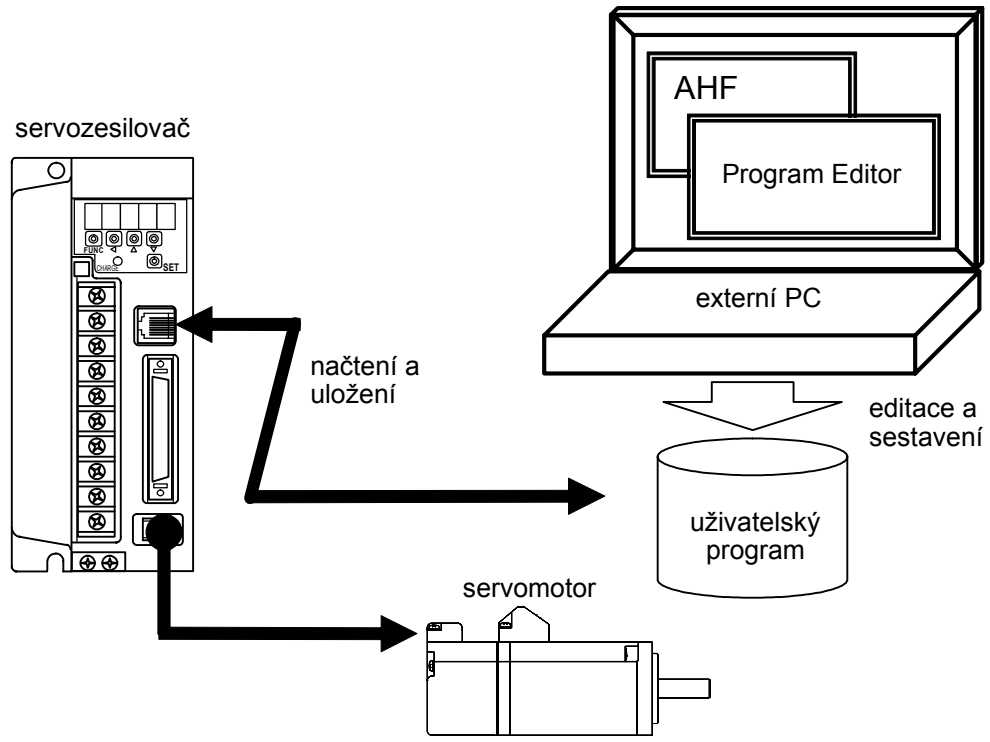
2.1 Specifikace jazyka

Následující tabulka obsahuje specifikaci programovacího jazyka pro programovatelné funkce HITACHI AC servopohonu.

Pojem		Specifikace
specifikace jazyka	typ jazyka	Jazyk příbuzný BASUCu (včetně operací - nájezd na VP, polohování serva, řízení rychlosti serva)
	Vstupní předpoklady	PC kompatibilní s IBM PC/AT, operační systém Windows 95/98/Me, Windows NT4.0, Windows 2000
	Velikost programu	512 steps (Paměť pohonu pojme 512 kroků, nebo 6KB.)
	Programové podpůrné funkce	• textový vstup • zobrazení (v oknech) • prověření syntaxe (v oknech) • nahrání a načtení programu (PC ←→ pohon) • jednotlivý krok • přerušení (až 4 body)
	Forma provádění	interpretační metoda, 1.12 ms/příkaz (dovolené volání podprogramů až do úrovně vnoření 8)
Vstup/výstupní funkce	Externí dvoustavový kontaktní vstup	Kontaktní signál, nebo signál z otevřeného kolektoru (vnitřní zdroj 24V_{DC}). servo ON, reset poruchy, a 12 obecných vstupů označených X(0) až X(11)
	externí výstup	8 dvoustavových výstupů označených (Y(0) až Y(7))
	externí analogový výstup	2 analogové výstupy označené (XA(0) a XA(1))
Rezervovaná slova	Proměnné	• poloha: P(00) až P(99) (100 bodů) • rychlost: N(00) až N(15) (16 bodů) • moment: T(00) až T(15) (16 bodů) • doba rozběhu: ACC(0), ACC(1) (2 body) • doba doběhu: DEC(0), DEC(1) (2 body) (zapamatovatelné) • režim regulace: MOD • zesílení regulace: KSP, KSI, KP, atd. • zobrazení: IFB, IRF, NFB, NRF, POS, PRF, atd. • uživatelsky definované proměnné: U(00) až U(15) (16 bodů)
	Povely	povel provádění programu, povel pohybu (povel provozu servopohonu)
Ostatní	Uložení programu	• uložit do poměti EEPROM hlavní jednotky • uložit jako zdrojový soubor a soubor přechodových kódů
	Přerušení	až 4 přerušení (set, reset, enable, nebo disable)

2.2 Přehled konfigurace

Software pro vytváření programu servopohonu (Program Editor) provádí mnoho různých úkonů spojených s prací na programu včetně editace a sestavení programu načtení a uložení programu do paměti servopohonu.



Editační funkce programu jsou uvedeny níže:

Funkce	prováděný úkon
programování/editace	programování vstupů, editace, ukládání, načítání, tisk
sestavení a rozložení	sestavení editovaného programu a rozložení načteného programu
načtení a uložení	uložení programu do paměti servopohonu, načtení programu z paměti servopohonu
podpora ladění	provádění načteného programu, provádění po 1 kroku, nastavení přerušení atd.

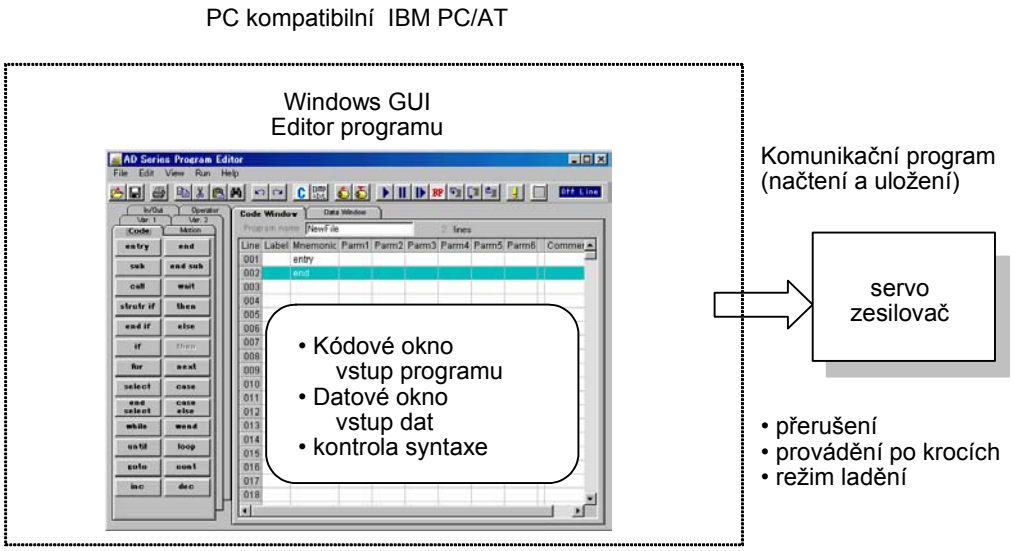
2.3 Vytváření programu

Vstupem uživatelského programu je editační okno.

Editace sestává z kódové oblasti (kódové okno) a datové oblasti (datové okno).

Kódové okno umožňuje vložit zdrojové kódy pro vytvoření programu. Datové okno dovoluje nastavení počátečních hodnot programu.

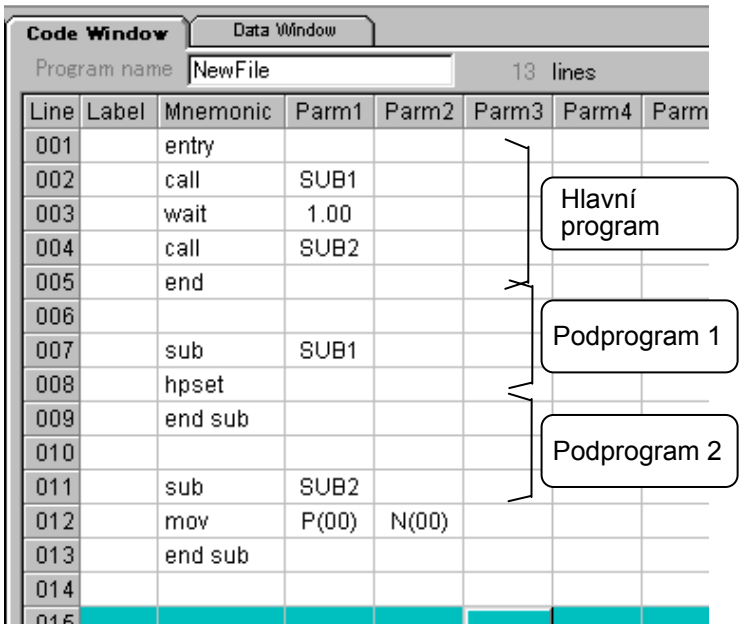
Můžete provádět kontrolu syntaxe, krokový výpočet programu, krokové provádění programu, přerušení provádění komunikačním programem (AHF) a uložení programu do paměti pohonu



Provozní procedury a další detailní instrukce naleznete v kapitole 5, Editování programu.

(1) Kódové okno

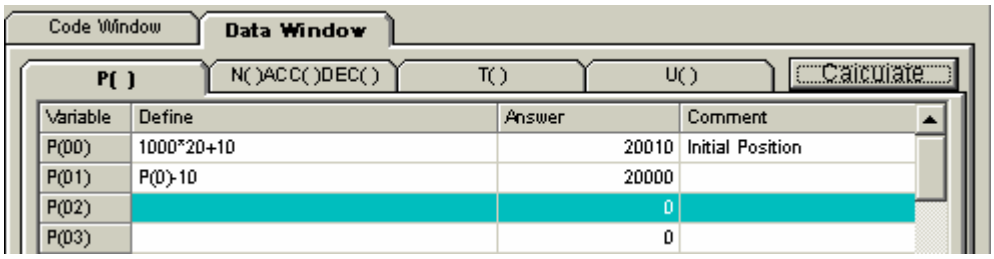
Hlavní program začíná "entry" a končí "end".
V programu může existovat pouze jeden hlavní program
Podprogram začíná "sub" a končí "end sub".
V programu může být jeden nebo více podprogramů.



(2) Datové okno

Datové okno umožňuje nastavení počátečních hodnot proměnných na počátku programu. Každé proměnné je přiřazen jeden řádek, který se skládá z definice proměnné, a výsledku kalkulace. Proměnná může být určena i jinými proměnnými použitými v definici.

(Příklad)



P(00)	1000*20+10	20010	název proměnné
definice proměnné	výsledek kalkulace	počáteční hodnota	poznámka

2.4 Kontrola syntaxe

Když jsou data zadána do kódového okna, je prověřena jejich platnost. Je-li v zadání nějaká chyba syntaxe, v kódovém okně se zobrazí chybové hlášení a stav se vrátí k hodnotám před provedením změny.

Kódové okno také sleduje zda při zadání nedochází k překročení mezí parametrů nebo počtu kroků. Po ukončení zadávání programu generuje kódové okno přenosový kód pro prováděcí program.

Zadání parametrů v kódovém okně má následující omezení

(1) Značka

lze vkládat alfanumerické znaky.

(2) Mnemonické retězce

může obsahovat pouze příkazy a přepisovatelné proměnné.

(3) Sloupec parametrů (Parm 1 až Parm 6)

Zpracovávaná data společně s příkazy v mnemonických řetězcích.

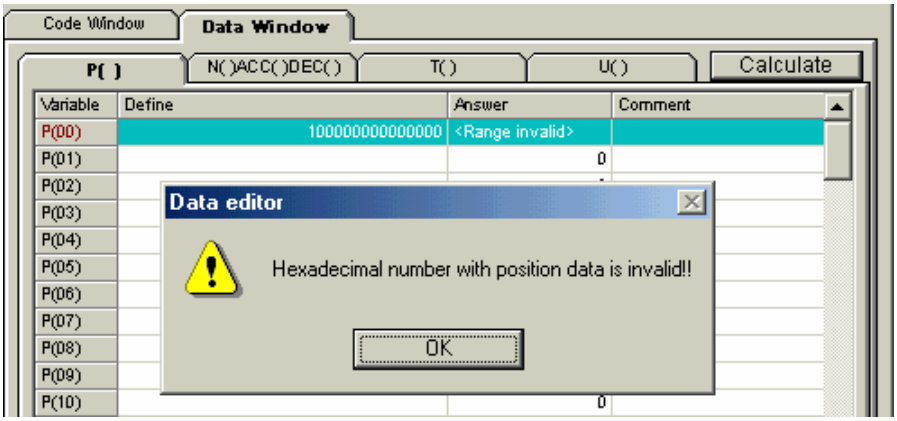
Pokud je informace v mnemonickém řetězci prázdná, nebo "<skupina příkazů>", není akceptován žádný řetězec parametrů.

(4) Sloupec poznámek

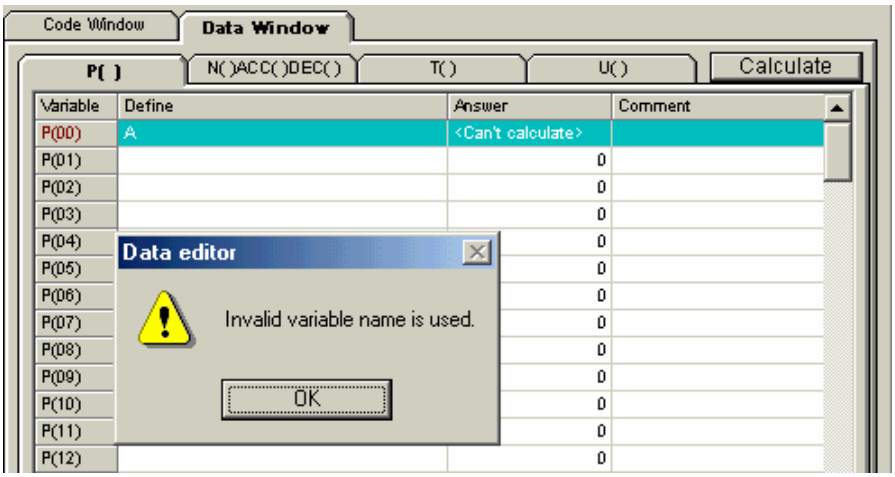
lze vkládat alfanumerické znaky

K prověření zadávaných dat v datovém okně dochází při stisknutí tlačítka kalkulace nebo návrat (return). Musí být zadána hodnota v definičním sloupci.

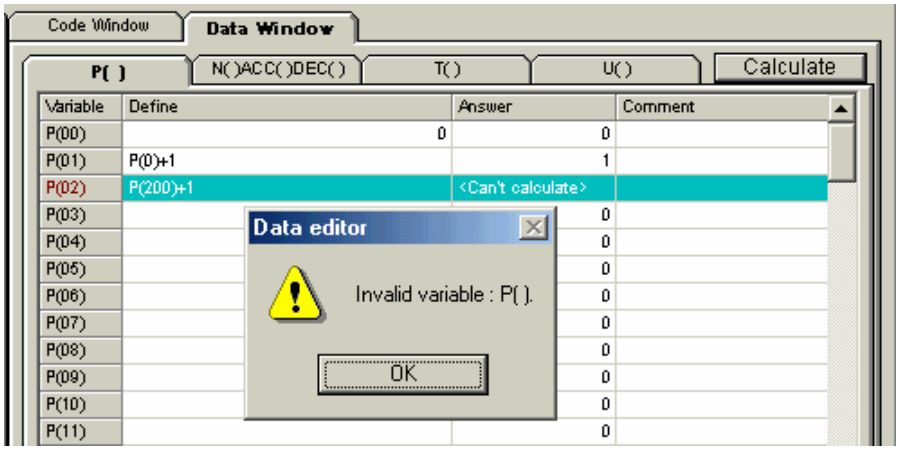
(1) Například uveďme chybu dat polohy. Pokud se při kalkulaci polohy výsledek ocitne mimo povolený rozsah polohy, je zobrazeno hlášení "Hexadecimal number with position data is invalid!"(hexadecimální číslo určení polohy je nepřípustné) a v odpovědním sloupci je hlášení "<Range invalid>" (nepřípustný rozsah).



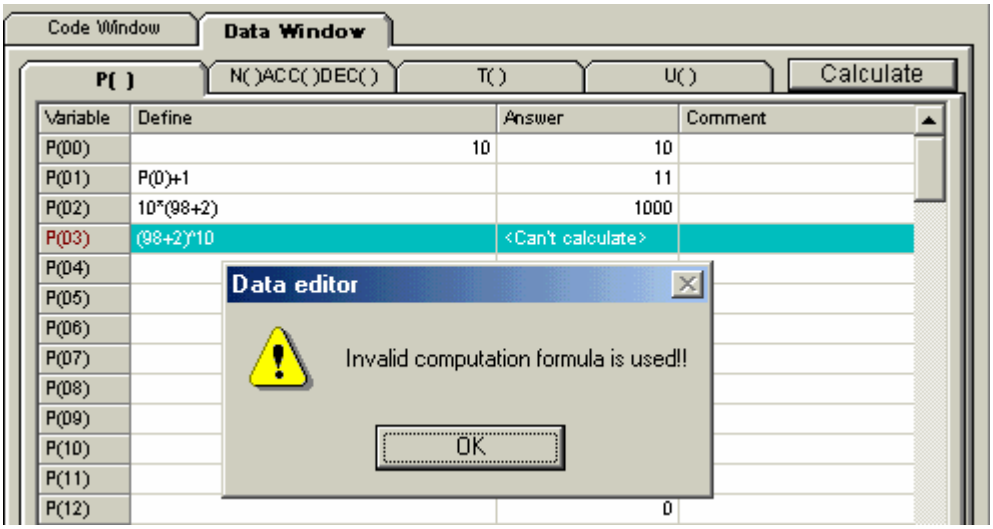
(2) Je-li použit nesprávný název proměnné, je hlášeno "Invalid variable name is used" (použito nepřípustný název proměnné) a v odpovědním sloupci je hlášení "<Can't calculate>" (nemohu spočítat).



(3) Je-li číslo proměnné mimo povolený rozsah je hlášeno "Illegal parameter: X()" (X: symbol proměnné P, N, ACC, DEC, T, nebo U) a v odpovědním sloupci je hlášení "<Can't calculate>" (nemohu spočítat).



(4) Je-li použita jiná značka než matematický operátor (značka plus, minus, násobení, dělení) a umístěna bezprostředně před nebo za závorku je hlášeno "Invalid computation formula is used!" (nepřípustný kalkulační vzorec) a v odpovědním sloupci je hlášení "<Can't Calculate>" (nemohu spočítat).

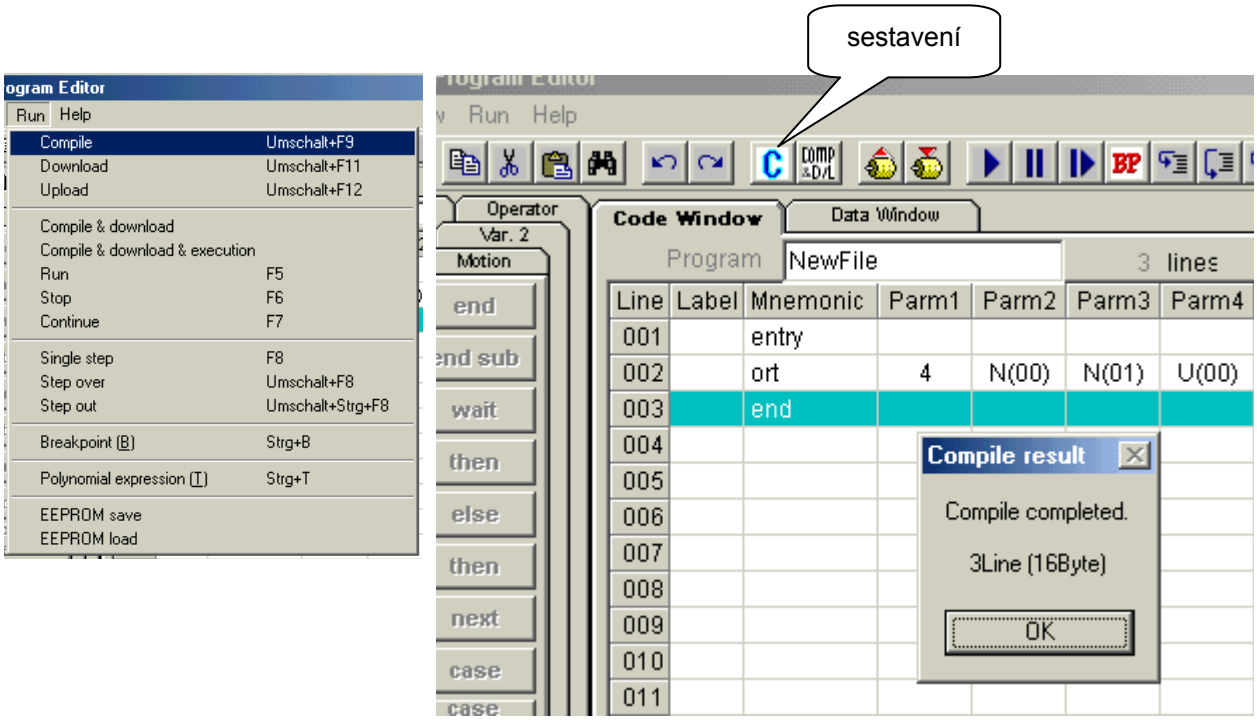


Přípustný rozsah proměnných je zobrazení níže:

Proměnná	Název proměnné	možný rozsah
proměnné polohy	P(00) až P(99)	-0x7fffffff až 0x7fffffff (-2147483648 až 2147483647)
rychlostní proměnná	N(00) až N(15)	-5000 až 5000
proměnné rozběhu	ACC(0), ACC(1)	0.00 až 99.99 je-li pro proměnnou použita přímo číselná hodnota)
proměnné doběhu	DEC(0), DEC(1)	0000 až 9999 1s=100dig (je-li zástupně použita proměnná U()
proměnná momentu	T(00) až T(15)	-300 až 300
uživatelské proměnné	U(00) až U(15)	-0x7fffffff až 0x7fffffff (-2147483648 až 2147483647)

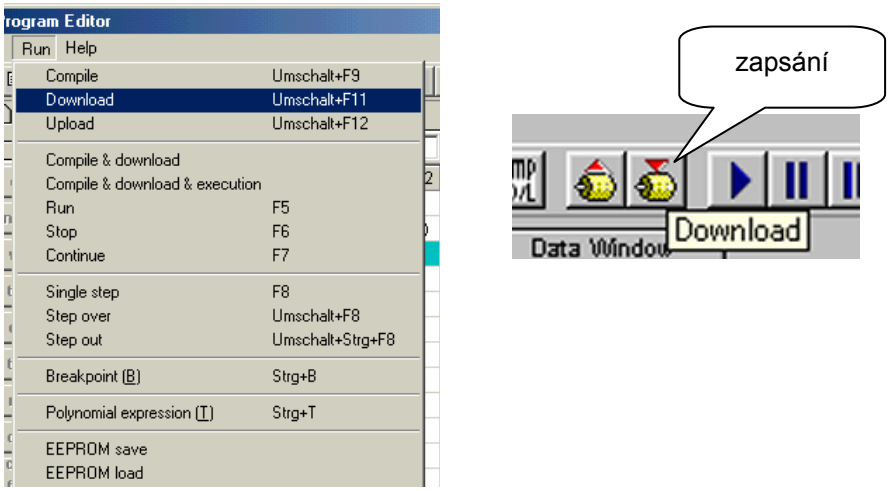
2.5 Provádění programu

(1) Po vytvoření prováděcího programu v kódovém okně proveďte sestavení programu. Pokud je program bez chyb je generován převodní kód.
Sestavení programu proveďte příkazy "Compile" v nabidce "Execute" nebo prostě stiskněte tlačítko sestavení.



(2) Prováděcí program vytvořený editačním programem lze nyní zapsat do paměti pohonu. v nabídce "Execute" zvolte "Download" nebo prostě stiskněte tlačítko zapsání

(3) Zapsaný program bude prováděn, pokud je pohon ve stavu „servo ON“ a pokud je nastaven v režimu provádění programových operací.



KAPITOLA 3 SYNTAXE

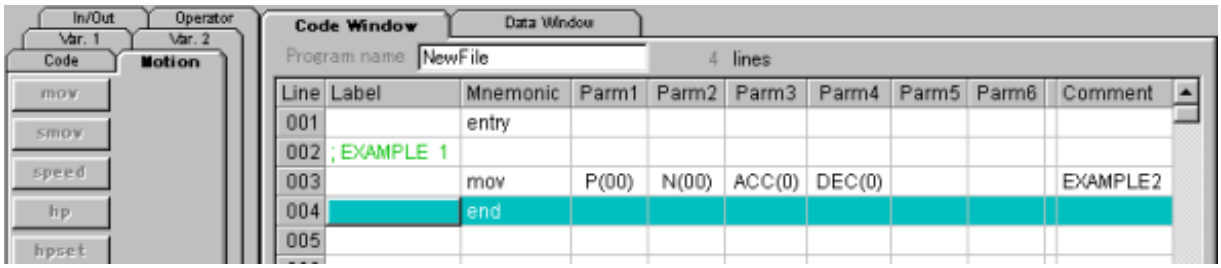
Tato kapitola pojednává o tom jak napsat program.

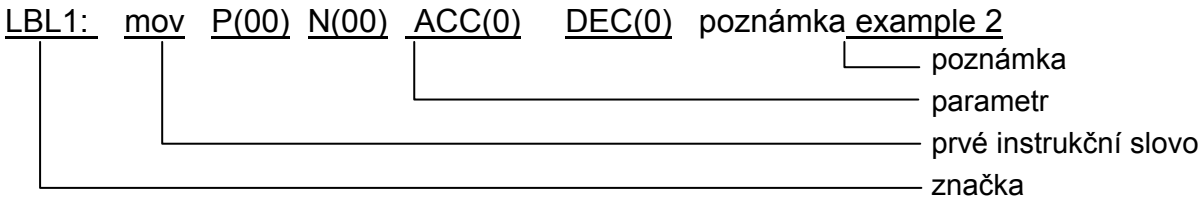
3.1 Kódová forma 3-2
3.1.1 Řádek....3-2
3.1.2 Značka 3-2
3.1.3 Instrukční slovo 3-2
3.1.4 Parametry 3-2
3.1.5 Poznámka 3-2
3.2 Okno pro zadávání dat 3-3
3.2.1 Název proměnné....3-3
3.2.2 Definice 3-3
3.2.3 Výsledek výpočtu 3-3
3.2.4 Poznámka 3-3
3.3 Numerické hodnoty 3-4
3.4 Operátory....3-4
3.5 Podmínky....3-5

3.1 Kódová forma

Každý řádek sestává ze značky, instrukčního slova, parametrů instrukčního slova a pole pro poznámku. Některá instrukční slova nemusí mít žádné parametry.

(Příklad)





Příklad výše zobrazuje případ, kdy povel rychlosti N(00) je použit k dokončení polohování na cílovou pozici P(00).

3.1.1 Řádek

Řádek je jednotkou programové instrukce. Na každém řádku je napsáno jedno instrukční slovo. Provedení trvá 1,12 ms na řádek. Tato jedna instrukce je nazývána krokem.

3.1.2 Značka

Do pole značka zapisujeme okruh do kterého instrukce spadá, nebo jiné označení určení instrukce.

Je-li na začátku pole značky zapsán středník (;) řádek se neprovádí. Tato možnost se využívá při ladění programu k podmínečnému vyloučení některých řádků.

3.1.3 Instrukční slovo

Napište instrukci, která má být provedena. Detaily různých instrukcí naleznete v kapitole 6 „Instrukční slova – popis, použití“.

3.1.4 Parametry

Zapište parametry potřebné k provedení instrukce. Lze zapsat až 6 parametrů v závislosti na použité instrukci.

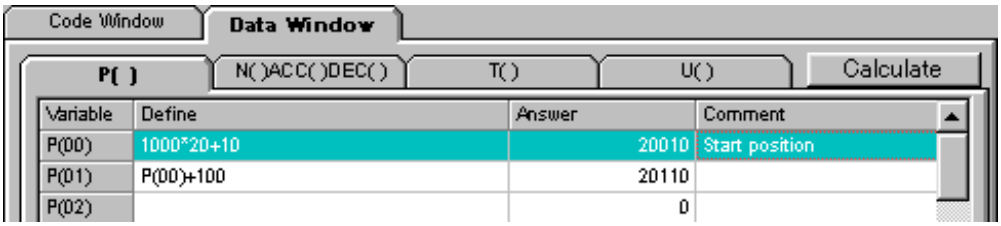
3.1.5 Poznámka

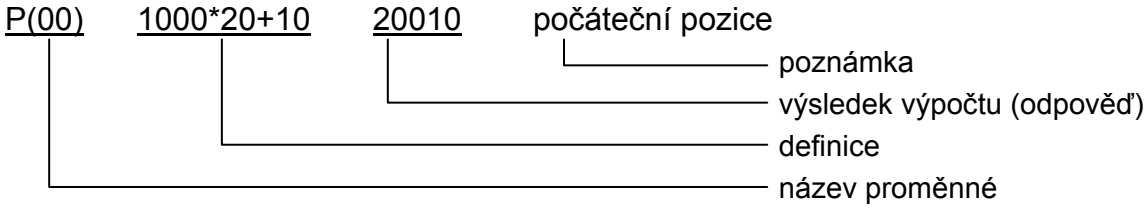
Do pole „poznámka“ je možné napsat další upřesňující informace ke každému instrukčnímu řádku.

3.2 Okno pro zadávání dat

Každá proměnná je určena jedním řádkem, který obsahuje název proměnné, definici proměnné, výsledek výpočtu a pole pro poznámku

(Příklad)





3.2.1 Název proměnné

Do pole název zapisujeme název proměnné , která má být definována. Použitelné proměnné jsou uvedeny níže:

Typ	Název proměnné
poloha	P(00) až P(99)
rychlost	N(00) až N(15)
moment	T(00) až T(15)
doba rozběhu	ACC(0), ACC(1)
doba doběhu	DEC(0), DEC(1)
uživatelsky definovaná proměnná	U(00) až U(15)

3.2.2 Definice

Napište definiční vztah proměnné. V definici lze použít i názvy proměnných. Po ukončení definice a stisknutí tlačítka "Calculate" v datovém okně se provede výpočet hodnoty nadefinované proměnné.

(Příklad)

P(00) 1000*20+10 20010
P(01) P(00)+100 20110
definice

3.2.3 Výsledek výpočtu

Je zobrazen výsledek výpočtu (nelze přepsat).

3.2.4 Poznámka

Lze uvést poznámku vztahující se k proměnné.

3.3 Číselné hodnoty

V následující části jsou uvedeny číselné hodnoty, které lze uvést v programu a při definici proměnných.

<Rozsah použitelných číselných hodnot>

Název proměnné	číselný rozsah
bitová proměnná	0, 1
uživatelská proměnná	-2147483648 to 2147483647
proměnná volby polohy	-2147483648 to 2147483647
proměnná volby rychlosti	-5000 to 5000
proměnná rozběhu a doběhu	0.00 to 99.99
proměnná volby momentu	-300 to 300
proměnná volby režimu regulace	0, 1, 2

<Zápis numerické hodnoty>

značka	druh	poznámka
(není)	dekadická	číslo
&H	hexadecimální	hexadecimální data
&B	binární	binární data

- Je-li výsledek výpočtu proměnné mimo její dovolený rozsah, zobrazí se hlášení "Outside the acceptable data range!" (mimo dovolený rozsah dat) a ve sloupci odpověď (answer) bude zápis "<Out of range>".
- Je-li číslo proměnné mimo dovolený rozsah, zobrazí se hlášení "Illegal variable: *()" (*:symbol proměnné) a ve sloupci odpověď bude zápis "<Calculation impossible>".

3.4 Operátory

Při tvorbě programu lze použít následující operátory:

Formát	Operace
<proměnná 1> = <proměnná 2> + <proměnná 3>	sčítání
<proměnná 1> = <proměnná 2> – <proměnná 3>	odčítání
<proměnná 1> = <proměnná 2> * <proměnná 3>	násobění
<proměnná 1> = <proměnná 2> / <proměnná 3>	dělení
<proměnná 1> = <proměnná 2>	přiřazení
<proměnná 1> = <proměnná 2> mod <proměnná 3>	modulo
<proměnná 1> = <proměnná 2> or <proměnná 3>	logická operace OR
<proměnná 1> = <proměnná 2> and <proměnná 3>	logická operace AND
<proměnná 1> = <proměnná 2> xor <proměnná 3>	logická operace XOR
<proměnná 1> = not <proměnná 2>	logická operace NOT
<proměnná 1> = abs <proměnná 2>	absolutní hodnota

3.5 Podmínky

Podmínkové operace se užívají k porovnámí mezi dvěma proměnnými. Výsledkem operace je „pravda“ je-li podmínka splněna nebo „nepravda“ není-li podmínka splněna.

Podmínkové vyjádření "If", "While", nebo jiné se používají k ovlivnění toku programu v závislosti na právě nastavších podmínkách.

Formát	Operace
<Proměnná 1> = <Proměnná 2>	pravda, pokud se proměnné rovnají
<Proměnná 1> < <Proměnná 2>	pravda, pokud je proměnná 1 menší něž proměnná 2
<Proměnná 1> < = <Proměnná 2>	pravda, pokud je prom.1 menší nebo rovna prom.2
<Proměnná 1> > <Proměnná 2>	pravda, pokud je proměnná 1 větší než proměnná 2
<Proměnná 1> > = <Proměnná 2>	pravda, pokud je prom.1 větší nebo rovna než prom.2
<Proměnná 1> < > <Proměnná 2>	pravda, pokud se proměnné nerovnají

POZNÁMKY

KAPITOLA 4 ZÁKLADY PROGRAMOVÁNÍ

Tato kapitola uvádí a vysvětluje některé příklady ze základního programování.

4.1 Základní polohování 4-2
4.2 Volba polohy (jednobodový vstup).....4-4
4.3 Volba polohy (binární vstup).....4-8
4.4 Volba polohy (zadání stavem vstupů).....4-12

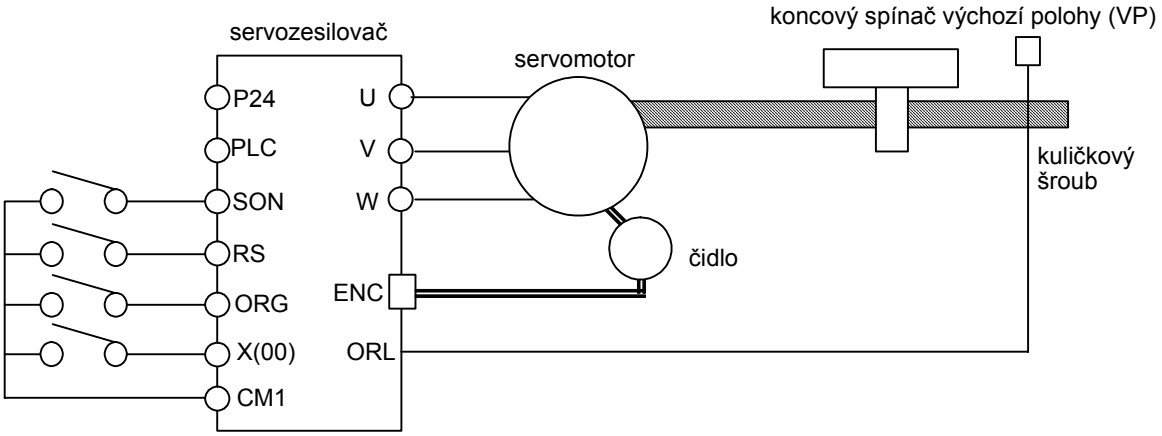
Příklady programu použití v následující stati jsou uloženy v sekci "Sample" na adrese, na které je nainstalován software AHF.

4.1 Základní polohování

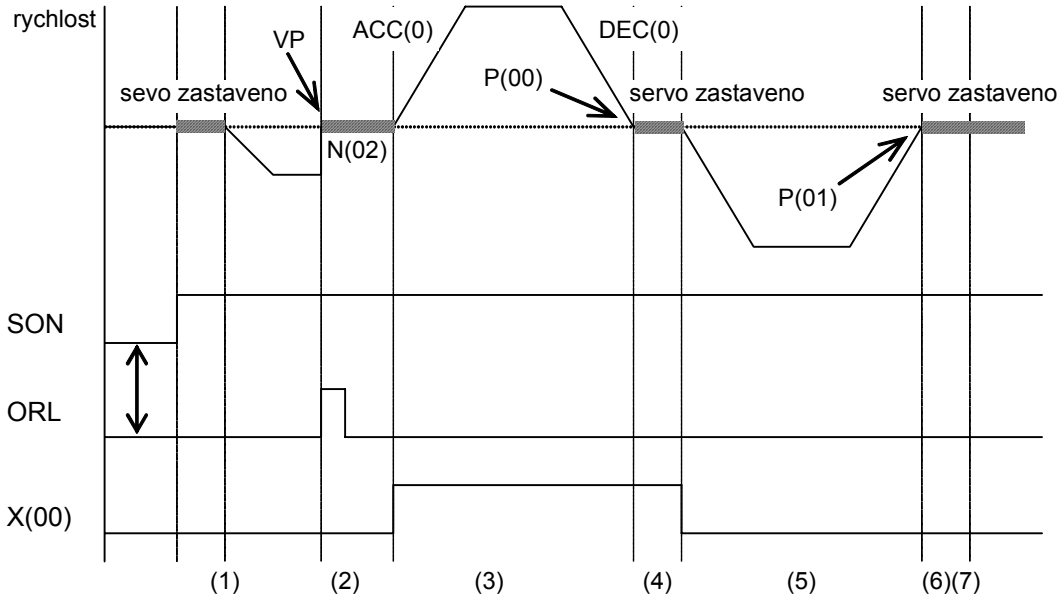
(1) Popis příkladu

Je-li svorka X(00) = 1 (ON) (po nájezdu na VP), pak pohon najede na polohu P(00). Je-li následně X(00) = 0 (OFF), pohon změní polohu na P(01).

(2) Zapojení svorek



(3) Časový diagram průběhu rychlosti



(4) Příklad programu

• Kódové okno

řádek	značka	instr. slovo	parm. 1	parm. 2	parm. 3	parm. 4	parm. 5	parm. 6	
001		entry							
002		ort	1	N(00)		ACC(0)	DEC(0)		←(1)
003	LOOP	wait	X(00)	=	1				←(2)
004		mov	P(00)	N(00)	ACC(0)	DEC(0)			←(3)
005		wait	X(00)	=	0				←(4)
006		mov	P(01)	N(01)	ACC(0)	DEC(0)			←(5)
007		goto	LOOP						←(6)
008		end							←(7)
009									

• Datové okno

proměnná	definice	výsledek výpočtu	poznámka	
P(00)	&H8000*10	327680		← 10 otáček= 32768 × 10
P(01)	0	0		

proměnná	definice	výsledek výpočtu	poznámka	
N(00)	1000	1000		←1000min^{-1}
N(01)	3000	3000		←3000min^{-1}

proměnná	definice	výsledek výpočtu	poznámka	
ACC(0)	1.0	1.00		← čas rozběhu na maximální otáčky 5000 min^{-1}

proměnná	definice	výsledek výpočtu	poznámka	
DEC(0)	1.0	1.00		← čas doběhu z maximálních otáček 5000 min^{-1}

(5) Provedení a popis programu

(1) Je proveden nájezd na VP nízkou rychlostí (směr zpět).
Sekvence nájezdu na VP začne při spuštění serva (servo ON). Výchozí poloha (VP) je dosažena v okamžiku kdy se hodnota signálu ORL změní z 0FF na ON.

(2) Program čeká, dokud není sepnuta svorka X(00).

(3) Po zapnutí X(00) se provádí nájezd na polohu P(00).
Servopohon se rozbíhá dle ACC(0) na provozní rychlost (N(00) a dobíhá dle DEC(0), aby dojel na polohu P(00) (optimální čas k dosažení provozní rychlosti a polohy lze dosáhnout nastavením rozběhu a doběhu).

(4) Program čeká až bude dál povel k další akci (rozepnutí svorky X(00).

(5) Po přechodu svorky X(00) do stavu OFF je proveden nájezd na polohu P(01) (stejným způsobem jako v případě nájezdu na P(00) se uplatní parametry N(01), ACC(0) a DEC(0))

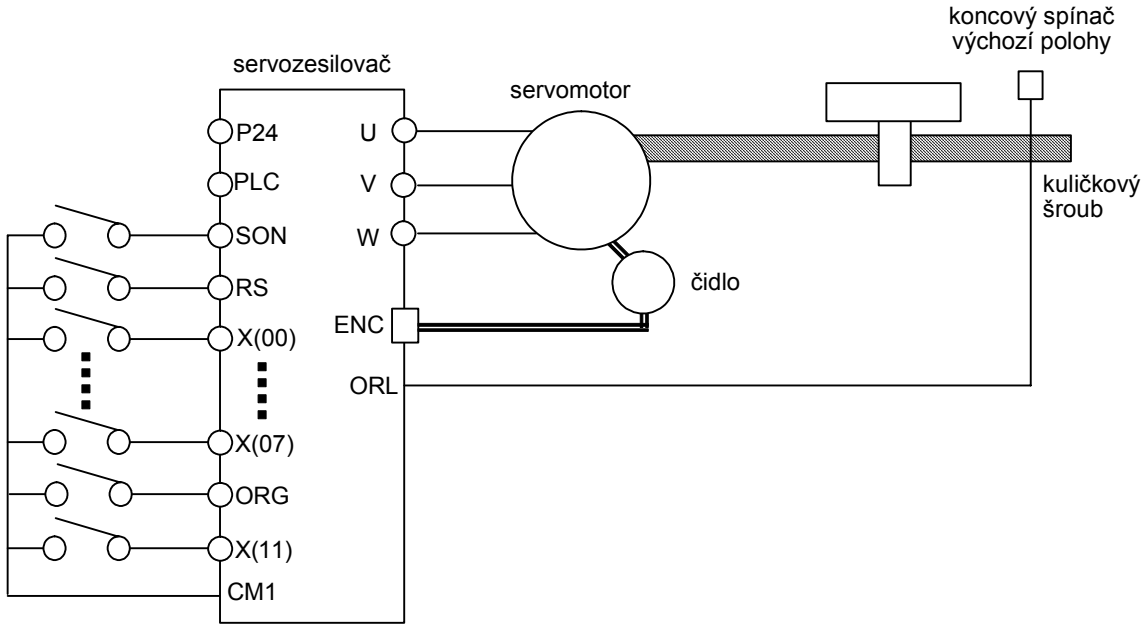
(6) Program se vrací ke kroku (2).

4.2 Volba polohy (jednobodový vstup)

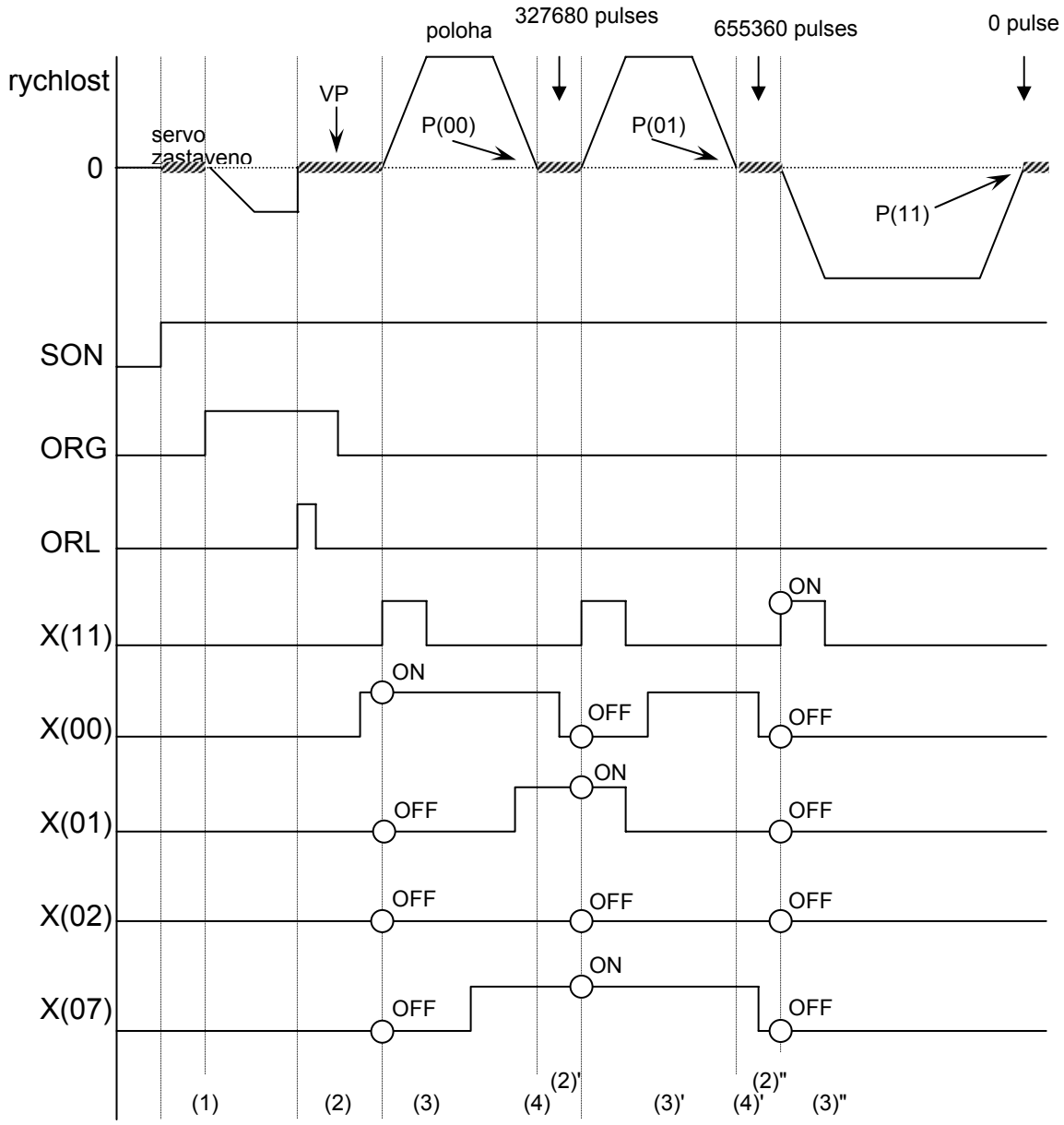
(1) Popis příkladu

Je-li po dosažení VP sepnuta svorka X(11) = 1 servopohon se posune na polohu P(Xn), kde n je číslo svorky v intervalu X(00) až X(07), která je sepnuta. Je-li např. sepnuta svorka X(07) = ON, servopohon se přesouvá na polohu P(07), jsou-li sepnuty současně svorky X(07)=ON a X(01)=ON pak se servopohon přesune na polohu P(01), protože instrukce mov P(Xn) předpokládá vždy přesun na polohu odpovídající sepnuté svorce s nejnižším pořadovým číslem.

(2) zapojení svorek



(3) Časový diagram průběhu rychlosti



(4) Příklad programu

• Kódové okno

řádek	značka	instr. slovo	parm. 1	parm. 2	parm. 3	parm. 4	parm. 5	parm. 6	
001		entry							
002		chgORG=	default						
003		ort	1	N(01)		ACC(0)	DEC(0)		←(1)
004	LOOP	wait	X(11)	=	1				←(2)
005		mov	P(Xn)	N(00)	ACC(0)	DEC(0)			←(3)
006		goto	LOOP						←(4)
007		end							

• Datové okno

proměnná	definice	výsledek výpočtu	poznámka	
P(00)	&H8000*10	327680		←10 otáček=32768 × 10
P(01)	&H8000*20	655360		←20 otáček=32768 × 20
P(02)	&H8000*30	983040		←30 otáček=32768 × 30
P(03)	&H8000*40	1310720		←40 otáček=32768 × 40
P(04)	&H8000*50	1638400		←50 otáček=32768 × 50
P(05)	&H8000*60	1966080		←60 otáček=32768 × 60
P(06)	&H8000*70	2293760		←70 otáček=32768 × 70
P(07)	&H8000*80	2621440		←80 otáček=32768 × 80
P(11)	0	0		← není-li sepnuta žádná svorka pak se provede návrat na VP

proměnná	definice	výsledek výpočtu	poznámka	
N(00)	1000	1000		←1000min^{-1}
N(01)	30	30		← 30min^{-1}

proměnná	definice	výsledek výpočtu	poznámka	
ACC(0)	1.0	1.00		← čas rozběhu na max. rychlost 5000 min^{-1}

proměnná	definice	výsledek výpočtu	poznámka	
DEC(0)	1.0	1.00		← čas doběhu z max. rychlosti 5000 min^{-1}

(5) Provedení a popis programu

(1) Je proveden nájezd na VP nízkou rychlostí (směr zpět).
Sekvence nájezdu na VP začne při spuštění serva (servo ON). Nájezd na VP začíná, když je aktivován signál ORG. Výchozí poloha (VP) je dosažena v okamžiku kdy se hodnota signálu ORL změní z 0FF na ON.

(2) Program čeká, dokud není aktivován vstup X(11).

(3) Dokud je aktivní svorka X(00) ze skupiny X(00)-X(07), servomotor se najede na pozici P(00).

(4) Program se vrátí ke kroku (2).

(2)’ Program čeká, dokud není aktivován vstup X(11).

(3)' Je-li svorka X(00) = OFF, X(01) = ON, X(02) až X(06) = OFF, a X(07) = ON, servopohon najede na polohu P(01).

(4)' Program se vrátí ke kroku (2).

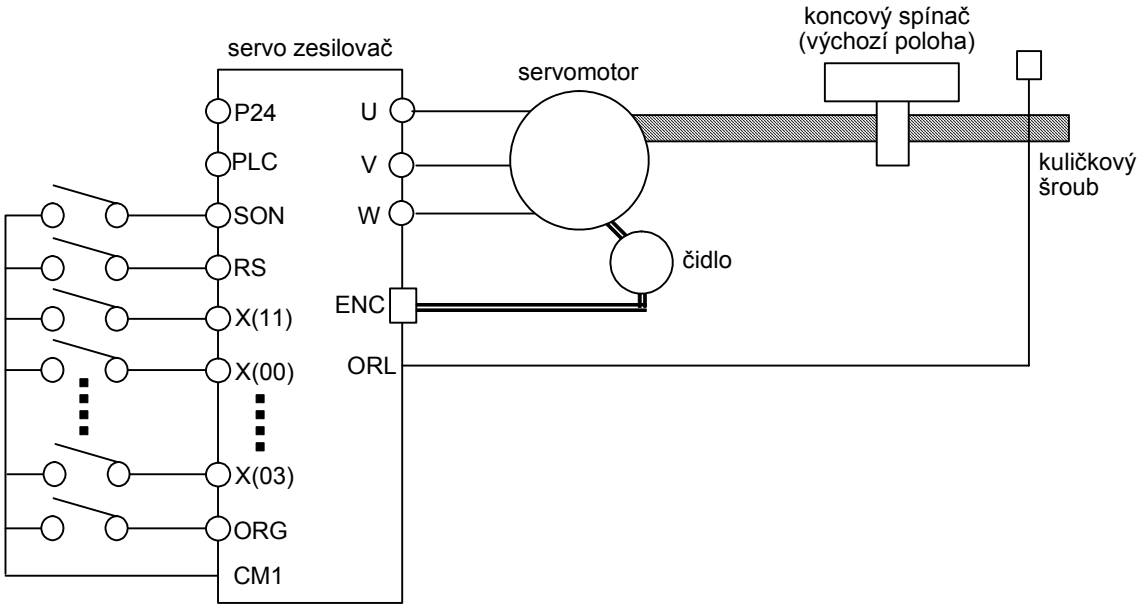
(2)'' Je-li vstup X(00) až X(07) = OFF a X(11) = ON, servopohon najede na pozici P(11) = 0.

4.3 Volba polohy (binarní vstup)

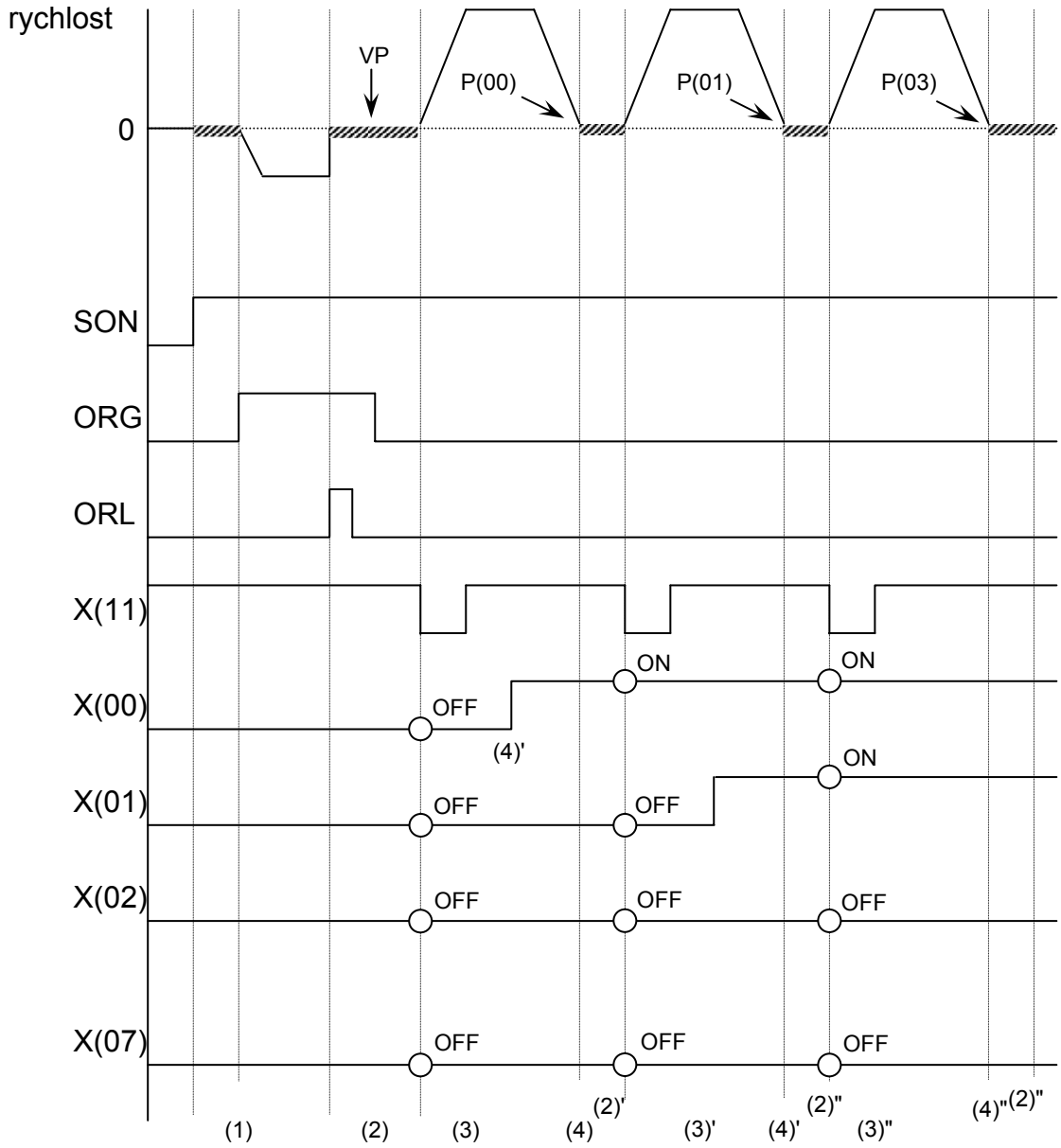
(1) Popis příkladu

Po dosažení VP je vstupní svorka X(11) = 0, binární hodnota zadaná vstupy X(00) až X(03) určuje výslednou polohu pohonu P(Xw).

(2) Zapojení svorkovnice



(3) Časový diagram průběhu rychlosti



(4) Příklad programu

• Kódové okno

001		entry							
002		chgORG=	default						
003		ort	1	N(01)		ACC(0)	DEC(0)		←(1)
004	LOOP	wait	X(11)	=	0				←(2)
005		mov	P(Xw)	N(00)	ACC(0)	DEC(0)			←(3)
006		goto	LOOP						←(4)
007		end							

• Datové okno

proměnná	definice	výsledek výpočtu	poznámka	
P(00)	&H8000*10	327680		←10 otáček=32768 × 10
P(01)	&H8000*20	655360		←20 otáček=32768 × 20
P(02)	&H8000*30	983040		←30 otáček=32768 × 30
P(03)	&H8000*40	1310720		←40 otáček=32768 × 40
P(04)	&H8000*50	1638400		←50 otáček=32768 × 50
P(05)	&H8000*60	1966080		←60 otáček=32768 × 60
P(06)	&H8000*70	2293760		←70 otáček=32768 × 70
P(07)	&H8000*80	2621440		←80 otáček=32768 × 80
P(08)	&H8000*90	2949120		←90 otáček=32768 × 90
P(09)	&H8000*100	3276800		←100 otáček=32768 × 100
P(10)	&H8000*110	3604480		←110 otáček=32768 × 110
P(11)	&H8000*120	3932160		←120 otáček=32768 × 120
P(12)	&H8000*130	4259840		←130 otáček=32768 × 130
P(13)	&H8000*140	4587520		←140 otáček=32768 × 140
P(14)	&H8000*150	4915200		←150 otáček=32768 × 150
P(15)	&H8000*160	5242880		←160 otáček=32768 × 160

proměnná	definice	výsledek výpočtu	poznámka	
N(00)	1000	1000		←1000min^{-1}
N(01)	30	30		← 30min^{-1}

proměnná	definice	výsledek výpočtu	poznámka	
ACC(0)	1.0	1.00		← čas rozběhu na max. rychlost 5000 min^{-1}

proměnná	definice	výsledek výpočtu	poznámka	
DEC(0)	1.0	1.00		← čas doběhu z max. rychlosti 5000 min^{-1}

(5) Provedení a popis programu

(1) Je proveden nájezd na VP nízkou rychlostí (směr zpět).
Sekvence nájezdu na VP začne při spuštění serva (servo ON). Nájezd na VP začíná, když je aktivován signál ORG. Výchozí poloha (VP) je dosažena v okamžiku kdy se hodnota signálu ORL změní z 0FF na ON.

(2) Program čeká, dokud nepřejde vstup X(11) do stavu OFF.

(3) Jsou-li svorky X(00) až X(11) = OFF, sevopohon se přesune na polohu P(00).

(4) Program se vrátí ke kroku (2).

(2)' Program čeká až vstup X(11) přejde do stavu OFF.

(3)' Je-li vstup X(00) = ON a vstupy X(01) až X(11) = OFF, servopohon se přesune na polohu P(01).

(4)' Program se vrátí ke kroku (2).

(2)“ Program čeká až vstup X(11) přejde do stavu OFF.

(3)'' Jsou-li vstupy X(00) a X(01) = ON a X(02) až X(11) = OFF, servopohon přejede na polohu P(03).

(4)'' Program se vrátí ke kroku (2).

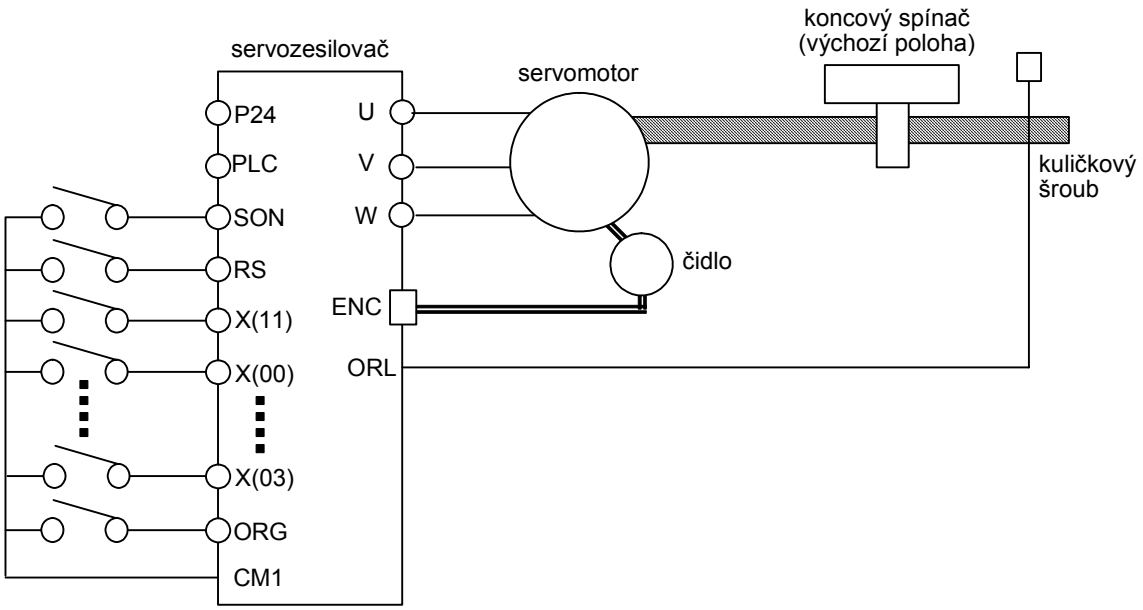
4.4 Volba polohy (zadání stavem vstupů)

(1) Popis příkladu

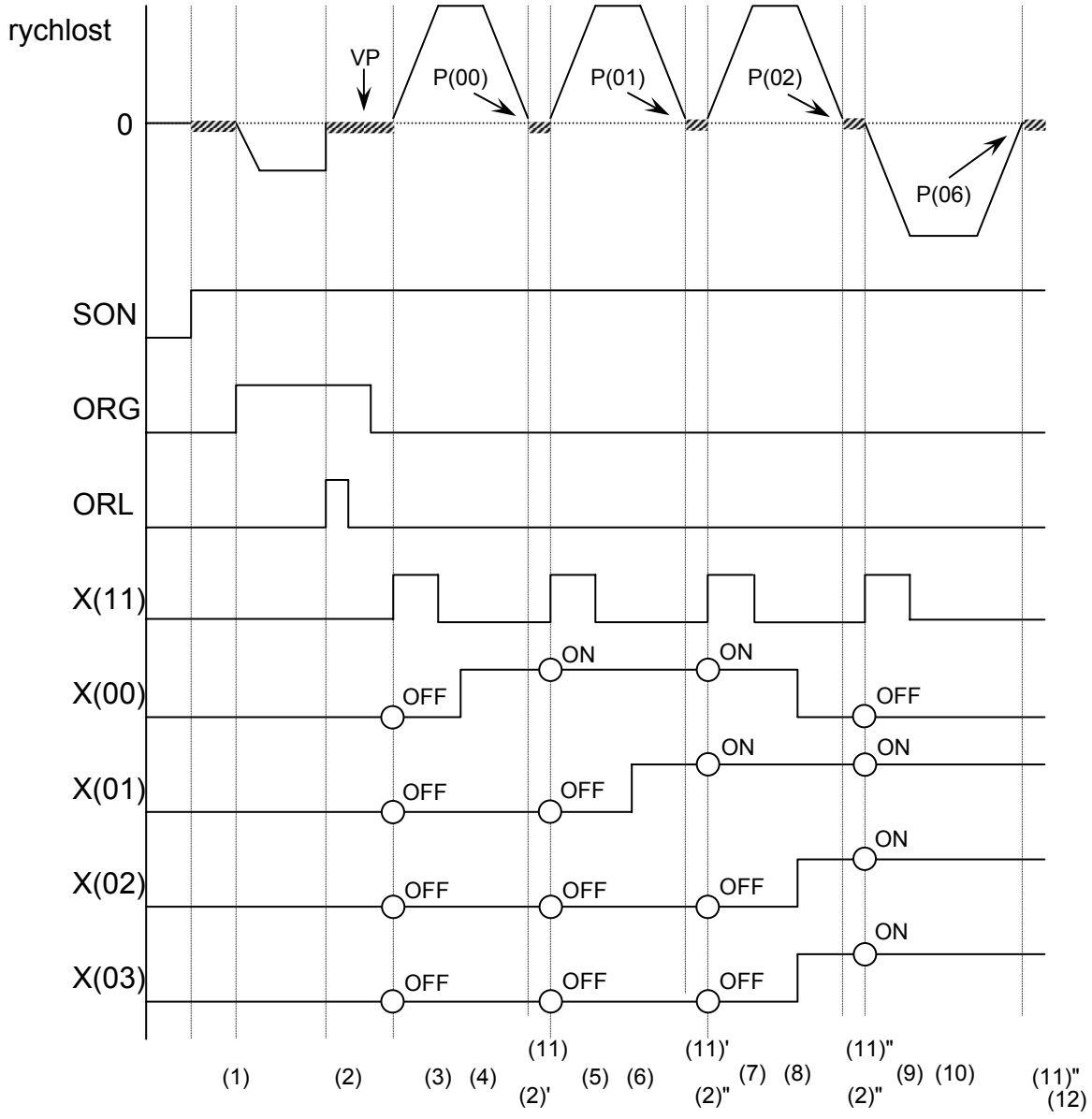
Je-li po dosažení VP vstupní svorka X(11) = 1, určuje stav vstupů X(00) až X(03) výslednou polohu pohonu P(Xw).

Vstupní svorky					cílová pozice
X(03)	X(02)	X(01)	X(00)	Data	
0	0	0	0	0	P(00)
0	0	0	1	1	P(01)
0	0	1	1	3	P(02)
0	1	1	0	6	P(03)
1	1	0	1	13	P(04)
1	1	1	1	15	P(05)
ostatní					P(06)

(2) Zapojení svorkovnice



(3) Časový diagram průběhu rychlosti



(4) Příklad programu

• Kódové okno

řádek	značka	instr. slovo	parm. 1	parm. 2	parm. 3	parm. 4	parm. 5	parm. 6	
001		entry							
002		chgORG=	default						
003		ort	1	N(02)		ACC(0)	DEC(0)		←(1)
	LOOP	wait	X(11)	=	1				←(2)
		U(00)=	Xw	and	15				
		select	U(00)						
		case	0						←(3)
		mov	P(00)						←(4)
		case	1						←(5)
		mov	P(01)						←(6)
		case	3						←(7)
		mov	P(02)						←(8)
		case	6						
		mov	P(03)						
		case	13						
		mov	P(04)						
		case	15						
		mov	P(05)						
		case else							←(9)
		mov	P(06)						←(10)
		end select							
		goto	LOOP						←(11)
		end							

• DataWindow

proměnná	definice	výsledek výpočtu	poznámka	
P(00)	&H8000*10	327680		←10 otáček=32768 × 10
P(01)	&H8000*20	655360		←20 otáček=32768 × 20
P(02)	&H8000*30	983040		←30 otáček=32768 × 30
P(03)	&H8000*40	1310720		←40 otáček=32768 × 40
P(04)	&H8000*50	1638400		←50 otáček=32768 × 50
P(05)	&H8000*60	1966080		←60 otáček=32768 × 60
P(06)	0	0		

proměnná	definice	výsledek výpočtu	poznámka	
N(00)	1000	1000		←1000min^{-1}
N(01)	3000	3000		←3000min^{-1}
N(02)	30	30		← 30min^{-1}

proměnná	definice	výsledek výpočtu	poznámka	
ACC(0)	1.0	1.00		← čas rozběhu na max. rychlost 5000 min^{-1}

proměnná	definice	výsledek výpočtu	poznámka	
DEC(0)	1.0	1.00		← čas doběhu z max. rychlosti 5000 min^{-1}

(5) Provádění a popis programu

(1) Dosažen výchozí pozice (VP).

(2) Program čeká až svorka X(11) přejde do stavu ON.
Bity X(11) až X(04) jsou překryty, aby neměly vliv na rozhodování.

(3)(4) Jsou-li svorky X(00) až X(3) = OFF, servopohon se přesune na polohu P(00).

(11) Program se vrátí na krok (2).

(2)' Program čeká, dokud svorka X(11) nepřejde do stavu ON.

(5)(6) Je-li svorka X(00) = ON a svorky X(01) až X(03) = OFF, servopohon se přesune na polohu P(01).

(11)' Program se vrátí na krok (2).

(2)'' Program čeká, dokud svorka X(11) nepřejde do stavu ON.

(7)(8) Jsou-li svorka X(00) = ON, X(01) = ON, a X(02) až X(3) = OFF, servopohon se přesune na polohu P(02).

(11)" Program se vrátí na krok (2).

(2)''' Program čeká, dokud svorka X(11) nepřejde do stavu ON.

(9)(10) Je-li svorka X(00) = OFF a svorky X(01) až X(3) = ON, servopohon se přesune na polohu P(06).

(11)''' Program se vrátí na krok (2).

(2)'''' Program čeká, dokud svorka X(11) nepřejde do stavu ON.

Poznámky

KAPITOLA 5 EDITOVÁNÍ PROGRAMU

Tato kapitola popisuje funkce editoru programu a jejich použití.

5.1 Zahájení a ukončení 5-2
5.2 Postup vypracování programu 5-4
5.3 Práce se soubory 5-5
5.4 Kódové okno – postupy zadávání 5-7
5.5 Editování programu 5-14
5.6 Datové okno – postupy zadávání 5-19
5.7 Sestavování programu 5-22
5.8 Nahrání programu do servopohonu 5-23
5.9 Dávkové zpracování programu
– sestavení / nahrání / chod 5-24
5.10 Načtení programu ze servopohonu 5-25
5.11 Ladění / chod programu 5-27
5.12 Vytištění programu 5-35
5.13 Zobrazení pomoci 5-36
5.14 Informace o verzi programu 5-36

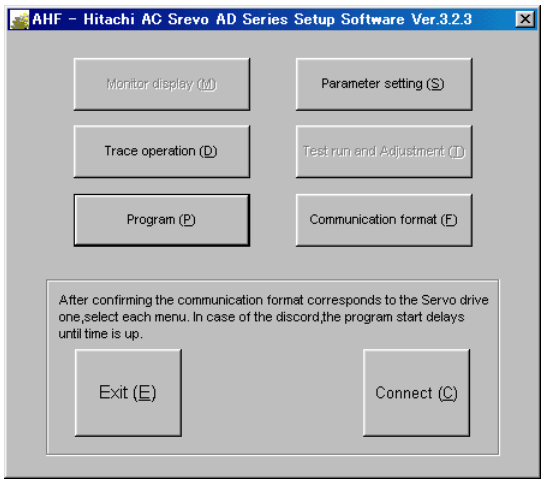
5.1 Zahájení a ukončení

Tato sekce znázorňuje, jak zahájit a ukončit vytváření programu.

(1) Zahájení

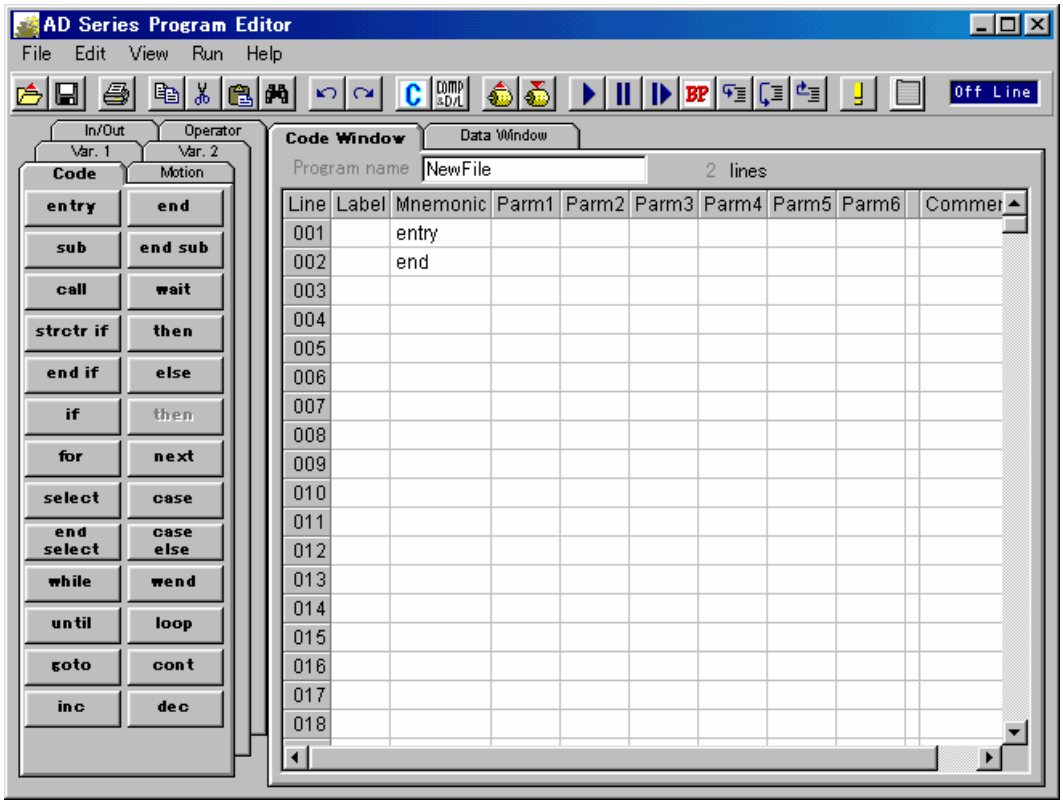
(1) V hlavní nabídce zvolte Programy (P) a následně přejeďte myší na AHF a zvolte kliknutím AHF.

(2) Program AHF se vám otevře základním oknem



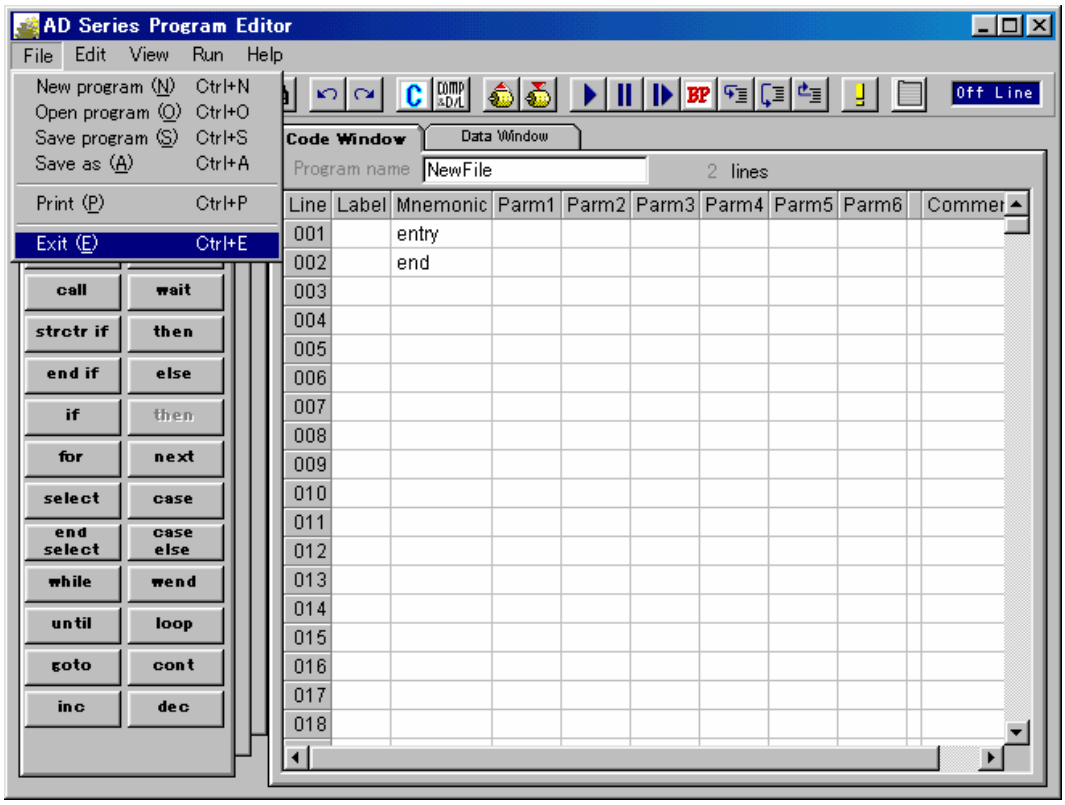
(3) Chcete-li mít při vytváření programu připojen pohon, klikněte na tlačítko připojit (Connect).

(4) Pokud vytváříte program bez připojení pohonu klikněte na program (P).

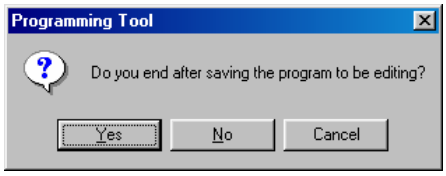


(2) Ukončení

(1) Stiskněte tlačítko Exit (E) v nabídce Soubor [File], nebo jednoduše stiskněte tlačítko ✕ v pravém horním rohu okna.



(2) Byl-li v rogramu proveden nějaký zásah objeví se následující dotaz:



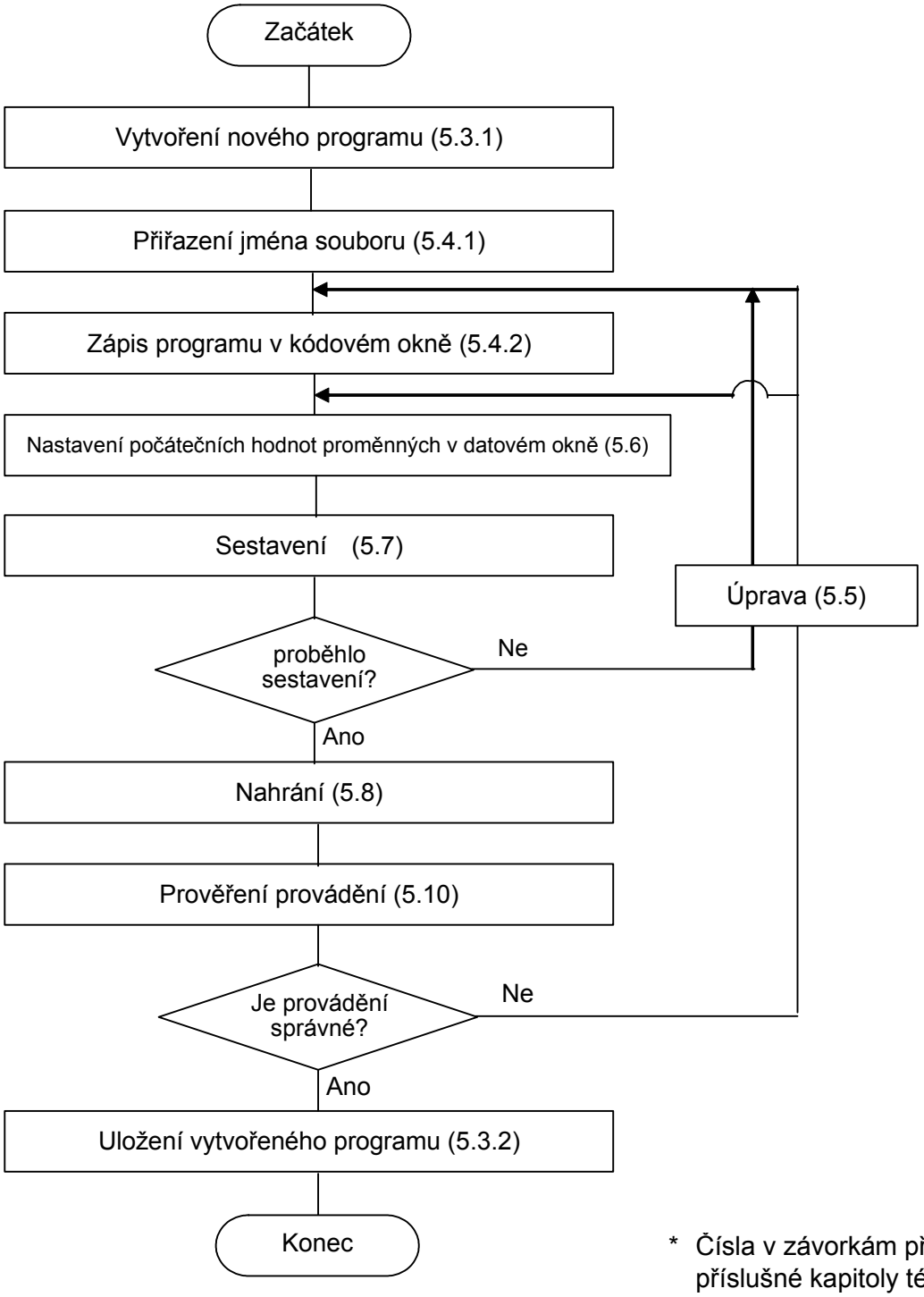
Yes (Y): Ano - ulož provedené změny a ukonči editaci.
No (N): Ne - ukonči editaci bez zápisu provedených změn.
Cancel: Přeruš - proces ukončení editoru se přeruší.

(3) Nebyla–li provedena žádná změna objeví se následující zpráva:

Yes (Y): Ano - ukonči editační program.
No (N): Přeruš - proces ukončení editoru se přeruší.

5.2 Postup vypracování programu

Níže je vývojový diagram znázorňující postup při vytváření uživatelského programu:



* Čísla v závorkám představují příslušné kapitoly této příručky

5.3 Práce se soubory

Tato část vysvětluje, jak uložit nebo načíst program vypracovaný v editoru.

(1) Vypracování nového programu

(1) V menu „Soubor“ [File] zvolte nový soubor [New (N)]. (Stejná akce se provede současným stisknutím kláves [Ctrl] a [N] na klávesnici.)

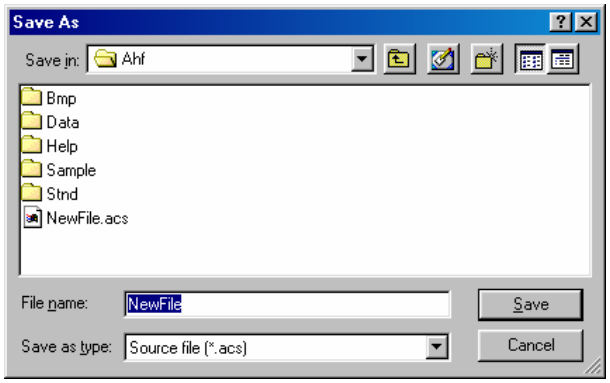
(2) Odložte právě editovaný program a začněte tvořit nový.

(2) Uložení souboru s novým programem

Za účelem uložení nového programu proveďte následující úkony:

(1) V nabídce „soubor“ [File] zvolte „uložit jako“ [Save As (A)]. (Stejná akce se provede současným stisknutím kláves [Ctrl] a [A].)

(2) Otevře se dialogové okno:



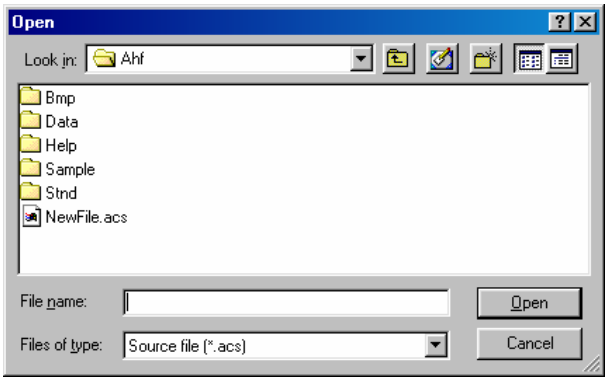
(3) Stisknutím tlačítka „ulož“ [Save (S)] je právě editovaný program uložen. Název souboru je shodný s názvem programu . Je možné zvolit dvě přípony "acs" nebo "csv" označující typ souboru.

Pozn.: Standardně používejte pro označení typu souboru příponu "acs". Příponu "csv" použijte, pokud vytváříte nebo editujete soubory v programu Excel nebo podobném.

(3) Otevření existujícího souboru

(1) V nabídce „soubor“ [File] zvolte „otevřít“ [Open (O)]. (Stejná akce se provede současným stisknutím kláves [Ctrl] a [O], nebo volbou tlačítka  .)

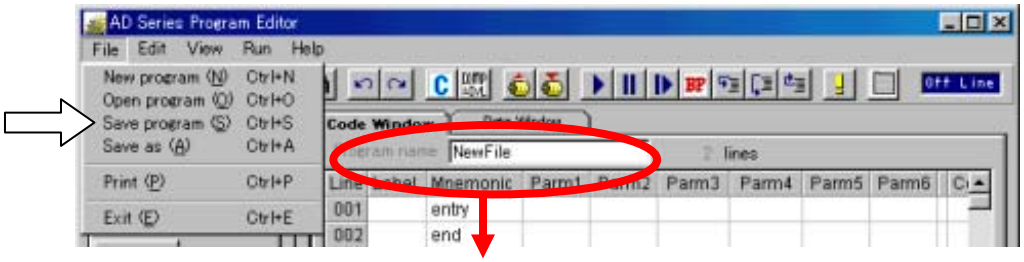
(2) Dialogové okno otevření souboru.



(3) Stiskem tlačítka „otevři“ [Open (O)] je načten vybraný program. Je možné otevřít typy souborů "acs" nebo "csv".

(4) Uložení souboru (přepsání původního souboru)

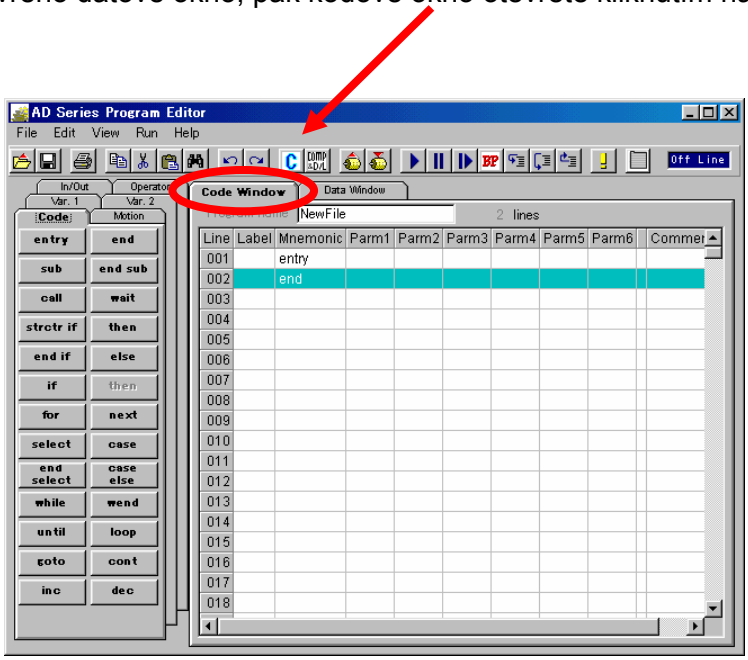
(1) V nabídce „soubor“ [File] zvolte „uložit“ [Save (S)]. (Stejná akce se provede současným stisknutím kláves [Ctrl] a [O], nebo volbou tlačítka  .)

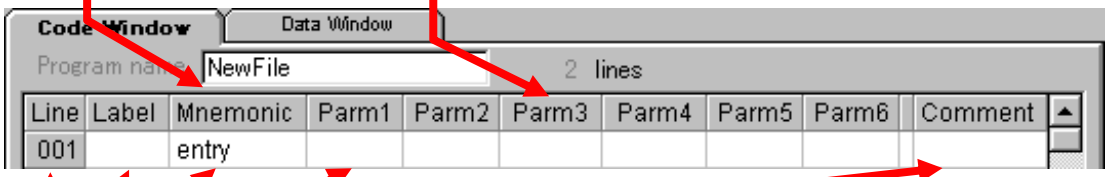


(2) Vámi vytvořený program je uložen do souboru se jménem právě užívaného programu.

5.4 Kódové okno – postupy zadávání

Je-li otevřeno datové okno, pak kódové okno otevřete kliknutím na tabulku kódového okna.





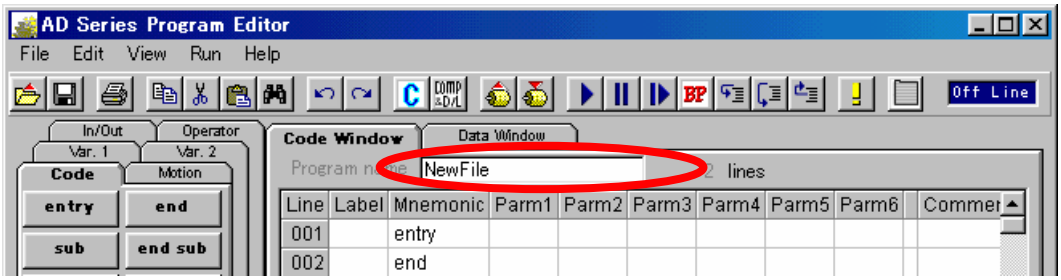
- zobrazuje se název programu
- zobrazuje se číslo právě editovaného programového řádku
- Do sloupce poznámka můžete zapsat jakékoliv poznatky k příslušné operaci. Obsah sloupce poznámka je uložen spolu programem nemá však žádný vliv na jeho sestavení a provádění. Proto je-li program nahrán do paměti servozesilovače a následně načten zpět do PC je obsah sloupce poznámka ztracen.
- Ve sloupcích Parm1 až Parm6, vložte parametry související s instrukcí vloženou ve sloupci instrukce (Mnemonic).
- Ve sloupci instrukce (Mnemonic) vložte proměnnou, do které se ukládá výsledek instrukce nebo výpočtu.
- Ve sloupci značka uveďte název nebo značku která označuje aktuální řádek. Přiřazený název nebo značka specifikuje adresu skoku realizovaného instrukcí IF nebo GOTO. Zvolená značka musí být pro každý řádek jiná (unikátní). Výskyt dvou stejných označení v programu je považováno za chybu, která se projeví při sestavování programu.
- V kódovém okně je možné vytvořit až 512 programových řádků. Výsledek sestavování programu nesmí překročit 512 kroků. Skutečný počet kroků se zobrazí na obrazovce po úspěšném ukončení sestavování programu.

- Platnost instrukcí a parametrů je prověřována ihned při jejich zadání. Je-li některé zadání nesprávné objeví se chybové hlášení a zadané hodnoty nejsou přijaty. Nelze zadávat žádné hodnoty do buněk, které žádné zadání nevyžadují

(1) Zadání názvu souboru

Pole název programu zobrazuje jméno právě editovaného programu. V případě nového započetí práce s editorem, nebo zvolení nového souboru se objeví "NewFile".

Nejdříve proveďte zadání jména souboru, aby nemohlo dojít nechtěnému vymazání, nebo nabourání dříve editovaného programu.



(I v případě uložení již pojmenovaného souboru lze jeho název změnit (viz odstavec 5.2.2))

(2) Zápis programu

A) Zadávání hodnot z tabulky kódů

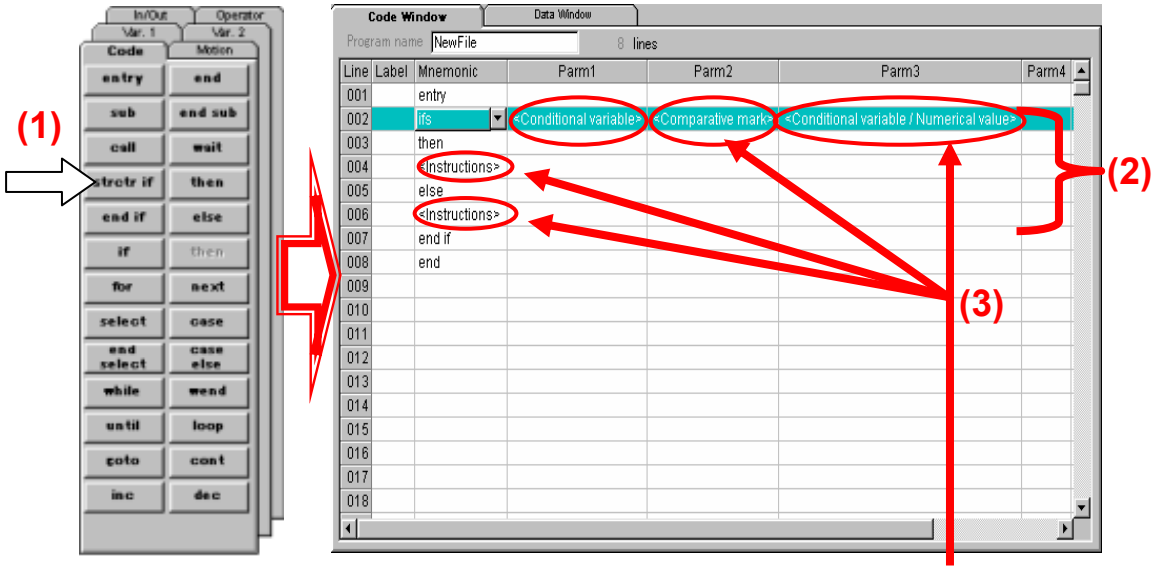
Editovací program Vám umožní velmi jednoduché zadávání pomocí tabulky kódů. Jste-li nejistí v zadávání syntaxe, prosím použijte tuto metodu zadání.

Kliknutím na tlačítko kódu v kódové tabulce jej přenesete do zvolené buňky kódového okna.

<Příklad provedení>

(1) Klikněte na instrukční tlačítko [Structure if] v kódové tabulce.

(2) Instrukce [Structure if] se přenese do právě aktivního řádku kódového okna.



(3) U příkazů, které jsou napsány v kódovém okně a mají připojenu buňku s vysvětlením v závorkách (< >), musíte vložit instrukci parametr atd.

(Pokud neprovedete do takového buňky potřebný zápis, program usoudí při sestavování, že buňka nemá žádný vstup)

Příklad)

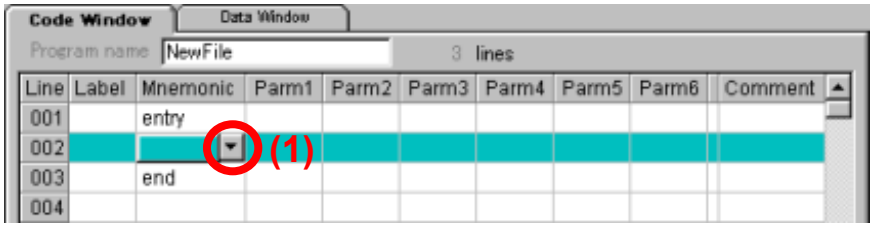
<podmíněná proměnná>
<podmíněná proměnná/číselná hodnota> ... proměnná nebo číslo (-128 až +127)
<kód porovnání>..... "=","<","<=",">",">=","<>"
<instrukce> ..... některá instrukce nebo výpočtová proměnná

* Nelze kliknout na tlačítka instrukcí / proměnných, které nejsou určeny k zadání do zvolené buňky.

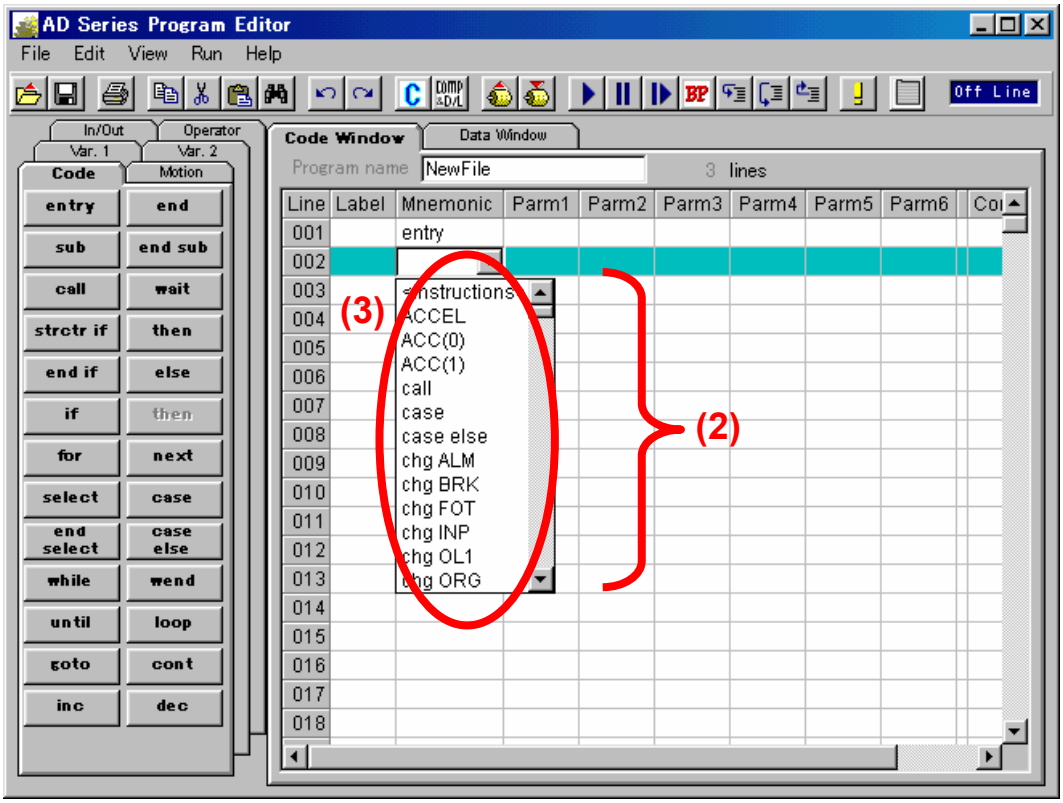
B) Vkládání hodnot ze seznamu

Zvolíte-li buňku, která vyžaduje zadání, může se v pravém konci buňky objevit tlačítko , které značí , že je možné vybrat vstupující hodnoty ze seznamu. Instrukce, které vstupují do instrukčního pole (Mnemonics) nelze vybrat ze seznamu. Musíte je vložit z tabulky kódů, nebo zadat z klávesnice.

(1) Klikněte myší na tlačítko ,nebo stiskněte tlačítko [Enter] na klávesnici.



(2) Otevře se seznam parametrů, které lze vložit do zvolené buňky.



(3) Tlačítky nahoru/dolů je možné zobrazit další skryté parametry seznamu.

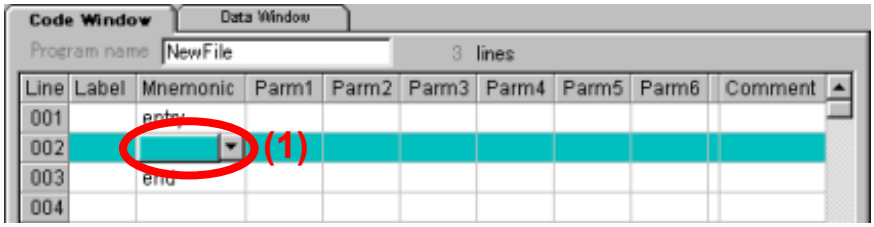
(4) Najděte parametr, který hodláte vložit a klikněte na něj myší. Stejného cíle dosáhnete použijete-li tlačítka [↑] a [↓] a [Enter] na klávesnici.

(5) Zvoleny parametr je vložen do příslušného pole.

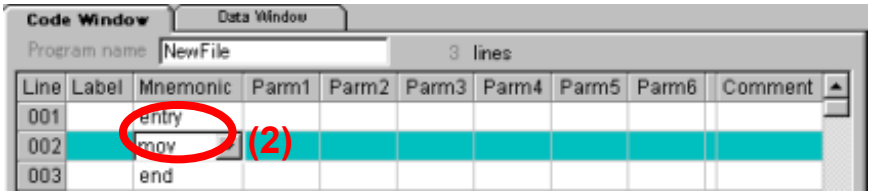
C) Vkládání hodnot z klávesnice

Je možné také vkládat obsah buňky přímo z klávesnice.

(1) Najeďte kursorem na buňku, do které chcete provést zápis a klikněte.

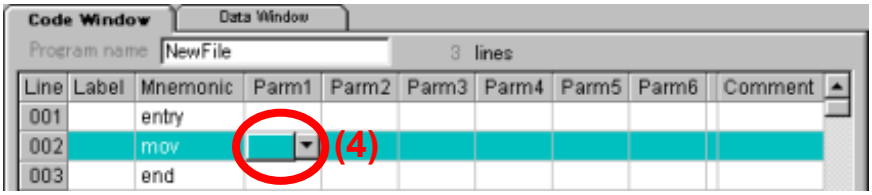


(2) Proveďte zápis hodnot z klávesnice.



(3) Po dokončení zápisu instrukce stiskněte tlačítko [Tab] nebo [→] key.

(4) Aktivní se stane buňka pro zápis parametrů dané instrukce.



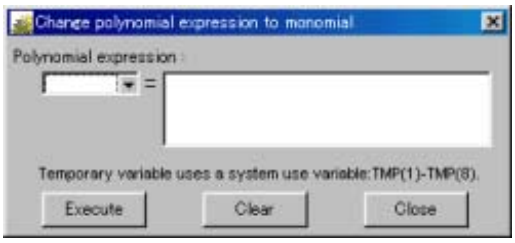
* Pokud uděláte v zadání nějakou chybu objeví se hlášení chyby a zápis se neprovede.

(3) Zadání výpočtu polynomu

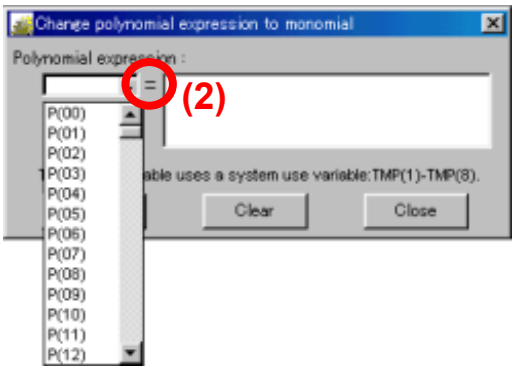
Servopohon je schopen přímo zpracovávat funkce zadané polynomem. Zpracování polynomu se děje automatickým převodem na dvoučlenný výraz, který je zpracován servopohonem. Pro převod polynomu na dvoučlennou funkci jsou použity dočasné (systémové) proměnné TMP(1) až TMP(8), kde jsou uchovány potřebné mezi výsledky.

* Systémové proměnné jsou používány pouze překladačem a na rozdíl od uživatelských proměnných je nelze využívat pro jiné operace.

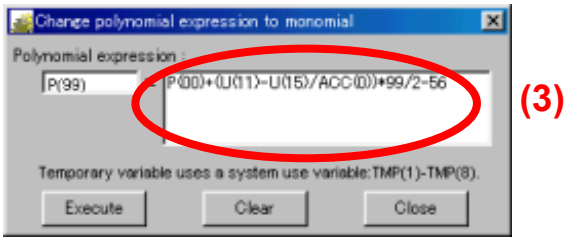
(1) V nabídce „Proveď “ [Execute] zvolte „Polynom“ [Polynomial]. Otevře se Vám okno převodu polynomu na dvoučlen. (stejného výsledku dosáhnete současným stiskem kláves [Ctrl] a [T] na klávesnici).



(2) Na levou stranu rovnice zvolte proměnnou, do které bude zapsán výsledek výpočtu. Kliknutím na klávesu ▼ se vám otevře nabídka proměnných.

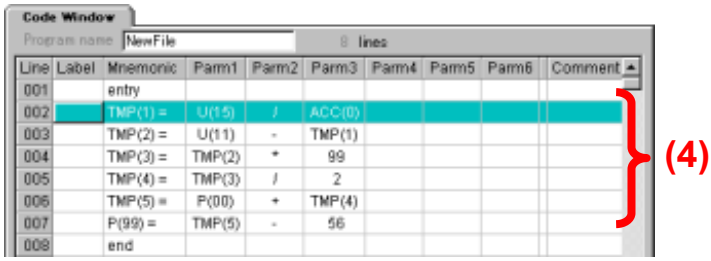


(3) Na pravou stranu rovnice pište polynomický výraz.



Klávesou „Vymaž“ [Clear] lze vymazat celý polynom.

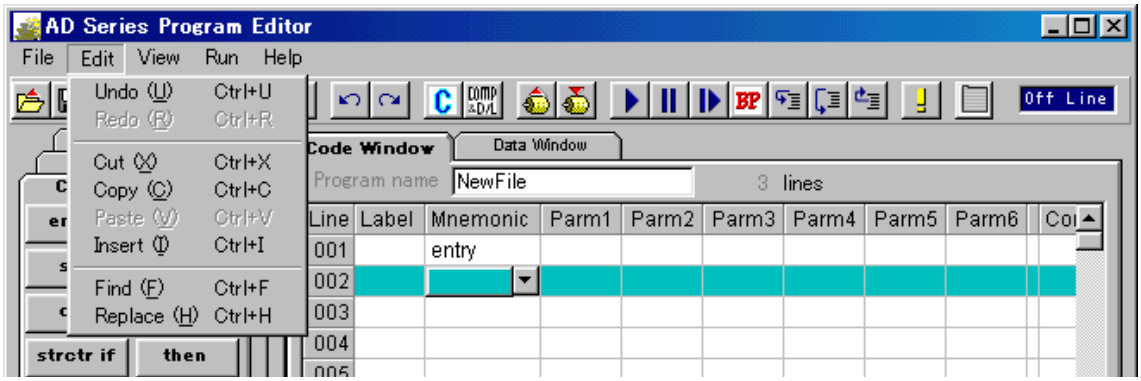
(4) Klikněte na tlačítko „Dvoučlen“ [Binomial]. V kódovém okně se Vám zobrazí převod Vašeho polynomu na dvoučlenné výrazy.



* Pokud je polynom příliš dlouhý zobrazí se chybové hlášení vzniklé nedostatkem systémových proměnných.

5.5 Editování programu

Tato sekce se zabývá editačními funkcemi programování.

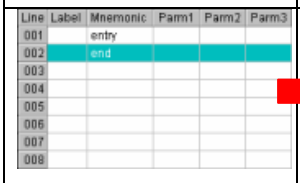
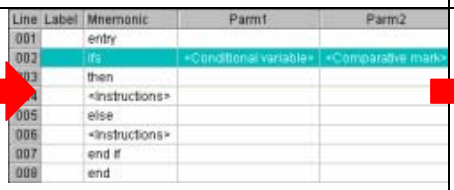
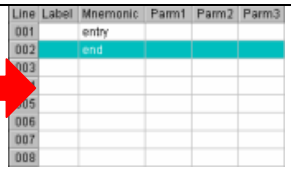


Funkce editace lze použít nejen v kódovém ale i v datovém okně (viz část 5.6).

(1) Zpět (Undo)

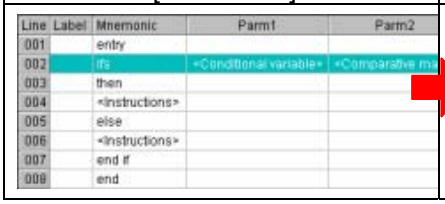
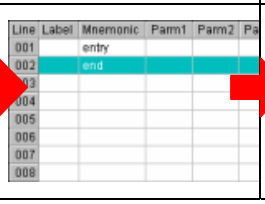
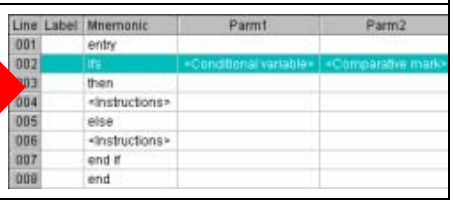
Tato funkce je dostupná po provedení jakékoliv vstupní akce. Jejím úkolem je vrátit právě provedenou akci zpět do předchozího stavu. Funkci aktivujete kliknutím na tlačítko [Undo (R)] v nabídce [Edit] (stejnou akci je možné provést kliknutím na tlačítko  na liště, nebo současným stisknutím kláves [Ctrl] a [R] na klávesnici).

<Příklad provedení>

(1) počáteční stav	(2) kliknutí na tlačítko [Structure if] v tabulce vstupních kódů	(3) provedení funkce zpět "Undo"
		

(2) Vpřed (Redo)

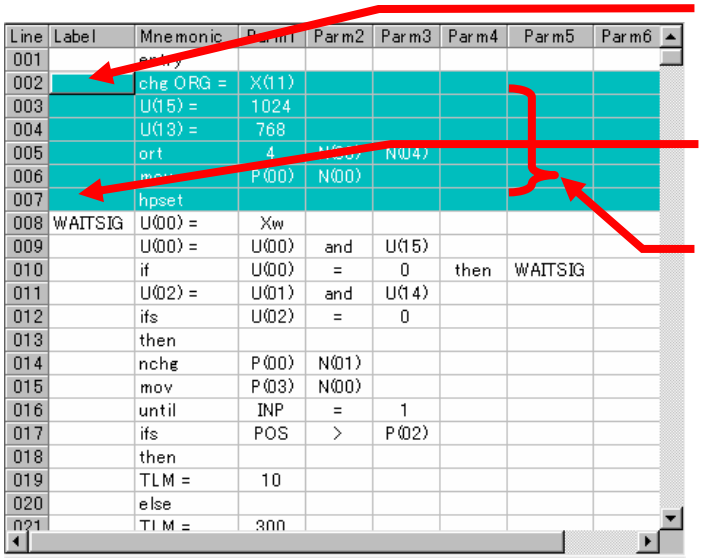
Tuto funkci lze použít po využití funkce "Undo", k opětnému provedení dříve vrácené akce. Funkci aktivujete kliknutím na tlačítko [Redo (U)] v nabídce [Edit] (stejnou akci je možné provést kliknutím na tlačítko  na liště, nebo současným stisknutím kláves [Ctrl] a [U] na klávesnici).

(1) počáteční stav po kliknutí na tlačítko [Structure if]	(2) po provedení funkce "Undo"	(3) po provedení funkce "Redo"
		

(3) Vyjmout (Cut)

Tato funkce vyjme z programu jeden nebo více zvolených řádků. Funkci lze využít k vymazání nepotřebných řádku (ev. k přenosu na jiné místo).

(1) e řádky které hodláte vyjmout.



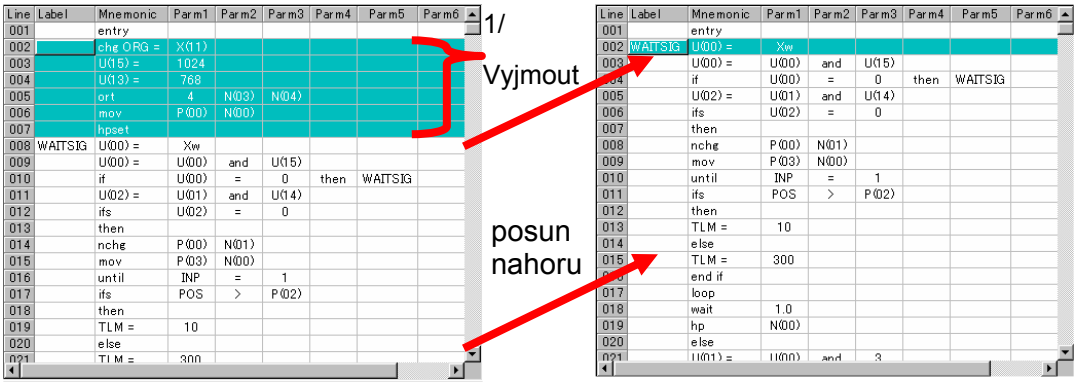
(1) Klikněte a držte levé tlačítko myši na prvním řádku, který má být vyjmut.

(2) Táhněte myší až na poslední řádek určený k vyjmutí (levé tlačítko myši pořád držíte).

(3) Pustíme levé tlačítko myši. Zvolené řádky jsou probarveny modře.

(2) V nabídce úprav [Edit], klikněte na tlačítko vyjmout [Cut (X)] (stejnou akci je možné provést kliknutím na tlačítko  na liště, nebo současným stisknutím kláves [Ctrl] a [X]).

(3) Probarvené řádky jsou nyní vyjmuty a zůstávají uloženy pro pozdější využití funkce „Vložit“ [Paste]. Řádky, které následují po vyjmutých řádcích se posunou



výše.

(4) Kopírovat (Copy)

Tato funkce umožňuje kopírovat jeden nebo více zvolených řádků.

(1) Zvolte řádky, které hodláte kopírovat.

(2) V nabídce úprav [Edit], klikněte na tlačítko kopírovat [Copy (C)] (stejnou akci je možné provést kliknutím na tlačítko  na liště, nebo současným stisknutím kláves [Ctrl] a [C] na klávesnici).

(3) Zvolené řádky jsou nyní zapamatovány a pro pozdější vložení na zvolené místo při aktivaci funkce vložit [Paste].

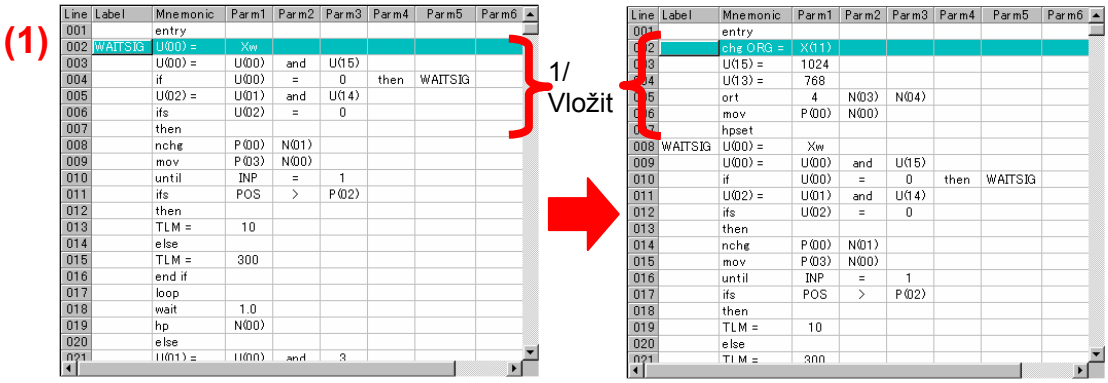
(5) Vložit (Paste)

Tato funkce se používá k vložení jednoho nebo více vyjmutých (Cut) nebo kopírovaných (Copy) řádků.

(1) Zvolte řádky, které budou přepsány vyjmutými nebo kopírovaným řádky.

(2) V nabídce úprav [Edit], klikněte na tlačítko vložit [Paste (V)] (stejnou akci je možné provést kliknutím na tlačítko  na liště, nebo současným stisknutím kláves [Ctrl] a [V]).

(3) Zvolené řádky jsou přepsány vyjmutými nebo kopírovanými řádky.



(6) Vlož řádek

Tato funkce vloží na zvolené místo jeden volný řádek.

(1) Zvolte místo, kam hodláte vložit volný řádek.

(2) V nabídce úprav [Edit], klikněte na tlačítko vložit řádek [Insert One Line (I) (stejnou akci je možné provést současným stisknutím kláves [Ctrl] a [I]).

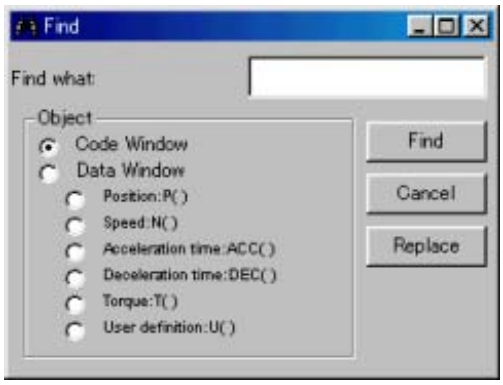
(3) Volný řádek je vložen bezprostředně před řádek označující místo k vložení.

(7) Vyhledat (Find)

Tato funkce slouží k vyhledávání znakových řetězců.

(1)

(2) Otevře se dialogové okno „vyhledat“ [Find].



(3) Vložte znakový řetězec, který chcete vyhledat. Stisknutím tlačítka "Find" započne vyhledávání.

(4) Pokud stiskněte tlačítko přeruš [Cancel] dialogové okno vyhledávání se zavře.

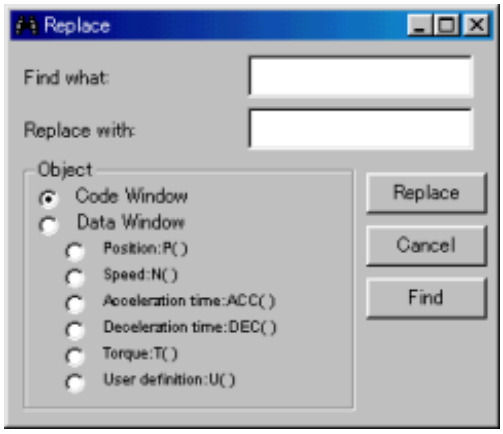
(5) Stisknete-li tlačítko nahradit [Replace] tak systém přejde do režimu nahrazení (viz odstavec 5.5.8).

(8) Nahradit (Replace)

Tato funkce vyhledá zadaný znakový řetězec a nahradí jej jiným řetězcem.

(1) V nabídce úprav [Edit], klikněte na tlačítko nahradit [Replace (H)] (stejnou akci je možné provést současným stisknutím kláves [Ctrl] a [H]).

(2) Otevře se dialogové okno „nahradit“ [Replace].



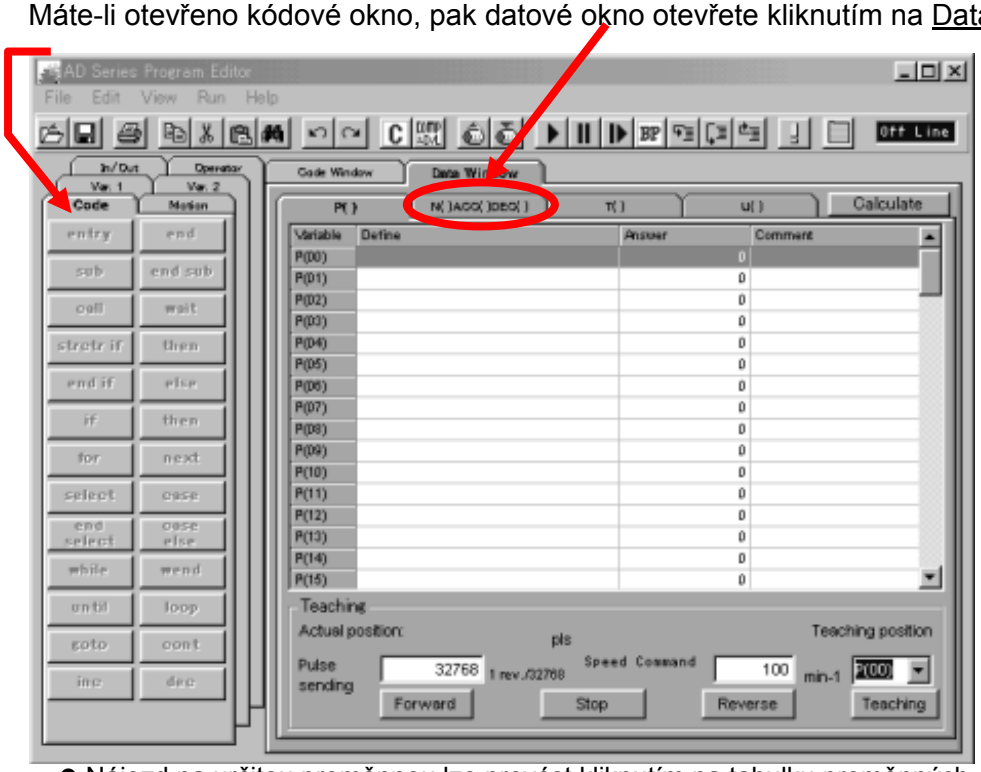
(3) Vložte hledaný znakový řetězec a řetězec, kterým má být nahrazen. Stisknutím tlačítka „nahradit“ [Replace] se akce provede.

(4) Kliknete-li na tlačítko „přerušit“ [Cancel], dialogové okno nahrazení se uzavře.

(5) Kliknete-li na tlačítko „vyhledat“ [Find], systém přejde do režimu vyhledávání (viz odstavec 5.5.7).

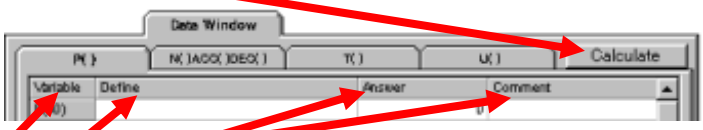
5.6 Datové okno – postupy zadávání

Datové okno (okno pro zadávání dat) slouží k zadávání počátečních hodnot různých proměnných na začátkem provádění programu.
Máte-li otevřeno kódové okno, pak datové okno otevřete kliknutím na Data Window tab.



● Nájezd na určitou proměnnou lze provést kliknutím na tabulku proměnných P()/N()ACC()DEC()/T()/U().

● Kliknutím tlačítka "Calculate" se provede výpočet a nastavení proměnné.



● Do sloupce poznámka můžete zapsat jakékoliv poznatky k příslušné operaci. Obsah je uložen spolu programem nemá však žádný vliv na jeho sestavení a provádění. Proto je-li program nahrán do paměti servozesilovače a následně načten zpět do PC je obsah sloupce poznámka ztracen.

● Sloupec odpověď ukazuje nastavení před započetím provádění programu.

● Do sloupce definice zapište nastavení a výpočtový vzorec proměnné.

● Sloupec proměnné obsahuje názvy proměnných. Pro pohyb ve sloupci proměnných použijte šipek nahoru a dolů v pravé okraji okna.

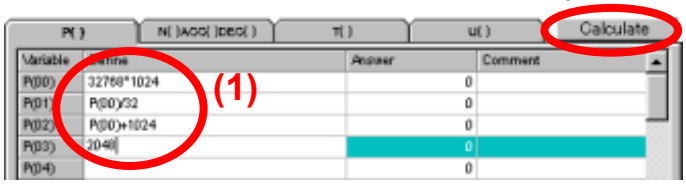
(1) Vkládání proměnných

Nastavení proměnné, vložení definičního vzorce a okamžité hodnoty.

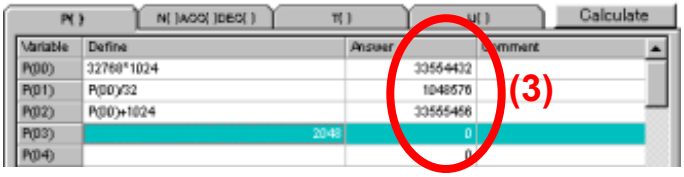
(1) Vložte definiční vzorec proměnné a její okamžitou hodnotu do definičního sloupce.

Výpočtový vzorec může obsahovat znaménka operace sčítání, odečítání násobení, dělení a závorky. Okamžitá hodnota a výsledky výpočtu se musí pohybovat v rámci nastavení horního a dolního omezení.

(2) Po provedení zadání klikněte na tlačítko „výpočet“ [Calculate].



(3) Výsledek výpočtu se zobrazí v odpovědním sloupci (Answer).



* Je-li v definičním sloupci proměnné odkaz na další proměnnou, zobrazí se při výpočtu v odpovědním sloupci chybové hlášení „výpočet není možný“ "<Calculation impossible>".

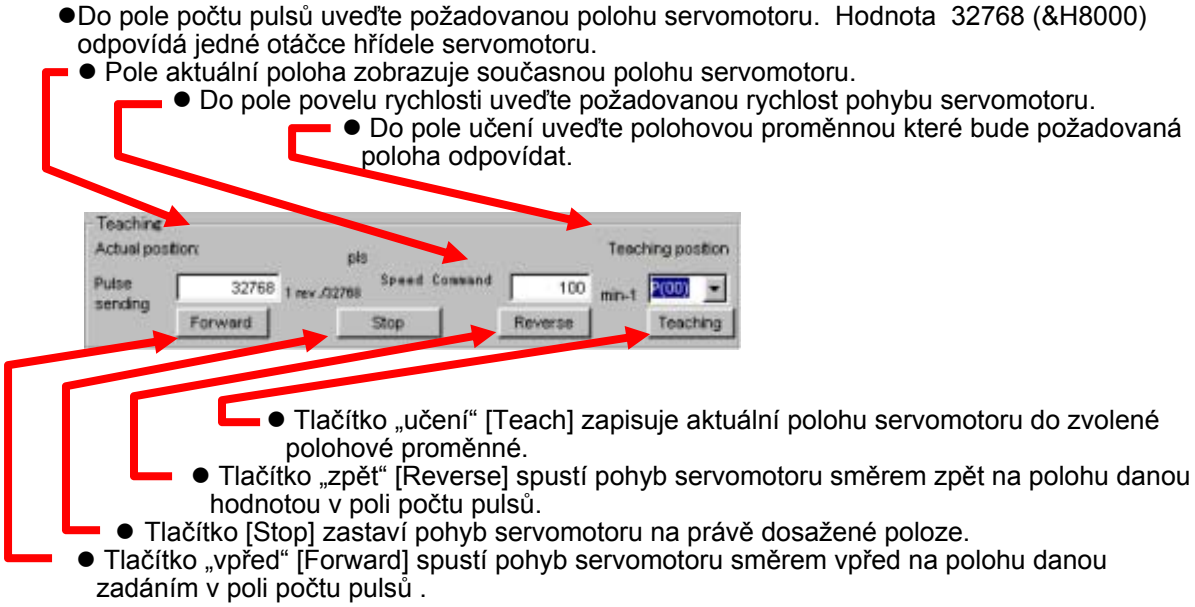
* Je-li nastavená hodnota proměnné mimo dovolené meze zobrazí se v odpovědním sloupci zpráva „ mimo rozsah“ "<Out of range>".

Horní dolní omezení proměnných obsahuje následující tabulka:

pořadové číslo	Proměnná	Horní a dolní omezení
1	polohové proměnné (P(00) až P(99))	+2147483647 až -2147483648(± 0x7FFFFFFF)
2	rychlostní proměnné (N(00) až N(15))	5000 až 0
3	doba rozběhu (ACC(0) až ACC(1))	99.99 až 0
4	doba doběhu (DEC(0) až DEC(1))	99.99 až 0
5	proměnné momentu (T(0) až T(15))	+300 až -300
6	uživatelské proměnné (U(0) až U(15))	+2147483647 až -2147483648(± 0x7FFFFFFF)

(2) Učení se

Použití funkce učení Vám umožní přímo dovést servomotor na požadovanou polohu. Proces učení lze řídit programem AHF nebo operátorským panelem (OP). Popis ovládání pomocí OP je uveden v kapitole 4 uživatelské příručky servopohonu.



(1) Připojte servopohon a zapište polohu do pole počtu pulsů [Pulse Feed].

(2) Je-li požadovaná poloha z hlediska současné polohy směrem vpřed klikněte na tlačítko „vpřed“ [Forward], je-li požadovaná poloha vzhledem k současné směrem vzad, klikněte na tlačítko „vzad“ [Reverse].

(3) Servomotor se roztočí dle provedeného nastavení.

(4) Jakmile se v poli aktuální poloha [Current Position] zobrazí poloha shodná s polem počtu pulsů [Pulse Feed] servomotor se zastaví.

(5) Opakujte kroky (1) až (4) dokud neodpovídá poloha servomotoru Vámi požadované poloze.

(6) Do pole učení [Teaching Position] zapište proměnnou, do které bude uložena aktuální poloha.

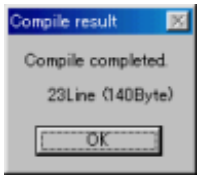
(7) Klikněte tlačítko učení [Teach]. Hodnota zobrazená v poli aktuální polohy [Current Position] je přepsána do definičního pole proměnné určené v kroku 6.

5.7 Sestavování programu

Po dokončení zadávání programu započněte se sestavováním.

(1) V nabídce provádění [Execute] klikněte tlačítko sestavení [Compile]. (stejnou akci je možné provést kliknutím na tlačítko  na liště, nebo současným stisknutím kláves [Shift] a [F9]).

(2) Provede se sestavení programu. Nedojde-li v průběhu sestavení k žádné chybě zobrazí se hlášení úspěšného ukončení sestavování a můžeme přistoupit k nahrání programu do servopohonu.



Pokud zvolíte v nabídce zobrazení [View] možnost zobrazení úrovně indexace (vnoření) [Show Nesting Structure for Compiling], pak se Vám ve sloupci značka zobrazí barevné rozlišení jednotlivých vnořených podprogramů (smyček), což usnadní porozumění celkové struktuře programu.

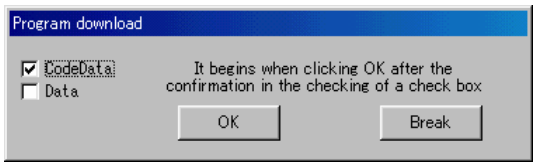
* Dojde-li v průběhu sestavování k chybě, zobrazí se chybové hlášení, které obsahuje příčinu chyby. Proveďte korekci zjištěné chyby a proces sestavení zopakujte.

5.8 Nahrání programu do servopohonu

Proběhlo-li sestavení programu úspěšně, můžeme program nahrát do paměti servopohonu. Nahrání programu lze provést pouze v případě je-li servopohon v klidu (servo OFF). Je-li servopohon v chodu (servo ON) nelze nahrání programu provést.

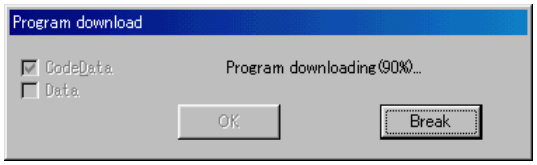
(1) V nabídce provádění [Execute] klikněte tlačítko nahrání [Download] (stejnou akci je možné provést kliknutím na tlačítko  na liště, nebo současným stisknutím kláves [Shift] a [F11]).

(2) Otevře se dialogové okno nahrávání programu.

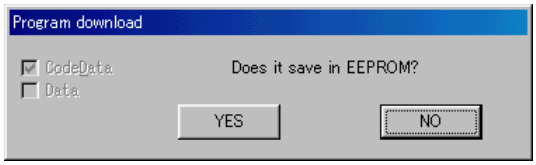


(3) Lze kontrolovat nahrávání obsahu programu datového i kódového okna, nebo pouze datového okna.

(4) Zahájení nahrávání provedete stiskem tlačítka [OK].



(5) Je-li nahrávání programu úspěšně dokončeno, otevře se dialogové okno ve kterém je dotaz, zda požadujete nahraný program uložit do paměti EEPROM servopohonu. Normální odpověď je ano [YES].



* Pokud jste při spuštění programu AHF nestiskli lačítko připojit [Connect] dojde k chybě. Je proto potřeba provést připojení (viz odstavec 5.1.1).

*Nelze-li provést nahrání programu (např. proto, že je servopohon v chodu) zobrazí se dialogové okno s následující zprávou. V tomto případě proveďte úkony zobrazené ve zprávě a pokus o nahrání opakujte.

5.9 Dávkové zpracování programu – sestavení / nahrání / chod

Sestavení programu, nahrání a ověření běhu lze provést v jednom kroku.

Funkce dávkové zpracování programu - sestavení a nahrání

V nabídce provádění [Execute] klikněte tlačítko sestavení a nahrání [Compile & Download] (stejnou akci je možné provést kliknutím na tlačítko  na liště.

Sestavení programu přejde plynule v jeho nahrání (pokud se nevyskytne chyba při sestavení). V případě chyby postupujte dle návodu v hlášení.

Funkce dávkové zpracování programu - sestavení a nahrání a chodu

V nabídce provádění [Execute] klikněte tlačítko sestavení a nahrání a chod [Compile & Download & Run].

Sestavení a nahrání programu přejde plynule v jeho provádění (pokud se nevyskytne chyba v sestavení). V případě chyby postupujte dle návodu v hlášení

5.10 Načtení programu ze servopohonu

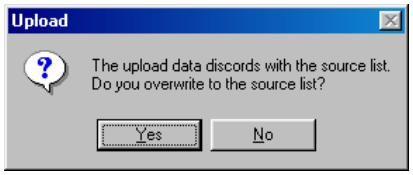
Načtení programu ze servopohonu lze provést následujícím způsobem:

(1) V nabídce provádění [Execute] klikněte tlačítko načtení [Upload] (stejnou akci je možné provést kliknutím na tlačítko  na liště, nebo současným stisknutím kláves [Shift] a [F12]).

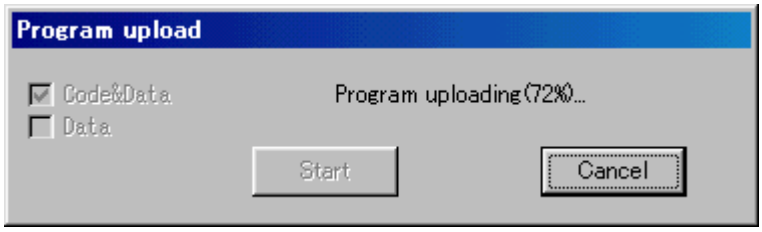
(2) Otevře se dialogové okno načtení programu.

(3) Lze kontrolovat nahrávání obsahu programu datového i kódového okna, nebo pouze datového okna.

(4) Po stisknutí tlačítka [Yes] budete dotázání, zda chcete přepsat právě editovaný program programem ze servopohonu.

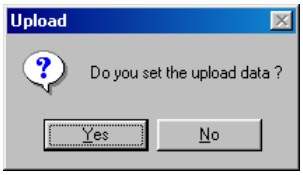


(5) Stisknutím tlačítka [Start] začne načítání.



(6) Pokud je načítání ukončeno korektně, objeví se hlášení správného ukončení.

(7) Nyní jste dotazování, zda chcete finalizovat načtená data.



(8) Stisknete-li tlačítko [Yes], načtená data jsou finalizována a zobrazena.
- Veškerý obsah sloupce poznámek je převeden jako prázdný (ztracen).
- Označení a názvy podprogramů jsou převedeny dle konvence uvedené níže, tak že se odlišují od nahraných zdrojových kódů.

  Názvy značek: První značka má název LBL00. Pořadové číslo značky je automaticky zvyšováno s každým následujícím řádkem.

  Názvy podprogramů: První podprogram má název SUB00. Pořadové číslo podprogramů je automaticky zvyšováno s každým následujícím podprogramem.

(9) Pokud kliknete na tlačítko [No] místo [Yes], systém zruší načtená data a vrátí se k původně zobrazeným datům (před načtením).

5.11 Ladění / chod programu

(1) Hlavní funkce ladění

Lze použít následující funkce ladění. použijte funkce, které jsou potřeba.

Název ladící funkce	Popis funkce	Prověřované skutečnosti
přerušení	zastaví provádění programu v určeném bodě Lze nastavit až čtyři přerušení (lze prohlédnout hodnoty proměnných)	• prověření běhu programu • prověření hodnot proměnných
jednotlivý krok	prověřuje provádění programu krok po kroku	• prověření běhu programu • prověření hodnot proměnných
zobrazení prováděného řádku	umožňuje prohlédnutí právě prováděného řádku programu	• hrubé prověření chodu programu
hlídání proměnné	umožňuje prověřovat hodnotu proměnné označené kursorem při přerušení, nebo provádění pojednotlivých krocích	• prověření hodnot proměnné
zobrazení vnoření (při sestavování)	zobrazuje, jaké je vnoření podprogramů	• vizuální prověření struktury programu
okno ladění komunikace	prověřuje chod uživatelského programu, pokud obsahuje komunikační funkce (program AHF řídí tiskové (print#) a vstupní (input#) funkce V/V dat při ladění komunikace)	• okno ladění je použito k vizuální kontrole dat komunikačního výstupu (instrukce print#) ze servopohonu. • vložením numerické hodnoty v okně ladění je prověřována reakce servopohonu (instrukce input#)

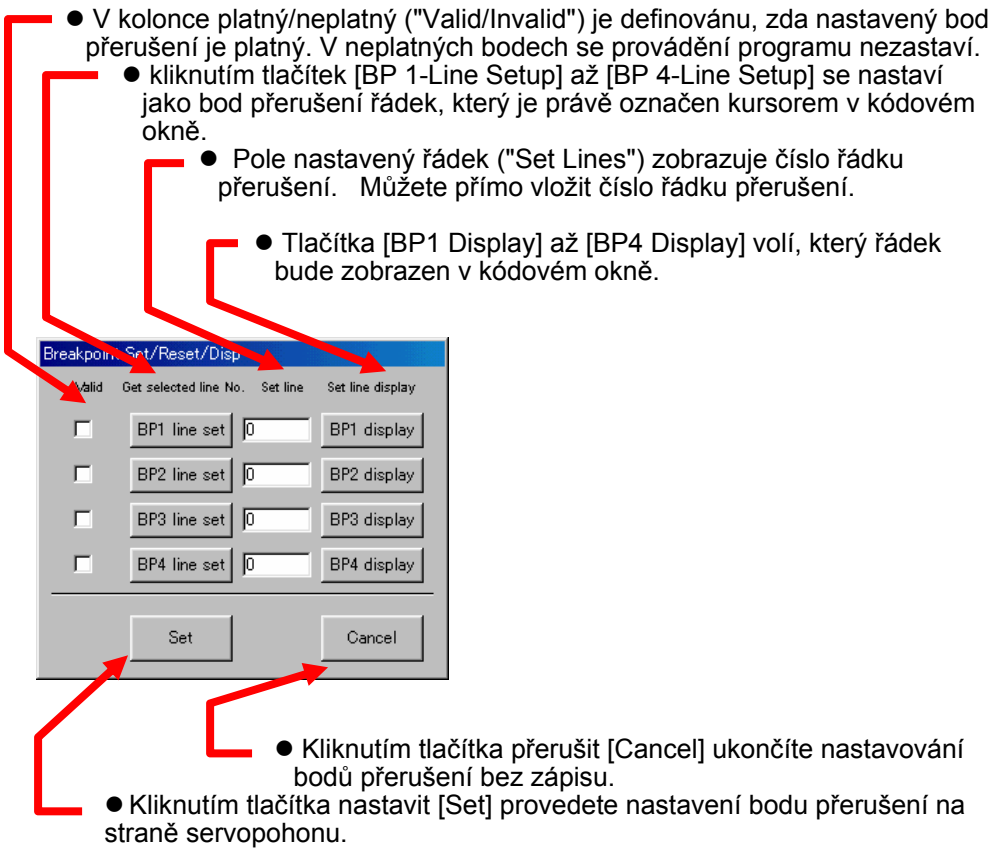
(2) Nastavení přerušení

Nastavit body přerušení je možné pro program, který je nahrán do servopohonu. Body přerušení jsou místa, ve kterých má být provádění programu zastaveno.

(1) V nabídce provádění [Execute] klikněte tlačítko body přerušení [Breakpoints (B)]

(stejnou akci je možné provést kliknutím na tlačítko  na liště, nebo současným stisknutím kláves [Ctrl] a [B]).

(2) Otevře se dialogové okno nastavení přerušení.



Pozn.1: V režimu přerušení nebo provádění po krocích se program zastaví na zvoleném řádku, bez toho, že by tento řádek provedl. Instrukce se provede až po proběhnutí kontroly správnosti.

Pozn.2: Nezapomeňte na to, že pohybové instrukce (např., mov) se i při režimu přerušení, nebo provádění po krocích, provádějí kontinuálně.

NOTE 3: V bodě přerušení změní kursor myši barvu na žlutou. V tomto okamžiku můžete zkontrolovat hodnotu proměnné tím, že na ní najedete takto zbarveným kursorem.

(2) Provádění a zobrazení prováděného řádku

Provádění programu nahraného do servopohonu započne od příkazu "entry" a skončí příkazem "end" nebo prvním nastaveným bodem přerušení.

- V nabídce provádění [Execute] klikněte tlačítko chod [Run]. (stejnou akci je možné provést kliknutím na tlačítko  na liště, nebo stisknutím klávesy [F5]).

Je-li v nabídce zobrazení [View] zvoleno zobrazení prováděného kroku [Display Run Steps], pak je prováděny řádek zobrazen modře. Nechcete-li modré zobrazení prováděných řádku, pak v nabídce [View] zobrazení zrušte volbu zobrazení prováděného kroku [Display Run Steps

Jakmile začne provádění programu bude obrazovka vypadat tak, jak je uvedeno na obrázku dále a objeví se upozornění chod (On Line).

Pozn.1: Je-li zobrazovací rychlost monitoru nízká, může se zdát, že některé instrukce byly přeskočeny.

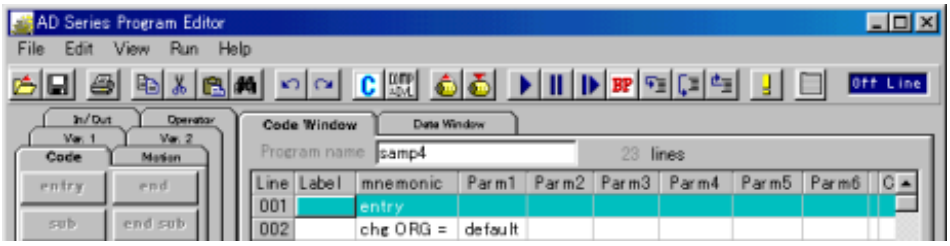
Pozn.2: Přerušíte-li v průběhu provádění programu (stav "On Line") napájení servopohonu nebo spojení s PC, objeví se hlášení komunikační chyby. V tomto případě ukončete program AHF, znovu spusťte AHF a vypněte a zapněte servopohon.

(3) Zastavení

Právě prováděny program může být zastaven v pozici, kdy byl obdržen povel k zastavení.

- V nabídce provádění [Execute] klikněte tlačítko zastavit [Halt] (stejnou akci je možné provést kliknutím na tlačítko  na liště, nebo stisknutím klávesy [F6]).

Prověřte, že byl zobrazen stav „zastaveno“ (Off Line) , viz níže:



(4) Pokračovat

Program zastavený příkazem zastavení nebo bodem přerušení lze spustit dále po takovémto zastavení.

- V nabídce provádění [Execute] klikněte tlačítko pokračovat [Continue] (stejnou akci je možné provést kliknutím na tlačítko  na liště, nebo stisknutím klávesy [F7]).

(5) Jednotlivý krok

Je možné spuštění zastaveného programu pouze na jeden krok.

- V nabídce provádění [Execute] klikněte tlačítko jeden krok [Single Step] (stejnou akci je možné provést kliknutím na tlačítko  na liště, nebo stisknutím klávesy [F8]).

Pozn.: Tak jako v případě bodu přerušení umožňuje provádění po jednom kroku kontrolu proměnných přiložením myši na pozici proměnné.

(6) Krok přes

Je možné spuštění zastaveného programu pouze na jeden krok. Je-li následujícím krokem volání podprogramu , pak je program zastavení až po provedení podprogramu.

- V nabídce provádění [Execute] klikněte tlačítko krok přes [Step Over] (stejnou akci je možné provést kliknutím na tlačítko  na liště, nebo současným stisknutím kláves [Shift] a [F8]).

Pozn.: Tak jako v případě bodu přerušení umožňuje provádění po jednom kroku kontrolu proměnných přiložením myši na pozici proměnné.

(7) Krok ven

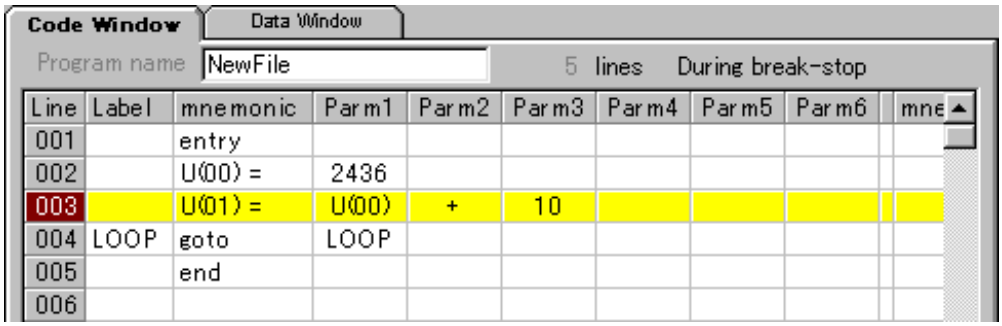
Je možné spuštění zastaveného programu pouze na jeden krok. Byl-li program zastaven při provádění podprogramu, pak program pokračuje dalším krokem po instrukci volání tohoto podprogramu

- V nabídce provádění [Execute] klikněte tlačítko krok ven [Step Out] (stejnou akci je možné provést kliknutím na tlačítko  na liště, nebo současným stisknutím kláves [Shift] [Ctrl] a [F8]).

Pozn.: Tak jako v případě bodu přerušení umožňuje provádění po jednom kroku kontrolu proměnných přiložením myši na pozici proměnné.

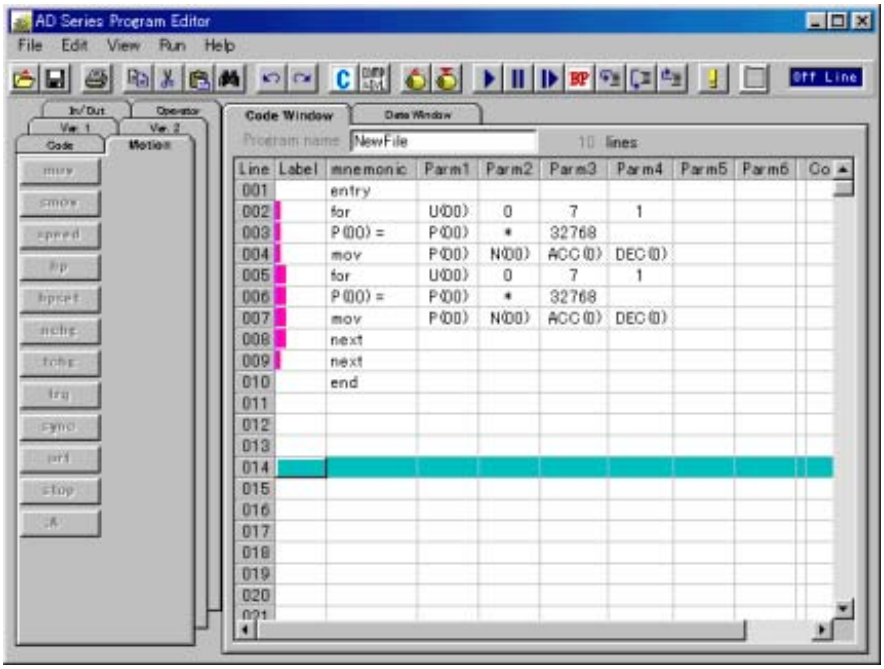
(8) Sledování proměnné

Tím, že umístíte kursor na určitou proměnnou (řádek je žlutě zbarvený) můžete sledovat její stav při přerušení, nebo v provádění po jednotlivých krocích.



(9) Zobrazení vnoření

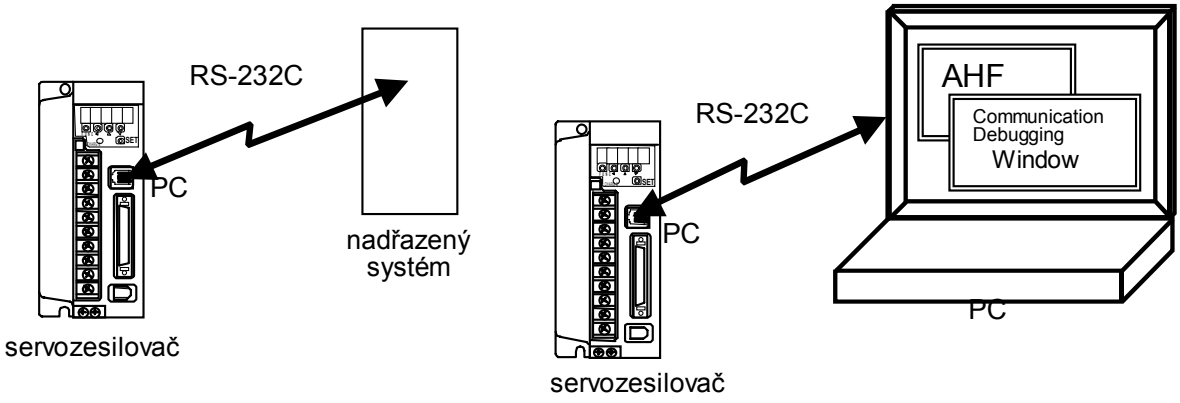
V průběhu sestavování programu zvolíme v nabídce zobrazení [View] povel zobraz strukturu podprogramů [Show Nesting Structure for Compiling] a ve sloupci značka (LABEL) se zobrazí úroveň vnoření.



(10) Okno ladění komunikace

Je-li nastaveno spojení mezi programem AHF a servopohonem, je možné zadat příkaz "open com" a sledovat instrukce print# a input# z programu AHF. Tato funkce je dostupná v nabídce [View] příkazem ladění komunikace [Debug Communications].

Používáte-li komunikační funkce, není možné použít program AHF, protože je konektorem CNPC připojen nadřazený systém (viz obrázek a níže). V důsledku toho se stává ladění daleko časově náročnější, protože nelze jednoduše určit zda je případná chyba v komunikaci způsobena nadřazeným systémem, přenosem RS232 nebo programem. Používáme-li okno ladění komunikace (Communication Debugging Window) (viz obrázek b níže) lze ladění komunikace a předávání dat spustit mezi programem AHF, PC a servopohonem a ladit pouze program. Po ukončení operace ladění lze připojit nadřazený systém a provést odladění v této konfiguraci. Časové nároky na odladění se tím sníží.

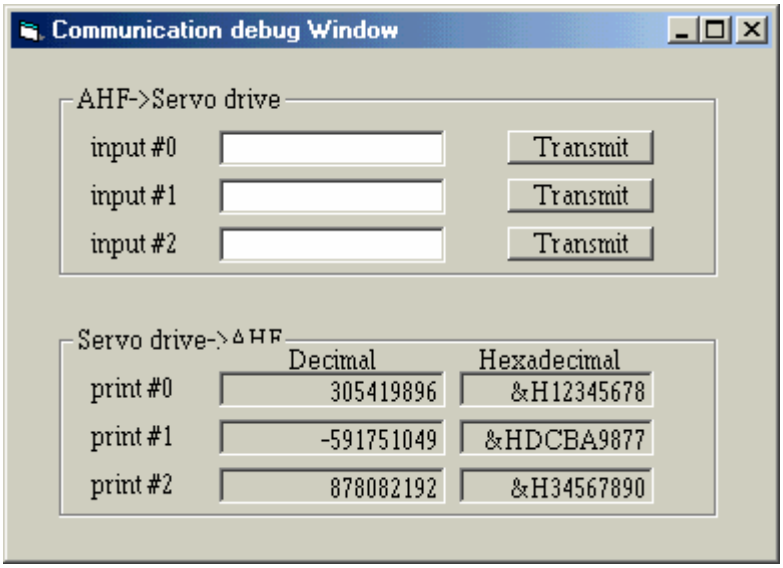


(a) standardní ladění (b) použití okna ladění komunikace

<Příklad použití 1>

Sledování přenosu komunikačních dat ze servopohonu

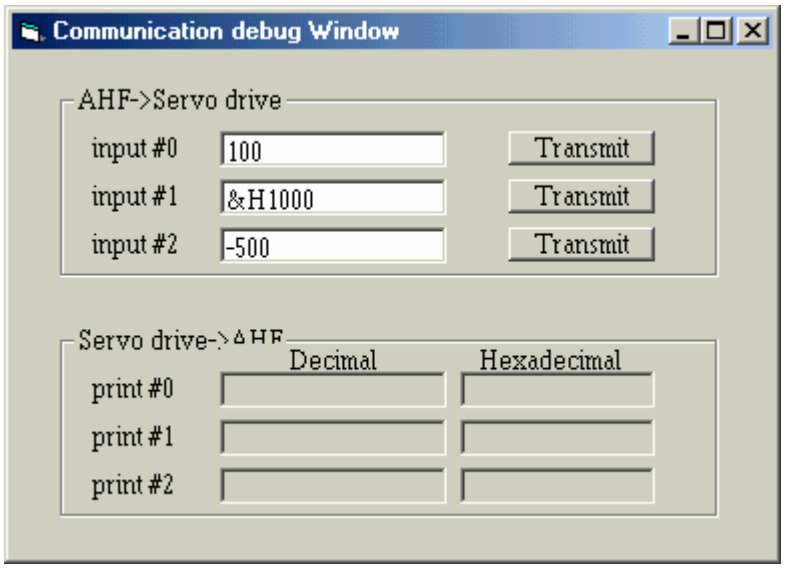
Zobrazení tiskové oblasti (print#) okna ladění komunikace (Communication Debugging Window) je ukázáno níže.



<Příklad použití 2>

Prostředky sledování přenosu dat do servopohonu

Zadáte-li v okně ladění komunikace do oblasti vstupů (input#) numerickou hodnotu a stisknete-li tlačítko přenos (Transmit), přenáší se zadaná data z PC do servozesilovače. okno ladění komunikace Vám umožní jednoduše sledovat tento přenos.



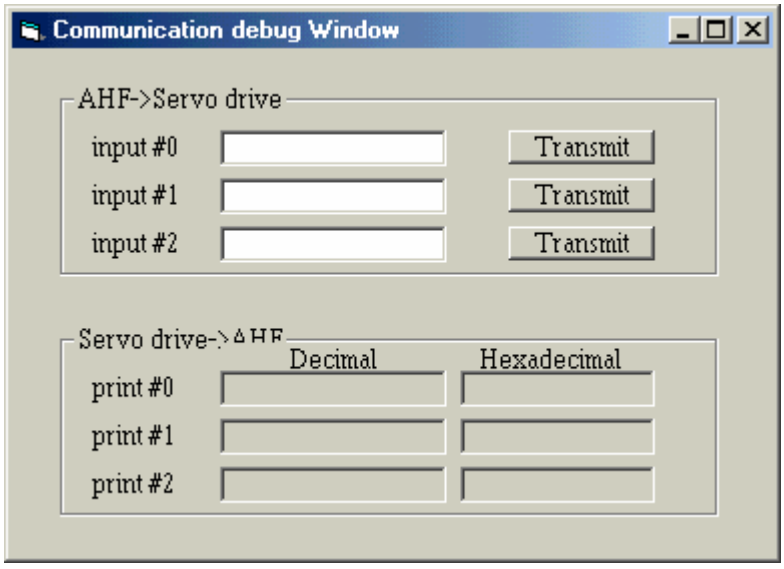
<Použití okna ladění komunikace>

(1) Vytvořte program s použitím komunikačních instrukcí com, print#, input#, a LOC. Vylaďte program tak aby při spuštění příkazu sestavení [Compile] v nabídce provádění [Execute] nedocházelo k chybám.

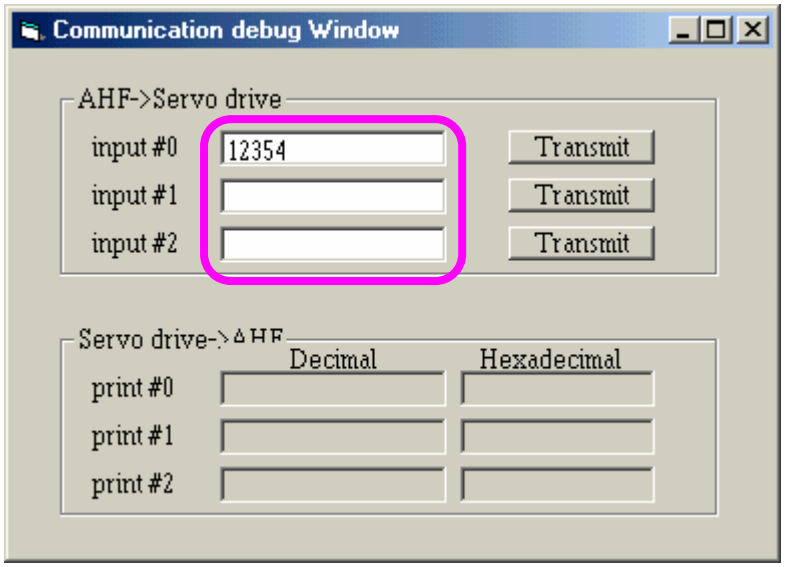
(2) Přeneste program do servopohonu použitím příkazu nahrát [Download] z nabídky provádění [Execute].

(3) Spusťte v nabídce zobrazení [View] příkaz ladění komunikace [Debug Communications].

(4) Spusťte v AHF provádění programu příkazem [Run] v nabídce provádění [Execute]. Otevře se Vám následující okno ladění komunikace (Communication Debugging Window)



(5) Do vstupní oblasti (input#) vložte numerickou hodnotu a stiskem tlačítka přenést [Transmit] spusťte operaci přenosu do servopohonu (viz níže):



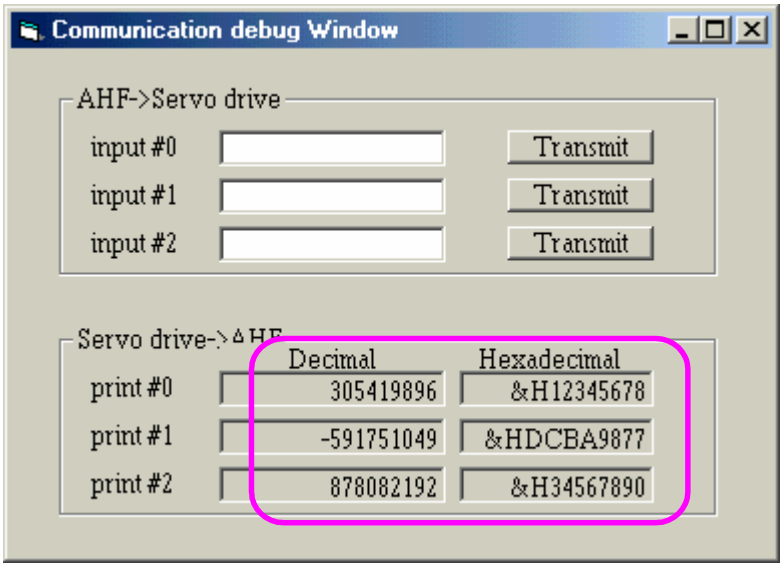
Pozn.1: Přenos se děje v okamžiku, kdy stisknete tlačítko přenést [Transmit]. Data ve vstupech jsou zpracovávána individuálně při stisknutí příslušného tlačítka „přenést“ (pro přenos dat ve vstupní oblasti 1 (input #1) slouží tlačítko „přenést“ situované vpravo od zadávacího pole oblasti

Pozn.2: V případě zadávání hexadecimálního čísla, vložte na začátek znaky "&H" (jedno-bytový řetězec znaků).

Příklad) hex 12345678, zadejte &H12345678.

(6) Když PC obdrží data ze servopohonu (instrukce print#), jsou tato zobrazena v příslušné oblasti okna ladění komunikace jako numerická hodnota (dekadicky a hexadecimálně).

Pozn.:Zobrazovací operace probíhá pomalu, proto se Vám může zdát, že když servopohon vysílá s vysokou přenosovou rychlostí, tak to vypadá, jakoby data nebyla přijata.



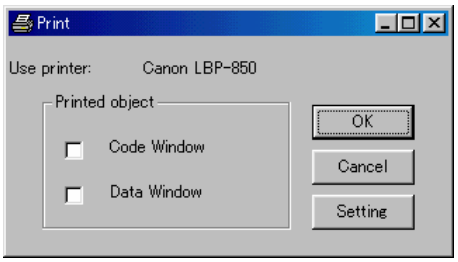
5.12 Vytištění programu

Lze vytisknout právě editovaný program.

(1) V nabídce soubor [File] klikněte tlačítko tisk [Print (P)] (stejnou akci je možné provést kliknutím na tlačítko  na liště, nebo současným stisknutím kláves [Ctrl] a [P]).

(2) Otevře se dialogové tiskové okno.

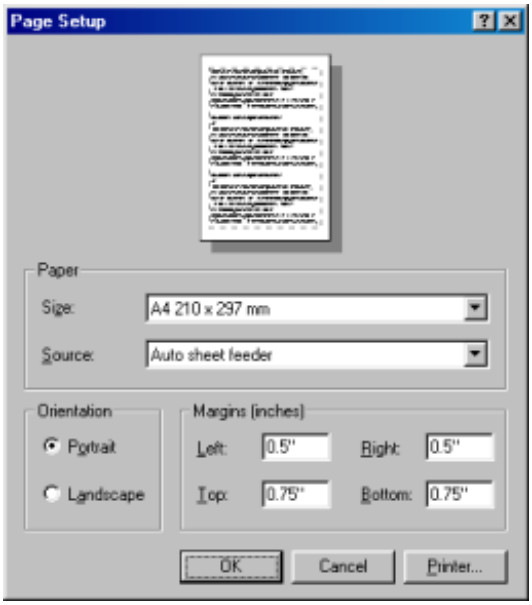
(4) Můžete tisknout zdrojový program i data zvolením příslušného pole.



(5) Spuštění tisku proveďte kliknutím na tlačítko [OK].

(6) Stisknete-li tlačítko přerušit [Cancel], tiskové okno se zavře.

(7) Kliknete-li na tlačítko nastavení [Setup], otevře se dialogové okno nastavení tisku.



5.13 Zobrazení pomoci

V nabídce pomoc [Help] klikněte na tlačítko pomoc [Help] (stejná akce se provede stisknutím tlačítka [F1] na klávesnici).

Na obrazovce se objeví okno pomoci editoru.

5.14 Informace o verzi programu

Můžete zobrazit informaci o verzi používaného editovacího programu.

(1) V nabídce pomoc [Help] klikněte tlačítko „o verzi“ [About].

(2) otevře se dialogové okno s informacemi o programu.

CHAPTER 6 INSTRUKČNÍ SLOVA – POPIS A POUŽITÍ

Tato kapitola popisuje funkci instrukčních slov, jejich programování a použití.

6.1 Vstupní a výstupní svorky 6-2
6.2 Rezervované proměnné 6-3
6.3 Seznam instrukčních slov 6-6
6.4 Řídící programové instrukce 6-10
6.5 Pohybové instrukce 6-25
6.6 Instrukce vstupů a výstupů 6-56

6.1 Vstupní a výstupní svorky

Významy přiřazené vstupním svorkám při použití programovatelných funkcí jsou odlišné od přiřazení při normálním provozu. Tabulka přiřazení významů vstupním a výstupním svorkám je uvedena níže:

	při normálním provozu	provoz s programovatelnými funkcemi	
		standardní instrukce	při nájezdu na VP
vstupní svorky	SON	SON	SON
	RS	RS	RS
	MOD	X(00)	X(00)
	TL	X(01)	X(01)
	FOT	X(02)	X(02)
	ROT	X(03)	X(03)
	SS1	X(04)	X(04)
	SS2	X(05)	X(05)
	PPI/GCH	X(06)	X(06)
	SRZ	X(07)	X(07)
	ORL	X(08)	ORL
	ORG	X(09)	ORG
	PEN/FWD	X(10)	X(10)
	CER/REV	X(11)	X(11)
výstupní svorky	SRD	Y(00)	Y(00)
	ALM	Y(01)	Y(01)
	INP	Y(02)	Y(02)
	SA	Y(03)	Y(03)
	SZD	Y(04)	Y(04)
	BRK	Y(06)	Y(06)
	TLM	Y(07)	Y(07)
	OL1	Y(08)	Y(08)
analogové vstupy	AI0	XA(0)	XA(0)
	AI1	XA(1)	XA(1)

Funkce přiřazené svorkám FOT, ROT, ORL, ORG, a všem ostatním výstupním svorkám lze změnit pomocí speciální funkce přiřazení výstupních svorek – instrukce změny (chg instruction) v programu. Tato změna přiřazení významů nastane po provedení instrukce změny.

6.2 Rezervované proměnné

Tabulka níže obsahuje rezervované proměnné, které je možné použít v programu:

kategorie	pojem - název	název proměnné	velikost	jednotka	R/W	přiřazený parametr	popis
nastavení k zapamatování	povel polohy	P(00) až P(99)	znaménko, 2 w	2^{15} pls/ot.	R/W	–	obsahuje povel polohy 32768dig=1 otáčka
	povel rychlosti	N(00) až N(15)	znaménko, 1 w	min^{-1}	R/W	–	obsahuje povel rychlosti 1dig=1min^{-1}
	povel momentu	T(00) až T(15)	znaménko, 1 w	%	R/W	–	obsahuje povel momentu 1dig=1%
	rozběhový čas	ACC(0) až ACC(1)	znaménko, 1 w	0.01s	R/W	–	obsahuje čas rozběhu 1dig=0.01s
	doběhový čas	DEC(0) to DEC(1)	znaménko, 1 w	0.01s	R/W	–	obsahuje čas doběhu 1dig=0.01s
uživatelské proměnné	uživatelská proměnná	U(00) až U(15)	znaménko, 2 w	–	R/W	–	uživatelem definovaná proměnná
zobrazení/ nastavení	nastavení času rozběhu	ACCEL	znaménko, 1 w	0.01s	R/W	Fb-04	indikuje nastavení doby rozběhu 1dig=0.01s
	LED znaky displeje	znaky 1 až 5	–	–	–	–	není dostupné
	LED znaménko displeje	DATR	–	–	–	–	není dostupné
	nastavení času doběhu	DECEL	znaménko, 1 w	0.01s	R/W	Fb-05	indikuje nastavení doby doběhu 1dig=0.01s
	LED data displeje	DISP	–	–	–	–	není dostupný
	výchozí poloha	HPOS	znaménko, 2 w	2^{15} rev/pls	R/W	–	indikuje druhou výchozí polohu, která je rozdílná od první VP (poloha 0 má při zapnutí sítě přednost)
	proudová zpětná vazba	IFB	bez znaménka, 1 w	%	R	d-02	indikuje proudovou zp. vazbu 50dig=1%
	ukončení polohování	INP	0.1	–	R	–	aktivní, když diference mezi hodnotou povelu polohy a skutečnou polohou klesne pod mez nastavenou v parametru Fb-23 (0 probíhá polohování, 1 polohování ukončeno)
	povel proudu	IRF	znaménko, 1 w	%	R	d-03	indikuje povel proudu 50dig=1%
	moment setrvačnosti	J	–	0.01×10^{-4} (0.01×10^{-5})	R/W	Fd-00	moment setrvačnosti zátěže 1dig=0.01^{-4} 1dig=0.01^{-5} (400 W max.)
	mezní frekvence rychlostní regulace	KFC	–	0.1Hz	R/W	Fd-01	frekvence odezvy ASR 1dig=0.1Hz
	mezní frekvence regulace polohy	KP	–	0.01Hz	R/W	Fd-09	frekvence odezvy ASR 1dig=0.1Hz
	zesílení polohové regulace FF	KPF	–	0.01	R/W	Fd-10	Zesílení kladné větve APR může být i 0. 1dig=0.01
	integrační zesílení ASR	KSI	–	0.01%	R/W	Fd-03	integrační zesílení regulace ASR PI 1dig=0.01%

kategorie	pojem - název	název proměnné	velikost	jednotka	R/W	přiřazený parametr	popis
zobrazení/ nastavení (pokračování)	proporcionální zesílení ASR	KSP	–	0.01%	R/W	Fd-02	proporcionální zesílení regulace ASR PI 1dig=0.01%
	způsob regulace	MODE	0 to 2	–	R	–	volba způsobu regulace (0: regulace momentu; 1: regulace rychlosti; 2: regulace polohy).
	okamžitá rychlost	NFB	znaménko, 1 w	min^{-1}	R	d-01	indikuje aktuální rychlost 1dig=1min^{-1}
	omezení rychlosti	NLM, NLM(0), NLM(1)	bez znaménka, 1 w	min^{-1}	R/W	– Fb-21 Fb-22	omezuje rychlost. Volba NLM přepisuje omezení v obou směrech NLM(0) a NLM(1). NLM(0): vpřed NLM(1): vzad 1dig=1min^{-1}
	přednastavený povel rychlosti	NRF	znaménko, 1 w	min^{-1}	R	d-00	indikuje přednastavený povel rychlosti 1dig=1min^{-1}
	povel posunu polohy	PBIAS	-128 to 128	1puls	R/W	–	je-li prováděna instrukce "sync" pak povel posunu polohy přidá v intervalu 140 μs nastavený počet pulsů.
	časová konstanta filtru povelu polohy	PFILT	–	0.1ms	R/W	Fd-36	
	přednastavená poloha	POS	znaménko, 1 w	2^{15} rev/pls	R/W	d-08	indikuje přednastavenou polohu. první výchozí poloha je považována za polohu 0 (má přednost při zapnutí sítě) 32768dig=1 otáčka
	míra křivky S	SCV	–	–	R/W	Fb-30	míra křivky S (0: žádná; 1: ostrá; 2: střední, 3: mírná).
	časová konstanta filtru povelu rychlosti	SFILT	–	1ms	R/W	Fd-20	povel rychlosti je neplatný, je-li časová konstanta 01dig=0.1ms
	detekce nulové rychlost	SZD	0.1	–	R	–	aktivní, je-li povel rychlosti nižší než hodnota nastavená v parametru Fb-22. (0: jiná rychlost; 1: nulová rychlost).
	výstup momentu	TFB	znaménko, 1 w	%	R	d-04	indikuje výstupní moment 50dig=1%
	časová konstanta filtru povelu momentu	TFILT	–	0.01ms	R/W	Fd-06	povel momentu je neplatný, je-li časová konstanta filtru 0. 1dig=0.01ms
	omezení momentu	TLM, TLM(0) až TLM(3)	znaménko, 1 w	%	R/W	– Fb-07 až Fb-10	omezuje moment. Volba TLM omezuje současně moment ve všech čtyřech kvadrantech TLM(0), TLM(1), TLM(2), a TLM(3). 50dig=1%
	Přednastavený povel momentu	TRF	zmanénko, 1 w	%	R	–	indikuje povel momentu (zadaný, před uplatněním omezení a filtru) 50dig=1%

kategorie	pojem - název	název proměnné	velikost	jednotka	R/W	přiřazený parametr	popis
vstup/výstup	bitový vstup	X(00) až X(11)	0,1	–	R	–	představuje bitový vstup pro zadání pomocí kontaktních vstupů.
	vstup slova	Xw	0 až 1024	–	R	–	představuje vstup slova pro zadání pomocí kontaktních vstupů X(00)=b0 ····X(11)=b11
	vstup čísla bitu	Xn	0 až 15	–	R	–	představuje vstup čísla bitu pro způsob zadávání pomocí kontaktních vstupů (k dispozici pouze pro instrukce pohybu – mov)
	bitový výstup	Y(00) až Y(07)	0,1	–	R/W	–	přestavuje bitový výstup pomocí kontaktních výstupů
	výstup slova	Yw	bez znaménka, 1 byte	–	R/W	–	představuje výstup slova pomocí kontaktních výstupů Y(00)=b0 ····Y(07)=b7
	analogový vstup	XA(0) až XA(1)	znaménko, 1 w	0.10%	R	–	představuje analogový vstup, plná výchylka je 100% 1dig=0.01%
komunikace	stav komunikačního vstupu dat	LOC(0) až LOC(2)	0,1	–	R	–	ověřuje, zda přišla po sběrnici RS 232 data z PC (0 nepřijato, 1 přijato)
informace o chybě	kód příčiny chyby	ERR(0) až ERR(3)	–	–	R	d-11 d-12	kód příčiny chyby

• Proměnné, které jsou k dispozici v programu se mění se zvoleným instrukčním slovem.
• V programovém editoru se akční klávesy automaticky mění se změnou instrukčního slova.

6.3 Seznam instrukčních slov

(1) Řízení programu

Název instrukce	Formát instrukce						Popis
	Mnemonické označení	prvý argument	druhý argument	třetí argument	čtvrtý argument	pátý argument	
základní stav programu	entry						značí začátek hlavního programu
	end						značí konec hlavního programu
stav podprogramu	sub	<název pod-programu>					značí začátek podprogramu
	end sub						značí konec podprogramu (návrat do místa volání)
volání podprogramů	call	<název pod-programu>					odskok z hlavního programu do podprogramu, jehož jméno je uvedeno v prvém argumentu; návrat následuje po příkazu "end sub".
řídící příkaz	cont	on					volba příkazu, který má být proveden při přerušení; on znamená provedení v bodě přerušení z PC.
		off					off = znamená provedení na začátku
inkrementální smyčka	for	<proměnná>	<počáteč-ní hodnota>	<konečná hodnota >	[krok <hodnota inkrementa-ce>]		inkrementální smyčka mezi počáteční hodnotou a konečnou hodnotou s určeným krokem. základní hodnota kroku je 1.
	<instrukce>						opakovací instrukce smyčky
	Next						ukončení inkrementální smyčky
nepodmíněný skok	goto	<název značky>					provede nepodmíněný skok na pozici zadanou názvem značky
podmíněný podprogram - ifs (struktura if)	Ifs	<podmínka>					počátek podmíněného skoku
	[then]						počátek instrukcí, které se mají provést, pokud je splněna podmínka
	<instrukce>						instrukce prováděné při splnění podmínky
	[else]						počátek instrukcí, které se provedou není-li splněna podmínka
	<instrukce>						instrukce prováděné při nesplnění podmínky
	end if						ukončení odstavce podmíněného skoku
podmíněný skok	if	<podmínka>			then	<název značky>	při splnění podmínky je proveden skok na pozici zadanou značkou
načti stav (volitelné načtení)	load	[P]	[N]	[T]	[ACC]	[DEC]	načte specifikované parametry; základní nastavení pro čtení je P(), N(), T(), ACC(), a DEC()
ulož stav (volitelné uložení)	save	[P]	[N]	[T]	[ACC]	[DEC]	uloží specifikované parametry; základní nastavení pro uložení je P(), N(), T(), ACC(), a DEC()

Název instrukce	Formát instrukce						Popis
	Mnemonické označení	prvý argument	druhý argument	třetí argument	čtvrtý argument	pátý argument	
vícenásobný podmíněný skok	select	<určující proměnná>					provedou se instrukce následující za příkazem „case“, pokud „určující proměnná“ dosáhne „hodnoty podmínky“.
	case	<hodnota podmínky>					hodnota podmínky a instrukce, které se mají provést, jestliže je podmínka splněna (odstavců „case“ může být několik s různými „hodnotami podmínky“ – viz vysvětlení dále)
	[case else]						počátek instrukcí, které se provádějí, když podmínka není splněna („určující proměnná“ má jinou hodnotu než „hodnotu podmínky“)
	……						instrukce podprogramu
	end select						ukončení podprogramu „vícenásobný skok a návrat do hlavního programu
Provádění podprogramu „až do“	Until	<podmínka>					provádí se následující instrukce až do doby, než je splněna podmínka
	<instrukce>						instrukce, které se mají provádět v případě nesplnění podmínky
	loop						ukončení odstavce instrukcí uzavřených ve smyčce
pozastavení provádění (řízení času)	wait	iii.ii					program čeká po dobu iii.ii (s).
		<podmínka>					program čeká do doby dokud není splněna podmínka
smyčka „dokud ..“	while	<podmínka>					provádějí se následující instrukce, dokud je splněna podmínka. Před každým prováděním testuje splnění podmínky
	<instructions>						instrukce, které mají být prováděny při splnění podmínky
	wend						konec smyčky „dokud“

(2) Pohybové instrukce

Název instrukce	Formát instrukce						Popis
	Mnemonické označení	prvý argument	druhý argument	třetí argument	čtvrtý argument	pátý argument	
výchozí poloha (návrat na VP)	hp	N(i)	[ACC(k)]	[DEC(m)]			instrukce vyvolá návrat na VP (druhá VP); parametry pohybu N(i), ACC(k(a DEC(m)
nastavení výchozí polohy (VP)	hpset						označí okamžitou polohu jako výchozí polohu (instrukce nevyvolá žádný pohyb)
instrukce pohybu na zadanou polohu	mov	P(i)	[N(j)]	[ACC(k)]	[DEC(m)]	[;&]	instrukce vykoná pohyb na polohu P(i) rychlostí N(j);. další parametry určují rozběhový a doběhový čas
		P(Xn)	[N(j)]	[ACC(k)]	[DEC(m)]	[;&]	instrukce vykoná pohyb na polohu P(Xn), kde číslo polohy je specifikováno vstupem X(j) a rychlost pohybu je N(j);.další parametry určují rozběhový a doběhový čas (když X(7)=1, pak X(n)=7)
		P(Xw)	[N(j)]	[ACC(k)]	[DEC(m)]	[;&]	instrukce vykoná pohyb na polohu P(Xw) , kde číslo polohy je specifikováno X(w) a rychlost pohybu je N(j).; další parametry určují rozběhový a doběhový čas (X(w) je číslo určené binární kombinací aktivních vstupů)
		Z	[N(j)]	[ACC(k)]		[;&]	vyvolá rotaci motoru rychlostí a N(j) až do příchodu prvního pulsu Z; další parametr určuje rozběhový čas
změna rychlosti	nchg	P(i)	[N(j)]	[ACC(k)]	[DEC(m)]		instrukce specifikuje změnu rychlosti v průběhu vykonávání instrukcí „mov“ nebo „speed“;další parametry určují rozběhový a doběhový čas při změně rychlosti (instrukce nchg musí stát vždy před instrukcí mov/speed)
nájezd na výchozí polohu	ort	0 1	[N(j)]		[ACC(m)]	[DEC(n)]	vyvolá nájezd na VP nízkou rychlostí
		2 to 5	[N(j)]	[N(k)]	[ACC(m)]	[DEC(n)]	vyvolá nájezd na VP vysokou rychlostí 1/2.
		6					vyvolá nájezd na VP způsobem dle potřeby
změna mezi regulací rychlosti a polohy	smov	P(i)					vyvolá otáčení motoru rychlostí N(i) dokud nedosáhne polohy P(i); (zařazením této instrukce za instrukce „speed“ přejde pohon do regulace polohy a najede na polohu P(i))
instrukce rychlostní regulace	speed	N(j)	[ACC(k)]	[DEC(m)]			pohon se pohybuje určenou konstantní rychlostí N(j) s rozběhovým a doběhovým časem ACC(k) a DEC(m)
zastavení pohybové operace	stop						instrukce vyvolá doběh a stop a ukončí pohybovou operaci; při provozu „sync“ vyvolá okamžité zastavení
instrukce synchronizace	sync	1					1: synchronizace bez posunu rychlosti; 2: synchronizace s využitím posunu rychlosti; s: rychlostní řízení s použitím analogového vstupu 1. ukončení provozu synchronizace se provede instrukcí stop
		2					
		s					
přechod na momentové řízení v zadané poloze	tchg	P(i)	T(k)				instrukce provede přechod na momentové řízení nebo změnu velikosti povelu momentu po dosažení zadané polohy P(i)
momentové řízení	trq	T(j)					instrukce zadání momentového řízení, nebo změny povelu momentu na hodnotu T(j)

(3) Řízení vstupů a výstupů

Název instrukce	Formát instrukce						Popis
	Mnemonické označení	prvý argument	druhý argument	třetí argument	čtvrtý argument	pátý argument	
X() (kontaktní vstup)	<proměnná>=	X(i)					sejme informaci z kontaktních vstupů a uloží ji to zvolené proměnné (0 = off, 1 = on).
	<proměnná>=	XW					sejme informaci z kontaktních vstupů a uloží ji jako datové slovo do zvolené proměnné
Y() (kontaktní výstupy	Y(i)=	<proměnná>					nastaví kontaktní výstupy Y(i) dle stavu zvolené proměnné (bit po bitu) (0 = off, 1 = on).
	YW=	<proměnná>					přenese stav zvolené proměnné na kontaktní výstupy jako datové slovo
změna přiřazení významu svorek	chg FOT= ROT=	X(i)					změna nastavení významu svorek FOT a ROT
		non					zrušení změny významu svorek FOT a ROT
	chg ORL= ORG=	X(i)					Změna významu svorek ORL a ORG
		základní					návrat významu svorek ORL a ORGdo základního přiřazení
	chg SRD= ALM= INP= SA= SZD= BRK= TLM= OL1=	Y(i)					změna významu výstupních svorek
		non					zrušení změny významu výstupních svorek

(4) Řízení komunikace

Název instrukce	Formát instrukce						popis
	Mnemonické označení	prvý argument	druhý argument	třetí argument	čtvrtý argument	pátý argument	
otevření komunikačního portu (počátek užití komunikace)	open com						deklaruje použití komunikačního portu RS-232C (PC).
instrukce tisku (výstup proměnných na komunikační rozhraní)	print #0 #1 #2	<proměnná>					určená proměnná je přivedena na výstup komunikace RS-232C (PC).
instrukce načtení (prove-de se načtení proměnné z komunikačního rozhraní)	input #0 #1 #2	<proměnná>					načte proměnnou z komunikačního rozhraní RS-232C (PC). K provedení dalšího kroku nedojde dokud není dokončeno načtení dat.

6.4 Řídící programové instrukce

Tato stať vysvětluje blíže řídící programové instrukce.

základní stav programu | **Začátek / konec hlavního programu**

- **Formát**

Format	popis operace
entry	označuje začátek hlavního programu (tato instrukce musí být umístěna vždy na začátek hlavního programu)
end	označuje konec hlavního programu. Při provedení instrukce „end“ se program zastaví, servopohon přeruší provádění pohybů a přejde do stavu servo uzamknuto.

- **Popis**

označuje začátek/konec hlavního programu.

- **Příklad použití**

```
entry                   začátek hlavního programu
    mov     P(00) N(00) ACC(0) DEC(0)
    ……
end                     konec hlavního programu
```

stav podprogramu | **začátek/konec podprogramu**

- **Format**

Format	popis operace
sub <název podprogramu>	označuje počátek podprogramu
end sub	označuje konec podprogramu, vyvolá návrat do hlavního programu na místo volání podprogramu

- **Popis**

označuje začátek/konec podprogramu

<název podprogramu>: označuje název podprogramu, který je pak prvním parametrem instrukce volání podprogramu (call).

Pozn.:: Dovolená hloubka vnoření (počet volání podprogramu v podprogramu) je 8. instrukce "sub", "for", "while", "until", "select", a "ifs" se počítají jako jedna úroveň. Proto použijeme-li v podprogramu instrukci "for" znamená to že máme úroveň vnoření 2.

- **Příklad použití**

```
sub  sub1               začátek podprogramu
    mov     P(00) N(00) ACC(0) DEC(0)
    ……
end sub                 konec podprogramu
```

volání podprogramů | **Nepodmíněné volání podprogramu**

- **Formát**

Formát	popis operace
call <název podprogramu>	Provede nepodmíněné volání podprogramu s uvedeným názvem

- **Popis**

Provede nepodmíněné volání podprogramu s uvedeným názvem. Další postup v provádění programu je povolen až po ukončení provádění podprogramu. Konec podprogramu je označen instrukcí "end sub", pak následuje návrat k instrukcí následující za instrukcí volání (call)..

<název podprogramu>: specifikace podprogramu, který má být proveden, musí být shodná s názvem uvedeným v instrukci "sub".

Pozn.: Dovolená hloubka vnoření (počet volání podprogramu v podprogramu) je 8 vč. strukturovaných instrukcí.

- **Příklad použití**

```
call  SUB1          provede volání podprogramu s názvem SUB1.
……
sub     SUB1
……
end sub
```

řídící příkaz | **volba režimu chování po přerušení programu**

- **Format**

Format	popis operace
cont on	znamená pokračování programu dalším řádkem následujícím po řádku kdy nastalo přerušení
cont off (základní)	znamená provádění programu od počátku

- **Popis**

Tyto instrukce definují chování servopohonu v případě, že signál SON přejde při provádění programu do stavu OFF a následně opět do stavu ON.. Základní nastavení je "cont off", to znamená, že po opětovném spuštěni servopohonu (SON = ON) se program provádí od začátku.

- **Příklad použití**

Příklad 1)

```
        cont on          volí pokračování programu po přerušení na dalším řádku
        ……
Servo   mov P(00)        při provádění této instrukce přejde signál SON do stavu OFF
OFF     Yw=1             po opětovné aktivaci signálu SON pokračuje provádění programu touto instrukcí
```

Start of execution

Příklad 2)

```
        entry            při opětovné aktivaci signálu SON bude program pokračovat touto instrukcí
        cont off         volí provádění programu po přerušení od začátku
        ……
Servo   mov P(00)        při opětovné aktivaci signálu SON bude program pokračovat touto instrukcí
OFF
```

Start of execution

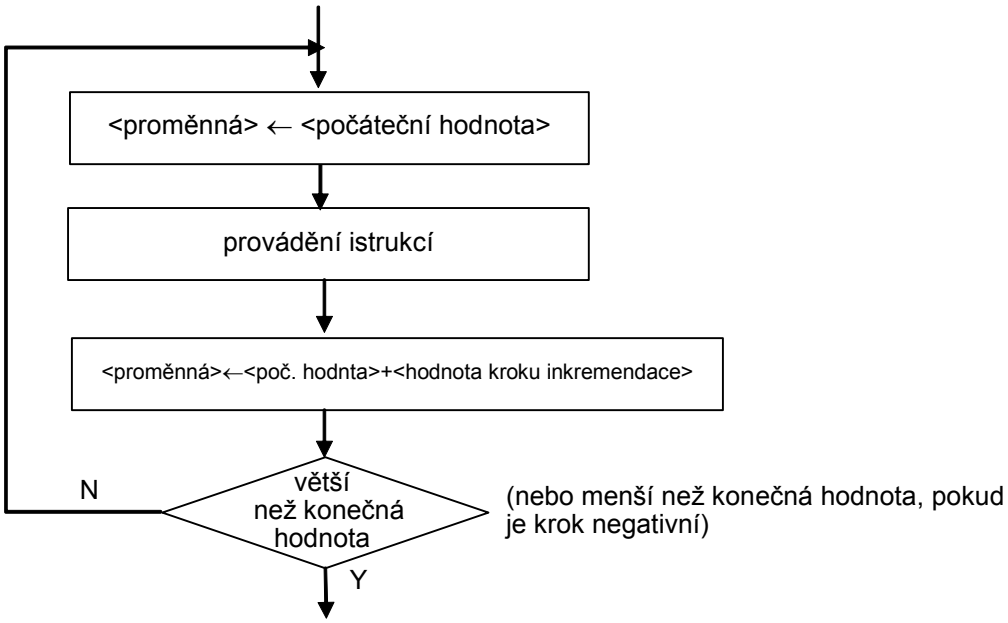
instrukce for — inkrementální smyčka

- **Format**

Format	popis operace
for <proměnná><počáteční hodnota> <konečná hodnota> [<velikost kroku>] <instrukce> next	provádění inkrementální smyčky probíhá mezi počáteční hodnotou a konečnou hodnotu. K ukončení provádění smyčky dojde v okamžiku, kdy proměnná dosáhne nebo překročí konečnou hodnotu.

- **Popis**

Je-li předem známo kolikrát se má určitý úsek programu provést, je možné s výhodou použít inkrementální smyčky „for“. Při každém provedení instrukcí smyčky je sledovaná proměnná zvýšena [snížena] o hodnotu (v jednotkách) zadanou krokem inkrementace oproti předchozí hodnotě (v prvém kroku oproti počáteční hodnotě; hranaté závorky jsou užity pro případ negativního kroku). Když výsledná hodnota proměnné bude vyšší [nižší] než zadaná konečná hodnota je provádění smyčky ukončeno. K provedení smyčky dojde minimálně jednou. Vývojový diagram je uveden níže:



<proměnná>: specifikace názvu proměnné použité ve smyčce. Je možné zvolit proměnné P(), N(), T(), ACC(), DEC(), a U().

<počáteční hodnota>: specifikace počáteční hodnoty proměnné při vstupu do smyčky. Tuto hodnotu lze specifikovat pomocí jiné proměnné nebo přímo zadané číselné hodnoty. Pevně zadaná číselná hodnota musí být v intervalu -128 až 127. Vyšší hodnotu lze zadat nepřímo předem pomocí proměnné.

<konečná hodnota>: specifikace hodnoty pro ukončení smyčky. Překročí-li (podkročí-li) sledovaná proměnná nastavenou hodnotu program ukončí provádění smyčky a pokračuje dále. Tuto hodnotu lze specifikovat pomocí jiné proměnné nebo přímo zadané číselné hodnoty. Pevně zadaná číselná hodnota musí být v intervalu -128 až 127. Vyšší hodnotu lze zadat nepřímo předem pomocí proměnné.

<krok inkrementace>: specifikace kladné nebo záporné hodnoty o kterou bude sledované proměnná zvyšována (snižována) při každém průchodu smyčky. Základní hodnota kroku je 1. Tuto hodnotu lze specifikovat pomocí jiné proměnné nebo přímo zadané číselné hodnoty. Pevně zadaná číselná hodnota musí být v intervalu -128 až 127. Vyšší hodnotu lze zadat nepřímo předem pomocí proměnné.

<instrukce>: instrukce, které mají být prováděny při průchodu smyčkou. lze vložit jednu nebo více instrukcí. Nezapomeňte, že každý řádek instrukce zabere jeden cyklus.

- **Prováděcí cyklus**

popis příkladu použití uvedeného níže

(1): řádek obsahující "for" se provádí pouze jednou. (v následujících průchodech se provádějí pouze <instrukce> a příkaz "next")

(2) až (4): provádění instrukcí

(5): v cyklu následujícím po poslední instrukci se inkrementuje proměnná a prověří se zda již nepřekročila koncovou hodnotu, Je-li proměnná v intervalu mezi počáteční a koncovou hodnotou, pokračuje se v provádění smyčky dalším průchodem instrukcí.

(6): další průchod instrukcí.

(7) to (13): následuje další opakování

(14): je proveden přechod na další instrukční řádek následující po instrukci "next".

- **Příklad použití**

Příklad) instrukce ve smyčce se provádějí třikrát

```
                                  pořadí provádění
for     U(00) 0 2                 '(1)    U(00)=0 až 2 s krokem 1
        mov  P(U(00))             '(2)    (6)    (10)
        Y(00)=1                   '(3)    (7)    (11)
        Y(00)=0                   '(4)    (8)    (12)
next                              '(5)    (9)    (13)
hpset                             '(14)
```

instrukce goto — nepodmíněný skok

- **Formát**

Formát	popis operace
goto <název značky>	provede se nepodmíněný skok na řádek označený názvem značky

- **Popis**

Instrukce „goto“ se používá když je potřeba v programu provést nepodmíněný skok na adresu instrukce označenou názvem značky.

<název značky>: vložte název řádku definovaný ve sloupci značka..

- **Příklad použití**

```
    goto  LBL1          provede se skok na řádek s názvem značky LBL1.
    ……
LBL1:hpset
```

instrukce if — podmíněný skok

- **Formát**

Formát	popis operace
if <podmínka> then <název značky>	Je-li splněna podmínka , provede se skok na řádek označený názvem značky. Není-li podmínka splněna, pokračuje program další instrukcí

- **Popis**

Instrukce "if" se používá je-li nutné provést větvení programu v závislosti na nějaké podmínce. Instrukce "if" provede při splnění zadané podmínky skok v programu na řádek označený uvedenou značkou.

<podmínka>:zapište podmínku skoku ve tvaru uvedeném v oddíle 3.5 Podmínky kapitoly 3 Syntaxe.

<název značky>: vložte název řádku definovaný ve sloupci značka.

- **Prováděcí cyklus**

prověření splnění podmínky a odskok se děje v jednom cyklu. V následujícím cyklu se provádí instrukce uvedená na adrese skoku.

Pro ilustraci jsou v následujícím příkladu použití uvedena čísla, která představují pořadí provádění jednotlivých instrukcí v různých situacích.

- **Příklad použití**

```
      Yw=0                          všechny svorky Y(00) až Y(07) přejdou do stavu OFF
      if  POS=0       then  LBL1    (1) Při splnění podmínky se provede skok na značku LBL1.
      if  POS=P(00)   then  LBL2    (2) Při splnění podmínky se provede skok na značku LBL2.
      Y(00)=1                       (3)
      goto LBL3                     (4)
LBL1: Y(01)=1                                 (2)
      goto LBL3                               (3)
LBL2: Y(02)=1                         (3)
LBL3: Y(03)=1                         (5)     (4)     (4)
```

instrukce ifs-then-else-end if — strukturovaná instrukce if

- **Formát**

Formát	popis operace
Ifs <podmínka> [then] <instrukce 1> [else] <instrukce 2> end if	Je-li splněna podmínka provedou se instrukce vřazené mezi "then" a "else". Není-li splněna podmínka provedou se instrukce vřazené mezí "else" a "end if".

- **Popis**

Tato instrukce se použije pro podmíněné provedení jedné nebo druhé části programu. Je-li určená podmínka splněna provádějí se instrukce zapsané v části 1, není-li podmínka splněna provádějí se instrukce zapsané v části 2.

<podmínka>: zapište podmínku skoku ve tvaru uvedeném v oddíle 3.5 Podmínky kapitoly 3 Syntaxe.
<instrukce 1>: zapište instrukce, které se mají provést pokud je podmínka splněna. Nezapomeňte, že každá instrukce zabere jeden cyklus.
<instrukce 2>: zapište instrukce, které se mají provést pokud není podmínka splněna. Nezapomeňte, že každá instrukce zabere jeden cyklus.

POZN.: Dovolená hloubka vnoření (počet volání podprogramu v podprogramu) je 8 vč. strukturovaných instrukcí.

- **Prováděcí cyklus**

V prvém cyklu se provádí ověření podmínky, v následném cyklu se provádí prvá instrukce z oddílu 1 nebo 2. V třetím cyklu následuje další instrukce z vybraného oddílu, pokud tato ještě existuje, pokud ne, tak se provede odskok na „end if“. Je-li tedy v oddíle 1 za podmínkou pouze jedna instrukce, zabere provedení instrukce „ifs“ 3 cykly.
Pro ilustraci jsou v následujícím příkladu použití uvedena čísla, která představují pořadí provádění jednotlivých instrukcí v různých situacích.

- **Příklad použití**

Příklad 1) Je využita celá šíře instrukce

```
                                    je-li splněna podmínka      není-li splněna podmínka
Ifs     POS=0                           ‘(1)                        (1)
then                                     ‘
        hpset                           ‘(2)
else                                     ‘
        ort  0 N(00) DEC(0)              ‘                          (2)
end if                                  ‘(3)                        (3)
```

Příklad 2) Je vynecháno "then" (při splnění podmínky není potřeba provádět žádné instrukce)

```
                                    není-li splněna podmínka      je-li splněna podmínka
Ifs     POS=0                       ‘(1)                          (1)
else                                ‘
        ort  0 N(00) DEC(0)         ‘(2)
end if                              ‘(3)                          (2)
```

Příklad 3) Je vynecháno "else" (při nesplnění podmínky není potřeba provádět žádné instrukce)

```
                                    je-li splněna podmínka        není-li splněna podmínka
Ifs     POS<>0                      ‘(1)                          (1)
then                                ‘
        ort  0 N(00) DEC(0)         ‘(2)
end if                              ‘(3)                          (2)
```

instrukce natažení | **Natažení proměnných**

- **Formát**

Formát	popis operace
load [P] [N] [T] [ACC] [DEC]	natáhne specifikované proměnné z EEPROM.
load [P(i)] [N(j)] [T(k)] [ACC(l)] [DEC(m)]	

- **Popis**

Tato instrukce provede natažení proměnných P(), N(), T(), ACC(), a DEC() z EEPROM servozesilovače. v základním nastavení instrukce natáhne všechny uvedené proměnné. Tuto instrukci použijte, pokud je potřeba přepsat stav proměnných základními hodnotami určenými pro zapnutí sítě (power ON).

P: Natáhne všechny P() proměnné (P(00) až P(99)).
P(i): Natáhne pouze zvolenou proměnnou P(). Např. je-li specifikováno P(01), pak je natažena pouze proměnná P(01).
N: Natáhne všechny N() proměnné (N(00) až N(15)).
N(j): Natáhne pouze zvolenou proměnnou N(). Např. je-li specifikováno N(01), pak je natažena pouze proměnná N(01).
T: Natáhne všechny T() proměnné (T(00) až T(15)).
T(k): Natáhne pouze zvolenou proměnnou T(). Např. je-li specifikováno T(01), pak je natažena pouze proměnná T(01).
ACC: Natáhne obě proměnné ACC(00) a ACC(01).
ACC(l): Natáhne buď proměnnou ACC(00) nebo ACC(01). Např. je-li specifikováno ACC(01), pak je natažena pouze proměnná ACC(01).
DEC: Natáhne obě proměnné DEC(00) a DEC(01).
DEC(m): Natáhne buď proměnnou DEC(00) nebo DEC(01). Např. je-li specifikováno DEC(01), pak je natažena pouze proměnná DEC(01).

- **Příklad použití**

Příklad 1)

```
load                    jsou nataženy všechny proměnné
```

Příklad 2)

```
load    P N ACC DEC     jsou nataženy proměnné P, N, ACC, a DEC
```

Příklad 3)

```
load    P(00)           je natažena pouze proměnná P(00)
```

instrukce ulož	Uložení proměnné

- **Formát**

Formát	popis operace
save [P] [N] [T] [ACC] [DEC]	uloží proměnné do EEPROM.
save [P(i)] [N(j)] [T(k)] [ACC(l)] [DEC(m)]	

- **Popis**

Tato instrukce uloží proměnné P(), N(), T(), ACC(), a DEC() do EEPROM. V základním tvaru jsou uloženy všechny proměnné. Při příštím zapnutí sítě můžete předefinovat proměnné, které jsou přednastaveny pro první zapnutí sítě.

P: Uloží všechny P() proměnné (P(00) až P(99)).
P(i): Uloží pouze specifikovanou proměnnou P(). Např. je-li specifikováno P(01), pak je uložena pouze proměnná P(01).
N: Uloží všechny N() proměnné (N(00) až N(15)).
N(j): Uloží pouze specifikovanou proměnnou N(). Např. je-li specifikováno N(01), pak je uložena pouze proměnná N(01).
T: Uloží všechny T() proměnné (T(00) až T(15)).
T(k): Uloží pouze specifikovanou proměnnou T(). Např. je-li specifikováno T(01), pak je uložena pouze proměnná T(01).
ACC: Uloží obě proměnné ACC(00) a ACC(01).
ACC(l): Uloží buď proměnnou ACC(00) nebo ACC(01). Např. je-li specifikováno ACC(01), pak je uložena pouze proměnná ACC(01).
DEC: Uloží obě proměnné DEC(00) a DEC(01).
DEC(m): Uloží buď proměnnou DEC(00) nebo DEC(01). Např. je-li specifikováno DEC(01), pak je uložena pouze proměnná DEC(01).

POZN.1: Paměť EEPROM má omezenou životnost na zhrub 100,000 zápisů. Pokud používáte zápis do EEPROM velmi často, může se stát, že po překročení životnosti paměť vyhlásí chybu a nebude již více možné do ní zapisovat. Měli by jste udělat taková opatření, aby byl počet zápisů do EEPROM minimální.
POZN.2: Nevypínejte napájení řídící části, pokud probíhá instrukce zápisu. Dojde-li při zápisu k vypnutí napájení řízení, nemusí být zápis proveden správně.

- **Příklad použití**

Příklad 1)
```
save                    uloží se všechny proměnné
```

Příklad 2)
```
save    P N ACC DEC     uloží se proměnné P, N, ACC, a DEC
```

Example 3)
```
save    P(00)           uloží se pouze proměnnou P(00)
```

instrukce select case — Vícenásobný promíněný skok

- **Formát**

Formát	popis operace
select <podmínková proměnná> case <hodnota podmínky 1> <instrukce 1> [case <hodnota podmínky 2> <instrukce 2> … [case else] <instrukce n> end select	Provádí se instrukce v odstavcích <instrukce 1 až n-1> podle toho jak proměnná uvedená jako <podmínková proměnná> nabývá <hodnot podmínky 1 až n-1> v jednotlivých případech (case) <hodnota podmínky> musí ležet v intervalu mezi -128 až 127. Je-li nutné použít vyšší hodnotu, je potřeba tuto specifikovat jako uživatelskou proměnnou U(). Pokud podmínková proměnná je rozdílná od všech „hodnot podmínky (nesplní žádný z „případů“) provádí se <instrukce n> uvedené za značkou "case else" („jinak proveď“).

- **Popis**

Nabude-li hodnota podmínkové proměnné některé z hodnot uvedených jako hodnota podmínky 1 až n-1, pak se provádí instrukce zapsané za příslušnou hodnotou podmínky (case) až po následující hodnotu podmínky. Následně se provede instrukce „end select“ a program pokračuje další instrukcí. Nesplňuje-li „podmínková proměnná“ žádnou z „hodnot podmínky“, provedou se instrukce uvedené v případě „jinak proveď“ (case else) až po instrukci „end select“ a pokračuje se další instrukcí.

<podmínková proměnná>: může být použita některá z proměnných P(), N(), nebo T().
<hodnota podmínky 1 až n-1>: <hodnota podmínky> musí ležet v intervalu mezi -128 až 127. Je-li nutné použít vyšší hodnotu, je potřeba tuto specifikovat jako uživatelskou proměnnou U().
<instrukce 1 až n>: Napište pro každý případ instrukci (nebo více instrukcí), která se má provést, když je splněna příslušná podmínka. Pro každý případ lze napsat až 30 instrukcí v závislosti na jejich typu.

POZN.1: Není-li mezi uvedenými „případy“(1 ÷ n-1) žádný, kterému by odpovídala podmínková proměnná, provádí se instrukce uvedené za příznakem „case else“ (jinak). Pokud v programu chybí případ „case else“ dojde k chybě provádění (E45). V tomto případě doplňte příznak „case else“ a proveďte předběžný test.
POZN.2: Zvyšování uvedených případů volby (možností skoků) zvyšuje i čas potřebný k provádění této instrukce a snižuje celkovou rychlost provádění programu. Pokud má některý z případů zvýšenou četnost výskytu, pak jej uveďte bezprostředně za příznak „select“ jako případ 1 (case),tímto zkrátíte čas potřebný k provedení instrukce „select“ na minimum.

- **Prováděcí cyklus**

V prvním cyklu je provedeno načtení hodnoty podmíněné proměnné. V dalším cyklu je provedeno porovnání s hodnotou v prvním „případě“ (case). Je-li podmínka splněna, pak je první instrukce „případu“ provedena ještě v tomtéž cyklu. V následujících cyklech jsou prováděny další instrukce uvedené v „případu“ řádek po řádku. V cyklu následujícím po zpracování poslední instrukce je proveden skok na „end select“. Pokud podmínka není splněna, pak se provede v dalším cyklu skok na následující „případ“ a test podmínky.
V následující příkladu použití jsou uvedena čísla vyjadřující pořadí provádění jednotlivých instrukcí v různých situacích.

- **Příklad použití**

Když Xw = 1

```
                                    pořadí provádění
                                    provedení "case 1"      provedení "case else"
select Xw                            '     (1)                     (1)
    case 0                           '     (2)                     (2)
        Yw=0                         '
        speed N(00)                  '
    case 1                           '     (3)                     (3)
        Yw=1                         '     (3)
        speed N(00)                  '     (4)
    case 4                           '                 (4)
        Yw=2                         '
        speed N(00)                  '
    case else                        '                 (5)
        Yw=3                         '                 (6)
end select                           '     (5)                     (6)
```

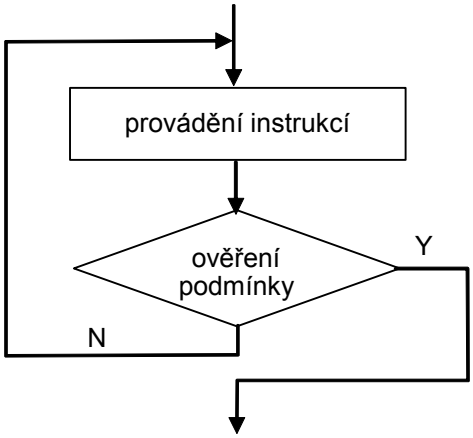
Smyčka „until“ — Provádění podprogramu „až dokud není splněna podmínka“

- **Formát**

Formát	popis operace
until <podmínka> <instrukce> loop	Provádí se <instrukce> dokud není splněna <podmínka>. <Splnění podmínky> se prověřuje až po provedení <instrukcí>.

- **Popis**

Příkaz „until“ představuje smyčku, ve které se provádí zapsané instrukce, do té doby, dokud není splněna předepsaná podmínka, přičemž test splnění podmínky se provádí až po provedení instrukcí (viz vývojový diagram níže)



<podmínka>: napište podmínku ve formátu uvedeném v odstavci 3.5, Podmínky, v kapitole 3, Syntaxe. Můžete napsat pevnou hodnotu přímým numerickým zadáním nebo název proměnné.

<instrukce>: Napište instrukce, které se mají provádět, dokud není splněna uvedená podmínka. Lze zadat jednu nebo více instrukcí viz příklad níže. Každý instrukční řádek představuje časově jeden cyklus.

- **Prováděcí cyklus**

První instrukce je provedena v cyklu bezprostředně následujícím po příkazu „until“. Druhá a další instrukce se provádějí řádek po řádku každá v jednom cyklu. Prověřování splnění podmínky se provede v cyklu následujícím za poslední instrukcí (společně s příkazem „loop“).V následujícím cyklu se program vrátí na příkaz „until“. Další cyklus představuje opět provádění první instrukce.

V následující příkladu použití jsou uvedena čísla vyjadřující pořadí provádění jednotlivých instrukcí v různých situacích.

- **Příklad použití**

```
                                    pořadí provádění
until       X(00)=1                 '(1)  (5)   (9)
hpset                               '(2)  (6)   (10)
Y(00)=1                             '(3)  (7)   (11)
loop                                '(4)  (8)   (12)
Y(01)=1                             '           (13)
```

příkaz čekej	pozastavení provádění / řízení času

- **Formát**

	Formát	popis operace
(1)	wait iii.ii	pozastaví provádění na dobu iii.ii [s] (i: integer).
(2)	wait <condition>	pozastaví provádění , dokud není splněna <podmínka>

- **Popis**

Formát (1): Zastaví provádění programu na nastavený čas. Po uplynutí nastaveného času pokračuje provádění programu dalším řádkem. Lze nastavit časové rozmezí od 000.00 [s] do 999.99 [s].

Formát (2): Zastaví provádění programu na dobu, dokud není splněna <podmínka>. Po splnění <podmínky> pokračuje program dalším řádkem.

iii.ii: Čas lze zadat přímo číselnou hodnotou v intervalu 000.00 [s] and 999.99 [s].

<podmínka>: napište podmínku ve formátu uvedeném v odstavci 3.5, Podmínky, v kapitole 3, Syntaxe.

- **Příklad použití**

Příklad 1)

```
wait   1.0          zastaví provádění na 1.0 [s].
```

Příklad 2)

```
wait   X(00)=1      zastaví provádění dokud X(00) = 1.
```

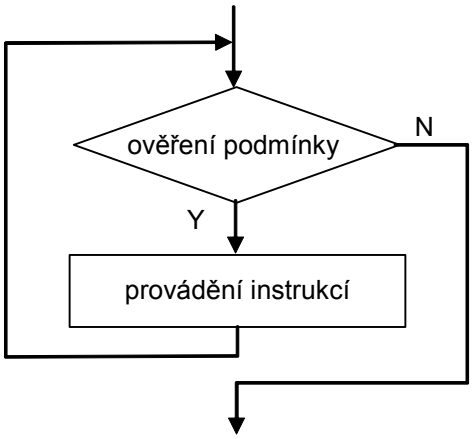
smyčka dokud — provádění smyčky při splnění podmínky

- **Formát**

Formát	popis operace
while <podmínka> <instrukce> wend	Když je splněna <podmínka> provádí se uvedené <instrukce>. Splnění podmínky se prověřuje před prováděním instrukcí.

- **Popis**

Instrukce „while“ určuje, že se budou opakovaně provádět dále zapsané <instrukce> až po příznak „wend“, pokud bude splněna <podmínka>. Splnění podmínky se prověřuje před prováděním instrukcí. Pokud není podmínka splněna provede se skok na příznak „wend“. (níže je uveden vývojový diagram)



<podmínka>: Napište podmínku ve formátu uvedeném v odstavci 3.5, Podmínky, v kapitole 3, Syntaxe. Můžete napsat pevnou hodnotu přímým numerickým zadáním nebo název proměnné.

<instrukce>: Napište instrukce, které se mají provádět, dokud je splněna uvedená podmínka. Lze zadat jedu nebo více instrukcí viz příklad níže. Každý instrukční řádek představuje časově jeden cyklus.

- **Prováděcí cyklus**

V prvním cyklu se provádí prověřování splnění podmínky. v dalším cyklu (je-li podmínka splněna) se provádí první z uvedených instrukcí. Další instrukce následují řádek po řádku v dalších cyklech. Není-li splněna podmínka, program pokračuje instrukcí následující po příznaku „wend“ (v cyklu následujícím po prověření podmínky).

V následující příkladu použití jsou uvedena čísla vyjadřující pořadí provádění jednotlivých instrukcí v různých situacích.

- **Příklad použití**

```
                                  pořadí provádění
while       X(00)=1               '(1)  ┌► (5)  ┌►  (9)
            hpset                 '(2)  │  (6)  │
            Y(00)=1               '(3)  │  (7)  │
wend                              '(4) ─┘  (8) ─┘
Y(01)=1                           '                (10)
```

6.5 Pohybové instrukce

výchozí poloha „hp“	**nájezd na výchozí polohu**

- **Formát**

Formát	popis operace
hp N(i) [ACC(k)] [DEC(m)]	vyvolá pohyb na výchozí polohu

- **Popis**

příkaz „hp“ vyvolá pohyb servopohonu na výchozí pozici (VP) (druhou VP) rychlostí specifikovanou jako N(i) a rozběhovým a doběhovým časem ACC(k) a DEC(m). Nejsou-li parametry ACC(k) a DEC(m) specifikovány, je platné stávající nastavení. Tak jako v případech standardních pohybových operací lze nastavit S-křivku pro rozběh a doběh.
Program nebude provádět další instrukcí dokud není dosaženo VP (resp. dokud se chyba polohy nesníží pod dovolený rozsah (Fb-23)). pokud je nutné. aby provádění programu ihned pokračovalo je nutné použít k návratu na VP funkci „mov“.

N(j): Specifikace rychlosti, jednotka 1 dig = 1 min^{-1}. Směr otáčení není ovlivněn znaménkem rychlosti

ACC(k): Specifikuje čas rozběhu z rychlosti 0 na rychlost N(j) (jednotka 1 dig = 0.01 s). Hodnota ACC(k) je hodnota času rozběhu z rychlosti 0 na maximální rychlost N_{max} .

DEC(m): Specifikuje čas doběhu z rychlosti N(j) na rychlost 0 (jednotka 1 dig = 0.01 s). Hodnota DEC(m) je hodnota času doběhu z maximální rychlosti N_{max} na rychlost 0 .

POZN.1: Nejsou-li parametry [N(j)], [ACC(k)], nebo [DEC(m)] specifikovány před první instrukcí pohybu , pak je indikována chyba E45. Proto je nutné specifikova uvedené parametry alespoň jednou na začátku.

POZN.2: Je-li N(j) = 0, následuje hlášení chyby E45.

POZN.3: Není-li vzdálenost mezi aktuální polohou a VP dostatečně veliká, pak nemusí dojít vůbec k dosažení zadané rychlosti a rozběh pohonu přejde plynule v doběh.

POZN.4: Jak je patrné z uvedeného diagramu, měl by mít průběh rychlosti při provádění instrukce "hp" lichoběžníkový tvar, plynoucí z procesu polohování na VP.Je-li nastaveno nízké zesílení rychlostní nebo polohové regulační smyčky může docházet v průběhu provádění k deviaci (nedostatečná dynamika pohonu), takže průběh rychlosti nebude lichoběžníkový.

- **Příklad použití**

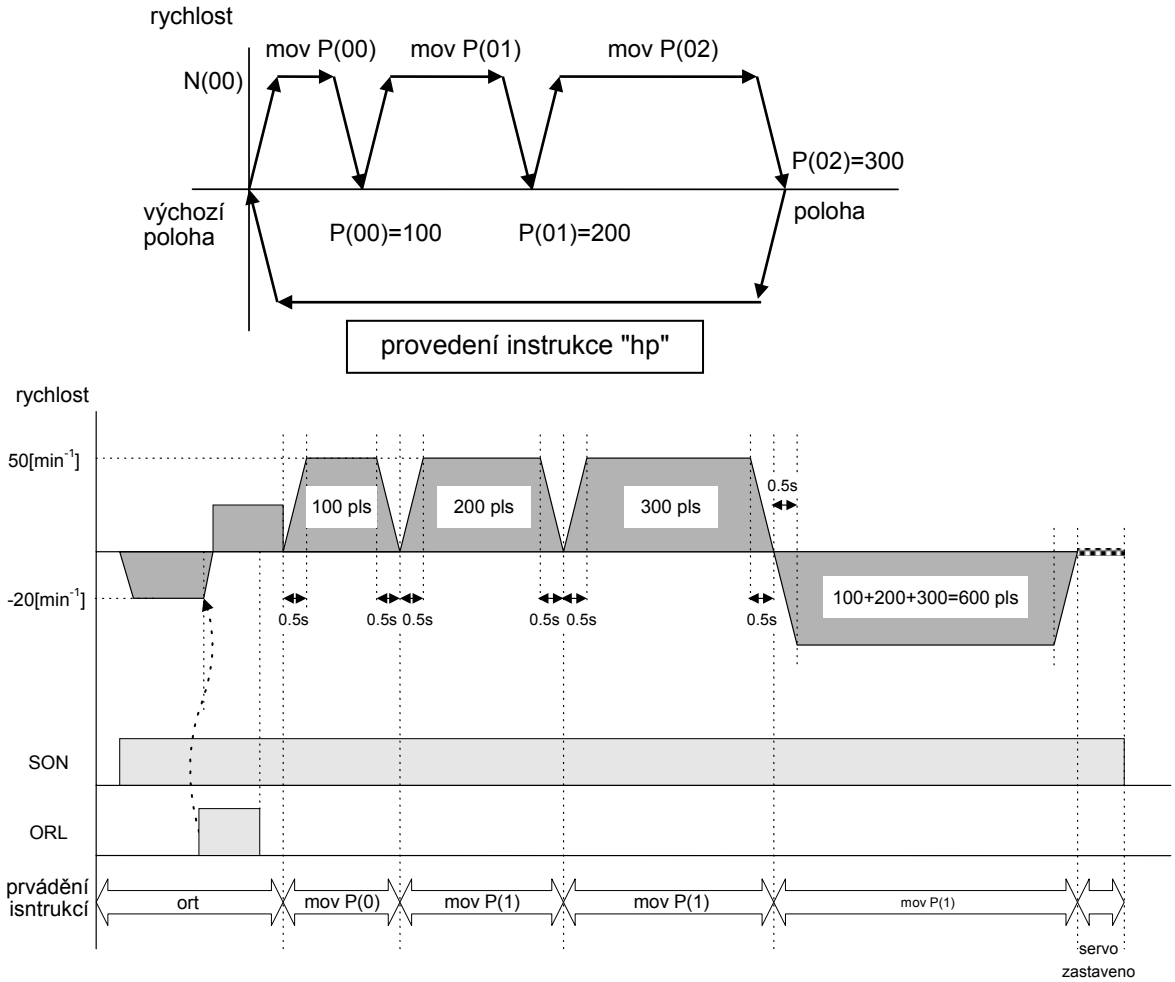
<datové okno>

ACC(0)=50[s] , DEC(0)=50[s] ,
N(00)=50 [min-^{1}] , N(01)=5[min^{-1}] , N(02)=20[min^{-1}] ,
P(00)=100[pls] , P(01)=200[pls] , P(02)=600 [pls] … (1)

<kódové okno>

```
entry                                   '
    ort 3  N(02) N(01) ACC(0) DEC(0)    'nájezd na VP                … (2)
    mov    P(00) N(00) ACC(0) DEC(0)    'nájezd na polohu P(00)      … (3)
    mov    P(01)                        'nájezd na polohu P(01)      … (4)
    mov    P(02)                        'nájezd na polohu P(00)      … (5)
    hp     N(00) ACC(0) DEC(0)          'nájezd na VP                … (6)
end                                     '
```

Provádění výše uvedené posloupnosti instrukcí vyvolá postupně následující změny polohy a rychlosti:



nastavení výchozí polohy	Přiřazení výchozí polohy (druhé VP)

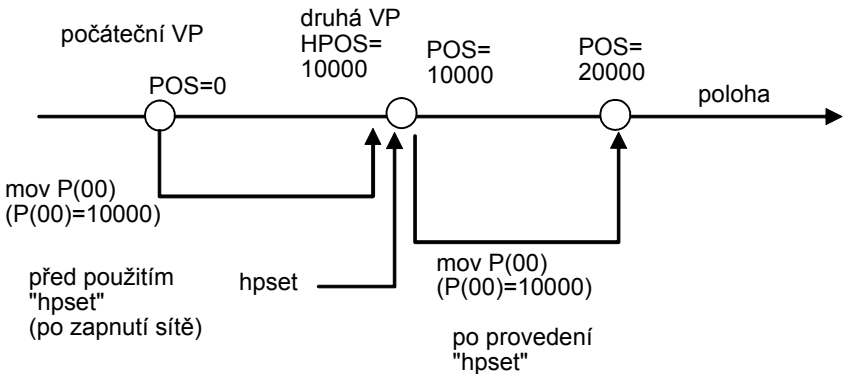
- **Formát**

Formát	popis operace
hpset	HPOS ← POS

- **Popis**

Instrukce "hpset" přiřadí aktuální polohu servopohonu jako výchozí polohu (druhá VP). Přiřazení druhé VP tímto způsobem nám umožní provádět identickou pohybovou operaci (sekvenci pohybových instrukcí) v různém místě pracovního rozsahu (absolutní hodnota VP se mění, ale relativní pohyby vůči zvolené VP zůstávají identické).
Druhá VP se liší od počáteční VP (pouze v počátečním stavu - najetí na VP po zapnutí sítě - jsou obě polohy identické. Příkaz „hpset“ umožní definovat aktuální polohu jako druhou VP. Proto stejná instrukce pohybu „mov“ bude mít před a po provedení příkazu „hpset“ rozdílnou výslednou polohu (viz obrázek níže). Druhou VP lze také přímo zadat jako HPOS.

<Výchozí poloha (druhá VP)>
Druhá VP se liší od počáteční VP (pouze v počátečním stavu - najetí na VP bezprostředně po zapnutí sítě - jsou obě polohy identické). Příkaz „hpset“ umožní definovat aktuální polohu jako druhou VP. Výsledkem je rozdíl mezi počáteční VP a aktuální VP po jeho provedení. Důsledky tohoto chování jsou patrné z obrázku uvedeného níže. Obrázek odpovídá situaci, ve které je použito inkrementální (relativní) čidlo polohy. Je-li použito absolutní čidlo polohy, bude poloha dosažená příkazem „mov“(ve kterém je číselné zadání polohy) vždy identická.

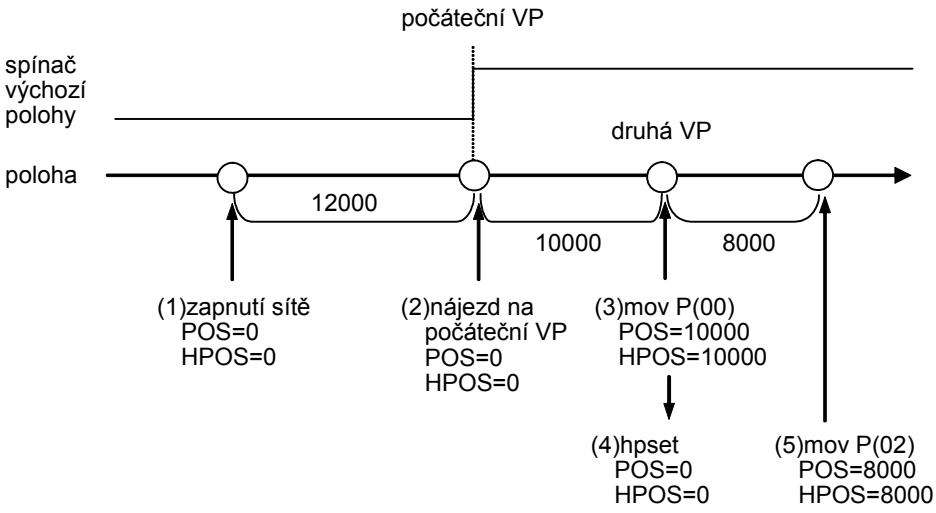


- **Příklad použití (použito inkrementální čidlo)**

  Následující obrázek znázorňuje vztah mezi počáteční VP, druhou VP a výslednou polohou.

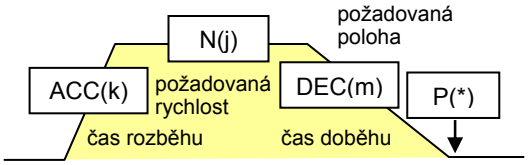
```
<datové okno>
    P(00)=10000        : N(00)=3000
    P(01)=9000         : N(01)=30
    P(02)=8000         : N(02)=8000      ' relativní poloha vůči druhé VP
    ACC(0)=50          : DEC(0)=50

<kódové okno>
    entry                                      '(1)
        ort 3    N(01) N(01) ACC(0) DEC(0)     '(2)
        nchg     P(01) N(01)                   '
        mov      P(00) N(00) ACC(0) DEC(0)     '(3)
        hpset                                  '(4) zadání druhé VP
        mov      P(02) N(02)                   '(5) specifikace relativní polohy vůči druhé VP
    end
```



instrukce mov	polohová pohybová instrukce

- **Formát**

Formát		popis operace
(1)	mov P(i) [N(j)] [ACC(k)] [DEC(m)][;&]	pohyb na polohu P(*) specifikovanou rychlostí N(j). ACC(k), DEC(m) zadání rozběhového a doběhového času (čas pro rozběh a doběh z 0 na N_{max} a opačně). je-li místo definice polohy P(*) použito Z, pak je dosaženo polohy při příchodu impulsu fáze Z. (* = i, Xn, Xw)
(2)	mov P(Xn) [N(j)] [ACC(k)] [DEC(m)][;&]	
(3)	mov P(Xw) [N(j)] [ACC(k)] [DEC(m)][;&]	
(4)	mov Z [N(j)] [ACC(k)][;&]	

- **Popis**

Instrukce "mov" provede přesun na požadovanou polohu P(*) požadovanou rychlostí N(j). Parametry ACC(k) a DEC(m) specifikují rozběhový a doběhový čas pro rozběh a doběh z 0 na N_{max} a opačně. Je-li místo definice polohy P(*) použito Z, pak je dosaženo polohy při příchodu impulsu fáze Z. Nejsou-li v příkazu „mov“ specifikovány parametry [N(j)], [ACC(k)], a [DEC(m)], pak se použije hodnot specifikovaných dříve. Pokud není zadán příznak [;&], pak nedojde k provedení dalšího instrukčního řádku dříve, než chyba polohy neklesne pod dovolenou toleranci specifikovanou v parametru Fb-23. Aby program pokračoval v provádění dalších příkazů je nezbytné zadat příznak [;&] pro každý příkaz „mov“ (paralelní provádění). Za těchto podmínek je nutné aby uživatelský program sledoval proměnnou INP zda je polohování ukončeno.

Instrukce "mov" je prováděna s lineárním průběhem rozběhu a doběhu. Lze použít i rozběhových a doběhových S-křivek. Hloubku zakřivení je možné zadat třemi úrovněmi v parametru Fb-30 nebo proměnné SCV. Bližší informace naleznete v kapitole 6 „popis parametrů“ v uživatelské příručce servopohonu. Popis proměnné SCV je uveden v této příručce.

Obecné předpoklady pro všechny formáty instrukce "mov" jsou popsány níže:

POZN.1: Nejsou-li parametry [N(j)], [ACC(k)] a [DEC(m)] alespoň jednou od začátku programu („entry“) nastaveny, dojde k chybě E45. Proto je nutné parametry [N(j)], [ACC(k)], a [DEC(m)] alespoň jednou nastavit.

POZN.2: je-li N(j) = 0, dojde k chybě provádění E45.

POZN.3: Překontrolujte, zda-li je velikost pohybu na jednu instrukci v rozsahu -2147483647 až 2147483647 pulsů (-65535 až 65535 otáček). Je-li zadána jakákoliv větší hodnota, servopohon změní směr otáčení.
Příklad) zvolíte-li nastavení polohy 2147483647 a je-li aktuální poloha -10000, pak servomotor pojede opačně, dokud čítač polohy nepřeteče a následně načte polohu 2147483647.

POZN.4: Je-li vzdálenost mezi aktuální a požadovanou polohou krátká, pohon se nemusí rozběhnout na požadovanou rychlost a rozběh plynule přejde v doběh.

POZN.5: V počátečním stavu bezprostředně po spuštění programu je počáteční výchozí poloha (VP) identická s druhou VP. Použitím instrukce "hpset" lze definovat druhou VP rozdílně od počáteční VP. Proto provedení instrukce „mov“ před instrukcí „hpset“ a po ní bude mít rozdílnou výslednou polohu

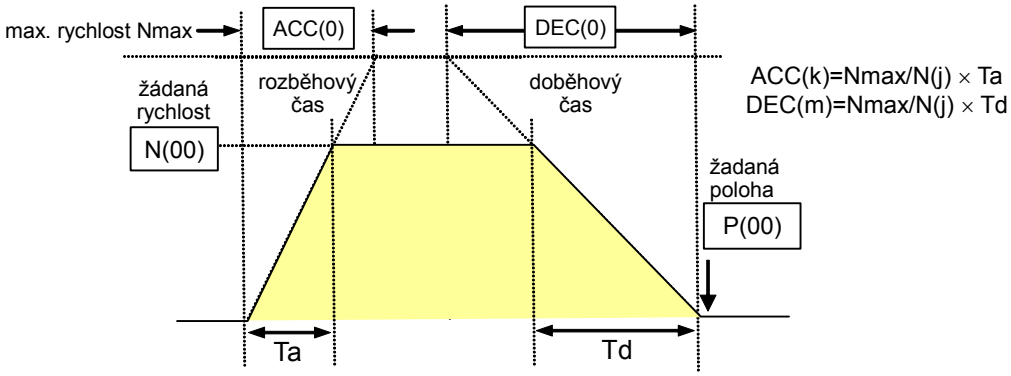
POZN.6: Jak je patrné z uvedeného diagramu, měl by mít průběh rychlosti při provádění instrukce "hp" lichoběžníkový tvar, plynoucí z procesu polohování na VP.Je-li nastaveno nízké zesílení rychlostní nebo polohové regulační smyčky může docházet v průběhu provádění k deviaci (nedostatečná dynamika pohonu), takže průběh rychlosti nebude lichoběžníkový.

(Pokračování z předchozí stránky – instrukce "mov")

Formát (1): Polohová instrukce s přímým zadáním polohy (mov P(j))

Tento formát umožňuje přímé zadání pohybu na požadovanou polohu P(i) rychlostí N(j) s rozběhovým ACC(k) a doběhovým DEC(m) časem. Grafické znázornění výsledného pohybu je na obrázku níže.

Formát (1): mov P(00) N(00) ACC(0) DEC(0)



P(i): Specifikujte požadovanou polohu, přičemž 1 dig = 1/32768 otáčky. Poloha musí být zadána relativně vůči druhé VP.

N(j): Specifikujte rychlost pohybu, přičemž 1 dig = 1 min^{-1}. Znaménko zadané rychlosti nemá vliv na směr pohybu. Směr pohybu je určen znaménkem hodnoty, která vznikne odečtením současné polohy pohonu (POS) od požadované polohy P(i).

ACC(k): Specifikuje čas rozběhu z rychlosti 0 na rychlost N(j) (jednotka 1 dig = 0.01 s). Hodnota ACC(k) je hodnota času rozběhu z rychlosti 0 na maximální rychlost N_{max} . Vztah mezi zadanou dobou rozběhu ACC(K) a skutečnou dobou rozběhu Ta na rychlost N(j) je názorně patrný z předchozího obrázku...

DEC(m): Specifikuje čas doběhu z rychlosti N(j) na rychlost 0 (jednotka 1 dig = 0.01 s). Hodnota DEC(m) je hodnota času doběhu z maximální rychlosti N_{max} na rychlost 0 . Vztah mezi zadanou dobou doběhu DEC(m) a skutečnou dobou doběhu Td z rychlosti N(j) je názorně patrný z předchozího obrázku.

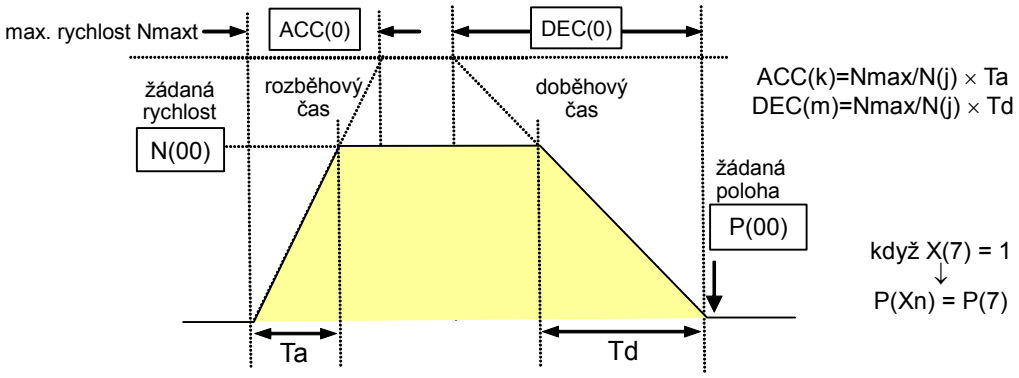
;&: Tímto příznakem lze specifikovat režim současného provádění. V režimu současného provádění se paralelně s instrukcí „mov“ provádí další instrukce na následujícím řádku (v cyklu následujícím po cyklu, ve kterém je uskutečněn start polohové instrukce). Pokud není zvolen režim současného provádění, pak se instrukční řádek následující po instrukci „mov“ bude provádět až tehdy, když hodnota chyby polohy bude nižší než nastavená hodnota tolerančního pásma Fb-23.

POZN.1: Je-li přednastavená poloha zvyšována nebo snižována proměnnou P(), je zvýšení nebo snížení provedeno i když dojde k překročení meze 2147483647 nebo -2147483647 (v případech provádění operace pevně dlouhého krok a pod.)
Příklad) je-li požadovaný posun o -2147418112 pulsů a přednastavená poloha je 2147418112, servomotor provede pohyb vpřed o +131072 pulsů a zastaví. Výsledná poloha bude -2147418110.

POZN.2: Je potřeba brát v úvahu i všechny předpoklady pro instrukcí „mov“ uvedené dříve.

Formát (2): Polohová instrukce, kdy žádaná poloha je zadána konfigurací logických kontaktních vstupů (mov P(Xn))

Tento formát umožňuje zadání pohybu na požadovanou polohu P(Xn) rychlostí N(j) s rozběhovým ACC(k) a doběhovým DEC(m) časem. Číslo polohy Xn je zadáno aktivním logickým vstupem. Je-li svorka X(j) aktivní (ON) v okamžiku provádění instrukce „mov“ pak je žádanou polohou poloha P(j) (např. X(7)=1, pak P(Xn)=P(7)). Je-li současně aktivních více vstupů X(), pak je vybrána jako žádaná poloha s číslem nejnižšího aktivního. Grafické znázornění výsledného pohybu je na obrázku níže.



P(Xn): Ku specifikaci žádané polohy použijte proměnnou P(), přičemž 1 dig = 1/32768 otáčky. Poloha musí být zadána relativně vůči druhé VP.

N(j): Specifikujte rychlost pohybu, přičemž 1 dig = 1 min^{-1}. Znaménko zadané rychlosti nemá vliv na směr pohybu. Směr pohybu je určen znaménkem hodnoty, která vznikne odečtením současné polohy pohonu (POS) od požadované polohy P(i).

ACC(k): Specifikuje čas rozběhu z rychlosti 0 na rychlost N(j) (jednotka 1 dig = 0.01 s). Hodnota ACC(k) je hodnota času rozběhu z rychlosti 0 na maximální rychlost N_{max} . Vztah mezi zadanou dobou rozběhu ACC(K) a skutečnou dobou rozběhu Ta na rychlost N(j) je názorně patrný z předchozího obrázku.

DEC(m): Specifikuje čas doběhu z rychlosti N(j) na rychlost 0 (jednotka 1 dig = 0.01 s). Hodnota DEC(m) je hodnota času doběhu z maximální rychlosti N_{max} na rychlost 0 . Vztah mezi zadanou dobou doběhu DEC(m) a skutečnou dobou doběhu Td z rychlosti N(j) je názorně patrný z předchozího obrázku.

;&: Tímto příznakem lze specifikovat režim současného provádění. V režimu současného provádění se paralelně s instrukcí „mov“ provádí další instrukce na následujícím řádku (v cyklu následujícím po cyklu, ve kterém je uskutečněn start polohové instrukce). Pokud není zvolen režim současného provádění, pak se instrukční řádek následující po instrukci „mov“ bude provádět až tehdy, když hodnota chyby polohy bude nižší než nastavená hodnota tolerančního pásma Fb-23.

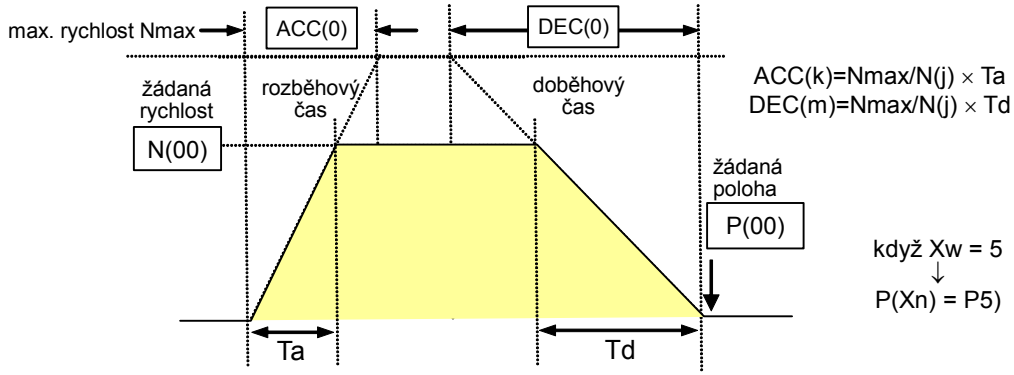
POZN.1: Pokud není žádná ze vstupních logických svorek X(j) aktivní, pak dojde k chybě E45. Proto zajistěte, aby v době provádění instrukce „mov“ byla některá ze svorek aktivní.

POZN.2: Doporučujeme použít před instrukcí „mov“ instrukci "wait X(j)" nebo "wait Xw", aby byl zaručeno, že vstup X() bude aktivní.

POZN.3: Je potřeba brát v úvahu i všechny předpoklady pro instrukcí „mov“ uvedené dříve.

Formát (3): Polohová instrukce, kdy žádaná poloha je zadána binární kombinací logických kontaktních vstupů (mov P(Xw)).
Tento formát umožňuje zadání pohybu na požadovanou polohu P(Xw) rychlostí N(j) s rozběhovým ACC(k) a doběhovým DEC(m) časem. Číslo polohy Xw je zadáno binární kombinací stavu logických vstupů. Žádaná poloha je zvolena podle toho jaké číslo představuje binární kombinace stavu vstupních svorek Xw (On i OFF). Je-li např. hodnota binární kombinace vstupů Xw=5, pak je žádaná poloha P(j)=P(5). Grafické znázornění výsledného pohybu je na obrázku níže.

Formát (3): mov P(Xw) N(00) ACC(0) DEC(0)



P(Xw): Ku specifikaci žádané polohy použijte proměnnou P(), přičemž 1 dig = 1/32768 otáčky. Poloha musí být zadána relativně vůči druhé VP.

N(j): Specifikujte rychlost pohybu, přičemž 1 dig = 1 min^{-1}. Znaménko zadané rychlosti nemá vliv na směr pohybu. Směr pohybu je určen znaménkem hodnoty, která vznikne odečtením současné polohy pohonu (POS) od požadované polohy P(i).

ACC(k): Specifikuje čas rozběhu z rychlosti 0 na rychlost N(j) (jednotka 1 dig = 0.01 s). Hodnota ACC(k) je hodnota času rozběhu z rychlosti 0 na maximální rychlost N_{max} . Vztah mezi zadanou dobou rozběhu ACC(K) a skutečnou dobou rozběhu Ta na rychlost N(j) je názorně patrný z předchozího obrázku.

DEC(m): Specifikuje čas doběhu z rychlosti N(j) na rychlost 0 (jednotka 1 dig = 0.01 s). Hodnota DEC(m) je hodnota času doběhu z maximální rychlosti N_{max} na rychlost 0 . Vztah mezi zadanou dobou doběhu DEC(m) a skutečnou dobou doběhu Td z rychlosti N(j) je názorně patrný z předchozího obrázku.

;&: Tímto příznakem lze specifikovat režim současného provádění. V režimu současného provádění se paralelně s instrukcí „mov“ provádí další instrukce na následujícím řádku (v cyklu následujícím po cyklu, ve kterém je uskutečněn start polohové instrukce). Pokud není zvolen režim současného provádění, pak se instrukční řádek následující po instrukci „mov“ bude provádět až tehdy, když hodnota chyby polohy bude nižší než nastavená hodnota tolerančního pásma Fb-23.

POZN.1: Rozsah zadání možných poloh je od P(00) do P(99), pokud hodnota Xw přesáhne dovolený rozsah dojde k chybě E45.

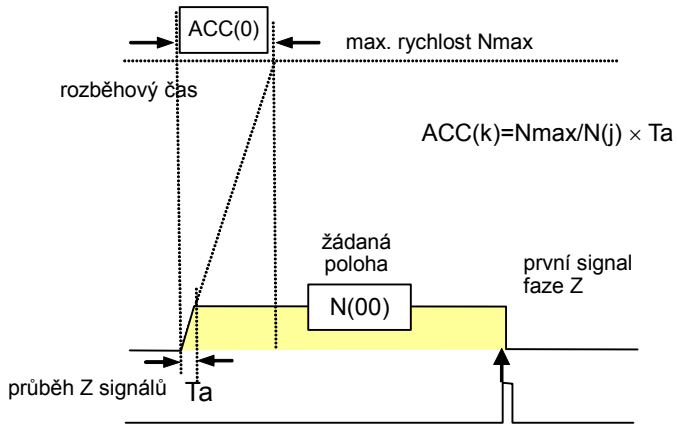
POZN.2: Doporučujeme použít před instrukcí „mov“ instrukci "wait X(j)" nebo "wait Xw", aby byl zaručeno, že vstup X() bude aktivní

POZN.3: Je potřeba brát v úvahu i všechny předpoklady pro instrukcí „mov“ uvedené dříve.

Formát (4): Polohová instrukce, kdy žádanou polohou je poloha při příchodu prvního pulsu Z (mov Z)

Tento formát umožňuje zadání pohybu rychlostí N(j) s rozběhovým časem ACC(k) do té doby, než je indikován první puls fáze Z. Pokud servomotor v době zadání instrukce „mov Z“ stojí na pulsu Z není proveden žádný pohyb. Grafické znázornění výsledného pohybu je na obrázku níže.

Formát (4): mov Z N(00) ACC(0)



N(j): Specifikujte rychlost pohybu, přičemž 1 dig = 1 min^{-1}. Oproti formátům 1÷3 má znaménko zadané rychlosti N(j) vliv na směr pohybu

ACC(k): Specifikuje čas rozběhu z rychlosti 0 na rychlost N(j) (jednotka 1 dig = 0.01 s). Hodnota ACC(k) je hodnota času rozběhu z rychlosti 0 na maximální rychlost N_{max} . Vztah mezi zadanou dobou rozběhu ACC(K) a skutečnou dobou rozběhu Ta na rychlost N(j) je názorně patrný z předchozího obrázku.

;&: Tímto příznakem lze specifikovat režim současného provádění. V režimu současného provádění se paralelně s instrukcí „mov“ provádí další instrukce na následujícím řádku (v cyklu následujícím po cyklu, ve kterém je uskutečněn start polohové instrukce). Pokud není zvolen režim současného provádění, pak se instrukční řádek následující po instrukci „mov“ bude provádět až tehdy, když hodnota chyby polohy bude nižší než nastavená hodnota tolerančního pásma Fb-23.

POZN.1: Mezi příchodem signálu fáze Z a jeho rozpoznáním v regulátoru pohonu je určitá časová prodleva, proto doporučujeme provádět pohyb instrukcí „mov Z“ při nízkých rychlostech

POZN.3: Je potřeba brát v úvahu i všechny předpoklady pro instrukcí „mov“ uvedené dříve.

- **Příklady použití (mov)**

Příklad 1) použití formátu (1)

<datové okno>

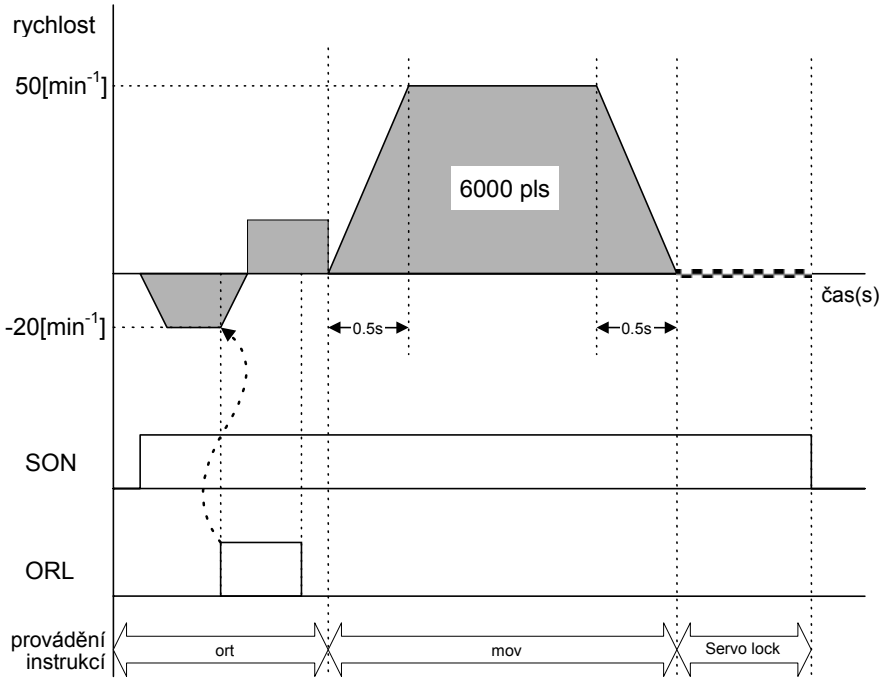
ACC(0)=50[s] , DEC(0)=50[s] ,
N(00)=50 [min^{-1}] , N(01)=5[min_{-1}] , N(02)=20[min^{-1}]
P(00)=6000 [pls] … (1)

<kódové okno>

```
entry
   ort 3  N(02)  N(01)  ACC(0)  DEC(0) dosažení VP.                  … (2)
   mov  P(00)  N(00)  ACC(0)  DEC(0) provedení přesunu na polohu    … (3)
end                                                                 … (4)
```

<Popis>

1) nastavení dat z datového okna editoru programu (1).
2) provedení nájezdu na výchozí polohu (ort).
3) Po nájezdu na VP se rychlost zvyšuje po rozběhové rampě ACC(0)=50 [s] až je dosaženo rychlosti N(00) = 50 [min^{-1}]. Jakmile se přiblíží indikovaná poloha k poloze P(00)=6000 pulsů započte snižování rychlosti po doběhové rampě DEC(0) a při dosažení polohy P(00) se servopohon zastaví.
4) Program končí (I když se program zastaví, servopohon zůstane zablokován na poloze 6000 [pls] dokud není deaktivován signál SON (SON = OFF)).



Příklad 2) použití formátu (2)

<datové okno>

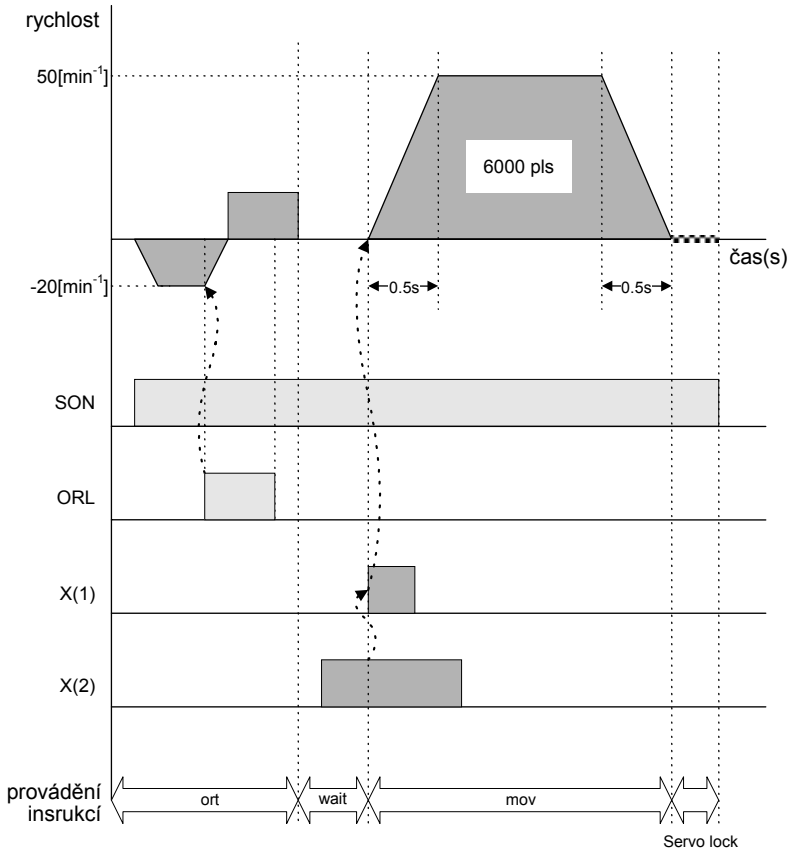
ACC(0)=50[s] , DEC(0)=50[s] ,
N(00)=50 [min^{-1}] , N(01)=5 [min^{-1}] , N(02)=20[min^{-1}] ,
P(02)=6000 [pls] … (1)

<kódové okno>

```
entry
   ort 3  N(02) N(01) ACC(0) DEC(0)   dosažení VP.                  … (2)
   wait   X(11)=1                     čeká dokud není  X(11) = 1.   … (3)
   mov    P(Xn) N(00) ACC(0) DEC(0)   provední přesunu na polohu    … (4)
end                                                                 … (5)
```

<Popis>

1) nastavení dat z datového okna editoru programu (1)
2) provedení nájezdu na výchozí polohu (ort)
3) Po nájezdu na VP servopohon čeká dokud není aktivován vstup X(11) = 1. Je-li v době aktivace vstupu X(11) již aktivní vstup X(02) =1 pak program provede další krok.
4) Rychlost se zvyšuje po rozběhové rampě ACC(0)=50 [s] až je dosaženo rychlosti N(00)= 50 [min^{-1}]. Jakmile se přiblíží indikovaná poloha k poloze P(00)=6000 pulsů započte snižování rychlosti po doběhové rampě DEC(0) a při dosažení polohy P(00) se servopohon zastaví.
5) Program končí (I když se program zastaví, servopohon zůstane zablokován na poloze 6000 [pls] dokud není deaktivován signál SON (SON = OFF)).



Příklad 3) požití formátu (3)

<datové okno>

ACC(0)=50[s] , DEC(0)=50[s] ,
N(00)=50[min^{-1}] , N(01)=5[min^{-1}] , N(02)=20[min^{-1}] ,
P(10)=6000 [pls] … (1)

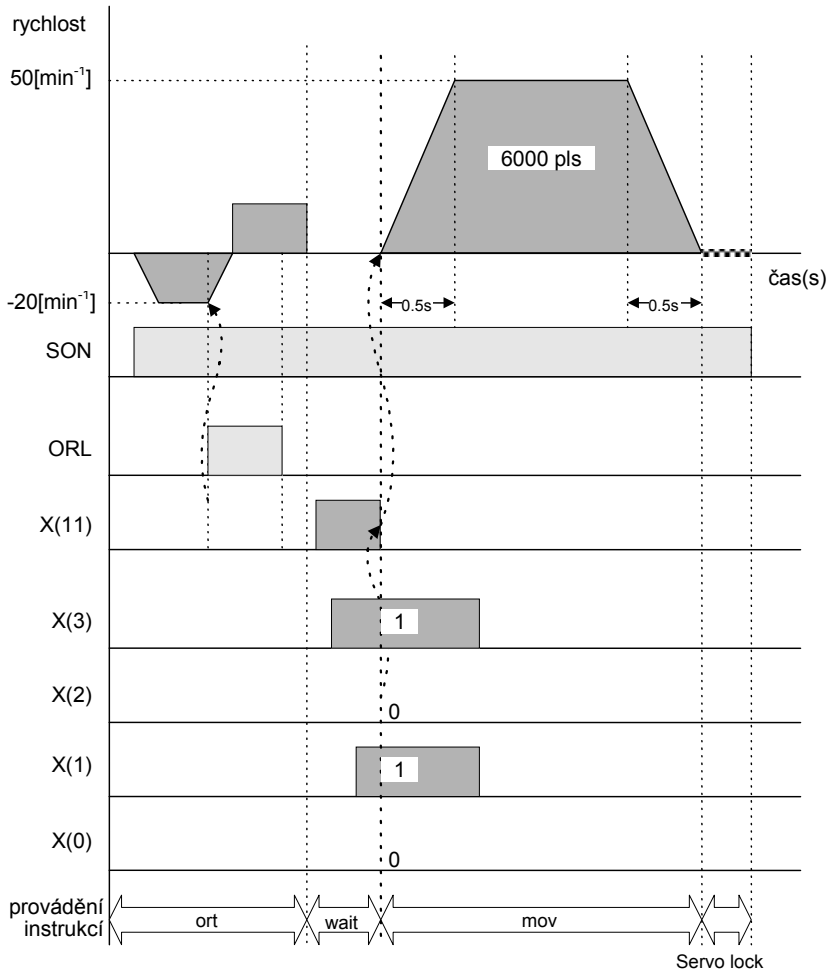
<kódové okno>

```
entry
   ort 3   N(02) N(01) ACC(0) DEC(0)     dosažení VP.                  … (2)

   wait    X(11)=1                       čeká dokud není X(11) = 1.
   wait    X(11)=0                       čeká dokud není X(11) = 0.    … (3)
   mov     P(Xw) N(00) ACC(0) DEC(0)     provedení přesunu na polohu   … (4)
end                                                                     … (5)
```

<Popis>

1) nastavení dat z datového okna editoru programu (1)
2) provedení nájezdu na výchozí polohu (ort)
3) Po dosažení VP a po aktivaci signálu "X(11) = 1" servo čeká dokud signál nezmění svoji hodnotu na X(11) = 0. V této době jsou nastaveny vstupy "X(00) = 0", "X(01) = 1", "X(02) = 0", a "X(03) = 1". Jakmile za tohoto stavu přejde signál X(11) do 0 program provede další krok.



4) Rychlost se zvyšuje po rozběhové rampě ACC(0)=50 [s] až je dosaženo rychlosti N(00)= 50 [min^{-1}]. Jakmile se přiblíží indikovaná poloha k poloze P(00)=6000 pulsů započte snižování rychlosti po doběhové rampě DEC(0) a při dosažení polohy P(00) se servopohon zastaví.
5) Program končí (I když se program zastaví, servopohon zůstane zablokován na poloze 6000 [pls] dokud není deaktivován signál SON (SON = OFF)).

Příklad 4) použití formátu (4)

<datové okno>

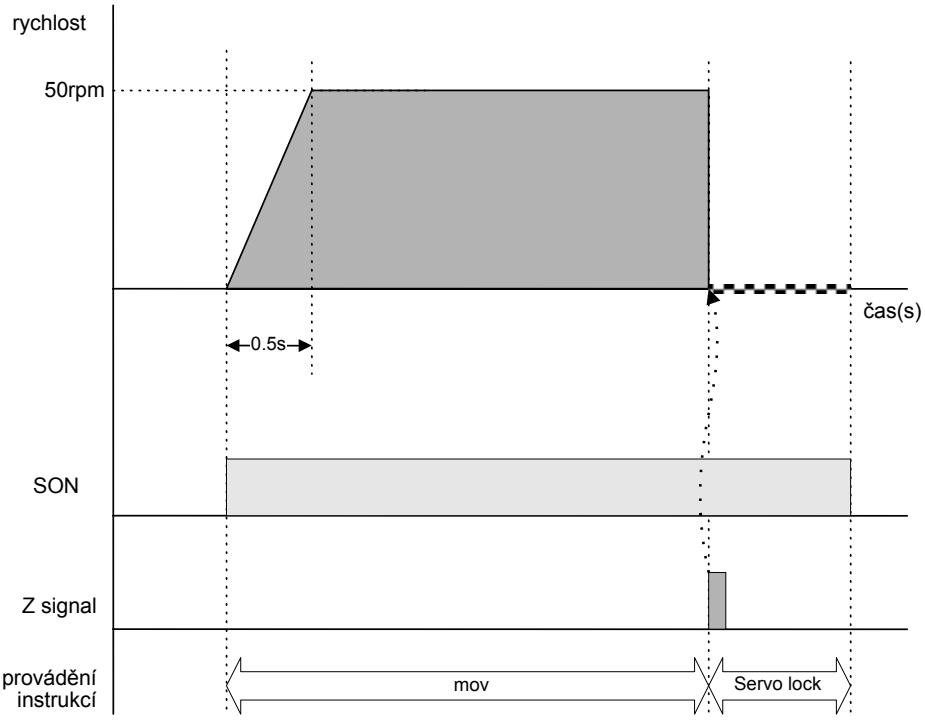
ACC(0)=50[s] , N(00)=50 [min^{-1}] ⋯ (1)

<kódové okno>

```
entry
   mov    Z  N(00) ACC(0)        provedení přesunu na polohu      ⋯ (2)
end                                                               ⋯ (5)
```

<Popis>

1) nastavení dat z datového okna editoru programu (1)
2) Servomotor se otáčí směrem a rychlostí nastavenou v N(00). Žádané polohy je dosaženo, jakmile přijde první signál fáze Z. (I když se program zastaví, servopohon zůstane zablokován na poloze 6000 [pls] dokud není deaktivován signál SON (SON = OFF)).



Příklad 5) současné provádění

Instrukce "mov" nedovolí pokračování programu další instrukcí, dokud se chyba polohy nesníží pod nastavenou hodnotu toleranční pásma Fb-23. Je-li zvolen režim současného provádění, pak instrukce "mov" spouští následující instrukci, která je provedena v dalším cyklu. Paralelní provádění lze využít např. k nastavení výstupů v průběhu polohování, nebo k zastavení polohování při příchodu vnějšího signálu.

Příklad 5-1) Povelování výstupních svorek v průběhu polohové operace

```
Y(00)=0
mov P(00) N(00) ACC(0) DEC(0) ;&        začíná polohová operace
Y(00)=1                                 v dalším cyklu se nastaví výstup Y(0) na 1
```

Příklad 5-2) Přerušení polohové operace při přechodu svorky X(00) do stavu "1"

```
    mov P(00) N(00) ACC(0) DEC(0) ;&
    while INP=0
    if  X(00)=0 then LBL1
    stop
LBL1:  wend
```

instrukce nchg — instrukce změny rychlosti

- **Formát**

Formát	popis operace
nchg P(i) N(j) [ACC(k)] [DEC(m)]	změní rychlost při dosažení nastavené polohy

- **Popis**

Tato instrukce zajistí změnu rychlosti při dosažení určité polohy, např. v průběhu provádění instrukce „mov“ nebo „speed“. Instrukce „nchg“ musí být provedena před započetím instrukce "mov"/"speed". Instrukce "nchg" se projeví pouze jedenkrát (změna rychlosti nastane pouze v následující instrukci "mov" nebo "speed") v následující pohybové instrukci. Aby byla instrukce správně provedena je nutné zadat polohu P(i), při jejímž dosažení má být provedena změna, rychlost N(j), která má následovat, rozběhový ACC(k) a doběhový DEC(m) čas, který se uplatní při změně. Požadujete-li změnu rychlosti vícekrát (při různých polohách) pak proveďte instrukci "nchg" tolikrát kolikrát potřebujete a nakonec proveďte pohybovou instrukci "mov"/"speed". Pokud nespecifikujete ve všech případech parametry ACC(k) a DEC(m), pak se použijí poslední zadané hodnoty.

P(i): Specifikujte požadovanou polohu, přičemž 1 dig = 1/32768 otáčky. Poloha musí být zadána relativně vůči druhé VP.

N(j): Specifikujte rychlost pohybu, přičemž 1 dig = 1 min^{-1}. Následuje-li pohybová instrukce „speed“ pak se směr otáčení řídí znaménkem rychlosti N(j) v instrukci “nchg“, následuje-li polohová instrukce „mov“, pak směr pohybu vyplývá z této instrukce.

ACC(k): Specifikuje čas rozběhu z rychlosti 0 na rychlost N(j) (jednotka 1 dig = 0.01 s). Hodnota ACC(k) je hodnota času rozběhu z rychlosti 0 na maximální rychlost N_{max} .

DEC(m): Specifikuje čas doběhu z rychlosti N(j) na rychlost 0 (jednotka 1 dig = 0.01 s). Hodnota DEC(m) je hodnota času doběhu z maximální rychlosti N_{max} na rychlost 0 .

POZN.1: Nejsou-li parametry ACC(k) nebo DEC(m) zadány, pak při provádění první instrukce "nchg" dojde k chybě provádění (E45). Proto je potřeba nastavit parametry ACC(k) a DEC(m) alespoň jednou na začátku programu (po „entry“).

POZN.2: Pokud je N(j) = 0, dojde k chybě provádění (E45) bez ohledu na instrukce "mov" nebo "speed".

POZN.3: Je-li vzdálenost mezi bodem změny rychlosti a koncem pohybu krátká, nemusí být ani zadané druhé rychlosti dosaženo, protože dříve dojde k doběhu a zastavení.

POZN.4: V počátečním stavu, bezprostředně po zapnutí se výchozí poloha (počáteční) shoduje s druhou VP. Instrukce "hpset" nám ale umožňuje zvolit jako VP právě dosaženou polohu. Proto i když provedeme stejnou instrukci polohy "mov" (stejné parametry a poloha) před a po instrukci "hpset", budou výsledné polohy odlišné.

POZN.5: Jak je patrné z uvedeného diagramu, měl by mít průběh rychlosti při provádění instrukce "hp" lichoběžníkový tvar, plynoucí z procesu polohování na VP.Je-li nastaveno nízké zesílení rychlostní nebo polohové regulační smyčky může docházet v průběhu provádění k deviaci (nedostatečná dynamika pohonu), takže průběh rychlosti nebude lichoběžníkový.

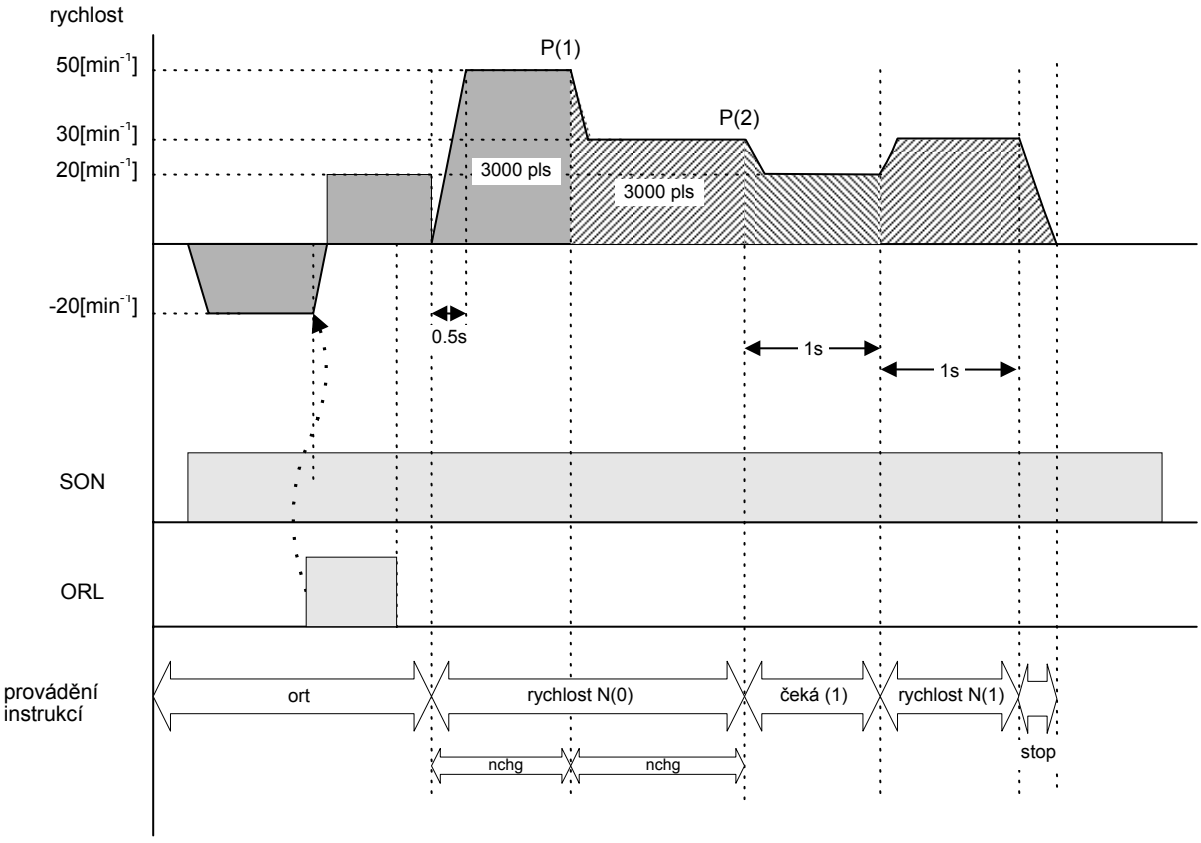
- **Příklad použití**

<datové okno>

P(00)=10000
P(01)=3000 : P(02)=P(01)+3000
N(00)=50 : N(01)=30
N(02)=20
ACC(0)=50 : DEC(0)=50

<kódové okno>

```
entry
    ort 3      N(02) N(02) ACC(0) DEC(0)
    nchg       P(02) N(02)
    nchg       P(01) N(01)
    speed      N(00) ACC(0) DEC(0)
    wait 1                                  čeká 1 s.
    speed      N(01)
    wait 1                                  čeká 1 s.
    stop
end
```



instrukce ort	povel návratu na VP

- **Format**

Formát	popis operace
ort 0 [N(j)] [ACC(l)] [DEC(m)]	otáčí servomotorem směrem vpřed pomalou rychlostí dokud nedosáhne VP.
ort 1 [N(j)] [ACC(l)] [DEC(m)]	otáčí servomotorem směrem vzad pomalou rychlostí dokud nedosáhne VP.
ort 2 [N(j)] [N(k)] [ACC(l)] [DEC(m)]	otáčí servomotorem vpřed vysokou rychlostí dokud nedosáhne VP způsobem 1 [N(j)]: rychlost 1 nájezdu na VP [N(k)]: rychlost 2 nájezdu na VP
ort 3 [N(j)] [N(k)] [ACC(l)] [DEC(m)]	otáčí servomotorem vzad vysokou rychlostí dokud nedosáhne VP způsobem 1 [N(j)]: rychlost 1 nájezdu na VP [N(k)]: rychlost 2 nájezdu na VP
ort 4 [N(j)] [N(k)] [ACC(l)] [DEC(m)]	otáčí servomotorem vpřed vysokou rychlostí dokud nedosáhne VP způsobem 2 [N(j)]: rychlost 1 nájezdu na VP [N(k)]: rychlost 2 nájezdu na VP
ort 5 [N(j)] [N(k)] [ACC(l)] [DEC(m)]	otáčí servomotorem vzad vysokou rychlostí dokud nedosáhne VP způsobem 2 [N(j)]: rychlost 1 nájezdu na VP [N(k)]: rychlost 2 nájezdu na VP
ort 6	libovolné dosažení VP

- **Popis**

Lze zvolit až sedm různých režimů nájezdu na výchozí polohu. Při dosažení VP jsou hodnoty parametrů Fb-14 a Fb-15 zapsány jako polohový údaj odpovídající VP (posun VP). Operace "ort" nedovolí pokračování provádění programu, dokud není nájezd na VP ukončen. Předchází-li instrukci "ort" instrukce "chgORG", pak nájezd na VP započne až svorka ORG přejde do stavu ON. Jakmile operace nájezdu na VP začne, vypnutím svorky ORG ji již nelze zastavit. Pokud není před instrukcí "ort" instrukce "chgORG“, pak se nájezd na VP provádí okamžitě. Bližší informace k nájezdu na VP naleznete v uživatelské příručce k servopohonu , včetně popisu významu svorek ORG a ORL. (Nezapomeňte však, ža svorkou ORG nelze nájezd na VP zrušit).

N(j): Nastavte rychlost 1 nájezdu na VP (vysoká rychlost) nebo rychlost nájezdu na VP pro režim pomalého nájezdu na VP, kdy 1 dig = 1 min^{-1}. Směr otáčení je určen režimem nájezdu na VP a stavem svorky ORL a není závislý na znaménku rychlosti N(j).

N(k): Nastavte rychlost 2 nájezdu na VP (pomalá rychlost), kdy 1 dig = 1 min^{-1}. Směr otáčení je určen režimem nájezdu na VP a stavem svorky ORL a není závislý na znaménku rychlosti N(k).

ACC(l): Specifikuje čas rozběhu pro operaci nájezdu na VP (jednotka 1 dig = 0.01 s). Hodnota ACC(k) je hodnota času rozběhu z rychlosti 0 na maximální rychlost N_{max} .

DEC(m): Specifikuje čas doběhu pro operaci nájezdu na VP (jednotka 1 dig = 0.01 s). Hodnota DEC(m) je hodnota času doběhu z maximální rychlosti N_{max} na rychlost 0 .

POZN.1: Nejsou-li parametry N(j), N(k), ACC(k), nebo DEC(m) specifikovány pro první pohybovou instrukci od počátku programu dojde k chybě provádění E45. Je proto nutné alespoň jednou hodnoty parametrů N(j), N(k), ACC(k), a DEC(m) zadat.

POZN.2: Je-li N(j) = 0, dojde k chybě provádění E45.

POZN.3: Je-li vzdálenost mezi aktuální polohou a výchozí polohou krátká, nemusí být ani zadané rychlosti dosaženo, protože dříve dojde k doběhu a zastavení.

POZN.4: Je-li operace nájezdu na VP provedena po nastavení VP (druhé VP), pak VP (druhá VP) odpovídá nastavenému posunu (Fb-14/Fb-15). Proto je nutné druhou VP nastavit pomocí operace "hpset".

POZN.5: Jak je patrné z uvedeného diagramu, měl by mít průběh rychlosti při provádění instrukce "hp" lichoběžníkový tvar, plynoucí z procesu polohování na VP.Je-li nastaveno nízké zesílení rychlostní nebo polohové regulační smyčky může docházet v průběhu provádění k deviaci (nedostatečná dynamika pohonu), takže průběh rychlosti nebude lichoběžníkový.

POZN.6: Zabezpečte, aby čas pohybu při nájezdu na VP nepřekročil 30 minut. Pokud je tento limit překročen, operace nájezdu na VP bude nesprávná (bludná, náhodná).

POZN.7: Operaci nájezdu na VP je nutné používat, pokud je připojeno inkrementální čidlo polohy. Pokud použijeme operaci nájezdu na VP a máme připojeno absolutní čidlo polohy, pak aktuální poloha druhá VP odpovídá nastavenému posunu VP (Fb-14 a Fb-15).

POZN.8: Pro nájezd na VP použijte nízkou rychlost, protože při zpracování signálu ze svorky ORL dochází k určitému malému zpoždění.

- **Příklady použití**

Příklad 1) Nájezd na VP nízkou rychlostí

<datové okno>

```
ACC(0)=50   : DEC(0)=50
N(00)=5000
N(02)=50
```

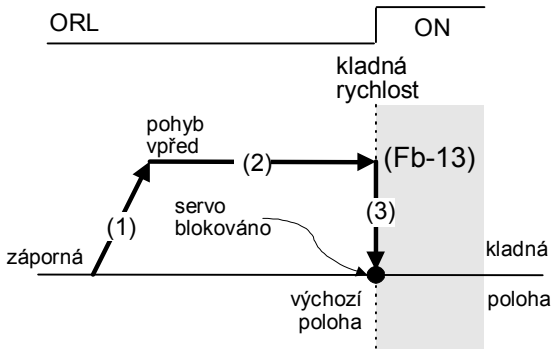
<kódové okno>

```
entry
    ort 0 N(02) ACC(0) DEC(0)
end
```

<popis>

(1) Servomotor se otáčí vpřed rychlostí N(02).

(2) V okamžiku, kdy se sepne signál ORL, servomotor zastaví a výsledná poloha je výchozí polohou.
(Po dokončení nájezdu na VP servomotor zastaví a zůstává ve stavu servo blokováno (servo-locked state).)



Příklad 2) Nájezd na VP vysokou rychlostí způsobem 1

<datové okno>

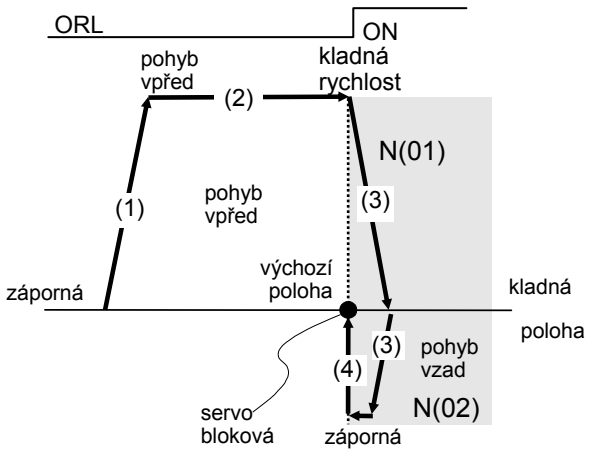
ACC(0)=50 : DEC(0)=50
N(00)=5000
N(01)=3000 : N(02)=50

<kódové okno>

```
entry
    ort 2 N(01) N(02) ACC(0) DEC(0)
end
```

<popis>

(1) Servopohon se otáčí rychlostí 1 směrem vpřed.
(2) Když se sepne signál ORL, servomotor dobíhá a zastaví.
(3) Servomotor se otáčí směrem vzad rychlostí 2.
(4) Když se signál ORL rozepne, servomotor zastaví.
(Po dokončení nájezdu na VP servomotor zastaví a zůstává ve stavu servo blokováno (servo-locked state).)



Příklad 3) Nájezd na VP vysokou rychlostí způsobem 2

<datové okno>

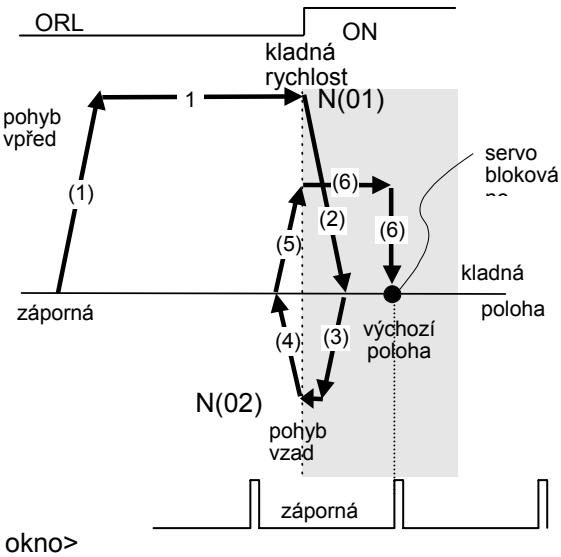
ACC(0)=50 : DEC(0)=50
N(00)=5000
N(01)=3000 : N(02)=50

<kódové okno>

```
entry
    ort 4 N(01) N(02) ACC(0) DEC(0)
end
```

<popis>

(1) Servopohon se pohybuje směrem vpřed rychlostí 1.
(2) Když se sepne signál ORL servopohon dobíhá a zastaví se.
(3) Servomotor se otáčí směrem vzad rychlostí 2.
(4) Když se signál ORL rozepne, servomotor dobíhá a zastaví se.
(5) Servomotor běží směrem vpřed rychlostí 2.
(6) První signál fáze Z po sepnutí signálu ORL je považován za výchozí polohu.



Příklad 4) libovolné dosažení VP

<datové okno>

ACC(0)=50 : DEC(0)=50
N(00)=5000 : P(00)=1000
N(01)=3000 : N(02)=50

<kódové okno>

```
entry
    ort 4 N(01) N(02) ACC(0) DEC(0)
    mov P(00)
    ort 6
end
```

<popis>

Přijde li signál ORL, pak je aktuální poloha považována za VP a jsou jí přiřazeny souřadnice dané parametry Fb-14/Fb-15.

instrukce smov	změna režimu regulace z rychlostní na polohovou

- **Formát**

Formát	popis operace
smov P(i)	přepne režim regulace z rychlostní na polohovou.

- **Popis**

Je-li tato operace uskutečněna při provozu v režimu regulace rychlosti (při provádění operace pohybu určitou rychlostí, neomezeného polohou), dojde v okamžiku (a poloze) provedení instrukce "smov" k přechodu na polohovou regulaci a pohyb je ukončen dosažením polohy P(i). Poloha P(i) je relativní poloha vůči poloze provedení instrukce "smov". Protože poloha zastavení se může měnit vlivem zpoždění v rozpoznání instrukce servopohonem, doporučujeme provádět tuto operaci při nízké rychlosti. Operace "smov" nedovolí provádění dalšího řádku programu, dokud není dokončeno dosažení polohy P(i) (aktuální poloha není v tolerančním pásmu dosažení polohy zadaném parametrem Fb-23).
Při provádění instrukce "smov" se změní režim regulace na polohovou regulaci a po dosažení určené polohy se motor okamžitě zastaví, bez doběhu. Tato instrukce vnitřně sníží omezení povelu rychlosti na hodnotu příslušnou pro polohovou regulaci N(), nastaví hodnotu P(i) jako povel polohy a přepne režim regulace na polohovou regulaci. Proto není potřeba nastavovat doběhový čas a doběh je určen zesílením rychlostní/polohové regulace. Nezapomeňte také, že chyba polohy v okamžiku provedení instrukce "smov" má obecně hodnotu polohy P(i).

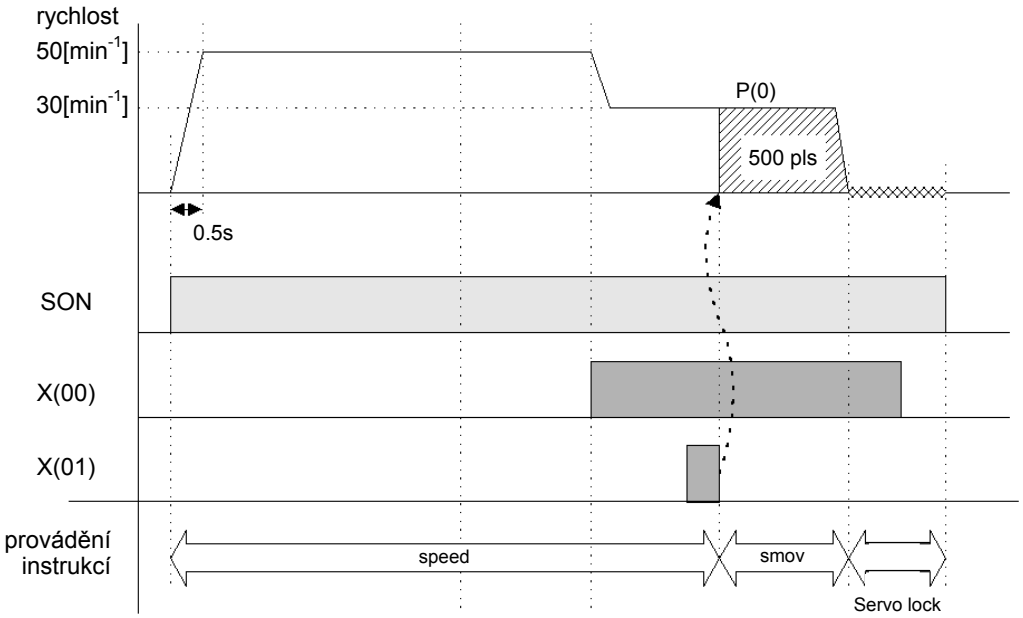
P(i): Nastavte polohovou vzdálenost mezi místem provedení instrukce "smov" a mistem zastavení pohonu za předpokladu že 1 dig = 1/32768 otáčky.

POZN.1: Je-li P(i) = 0, dojde k chybě provádění E45.

- **Příklad použití**

```
<datové okno>
    ACC(0)=50    : DEC(0)=50
    N(00)=50     : P(00)=1000
    N(01)=30     : P(01)=500

<kódové okno>
    entry
        speed N(00)
        wait X(00)=1
        speed N(01)
        U(00)=abs(NRF)  : wait U(00)=N(01)
        wait X(01)=1
        wait X(01)=0
        smov P(01)
    end
```


rychlost
50[min-1]
30[min-1]
P(0)
500 pls
0.5s
SON
X(00)
X(01)
provádění instrukcí
speed
smov
Servo lock

insrukce speed	instrukce pohybu v rychlostí regulaci

- **Formát**

Formát	popis operace
speed N(j) [ACC(k)] [DEC(m)]	provádí pohyb v rychlostí regulaci rychlostí N(j).

- **Popis**

instrukce "speed" provádí pohyb s konstantní rychlostí v rychlostí regulaci. Parametr N(j) specifikuje požadovanou rychlost pohybu, parametry ACC(k) a DEC(m) udávají rozběhový a doběhový čas. nejsou-li parametry specifikovány, použijí se dříve nastavené hodnoty. Tato instrukce je ukončena po uplynutí jednoho cyklu po začátku provádění. V dalším cyklu se provádí další instrukce a je dovoleno pokračování programu. Mezitím je dosaženo požadované rychlosti (odezní přechodové děje rozběhu nebo doběhu). Požadujeme-li změnu rychlost při dosažení určité polohy, je potřeba instrukci "speed" předřadit instrukci "nchg" (blíže odstavec "instrukce nchg). Je-li požadována změna rychlosti bezprostředně po instrukci "speed" v průběhu operace, pak lze takovéto změny dosáhnout přídavným provedením instrukce "speed". Při provádění instrukce "speed" dojde ke změně vnitřního režimu regulace z polohové regulace na regulaci rychlostní. Proto výsledná poloha zastavení bude posunuta, a to i v případě, že je operace prováděna nulovou rychlostí. Provádíme-li operaci zastavení, např. zastavení servopohonu v polohové regulaci jako obvykle, servopohon dobíhá v závislosti na doběhovém čase nastaveném v instrukci "speed

Rozběh a doběh je prováděn při instrukci "speed" lineárně. Abychom dosáhli hladkého průběhu přechodových dějů (snížení vibrací) je vhodné nastavit správnou hodnotu filtru povelu rychlosti (parametr Fd-20 nebo proměnnou NFILT). Parametr Fd-20 je popsán v kapitole 6 hlavní uživatelské příručky servozesilovače. Nastavení proměnné NFILT je popsáno v této uživatelské příručce.

N(j): Specifikujte rychlost pohybu, přičemž 1 dig = 1 min^{-1}. Směr pohybu odpovídá znaménku zadané rychlosti N(j).

ACC(k): Specifikuje čas rozběhu z rychlosti 0 na rychlost N(j) (jednotka 1 dig = 0.01 s). Hodnota ACC(k) je hodnota času rozběhu z rychlosti 0 na maximální rychlost N_{max} .

DEC(m): Specifikuje čas doběhu z rychlosti N(j) na rychlost 0 (jednotka 1 dig = 0.01 s). Hodnota DEC(m) je hodnota času doběhu z maximální rychlosti N_{max} na rychlost 0 .

POZN.1: Pokud hodláte provést operace v polohové regulaci jako "mov", "hp", nebo jiné v momentové regulaci jako "trq" nebo další, po instrukci "speed", proveďte napřed instrukci "stop" a vyčkejte dokud se pohon nezastaví (indikace nulové rychlosti).

POZN.2: Provádění instrukce "speed" probíhá i v bodě přerušení nebo v provádění programu po krocích (pohon se nezastaví !). Nezapomeňte na tuto skutečnost, aby jste se vyvarovali poškození stroje nebo dalším nepříjemnostem.

POZN.3: Pokud nejsou všechny parametry specifikovány před nebo v prvé instrukci "speed" po započetí programu ("entry") dojde k chybě provádění E45.

POZN.4: Je-li provedena instrukce "end", servopohon dobíhá a zastaví se bez ohledu na to zda je prováděna instrukce "speed". Aby pohon pracoval plynule a trvale vytvořte smyčku, ve které k provedení instrukce "end" nedojde.

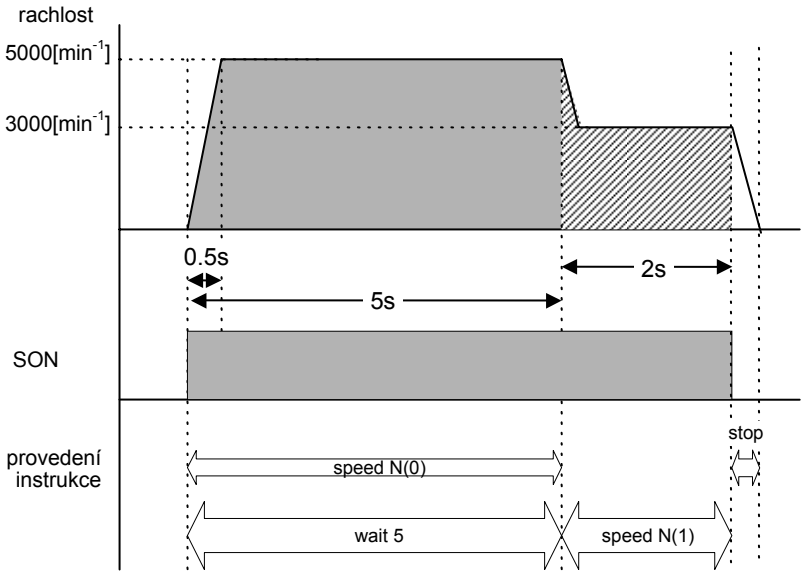
- **Příklad použití**

<datové okno>

ACC(0)=0.5 : DEC(0)=0.5
N(00)=5000 '5000min^{-1}
N(01)=3000 '3000min^{-1}

<kódové okno>

```
entry
   speed     N(00) ACC(0) DEC(0)
   wait 5.0                      Waits 5 s.
   speed     N(01)
   wait 2.0
   stop
end
```



instrukce stop	ukončení pohybové operace

- **Formát**

formát	popis operace
stop	ukončí pohybovou instrukci

- **Popis**

Instrukce "stop" ukončí pohybové instrukce "hp", "mov", "nchg", "smov", "speed", "tchg", "trq", nebo "sync". Je-li použita instrukce "stop" při vykonávání instrukcí "mov", "speed", nebo podobných, servopohon přejde do stavu zastaveno "servo-lock". Při provádění instrukce "sync" se operace okamžitě ukončí, což má za následek velmi rychlý doběh (pokud servopohon nestojí).

POZN.1: Pokud hodláte provést operace v polohové regulaci jako "mov", "hp", nebo jiné v momentové regulaci jako "trq" nebo další, po instrukci "speed", proveďte napřed instrukci "stop" a vyčkejte dokud se pohon nezastaví (indikace nulové rychlosti)..

instrukce sync	povel režimu synchronizace

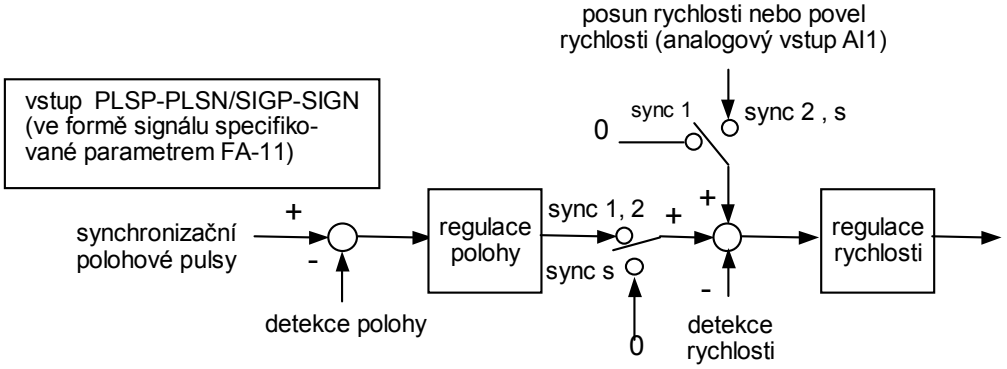
- **Formát**

Formát	popis operace
sync 1	Režim synchronizace (bez použití analogového posunu).
sync 2	Režim synchronizace (za použití analogového posunu).
sync s	Přejde do režimu regulace rychlosti ve shodě s povelem rychlost z analogového vstupu 1.

- **Popis**

Instrukce se používá, pokud synchronizujeme servopohon s jiným, nebo pokud řídíme servopohon posloupností pulsů nebo analogovým signálem, jako v případě standardního modelu servopohonu. Provedeme-li tuto instrukci a přivedeme synchronizační signál (vstupní posloupnost pulsů) na vstupy povelu polohy (PLSP-PLSN/SIGP-SIGN), servomotor běží synchronně s přivedeným signálem. Když je tato instrukce provedena servopohon pracuje v režimu polohové regulace a přikročí v dalším operačním cyklu k provedení dalšího řádku programu. Blokový diagram je uveden níže.

Aby jste se vrátili do normálního režimu řízení (ve kterém mohou být použity instrukce „mov“ a další) proveďte napřed instrukci „stop“. Přepínáte-li mezi jednotlivými typy instrukce "sync" není potřeba používat při přepnutí instrukci "stop". Při přechodu mezi instrukcemi „sync 1“/“sync 2“ je chyba polohy automaticky vynulována při provedení instrukce.



Formát (1): sync 1 (synchronizace bez analogového posunu)

Formát instrukce "sync 1" se použije, požadujeme-li prosté řízení polohy servopohonu přiváděnou posloupností polohových pulsů (jako v případě standardního modelu), nebo synchronizaci s jiným servopohonem. Tento formát instrukce představuje prostou synchronizaci s posloupnosti polohových pulsů. Je též možné korigovat synchronizovanou polohu pomocí přídavného polohového posunu (PBIAS). Lze zadat pozitivní nebo negativní polohovou korekci ve formě numerické hodnoty zapsané do proměnné PBIAS. PBIAS přidá nastavený počet pulsů (1 [dig] = 1 [pls]) v 140 [μs] intervalu.

Formát (2): sync 2 (synchronizace s analogovým posunem)

Formát instrukce "sync 2" použijeme tehdy, požadujeme-li rychlou synchronizaci pohonu s jiným. Na rozdíl od formátu "sync 1", přidává formát instrukce "sync 2" rychlostní posun v závislosti na hodnotě analogového signálu 1 (AI1) aby byla rychle korigována polohová chyba. Také tento tvar instrukce „sync“ vyžaduje přivedení analogové polohové posloupnosti pulsů. Lze též zadat pozitivní nebo negativní polohovou korekci ve formě numerické hodnoty zapsané do proměnné PBIAS. PBIAS přidá nastavený počet pulsů (1 [dig] = 1 [pls]) v 140 [μs] intervalu.

Formát (3): sync s (režim regulace rychlosti s analogovým vstupem)

Formát "sync s" představuje režim regulace rychlosti v závislosti na povelu rychlosti zadávaném ve formě analogového signálu na vstupu 1. V tomto případě nejsou použity žádné rozběhové a doběhové časy, proto výsledná rychlost se řídí přímo průběhem analogové signálu (stejně jako standardní provoz v rychlosti regulaci). Nezapomeňte, že analogový signál je vzorkován v intervalu 1.12 [ms], proto nelze dosáhnout vyšší rychlosti odezvy, i když zvýšíte parametr rychlost odezvy regulace rychlosti.

Pozn.: Nelze dosáhnout rychlé synchronizace pohonu, pokud je nastaveno nízké zesílení regulace polohy nebo rychlosti.

- **Příklad použití**

Příklad 1) Jednoduchá synchronizace s posunem rychlosti

<datové okno>

```
ACC(0)=50    : DEC(0)=50
N(00)=5000
P(00)=10000
```

<kódové okno>

```
entry
    sync 1
    ……
    stop
end
```

Příklad 2) Synchronizace bez specifikace polohy

<datové okno>

```
ACC(0)=50    : DEC(0)=50
N(00)=5000
P(00)=10000
```

<kódové okno>

```
entry
    sync 2
    ……
    stop
end
```

instrukce tchg — změna povelu momentu

- **Formát**

Formát	popis operace
tchg P(i) T(k) [N(l) N(m)]	Provádí regulaci momentu s tím, že v určené poloze dojde ke změně povelu momentu. Je nutné zadat omezení rychlosti [N(l) N(m)].

- **Popis**

Je-li po instrukci "tchg" použita některé z instrukcí "speed" nebo "trq", pak v průběhu této instrukce dojde ke změně povelu momentu. Při dosažení polohy P(i) dojde k přechodu do režimu momentové regulace s povelem momentu T(k) a rychlostním omezením N(l) a N(m). Instrukce "tchg" ovlivní pouze jednu následující instrukci ("speed" or "trq"). Pokud požadujete vícenásobnou změnu režimu regulace momentu, proveďte instrukci "tchg" tolikrát, kolikrát požadujete před instrukcemi "speed"/"trq". Provedení instrukce "tchg" je ukončeno v jednom cyklu, takže další řádek se provádí v dalším cyklu. Přerušení režimu regulace momentu a návrat do normálního stavu (kde lze provádět instrukce „mov“ a jiné) proveďte instrukcí stop.

P(i): Specifikace polohy, ve které dojde ke změně povelu momentu (přechodu na momentovou regulaci), za předpokladu že 1 dig = 1/32768 otáčky. Zadání polohy je relativní vzhledem k VP nebo druhé VP.

T(k): Číselná hodnota povelu momentu, která bude použita po přechodu na momentovou regulaci 1 dig = 1% momentu.

N(l): Omezení rychlosti ve směru „vpřed“ za předpokladu že 1 dig = 1 min^{-1}. Zadání se provádí s kladným znaménkem. Rychlost pohonu po přechodu na momentovou regulaci bude určována rovnováhou mezi momentem zátěže a povelem momentu, proto je nutné zadat omezení rychlosti, aby se pohon nedostal mimo kontrolu.

N(m): Omezení rychlosti ve směru „vzad“ za předpokladu že 1 dig = 1 min^{-1}. Zadání se provádí se záporným znaménkem. Rychlost pohonu po přechodu na momentovou regulaci bude určována rovnováhou mezi momentem zátěže a povelem momentu, proto je nutné zadat omezení rychlosti, aby se pohon nedostal mimo kontrolu.

Pozn.1: je-li některé z omezení rychlosti rovno 0, hlásí pohon chybu provádění E45.

Pozn.2: Omezení rychlost brání tomu, aby se pohon nedostal do oblasti nebezpečných rychlostí. Protože omezení rychlost nepracuje jako povel rychlosti, může se skutečná rychlost servopohonu lehce lišit v závislosti na zatížení od rychlost zadané omezením. Tuto diferenci lze ovlivnit zadáním P-zesílení regulace. Zvětšením P-složky zesílení regulace parametrem Fd-04 se zmenší rozdíl mezi rychlostí servopohonu a zadaným omezením, snížíme-li hodnotu parametru Fd-04, bude rozdíl větší. Nezapomeňte však, že příliš vysoká hodnota P složky zesílení může způsobovat kývání pohonu.

Pozn.3: Parametry omezení rychlost N(l) a N(m) lze při vícenásobném použití instrukce "tchg" vynechat. Pokut jsou však tyto parametry vynechány při prvním použití rychlosti "tchg" dojde k chybě provádění E45.

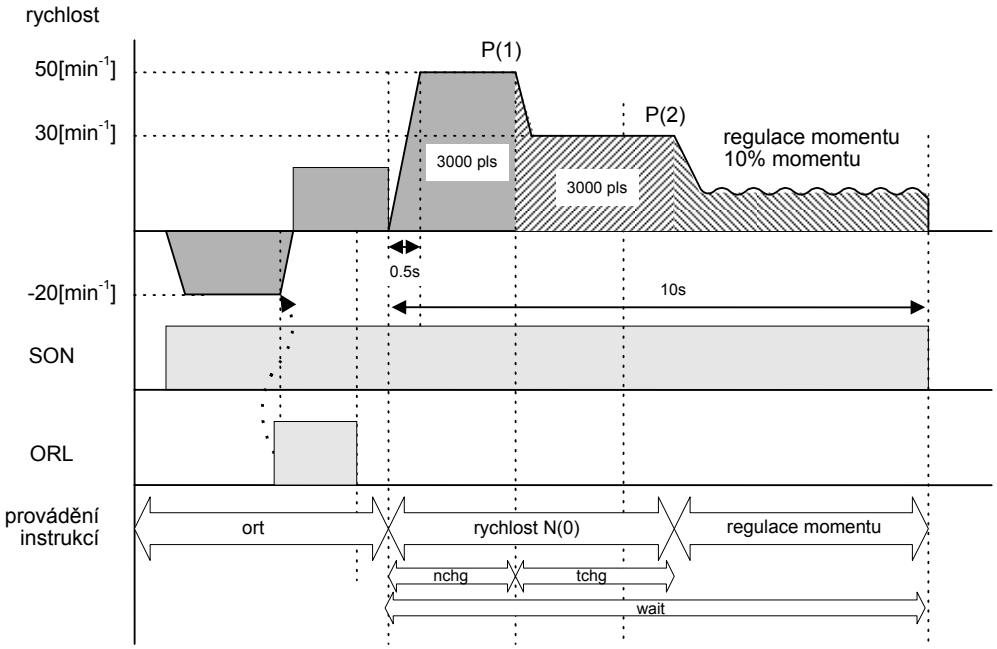
- **Příklad použití**

<datové okno>

P(00)=10000 : N(00)=50
P(01)=3000 : N(01)=30
P(02)=P(01)+3000 : N(02)=20
T(00)=10
ACC(0)=50 : DEC(0)=50

<kódové okno>

```
entry
   ort 3        N(02) N(02) ACC(0) DEC(0)
   nchg         P(01) N(01)
   tchg         P(02) T(00) N(02) N(02)
   speed        N(00) ACC(0) DEC(0)
   wait 10
   stop
end
```



instrukce trq	regulace momentu

- **Formát**

Formát	popis operace
trq T(j) [N(k) N(l)]	regulace momentu v závislosti na T(j).

- **Popis**

Instrukce "trq" nastavuje momentovou regulaci s povelem momentu T(k) a omezením rychlosti N(l) a N(m). Požadujete-li změnu povelu momentu v průběhu momentové regulace, proveďte instrukci „trq“ znovu. Požadujete-li změnu velikosti povelu momentu při dosažení určité polohy, použijte instrukci "tchg". Bližší vysvětlení na straně instrukce "tchg". Provedení instrukce "trq" je ukončenu v jednom cyklu, takže další řádek programu se provádí v následujícím cyklu, ale momentová regulace trvá. Chcete-li ukončit provoz v momentové regulaci a přejít do normálního stavu (kdy je možné provádět instrukce „mov“ a jiné pohybové instrukce) proveďte instrukci stop.

T(k): Specifikace číselné hodnoty povelu momentu, kdy 1 dig = 1% torque.

N(l): Omezení rychlosti ve směru „vpřed“ za předpokladu že 1 dig = 1 min^{-1}. Zadání se provádí s kladným znaménkem. Rychlost pohonu po přechodu na momentovou regulaci bude určována rovnováhou mezi momentem zátěže a povelem momentu, proto je nutné zadat omezení rychlosti, aby se pohon nedostal mimo kontrolu.

N(m): Omezení rychlosti ve směru „vzad“ za předpokladu že 1 dig = 1 min^{-1}. Zadání se provádí se záporným znaménkem. Rychlost pohonu po přechodu na momentovou regulaci bude určována rovnováhou mezi momentem zátěže a povelem momentu, proto je nutné zadat omezení rychlosti, aby se pohon nedostal mimo kontrolu.

Pozn.1: je-li některé z omezení rychlosti rovno 0, hlásí pohon chybu provádění E45.

Pozn.2: Omezení rychlost brání tomu, aby se pohon nedostal do oblasti nebezpečných rychlostí. Protože omezení rychlost nepracuje jako povel rychlosti, může se skutečná rychlost servopohonu lehce lišit v závislosti na zatížení od rychlost zadané omezením. Tuto diferenci lze ovlivnit zadáním P-zesílení regulace. Zvětšením P-složky zesílení regulace parametrem Fd-04 se zmenší rozdíl mezi rychlostí servopohonu a zadaným omezením, snížíme-li hodnotu parametru Fd-04, bude rozdíl větší. Nezapomeňte však, že příliš vysoká hodnota P složky zesílení může způsobovat kývání pohonu.

Pozn.3: Parametry omezení rychlost N(l) a N(m) lze při vícenásobném použití instrukce "trq" vynechat. Pokut jsou však tyto parametry vynechány při prvním použití rychlosti "trq" dojde k chybě provádění E45.

- **Příklad použití**

<datové okno>
P(00)=1000
ACC(0)=50
DEC(0)=50
T(00)=10 10% povel momentu

<kódové okno>
entry
mov P(00) N(00) ACC(0) DEC(0)
trq T(00) N(00) N(00)
wait 1
stop
end

6.6 Vstupní a výstupní instrukce

proměnné X()/Xw	ovládání kontaktních vstupů

- **Formát**

Formát		popis operace
(1)	<proměnná> = X(i) (i=0 to 11)	<proměnná> ← X(i)
(2)	<proměnná> = Xw	<proměnná> ← Xw

- **Popis**

Tato proměnná umožňuje sledování stavu kontaktních vstupů X(00) až X(11). Je určena pouze pro čtení a nelze do ní zapisovat. Použitelné formáty jsou uvedeny níže:

Formát (1): Umožní Vám získat <proměnnou> obsahující informaci o každém vstupu bit po bitu (0=OFF, 1=ON)
Příklad) Když X(00) je OFF: X(00)=0
Když X(01) je OFF: X(01)=1

Formát (2): Umožní Vám získat <proměnnou> obsahující informaci o každém vstupu ve formátu slova. Informační bit X(0) je nejnižším platným bitem.
Příklad) Když X(00) až X(03) = ON a X(04) až X(11) = OFF, pak Xw=7
Když X(00) až X(02) = OFF a X(03) až X(11) = ON, pak Xw=4088

POZN. 1: Polaritu vstupů nelze změnit nastavením funkce FC-01 (nastavení polarity vstupů). Polaritu lze změnit v uživatelském programu.
POZN. 2: Tato proměnná provádí vnitřně 2x čtení vstupních hodnot (2 prováděcí cykly) proto je další provádění zpožděno minimálně o dva cykly. Aby nemohlo dojít k případným problémům, vypracujte uživatelský program již s ohledem na toto zpoždění. k jeho odstranění
POZN. 3: Načtená hodnota může být chybná např. v důsledku zakmitání kontaktu. Abychom tento problém odstranili, vytvořte uživatelský program, který prověří načtené hodnoty před započetím provozu.

- **Příklad použití**

Příklad 1: Rozhodování dle stavu svorky X(01)

<kódové okno>

```
entry
U(00)=X(01)          X(01) načtení stavu svorky
ifs U(00)=1          podmíněný skok ‘if
then
Yw=4095
else
Yw=0
end if
```

<Popis>

stav vstupní svorky X(01) je určující pro stav výstupních svorek Y(), viz níže

stav svorky X(01)	všechny svorky Y()
OFF	OFF
ON	ON

Příklad 2: využití vstupních svorek X(07) až X(03) pomocí Xw.

<kódové okno>

```
        entry
MAIN:   U(00)=Xw              načtení stavu svorek X()
        U(00)=U(00) /16       X(11) až X(03) → b7 až b0
        U(00)=U(00) and 15    překryje ‘X(11) to X(8)
        Yw=U(00)              přepis dat na výstupní svorky Y()
        goto MAIN
        end
```

<Popis>

Stav svorek X(07) až X(03) je přenesen na výstupní svorky Y(03) až Y(00).

proměnné Y()/Yw | **Ovládání kontaktních výstupů**

- **Formát**

	Formát	popis operace
(1)	Y(i) = <proměnná> (i=0 to 7)	Y(i) ← <proměnná>
(2)	Yw = <proměnná>	Yw ← <proměnná>

- **Popis**

Tato proměnná Vám umožní řídit výstupní svorky Y(00) až Y(07) pomocí uživatelského programu. Použitelné formáty jsou uvedeny níže:

Formát (1): Nastaví hodnotu ON/OFF na kontaktním výstupu jednotlivě bit po bitu (0 = OFF, 1 = ON).

? Příklad) Když Y(00) má být OFF: Y(00) = 0

? Když Y(01) má být ON: Y(01) = 1

Formát (2): Nastaví hodnotu ON/OFF ve formě slova na kontaktní výstupy. Bit Y(0) je nejnižším platným bitem.

Příklad) Když Y(00) = ON a Y(01) až Y(07) = OFF, pak Yw = 1.
Když Y(00) až Y(06) = OFF a Y(07) = ON, pak Yw = U(00).
V U(00) musí být nastavena hodnota 128.

POZN.1: Polaritu výstupů nelze změnit nastavením funkce FC-02 (nastavení polarity výstupů). Polaritu lze změnit v uživatelském programu.

POZN.2: Nelze překrýt jednotlivé bity. Požadujete-li překrytí určitých bitů, je potřeba vytvořit uživatelský program, tak jak je ukázáno níže.

- **Příklad použití**

Příklad 1: příklad použití Yw a Y()

<kódové okno>

```
        entry
        Yw=0                                (1) všechny svorky Y() jsou OFF
MAIN:   Y(00)=1                             (2) svorka Y(00) je ON
        mov P(00) N(00) ACC(0) DEC(0)
        Y(00)=0                             (3) svorka Y(00) je OFF
        mov P(01) N(00) ACC(0) DEC(0)
        Yw=2                                (4) svorka Y(01) ON
        mov P(02) N(00) ACC(0) DEC(0)
        Yw=3                                (5) svorky Y(01), Y(00) jsou ON
        hp N(00) N(00) ACC(0) DEC(0)
        goto MAIN
        end
```

<datové okno>

P(00)=32768*10 [pulsů] ,P(01)=32768*20[pulsů]
P(02)=32768*30 [pulsů] ,P(03)=32768*30[pulsů]
N(00)=1000[min^{-1}] ,ACC(0)=0.1[s] ,DEC(0)=0.1[s]

<Popis>

Níže je uveden popis programu. Uvedená čísla (1) až (5) odpovídají shodným číslům v programu výše.

(1) nastaví všechny svorky do stavu OFF.
(2) nastaví svorku Y(00) do stavu ON a spustí polohovou operaci
(3) nastaví svorku Y(00) do stavu OFF a spustí další polohovou operaci
(4) nastaví svorku Y(01) do stavu ON, ostatní svorky do stavu OFF a spustí další polohovou operaci.
(5) nastaví svorku Y(01), Y(00) do stavu ON, ostatní svorky do stavu OFF a spustí operaci nájezdu do výchozí polohy

Příklad 2: Překrytí pouze svorky Y(06)

<kódové okno>

```
entry
Yw=0                    všechny výstupy jsou 0
U(04)= not U(05)        not &H40=4031
for U(02)=0 to U(07)
U(03)=Yw                Načtení stavu svorek Y()
U(03)=U(03) and U(05)   (1) načtení pozice b6 ze svorek Y() (všechny ostatní kromě b6
                        jsou 0)
U(02)=U(02) and U(04)   (2) vymazání pozice b6 v originálních datech
U(02)=U(02) or U(03)    (3) sloučení originálních dat a dat na svorkách Y()
Yw=U(02)
wait 1.0
U(06)=Y(06)             obrácená hodnota Y(06)
U(06)= not U(06)
Y(06)=U(06)
next
end
```

<datové okno >

U(07)=256, U(05)=64

<popis>

tato část programu provádí ve smyčce kódování výstupních svorek Y(). Svorka Y(06) obrací svůj stav při každém průchodu. Procedura překrytí se provádí v úkonech (1) až (3) výše uvedeného programu.

(1) nastavení všech svorek Y() kromě b6 na 0,
(2) operace AND s 2^6=64 aby byla vynulována pouze pozice b6 v originálních datech před překrytím
(3) operace OR sloučení dat z operací (1) a (2)

proměnné XA() — sledování analogové vstupní proměnné

- **Formát**

Formát	popis operace
<proměnná> = XA(i) (i=0 to 1)	<proměnná> ← XA(i)

- **Popis**

Tato instrukce sleduje stav analogové vstupní svorky AI1 a AI2. tato operace je pouze operací čtení, zápis proto není možný. Vztah mezi indexem i a příslušnou analogovou svorkou je uveden níže. Obě svorky jsou snímány s rozlišením 10V=100%=10000dig. Tento vztah lze změnit, pokud nastavíte zesílení FC-05 nebo posun FC-08. Blíže viz uživatelská příručka servopohonu.

XA(0): sledování hodnoty napětí na svorce AI1
rozlišení 10V= 100%= 10000dig
-10V=-100%=-10000dig
XA(1): sledování hodnoty napětí na svorce AI2
obě svorky XA(0) a XA(1) mají stejné rozlišení

- **příklad použití**

Zadání rychlosti pro polohovou operaci ze vstupu XA(00). Zadání rychlosti pro rychlostní operaci ze vstupu XA(01).

<kódové okno>

```
        entry
MAIN:   U(00)=XA(0)                     (1) načtení hodnoty z AI1
        U(00)=U(00)*N(00)               (2) nastavení rychlosti
        N(01)=U(00)/U(02)               (3) nastavení rozlišení 100%=10000dig
        if N(01)<>0 then RUN            (4) dolní omezení v případě provádění
                                            chyba E45 pokud N(01)=0
        N(01)=1                         (5)
RUN:    hp N(01) ACC(0) DEC(0)
        mov P(00) N(01) ACC(0) DEC(0)
        U(00)=XA(1)                     (6) načtení hodnoty z AI2
        U(00)=U(00)*N(00)               (7) nastavení rychlosti
        N(01)=U(00)/U(02)               (8) nastavení rozlišení 100%=10000dig
        speed N(01) ACC(0) DEC(0)
        wait 5.0
        stop
        goto MAIN
        end
```

<datové okno>

```
N(00)=3000                              100% rychlostní instrukce
U(02)=10000                             100%=10000dig
```

<Popis>

níže je popsáno provedení výše uvedeného programu. Čísla uvedená níže (1) až (5) odpovídají číslům uvedeným v programu.

(1) Načtení analogové hodnoty z AI1.

(2)(3) úprava analogového signálu na povel rychlosti
3000min^{-1} odpovídá 100% protože analogový signál je 10V=100%=10000dig.

(4)(5) v polohové regulaci dojde k chybě provádění E45, pokud nastavíme povel rychlosti 0
Proto je nastaven dolní limit 1min^{-1}.

(6) až (8) načtení dat z AI2 a provedení jako v (2) a (3).
v rychlostní operaci nedojde ani při zadání rychlosti 0 k problémům, proto není nutné nastavovat dolního omezení.

instrukce chg **Změna přiřazení významu svorek**

- **Formát**

Formát			popis operace
chg	FOT ROT	= X(i) non	X(i): změna významu svorky non: zrušení přiřazení
chg	SRD ALM INP SA SZD BRK TLM OL1	= Y(i) non	Y(i): změna významu svorky non: zrušení přiřazení
chg	ORL ORG	= X(i) default	X(i): změna významu svorky default: nastavení počátečního významu svorky (tovární nastavení

- **Popis**

Použijte instrukci chg pokud jsou vstupní svorky FOT, ROT, ORG, ORL a všechny výstupní svorky jako SDR a další použity v programu. Změna přiřazení je platná od bodu provedení instrukce "chg". Změna významu svorek zůstává v platnosti i při zastavení serva (při stavu servo OFF). Zneplatnění změny významu svorek lze dosáhnout jedině opětným použitím instrukce "chg" nebo vypnutím sítě. Bližší informace o funkci jednotlivých svorek naleznete v uživatelské příručce servo zesilovače.

non: udává, že V/V svorka není použitelná.
default: návrat významu svorek ORL a ORG do původního významu (tovární nastavení).
X(i): specifikace vstupní svorky u které má být změněn její význam
Y(i): specifikace výstupní svorky u které má být změněn její význam

Pozn.1: Polaritu vstupní svorky u které se mění význam nelze změnit X() významovým bitem FC-01.
Pozn.2: Polaritu výstupní svorky u které se mění význam lze změnit Y() významovým bitem FC-02.
Pozn.3: Následující tabulka uvádí, jak bude prováděna operace nájezdu na VP při přiřazení významu ORG a ORL (použití instrukce "chg") některé svorce.
Pozn.4: Svorky (významy) SDR a ALM jsou platné až po zapnutí serva (servo ON) a při provádění programu. Svorky SRD a ALM nejsou ovládány dokud není zapnuto napájení a není navozen stav servo ON.
V některých případech není možné zařízení uvést do chodu, pokud je zablokováno signálem z nadřazeného systému. Proto pokud hodláte pracovat s významy SRD a ALM, je nutné napřed odstranit nadřazené blokování, zapnout napájení serva a uvést jej do stavu servo ON.

název svorky	čas provádění, je-li použita instrukce "chg"	význam nepřiřazen
svorka ORG	význam je přiřazen určité svorce (operace nájezdu na VP se začne provádět až je svorka ve stavu ON)	význam nepřiřazen (nájezd na VP se začne provádět okamžitě)
svorka ORL	Význam je přiřazen určité svorce	svorka X(08) (počáteční přiřazení)

- **Příklad použití**

Příklad 1: význam ORG není přiřazen

<kódové okno>

```
entry                                  význam ORL je přiřazen svorce X(8) (dříve)
   chg ORL=X(00)                       změna přiřazení významu ORL na svorku X(0)
   ort  4  N(01) N(02) ACC(0) DEC(0)   nájezd na VP vysokou rychlostí je proveden bez
                                       čekání na zapnutí svorky ORG(protože není
                                       přiřazena)
end
```

<datové okno>

N(01)=1000[min^{-1}] , N(01)=10[min^{-1}]
ACC(0)=0.1[s] , DEC(0)=0.1[s]

<Popis>

Instrukce "chg" provede změnu přiřazení významu ORL na svorku X(00). Instrukce "ort" započne s nájezdem na VP okamžitě, protože význam ORG není přiřazen žádné svorce.

Příklad 2: význam ORG je přiřazen určité svorce

<kódové okno>

```
entry                                  význam ORL přiřazen svorce X(8) (dříve)
   chg ORL=X(00)                       přiřazení významu ORL svorce X(00)
   chg ORG=X(01)                       přiřazení významu ORG svorce X(01)
   ort  4  N(01) N(02) ACC(0) DEC(0)   provedení nájezdu na VP vysokou rychlostí s
                                       čekáním na přechod svorky ORG do stavu ON.
end
```

<datové okno>

N(01)=1000[min^{-1}] , N(01)=10[min^{-1}]
ACC(0)=0.1[s] , DEC(0)=0.1[s]

<Popis>

První instrukce "chg" provede změnu přiřazení významu ORL na svorku X(00). Druhá instrukce "chg" provede přiřazení významu ORG svorce X(01). Nájezd na VP započne až svorka ORG přejde do stavu ON (X(01)=ON).

6.7 Řízení komunikace

instrukce open com	deklarace použití komunikačnho rozhraní (RS232)

- **Formát**

Formát	popis operace
open com	deklaruje použití komunikačního rozhraní

- **Popis**

Tato instrukce říká, že po jejím provedení bude následovat použití komunikačního rozhraní. V tomto případě se stávají dostupné instrukce a proměnné vysílání a příjmu.

Pozn.1: Užití instrukce ukončení komunikace není zavedeno. Komunikační port se automaticky otevře při přerušení provádění (Servo OFF).

Pozn.2: Ke spojení mezi servo pohonem a PC (RS232) slouží prostředky popsané v "dodatku – kabel pro spojení s počítačem" obsaženém v uživatelské příručce servopohonu.

- **Příklad použití**

<kódové okno>

```
entry          začátek hlavního programu
open com
end            ukončení hlavního programu
```

<Popis>

Deklaruje dosažitelnost komunikačního rozhraní z uživatelského programu.

instrukce print # | **vyslání dat na komunikační sběrnici (RS-232C)**

- **Formát**

Formát	popis operace
print #0 <proměnná>	specifikovaná proměnná je vložena do výstupního komunikačního příkazu 80H servopohonu.
print #1 <proměnná>	specifikovaná proměnná je vložena do výstupního komunikačního příkazu 81H servopohonu.
print #2 <proměnná>	specifikovaná proměnná je vložena do výstupního komunikačního příkazu 82H servopohonu.

- **Popis**

Komunikační příkaz 80 až 82H vyšle na seriovou směrnici RS232C specifikovanou proměnnou. Přenos proměnné se děje ve formátu 4 bytů se znaménkem (signed long format). Proměnná jako např. datový byte je automaticky transformována do dlouhého formátu (4byte). Program se nepohne z příkazu "print" dále na další řádek , dokud není proveden kompletní přenos. Čas přenosu závisí na zvolené komunikační rychlosti (baud), stop bitu a start bitu. V následujícím je uveden pouze čas nutný pro přenos. Dále se přidá k času přenosu prodlení při zpracování. Komunikační formát je popsán na další straně.

Příklad: rychlost přenosu 19200bps, 8bitů, stop bit 2, bez parity
$(1+8+2+0) \times 16/19200=9.17$ms

#0 až #2: specifikace komunikačního příkazu. #0 odpovídá 80H a #2 odpovídá 82H.

<proměnná>: určení názvu proměnné, jejíž numerická hodnota má být odeslána po sběrnici RS-232C.

- **Příklad použití**

```
<kódové okno>
    entry
    open com
    U(15)=65536
    print #0 P(00)              příkaz vyslání údaje o poloze
    U(00)=N(00)*U(15)
    U(00)=U(00)+ACC(0)
    print #1 U(00)              příkaz vyslání údaje o rychlosti (vyšší řády) a o času
                                rozběhu a doběhu (nižší řády)
    U(01)=Xw
    print #2 U(01)              Příkaz vyslání informace o stavu vstupů
    end
```

<datové okno>

P(00)=32768 [pls] , N(00)=3000[min^{-1}] , ACC(0)=0.10[s] (0.01s=1dig)

<Popis>

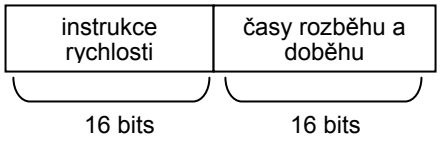
Uvedená část programu provede vyslání údaje o poloze, o aktuální rychlosti, o časech rozběhu a doběhu a o stavu vstupních svorek.

komunikační příkaz 80H ⋯údaj o poloze

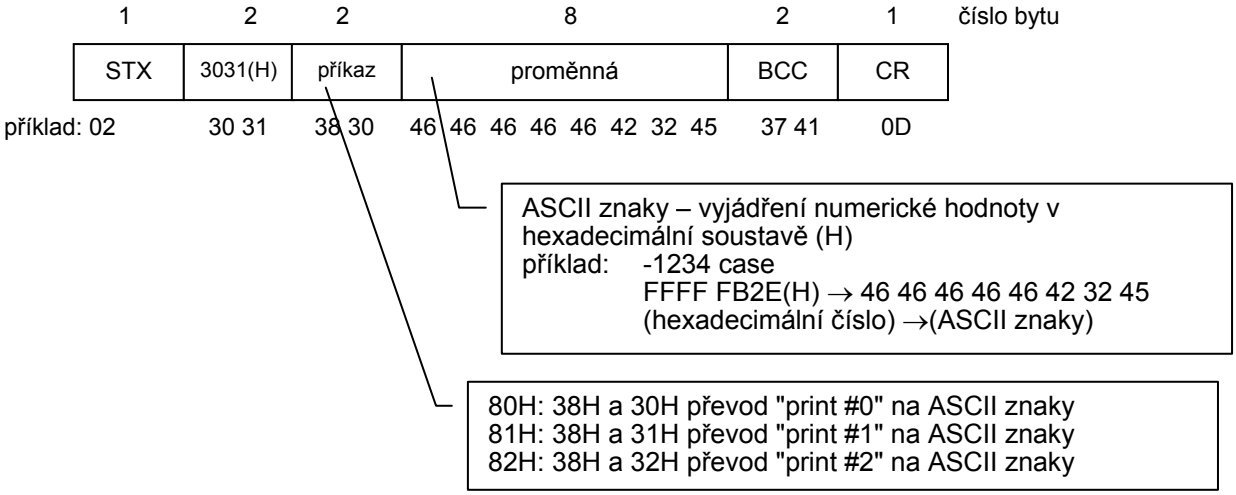
komunikační příkaz 81H ⋯údaj o rychlosti - vyšších 16 bitů

čas rozběhu a doběhu – nižších 16 bitů

komunikační příkaz 82H ⋯informace o stavu vstupních svorek



Komunikační formát instrukce "print"



BCC: "Exclusive OR" provedené s daty v bytu vpravo (od 30 (H)) a zapsáno do BCC.
příklad: BCC=30 ⊕ 31 ⊕ 38 ⊕ 30 ⊕ 46 ⊕ 46 ⊕ 46 ⊕ 46 ⊕ 46 ⊕ 42 ⊕ 32 ⊕ 45= 7A(H)
převod výsledku 7A na hodnotu 37(H) 41(H) do BCC

Následující tabulka obsahuje možné znakové kódy.

vysoký řád / nízký řád	0	1	2	3	4
0				0	
1				1	A
2	STX			2	B
3				3	C
4				4	D
5				5	E
6				6	F
7				7	
8				8	
9				9	
A					
B					
C					
D	CR				
E					
F					

Pozn.: možné kódové znaky jsou stejné jako pro instrukci print # tak i pro instrukci input #

instrukce input # | **načítá data z komunikační sběrnice (RS-232C)**

- **Formát**

Formát	popis operace
input #0 <proměnná>	načte proměnnou z komunikačního příkazu 90H a přiřadí ji specifikované proměnné
input #1 <proměnná>	načte proměnnou z komunikačního příkazu 91H a přiřadí ji specifikované proměnné
input #2 <proměnná>	načte proměnnou z komunikačního příkazu 92H a přiřadí ji specifikované proměnné

- **Popis**

Komunikační příkazy 90 až 92H načítají seriová data (proměnné) ze sběrnice RS232C do specifikované proměnné. Načtení proměnné se děje ve formátu 4 byte se znaménkem (signed long format). Příkaz "input" zabrání provádění dalších instrukcí, dokud nejsou načtena nová data. Obdržení nových dat lze sledovat pomocí funkce LOC. Jakmile jsou obdržena nová data, je jimi nahrazen obsah určené proměnné a program pokračuje další instrukcí. Následují-li přenášena data v zápětí po sobě, jsou vždy starší obdržená data nahrazena novými.

#0 až #2: Specifikace komunikačního příkazu. #0 odpovídá příkazu 90h a #2 příkazu 92H.

<proměnná>: specifikace názvu proměnné do které mají být uložena obdržená data.

- **Příklad použití**

<kódové okno>

```
        entry
        open com
        U(15)=65536
MAIN    input #1 P(00)
        input #2 U(00)
        N(00)=U(00)/U(15)
        ACC(0)=U(00) mod U(15)
        DEC(0)=ACC(0)
        mov P(00) N(00) ACC(0) DEC(0)
        goto MAIN
        end
```

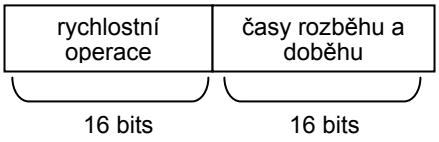
<Popis>

Polohovací operace začne, jakmile jsou následující data ze sběrnice RS232C načtena servopohonem.

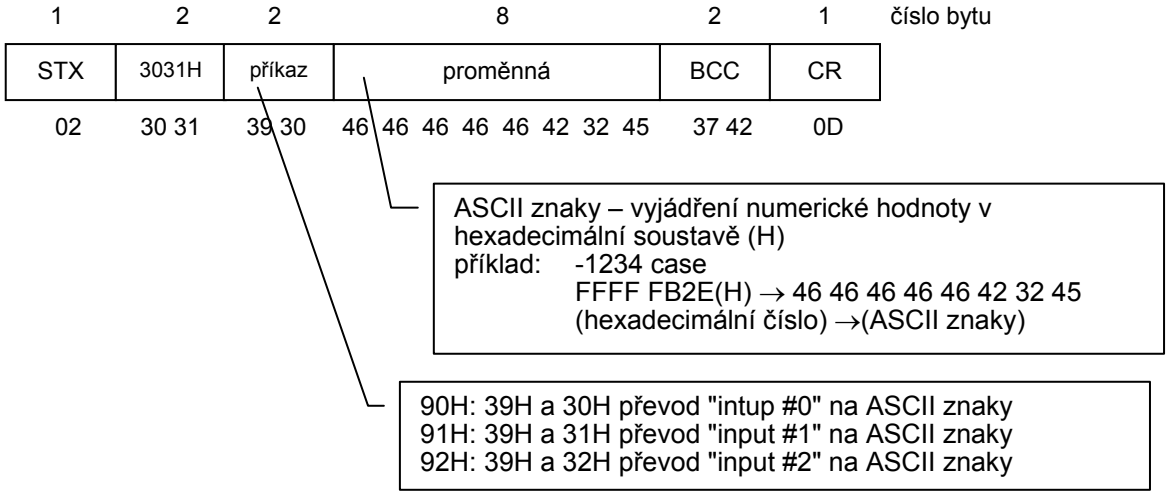
komunikační příkaz 91H data údaj o poloze

komunikační příkaz 92H data vyšších 16bitů – údaj o rychlosti

nižší bity – časy rozběhu a doběhu



Komunikační formát instrukce "input"



BCC: "Exclusive OR" provedené s daty v bytu vpravo (od 30 (H)) a zapsáno do BCC.
příklad: BCC=30 ⊕ 31 ⊕ 39 ⊕ 30 ⊕ 46 ⊕ 46 ⊕ 46 ⊕ 46 ⊕ 46 ⊕ 42 ⊕ 32 ⊕ 45=7BH
převod výsledku 7B na hodnotu 37(H) 42(H) do BCC

Použitá konverze mezi Hexadec. a ASCII znaky je v tabulce připojené k vysvětlení instrukce "print #" na straně 6-66.

funkce LOC | **Indikace příjmu dat po komunikační sběrnici RS-232C**

- **Formát**

Formát	popis operace
LOC(0)	Funkce nabude hodnoty 1, když jsou komunikačním příkazem 90H obdržena nová data, a hodnotu 0 pokud nejsou data obdržena nebo jsou již převzata příkazem "input #0" (a vložena do specifikované proměnné)
LOC(1)	Funkce nabude hodnoty 1, když jsou komunikačním příkazem 91H obdržena nová data, a hodnotu 0 pokud nejsou data obdržena nebo jsou již převzata příkazem "input #1" (a vložena do specifikované proměnné)
LOC(2)	Funkce nabude hodnoty 1, když jsou komunikačním příkazem 92H obdržena nová data, a hodnotu 0 pokud nejsou data obdržena nebo jsou již převzata příkazem "input #2" (a vložena do specifikované proměnné).

- **Popis**

Funkce LOC indikuje přijetí nových dat komunikačními příkazy 90 až 92H. Funkce nabude hodnoty 1 pokud byla nová data přijata ale ještě nebyla načtena příkazem "input #". Funkce nabývá hodnoty 0 pokud nová data nejsou přijata, nebo byla již po přijetí načtena funkcí "input #". Funkce LOC zajišťuje indikaci každého z komunikačních příkazů 90H až 92H jednotlivě.

- **Příklad použití**

<kódové okno>

```
            entry
            open com
SWAIT       if LOC(0)=1 then MOVGO          čekaní na přijetí dat
            Y(0)=0
            wait 0.1
            Y(0)=1
            wait 0.1
            goto SWAIT
MOVGO       input #1 P(00)                  načtení dat a započetí provozu
            mov P(00) N(00) ACC(0) DEC(0)
            goto SWAIT
            end
```

<Popis>

Tato funkce sleduje komunikační příkaz 90H a mění hodnotu svorky Y(00) tak dlouho, dokud nepřijdou data.

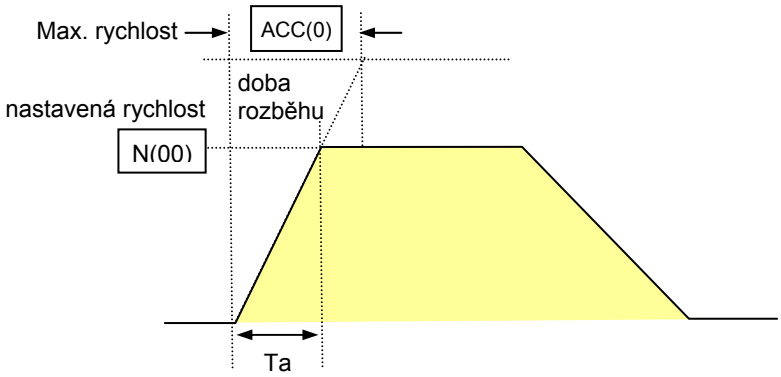
Jakmile je přijat komunikační příkaz 90H , provede se polohová operace dle načtené hodnoty polohy.

6.8 Ostatní rezervované proměnné

ACC(0) až ACC(1)	název proměnné	rozsah numerické hodnoty	počáteční hodnota	jednotky	velikost	atributy
	nastavení paměti proměnné rozběhového času	0.00 to 10.00	0.00 (nastavení datovým oknem)	0.01 s/dig	bez znaménka 1w	čtení zápis

- **Popis**

  Nastavení času rozběhu pro pohybové operace. Nastavení je s rozlišením 0.01s=1dig, nastavuje se čas rozběhu na max. rychlost. Datové okno nebo programový kód může definovat hodnotu na začátku programu. Pokud není doba rozběhu a doběhu přímo v pohybové instrukci specifikována, pak se použije nastavení bezprostředně předcházející.



ACC(k)=Nmax / N(j) × Ta

Pozn.1: Je-li nastavená hodnota mimo povolený rozsah, pak je použita maximální nastavitelná hodnota 10.0s.

Pozn.2: Pokud není čas rozběh a doběhu nastaven nejpozději v pohybové instrukci, pak nastane při provádění programu chyba provádění E45.

- **Příklad použití**

<kódové okno>

```
entry
ACC(0)=100                          ‘100   dig=1s
U(00)=ACC(0)/10                     ‘100/10dig=0.1s
ACC(1)=U(00)
ort    2 N(00) N(01) ACC(0) DEC(0)  ‘nájezd na VP
speed N(00) ACC(0) DEC(0)           rozběh 1s
wait 1.0
stop
mov P(00) N(00) ACC(1) DEC(0)       rozběh 0.1s
end
```

<datové okno>

P(00)=0[pls], N(00)=3000[min^{-1}], N(01)=10[min^{-1}], DEC(0)=0.1[s]

ACCEL	název proměnné	rozsah numerické hodnoty	počáteční hodnota	jednotka	velikost	atributy
	Nastavení času rozběhu dle Fb-04	0.00 to 10.00	0.00	0.01 s/dig	bez znaménka 1w	čtení zápis

- **Popis**

  Tato instrukce dovolí načíst a zapsat čas rozběhu specifikovaný v parametru doba rozběhu (Fb-04). Datový obsah je shodný jako u proměnných ACC(). Blíže viz popis proměnných ACC(). Je-li v Fb-04 nastavena hodnota1.00s, pak je do proměnné načtena hodnota 100.
  Pozn.1: Je-li zapsaná hodnota mimo numerický rozsah dojde k chybě provádění E45.

- **Příklad použití**

  <kódové okno>

```
entry
ACC(0)=ACCEL
mov         P(00) N(00) ACC(0) DEC(0)
end
```

  <datové okno>

  P(00)=0[pls], N(00)=3000[min^{-1}], DEC(0)=0.1[s]

CHR1 to CHR5	název proměnné	rozsah numerické hodnoty	počáteční hodnota	jednotka	velikost	atributy
	není k dispozici					

- **Popis**

  Tento příkaz není k dispozici.

- **Příklad použití**

  Žádný

DATR	název proměnné	rozsah numerické hodnoty	počáteční hodnota	jednotka	velikost	atributy
	není k dispozici					

- **• Popis**

  Tento příkaz není k dispozici.

- **Příklad použití**

  Žádný

DECEL	název proměnné	rozsah numerické hodnoty	počáteční hodnota	jednotka	velikost	atributy
	Nastavení času doběhu dle Fb-05	0.00 to 10.00	0.00	0.01 s/dig	bez znaménka 1w	čtení zápis

- **Popis**

  Tato instrukce dovolí načíst a zapsat čas doběhu specifikovaný v parametru doba doběhu (Fb-05). Datový obsah je shodný jako u proměnných DEC(). Blíže viz popis proměnných DEC(). Je-li v Fb-05 nastavena hodnota1.00s, pak je do proměnné načtena hodnota 100.
  Pozn.1: Je-li zapsaná hodnota mimo numerický rozsah dojde k chybě provádění E45.

- **Příklad použití**

  <kódové okno>

```
entry
DEC(0)=DECEL
mov        P(00) N(00) ACC(0) DEC(0)
end
```

  <datové okno>

  P(00)=0[pls], N(00)=3000[min^{-1}], ACC(0)=0.1[s]

DISP	název proměnné	rozsah numerické hodnoty	počáteční hodnota	jednotka	velikost	atributy
	není k dispozici					

- **Popis**

  Tento příkaz není k dispozici.

- **Příklad použití**

  žádný

ERR(0) to ERR(3)	název proměnné	rozsah numerické hodnoty	počáteční hodnota	jednotka	velikost	atributy
	kód chyby (d-11, d-12)	0 to 99	závisí na vzniklé chybě	–	–	čtení

- **Popis**

  tento příkaz dovoluje přečíst kód poslední chyby, jehož numerická hodnota se zobrazí na displeji (E**) operačního panelu. Přiřazení příčiny chyby a kódů chyb najdete v sekci 9 uživatelské příručky servopohonu. Následuje přiřazení významu ERR():
  ERR(0): poslední uložený kód chyby (d-11 na OP, znamená zobrazení chyby E**-1)
  ERR(1): předposlední kód chyby(první ze zobrazení d-12, znamená chybu E**-2)
  ERR(2): dřívější chyba (druhá ze zobrazení d-12, znamená chybu E**-3)
  ERR(3): dřívější chyba (třetí ze zobrazení d-12, znamená chybu E**-4)

- **Příklad použití**

  <kódové okno>

```
        entry
        Yw=0
        if ERR(0)=U(00) then MATCH    'E01=jeli nadproud, pak Y(0)=1
        if ERR(1)=U(00) then MATCH    'E01= jeli nadproud, pak Y(0)=1
        if ERR(2)=U(00) then MATCH    'E01= jeli nadproud, pak Y(0)=1
        if ERR(3)=U(00) then MATCH    'E01= jeli nadproud, pak Y(0)=1
        Y(0)=0                        'E01=není–li nadproud, pak Y(0)=0
        goto SKIP
MATCH:  Y(0)=1
SKIP:   Y(1)=1
        end
```

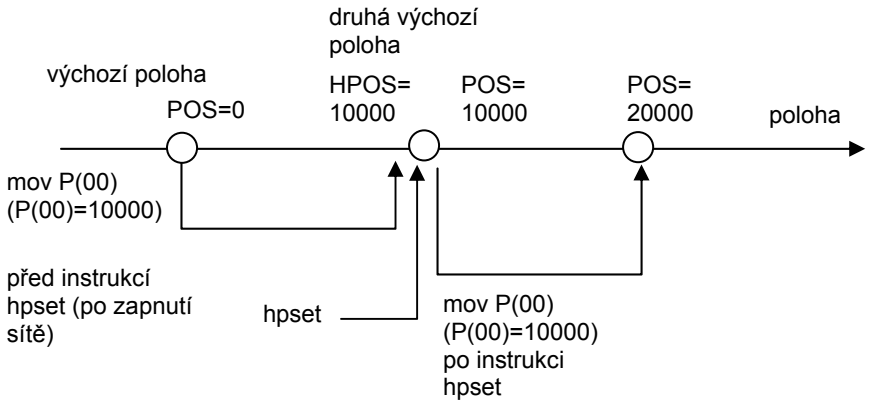
  <datové okno>

```
        U(00)=1                       '1=E01 chyba nadproud
```

HPOS	název proměnné	rozsah numerické hodnoty	počáteční hodnota	jednotka	velikost	atributy
	výchozí poloha	-2^{31} až $2^{31}-1$	0	2^{-15} ot./dig	2w se znaménkem	čtení

- **Popis**

  Tento příkaz je schopen načíst druhou výchozí polohu nastavenou příkazem "hpset". Jedna otáčka znamená 32768dig polohových jednotek. Následující diagram ozřejmuje vztah mezi druhou výchozí polohou a první výchozí polohou po zapnutí napájení.



- **Příklad použití**

  <kódové okno>

  ```
  entry
  mov     P(00) N(00) ACC(0) DEC(0)
  hpset
  mov     P(00) N(00) ACC(0) DEC(0)
  P(01)=HPOS+P(00)        vypočte vzdálenost od první VP do P(00) a vloží do ‘P(01).
  end
  ```

  <datové okno>

  P(00)=10000[pls], N(00)=3000[min^{-1}], ACC(0)=0.1[s], DEC(0)=0.1[s]

IFB	název proměnné	rozsah numerické hodnoty	počáteční hodnota	jednotka	velikost	atributy
	hodnota proudu (d-02)	0 to 32768	0	1/50 %/dig	1w bez znaménka	čtení

- **Popis**

  Tento příkaz umožní přečíst okamžitou hodnotu proudu motoru (zobrazení d-02) v provozu. Hodnota je načtena v 1%=50dig, bez znaménka (absolutní hodnota).

- **Příklad použití**

<kódové okno>

```
        entry
        speed  N(00) ACC(0) DEC(0)
MAIN:   ifs IFB > U(00)
        then
        Y(0)=1          je-li okamžitá hodnota proudu větší než U(00), je výstup Y(0) ON.
        else
        Y(0)=0          je-li okamžitá hodnota proudu menší než U(00), je výstup Y(0) OFF.
        end if
        goto   MAIN
        end
```

<datové okno>

U(00)=100*50[100%], N(00)=3000[min^{-1}], ACC(0)=0.1[s], DEC(0)=0.1[s]

INP	název proměnné	rozsah numerické hodnoty	počáteční hodnota	jednotka	velikost	atributy
	ukončení polohování	0, 1	–	–	0, 1	čtení

- **Popis**

Tento příkaz nám dovolí posoudit zda polování bylo ukončeno. Instrukce INP je pouze "čtení" a nabývá hodnot 0 a 1.

INP=0: probíhá-li polohování, nebo jiná operace než polohování jako "sped" nebo "trq".
1: polohování ukončeno (je-li odchylka polohy menší než dovolená hodnota nastavená v parametru Fb-23).

Výše uvedená instrukce nám umožní získat informaci o tom, zda bylo polohování ukončeno, při souběžném provádění instrukcí "mov".

- **Příklad použití**

<kódové okno>

```
          entry
MAIN:     mov   P(00) N(00) ACC(0) DEC(0) ;&    paralelní provádění
          Y(00)=0
          U(00)=0
BLINK:    inc U(00)                             Inkrementace
          if U(00)<100 then NOBLINK             bylo již provedeno 100 průchodů ?
          U(00)=not Y(00)                       když ano, změň stav výstupu Y(00)
          Y(00)=U(00)
          U(00)=0                               vynuluj proměnnou U(00) a pokračuj
NOBLINK:  if INP=0 then BLINK                   zjištění ukončení polohování
          stop
          Y(00)=0
          hp   N(01) ACC(0) DEC(0)
          goto MAIN
          end
```

<datové okno>

P(00)=327800[pls], N(00)=100[min^{-1}], N(01)=3000[min^{-1}]
ACC(0)=0.1[s], DEC(0)=0.1[s]

IRF	název proměnné	rozsah numerické hodnoty	počáteční hodnota	jednotka	velikost	atributy
	Okamžitá hodnota povelu momentu (d-03)	-32768 to 32767	0	1/50 %/dig	1w se znaménkem	čtení

- **Popis**

  Tento příkaz je schopen načíst okamžitou hodnotu povelu momentu (zobrazení d-03 na OP) v provozu. Hodnota je v 1%=50dig a kladné znaménko značí směr vpřed.

- **Příklad použití**

<kódové okno>

```
        entry
        speed  N(00) ACC(0) DEC(0)
MAIN:   ifs IRF > U(00)
        then
        Y(0)=1      Jeli okamžitý povel momentu větší než U(00), pak výstup Y(0) je ON.
        else
        Y(0)=0      Je-li okamžitý povel momentu menší než U(00), pak výstup Y(0) je OFF.
        end if
        goto   MAIN
        end
```

<datové okno>

U(00)=100*50[100%], N(00)=3000[min^{-1}], ACC(0)=0.1[s], DEC(0)=0.1[s]

	název proměnné	rozsah numerické hodnoty	počáteční hodnota	jednotka	velikost	atributy
J	setrvačný moment (Fd-00)	1 až 128 x moment setrvačnosti motoru	Fd-00 nastavená hodnota	0.01 kgm^2 $\times 10^{-n}$/dig*	–	čtení, zápis

- **Popis**

  Tento příkaz je schopen číst a zapisovat hodnotu setrvačného momentu soustavy (parametr Fd-00). Setrvačný moment soustavy představuje součet setrvačného momentu vlastního motoru a setrvačného momentu poháněného zařízení. Rozsah možného nastavení je násobkem setrvačného motoru (1 až 128x). Zapsáním numerické hodnoty setrvačného momentu touto instrukcí je okamžitě aktivní a je zapsáno do parametru Fd-00. Poslední zapsaná hodnota bude aktivní i při dalším zapnutí sítě.
  I drastická změna setrvačného momentu, ke které dochází např. při sepnutí spojky na zátěži, lze pomocí této funkce velmi dobře zregulovat. Příklady použití jsou uvedeny níže.

  Pozn.1: Příkaz změna setrvačného momentu může mít prodlevu 10 až 100ms (než se nová hodnota uplatní v regulaci).
  Pozn.2: Při přepisu hodnoty J je servo ve stavu zastaveno a zablokováno (servo lock stop state). Zesílení vnitřní regulace se při provedení instrukce zápisu změny setrvačného momentu skokově změní. Pokud použijete příkaz přepisu hodnoty J při běhu motoru, může dojít k poškození nebo zničení stroje vlivem "šoku při přepisu" a pod.
  Pozn.3: Zapíšete-li hodnotu mimo dovolený rozsah, může dojít k chybě provádění E45. Změny je možné provádět pouze v dovoleném rozsahu.

- **Příklad použití**

  Příklad aplikace, kdy dochází k připnutí zátěže spojkou a dochází k drastické změně setrvačného momentu

  <kódové okno>

```
        entry
MAIN:   J=100
        Y(00)=0                 rozepnutí spojky svorkou Y(00)
        wait 0.15               rozpínací čas spojky a prodleva za kterou se projeví
        změna
                                momentu J (150ms)
        mov P(00) N(00) ACC(0) DEC(0)
        J=300
        Y(00)=1                 sepnutí spojky svorkou Y(00)
        wait 0.15               spínací čas spojky a prodleva za kterou se projeví změna
                                momentu J (150ms)
        mov P(01) N(00) ACC(0) DEC(0)
        goto MAIN
        end
```

  <datové okno>

  P(00)=3276800[pls], P(00)=0[pls], N(00)=3000[min^{-1}], ACC(0)=0.1[s], DEC(0)=0.1[s]

KFC	název proměnné	rozsah numerické hodnoty	počáteční hodnota	jednotka	velikost	atributy
	Mezní frekvence regulace rychlosti (Fd-01)	1 to 5000	300	0.1 Hz/dig	–	čtení, zápis

- **Popis**

  Tento příkaz umožňuje načtení a zápis parametru mezní frekvence regulace rychlosti Fd-01, za předpokladu že 1dig=0.1Hz. Zapsání číselné hodnoty mezní frekvence tímto příkazem je okamžitě aktivní a uloží se též v parametru Fd-01. Poslední zapsaná hodnota bude uchována i pro opětovné zapnutí sítě.
  I drastická změna setrvačného momentu, ke které dochází např. při sepnutí spojky na zátěži, lze pomocí této funkce velmi dobře zregulovat. Příklady použití jsou uvedeny níže.

  Pozn.1: Příkaz změna setrvačného momentu může mít prodlevu 10 až 100ms (než se nová hodnota uplatní v regulaci).
  Pozn.2: Při přepisu hodnoty KFC je servo ve stavu zastaveno a zablokováno (servo lock stop state). Zesílení vnitřní regulace se při provedení instrukce zápisu změny KFC skokově změní. Pokud použijete příkaz přepisu hodnoty KFC při běhu motoru, může dojít k poškození nebo zničení stroje vlivem "šoku při přepisu" a pod.
  Pozn.3: Zapíšete-li hodnotu mimo dovolený rozsah, může dojít k chybě provádění E45. Změny je možné provádět pouze v dovoleném rozsahu.

- **Příklad užití**

  Příklad aplikace drastické změny zátěže na hřídeli a tuhosti mechanismu sepnutím spojky.

  <kódové okno>

```
        entry
MAIN:   J=100
        KFC=300              odezva regulace rychlosti30Hz
        Y(00)=0              sepnutí spojky výstupem Y(00)
        wait 0.15            spínací čas spojky a prodleva, za kterou se projeví změna
                              (150ms)
        mov P(00) N(00) ACC(0) DEC(0)
        J=300
        KFC=100              odezva regulace rychlosti 10Hz
        Y(00)=1              sepnutí spojky výstupem Y(00)
        wait 0.15            spínací čas spojky a prodleva, za kterou se projeví změna
                              (150ms)
        mov P(01) N(00) ACC(0) DEC(0)
        goto MAIN
        end
```

  <datové okno>

  P(00)=3276800[pls], P(00)=0[pls], N(00)=3000[min^{-1}], ACC(0)=0.1[s], DEC(0)=0.1[s]

KP	název proměnné	rozsah numerické hodnoty	počáteční hodnota	jednotka	velikost	atributy
	mezní frekvence regulace polohy (Fd-09)	1 to 9999	500	0.1 Hz/dig	–	čtení zápis

- **Popis**

Tento příkaz umožňuje načtení a zápis parametru mezní frekvence regulace polohy Fd-09, za předpokladu že 1dig=0.1Hz. Zapsání číselné hodnoty mezní frekvence tímto příkazem je okamžitě aktivní a uloží se též v parametru Fd-09. Poslední zapsaná hodnota bude uchována i pro opětovné zapnutí sítě.
I drastické změny tuhosti zátěže, ke kterým dochází např. při sepnutí spojky na zátěži, lze pomocí této funkce velmi dobře zregulovat. Příklady použití jsou uvedeny níže.

Pozn.1: Příkaz změna setrvačného momentu může mít prodlevu 10 až 100ms (než se nová hodnota uplatní v regulaci).
Pozn.2: Při přepisu hodnoty KP je servo ve stavu zastaveno a zablokováno (servo lock stop state). Zesílení vnitřní regulace se při provedení instrukce zápisu změny KP skokově změní. Pokud použijete příkaz přepisu hodnoty KP při běhu motoru, může dojít k poškození nebo zničení stroje vlivem "šoku při přepisu" a pod.
Pozn.3: Zapíšete-li hodnotu mimo dovolený rozsah, může dojít k chybě provádění E45. Změny je možné provádět pouze v dovoleném rozsahu.

- **Příklad použití**

Příklad aplikace drastické změny zátěže na hřídeli a tuhosti mechanismu sepnutím spojky.

<kódové okno>

```
        entry
MAIN:   J=100
        KFC=300                 zesílení polohové regulace 30Hz
        U(00)=KFC*10
        KP=U(00)/6              Hodnotu nastavte na 1/6 velikost odezvy regulace rychlosti
        Y(00)=0                 sepnutí spojky výstupem Y(00)
        wait 0.15               spínací čas spojky a prodleva, za kterou se projeví změna
                                (150ms)
        mov P(00) N(00) ACC(0) DEC(0)
        J=300
        KFC=100                 odezva regulace polohy 10Hz
        U(00)=KFC*10
        KP=U(00)/6              Hodnotu nastavte na 1/6 velikost odezvy regulace rychlosti
        Y(00)=1                 sepnutí spojky výstupem Y(00)
        wait 0.15               spínací čas spojky a prodleva, za kterou se projeví změna
                                (150ms)
        mov P(01) N(00) ACC(0) DEC(0)
        goto MAIN
        end
```

<datové okno>

P(00)=3276800[pls], P(00)=0[pls], N(00)=3000[min^{-1}], ACC(0)=0.1[s], DEC(0)=0.1[s]

KPF	název proměnné	rozsah numerické hodnoty	počáteční hodnota	jednotka	velikost	atributy
	zesílení smyčky polohové regulace (Fd-10)	0 to 100	0	0.01/dig	–	čtení zápis

- **Popis**

Tento příkaz umožňuje načtení a zápis parametru zesílení smyčky polohové regulace Fd-10, za předpokladu že 1dig=0.1Hz. Zapsání číselné hodnoty mezní frekvence tímto příkazem je okamžitě aktivní a uloží se též v parametru Fd-10. Poslední zapsaná hodnota bude uchována i pro opětovné zapnutí sítě.
I drastické změny tuhosti zátěže, ke kterým dochází např. při sepnutí spojky na zátěži, lze pomocí této funkce velmi dobře zregulovat. Příklady použití jsou uvedeny níže.

Pozn.1: Příkaz změna setrvačného momentu může mít prodlevu 10 až 100ms (než se nová hodnota uplatní v regulaci).
Pozn.2: Při přepisu hodnoty KPF je servo ve stavu zastaveno a zablokováno (servo lock stop state). Zesílení vnitřní regulace se při provedení instrukce zápisu změny KPF skokově změní. Pokud použijete příkaz přepisu hodnoty KPF při běhu motoru, může dojít k poškození nebo zničení stroje vlivem "šoku při přepisu" a pod.
Pozn.3: Zapíšete-li hodnotu mimo dovolený rozsah, může dojít k chybě provádění E45. Změny je možné provádět pouze v dovoleném rozsahu.

- **Příklad použití**

Příklad aplikace drastické změny zátěže na hřídeli a tuhosti mechanismu sepnutím spojky.

<kódové okno>

```
        entry
MAIN:   J=100
        KFC=300              nastavení mezní frekvence regulace rychlosti 30Hz
        U(00)=KFC*10
        KP=U(00)/6           nastavení mezní frekvence regulace polohy jako 1/6 mezní
                             frekvence regulace rychlosti
        KPF=90               nastavení zesílení Fd-10=0.9 (úprava charakteristiky
                             směrem k rychlejší odezvě)
        Y(00)=0              sepnutí spojky svorkou Y(00)
        wait 0.15            spínací čas spojky a prodleva, za kterou se projeví změna
                              (150ms)
        mov P(00) N(00) ACC(0) DEC(0)
        J=300
        KFC=100              nastavení mezní frekvence regulace rychlosti 10Hz
        U(00)=KFC*10
        KP=U(00)/6           nastavení mezní frekvence regulace polohy jako 1/6 mezní
                             frekvence regulace rychlosti
        KPF=0                nastaveno Fd-10=0.0
        Y(00)=1              sepnutí spojky svorkou Y(00)
        wait 0.15            spínací čas spojky a prodleva, za kterou se projeví změna
                              (150ms)
        mov P(01) N(00) ACC(0) DEC(0)
        goto MAIN
        end
```

<datové okno>

P(00)=3276800[pls], P(00)=0[pls], N(00)=3000[min^{-1}], ACC(0)=0.1[s], DEC(0)=0.1[s]

KSI	název proměnné	rozsah numerické hodnoty	počáteční hodnota	jednotka	velikost	atributy
	integrační složka regulace rychlosti (Fd-03)	1 to 30000	10000	0.01 %/dig	–	čtení zápis

- **Popis**

Tento příkaz umožňuje načtení a zápis parametru zesílení integrační složky regulace rychlosti Fd-03, za předpokladu že 1dig=0.01%. Zapsání číselné hodnoty mezní frekvence tímto příkazem je okamžitě aktivní a uloží se též v parametru Fd-03. Nastavujete-li hodnotu momentu setrvačnosti a odezvy rychlostí regulace, nastavte hodnotu asi 100%. Poslední zapsaná hodnota bude uchována i pro opětovné zapnutí sítě.
I drastické změny tuhosti zátěže, ke kterým dochází např. při sepnutí spojky na zátěži, lze pomocí této funkce velmi dobře zregulovat. Příklady použití jsou uvedeny níže.

Pozn.1: Příkaz změna setrvačného momentu může mít prodlevu 10 až 100ms (než se nová hodnota uplatní v regulaci).
Pozn.2: Při přepisu hodnoty KSI je servo ve stavu zastaveno a zablokováno (servo lock stop state). Zesílení vnitřní regulace se při provedení instrukce zápisu změny KSI skokově změní. Pokud použijete příkaz přepisu hodnoty KSI při běhu motoru, může dojít k poškození nebo zničení stroje vlivem "šoku při přepisu" a pod.
Pozn.3: Zapíšete-li hodnotu mimo dovolený rozsah, může dojít k chybě provádění E45. Změny je možné provádět pouze v dovoleném rozsahu.

- **Příklad použití**

Příklad aplikace drastické změny zátěže na hřídeli a tuhosti mechanismu sepnutím spojky.

<kódové okno>

```
        entry
MAIN:   J=100
        KSI=10000
        Y(00)=0                 sepnutí spojky výstupem Y(00)
        wait 0.15               spínací čas spojky a prodleva, za kterou se projeví změna
                                (150ms)
        mov P(00) N(00) ACC(0) DEC(0)
        KSI=8000                snížení velikosti
        Y(00)=1                 sepnutí spojky svorkou Y(00)
        wait 0.15               spínací čas spojky a prodleva, za kterou se projeví změna
                                (150ms)
        mov P(01) N(00) ACC(0) DEC(0)
        goto MAIN
        end
```

<datové okno>

P(00)=3276800[pls], P(00)=0[pls], N(00)=3000[min^{-1}], ACC(0)=0.1[s], DEC(0)=0.1[s]

KSP	název proměnné	rozsah numerické hodnoty	počáteční hodnota	jednotka	velikost	atributy
	proporcionální složka regulace (Fd-02)	1 to 30000	10000	0.01 %/dig	–	čtení zápis

- **Popis**

Tento příkaz umožňuje načtení a zápis parametru zesílení integrační složky regulace rychlosti Fd-02, za předpokladu že 1dig=0.01%. Zapsání číselné hodnoty mezní frekvence tímto příkazem je okamžitě aktivní a uloží se též v parametru Fd-02. Nastavujete-li hodnotu momentu setrvačnosti a odezvy rychlostí regulace, nastavte hodnotu asi 100%. Poslední zapsaná hodnota bude uchována i pro opětovné zapnutí sítě.
I drastické změny tuhosti zátěže, ke kterým dochází např. při sepnutí spojky na zátěži, lze pomocí této funkce velmi dobře zregulovat. Příklady použití jsou uvedeny níže.

Pozn.1: Příkaz změna setrvačného momentu může mít prodlevu 10 až 100ms (než se nová hodnota uplatní v regulaci).
Pozn.2: Při přepisu hodnoty KSP je servo ve stavu zastaveno a zablokováno (servo lock stop state). Zesílení vnitřní regulace se při provedení instrukce zápisu změny KSP skokově změní. Pokud použijete příkaz přepisu hodnoty KSP při běhu motoru, může dojít k poškození nebo zničení stroje vlivem "šoku při přepisu" a pod.
Pozn.3: Zapíšete-li hodnotu mimo dovolený rozsah, může dojít k chybě provádění E45. Změny je možné provádět pouze v dovoleném rozsahu.

- **Příklad použití**

Příklad aplikace drastické změny zátěže na hřídeli a tuhosti mechanismu sepnutím spojky.

<kódové okno>

```
        entry
MAIN:   J=100
        KSP=10000
        Y(00)=0                 sepnutí spojky výstupem Y(00)
        wait 0.15               spínací čas spojky a prodleva, za kterou se projeví změna
                                (150ms)
        mov P(00) N(00) ACC(0) DEC(0)
        KSP=8000                odezva je opožděna
        Y(00)=1                 sepnutí spojky svorkou Y(00)
        wait 0.15               spínací čas spojky a prodleva, za kterou se projeví změna
                                (150ms)
        mov P(01) N(00) ACC(0) DEC(0)
        goto MAIN
        end
```

<datové okno>

P(00)=3276800[pls],P(00)=0[pls], N(00)=3000[min^{-1}], ACC(0)=0.1[s], DEC(0)=0.1[s]

MODE	název proměnné	rozsah numerické hodnoty	počáteční hodnota	jednotka	velikost	atributy
	režim regulace	0 až 2	2	–	–	čtení

- **Popis**

Tento příkaz umožňuje přečtení režimu regulace servopohonu. Odezva je následující:
režim: 0: případ regulace momentu (provádění instrukcí trq a tchg)
1: regulace rychlosti (provádění instrukcí speed a nchg)
2: regulace polohy (provádění jiných instrukcí než výše uvedené)

- **Příklad použití**

Výše uvedenou instrukcí lze sledovat režim regulace, ve kterém se pohon nachází:

<kódové okno>

```
        entry
MAIN:   wait X(11)=1
        speed   N(00) ACC(0) DEC(0)          režim rychlostní regulace
        Yw=MODE
        wait X(11)=0
        stop
        mov P(00) N(00) ACC(0) DEC(0);&      režim polohové regulace
        Yw=MODE
        wait INP=1
        wait X(11)=1
        trq T(00) N(01) N(02)                režim momentové regulace
        Yw=MODE
        wait X(11)=0
        stop
        goto MAIN
        end
```

<datové okno>

P(00)=3276800[pls], P(00)=0[pls], N(00)=3000[min^{-1}], T(00)=5[%]
N(01)=100[min^{-1}], N(02)=100[min^{-1}], ACC(0)=0.1[s], DEC(0)=0.1[s]

N(00) to N(15) N(U(00)) to N(U(15))	název proměnné	rozsah numerické hodnoty	počáteční hodnota	jednotka	velikost	atributy
	Proměnné povelu rychlosti	0 až ±max. rychlosti	0	min^{-1}	1w se znaménkem	čtení zápis

- **Popis**

Tento příkaz se užívá k zadání hodnoty rychlosti v pohybové operaci jako „mov“ nebo „speed“ ev. „speed limit“ za předpokladu, že 1dig=1min^{-1}. Směr vpřed specifikovaný v parametru FA-14 je kladný, směr vzad je záporný. Směr pohybu vpřed nebo vzad však závisí i na pohybovém povelu a znaménko může být nevýznamné. Blíže prostudujte pokyny pro pohybové instrukce.

N(i): Přímá specifikace proměnné povelu rychlosti. může být od 0 do 15.
N(U(j)): Nepřímá N() specifikace hodnotou U(j). Značí N(11) pokud je U(j) = 11.

Pozn.1: Rozsah číselné hodnoty, který lze zapsat je až do 2w se znaménkem. Pokud proměnná povelu rychlosti přesáhne max. rychlost specifikovanou pro pohybovou instrukci, pak je pohyb prováděn dle nastaveného omezení rychlosti.

Pozn.2: Pokud je v pohybové instrukci jako „mov“ nastavena rychlost N()=0 dojde k chybě provádění E45. Je nezbytné, aby povel rychlosti v době provádění pohybové instrukce byl nenulový.

- **Příklad použití 1**

Následující příklad zvyšování rychlosti po 1000min^{-1}

<kódové okno>

```
entry
for U(00) 0 3 1
mov P(00) N(00) ACC(0) DEC(0)
P(00)=P(00)+U(00)
N(00)=N(00)+U(01)                    'zvýšení povelu rychlosti
next
end
```

<datové okno>

P(00)=327680[pls], U(00)=327680[pls], U(01)=1000[min^{-1}] ,
N(00)=1000[min^{-1}], ACC(0)=0.1[s], DEC(0)=0.1[s]

- **Příklad použití 2**

Další příklad ukazuje možnost zvyšování povelu rychlosti nepřímo.

<kódové okno>

```
entry
for U(00) 0 2 1
mov P(00) N(U(00)) ACC(0) DEC(0)
hp  N(U(00)) ACC(0) DEC(0)
next
end
```

<datové okno>

P(00)=327680[pls], , ACC(0)=0.1[s], DEC(0)=0.1[s]
N(00)=1000[min^{-1}], N(01)=2000[min^{-1}], N(02)=3000[min^{-1}]

NFB	název proměnné	rozsah numerické hodnoty	počáteční hodnota	jednotka	velikost	atributy
	okamžitý stav rychlosti	0 to ±32768	0	min^{-1}	1w se znaménkem	čtení

- **Popis**

  Tento příkaz Vám umožní zjistit okamžitý stav skutečné rychlosti. Jednotka rychlosti je 1dig=1min^{-1}. Směr vpřed specifikovaný v parametru FA-14 je kladný, směr vzad je záporný

- **Příklad použití**

  Příkazem NFB je božné identifikovat špatné nastavení rychlosti.

```
<kódové okno>
            entry
            Y(00)=0
            mov P(00) N(00) ACC(0) DEC(0);&
MAIN:       ifs NFB>U(00)
            then
            Y(00)=1                          rychlost dosažena
            else
            Y(00)=0                          rychlost nedosažena
            end if
            if INP=0 then MAIN               test ukončení polohování
            hp N(00) ACC(0) DEC(0)
            end

<datové okno>
    P(00)=3276800[pls], U(00)=1000[min-1], N(00)=3000[min-1],
    ACC(0)=1[s], DEC(0)=1[s]
```

NLM NLM(0), NLM(1)	název proměnné	rozsah numerické hodnoty	počáteční hodnota	jednotka	velikost	atributy
	proměnné omezení rychlosti (Fb-21, Fb-22)	0 to ±32768	max. rychlost	min^{-1}	1w se znaménkem	čtení zápis

- **Popis**

Tento příkaz umožňuje nastavení hodnoty omezení rychlosti (se znaménkem, za předpokladu že 1dig=1min^{-1}).

NLM: společné nastavení omezení rychlosti pro oba směry otáčení (zapisuje se do parametrů Fb-21 a Fb-22)

NLM(0): oddělené nastavení omezení rychlosti ve směru „vpřed“ (zapisuje se do parametru Fb-21)

NLM(1): oddělené nastavení omezení rychlosti ve směru „vzad“ (zapisuje se do parametru Fb-22)

Jsou-li k nastavení omezení rychlosti použity výše uvedené instrukce (NLM, NLM(0) a NLM(1)), pak se parametry Fb-21, Fb-22 přepisují. Rychlostní omezení jsou aktivní v cyklu následujícím po jejich nastavení. Omezení rychlosti se projeví ihned bez ohledu na aktuální stav prováděného pohybu (rozběh / doběh, setrvalá rychlost). Hodnota zapsaná instrukcemi NLM, NLM(0) a NLM(1) jsou brány jako absolutní hodnoty a jsou uloženy do parametrů Fb-21 a Fb-22 (NLM a NLM(0) jako kladná hodnota NLM(1) jako záporná hodnota). Znaménko hodnoty zapsané instrukcí není tedy podstatné.

Pozn.1: Pokud provedete instrukce nastavení omezení rychlost v průběhu pohybových instrukcí jako „speed“ a „mov“, projeví se tyto okamžitě a může dojít k chybě rychlosti (E84), chybě polohy (E83) nebo chybě překročení rychlosti (E85) ev. i jiné.

- **Příklad použití**

Příklad použití omezení rychlosti je uveden níže

<kódové okno>

```
        entry
MAIN:   U(00)=XA(0)
        U(00)=U(00)*N(01)
        N(00)=U(00)/10000
        NLM=N(00)
        speed N(01) ACC(0) DEC(0)
        goto MAIN
        end
```

<datové okno>

N(00)=3000[min^{-1}], N(01)=3000[min^{-1}], ACC(0)=0[s], DEC(0)=0[s]

NRF	název proměnné	rozsah numerické hodnoty	počáteční hodnota	jednotka	velikost	atributy
	okamžitý stav povelu rychlosti	0 to ±32768	0	min^{-1}	1w se znaménkem	čtení

- **Popis**

  Tento příkaz Vám umožní zjistit okamžitý stav povelu rychlosti. Jednotka rychlosti je 1dig=1min^{-1}. Směr vpřed specifikovaný v parametru FA-14 je kladný, směr vzad je záporný Lze přečíst povel rychlosti zadaný v pohybové instrukci „speed“ nebo „mov“.

- **Příklad užití**

  Následující příklad znázorňuje časování uvolnění brzdy dle stavu povelu rychlosti.

```
<kódové okno>
        entry
        Y(00)=0                                 ‘
        mov P(00) N(00) ACC(0) DEC(0);&
MAIN:   ifs NRF>U(00)
        then
        Y(00)=1                                 ‘uvolni brzdu
        else
        Y(00)=0                                 ‘sepnu brzdu
        end if
        if INP=0 then MAIN                      ‘test ukončení polohování
        hp N(00) ACC(0) DEC(0)
        end

<datové okno>
    P(00)=3276800[pls], U(00)=1000[min-1], N(00)=3000[min-1],
    ACC(0)=1[s], DEC(0)=1[s]
```

P(00) to P(99) P(U(00 to P(U(15))	název proměnné	rozsah numerické hodnoty	počáteční hodnota	jednotka	velikost	atributy
	Proměnné povelu polohy	-2^{31} to 2^{31}	0	2^{15} pls/ot.	2w se znaménkem	čtení zápis

- **Popis**

Tento příkaz nastavuje povel polohy stejně jako instrukce "mov". Jednotkou je 1/32768 dig/rev. Směr vpřed specifikovaný v parametru FA-14 je kladný, směr vzad je záporný. Pokud nastavíte hodnotu mimo interval C00000000 až 40000000 (hexadec.) a je zvoleno abs. čidlo polohy (FA-80=Abs) a provádíte instrukci "mov" dojde k přetečení a chybě provádění E45. Počáteční hodnota je nastavena v kódovém okně.

P(i): Přímá specifikace proměnné polohy. Může být i = 0 až 99.

P(U(j)): Nepřímá specifikace proměnné P() pomocí U(j). Např. pokud je U(j)=11, pak je zvoleno P(11).
Pokud není U(j)v intervalu 0 až 99 nebo není specifikováno dojde k chybě provádění E45.

Pozn.1: Chyba provádění nastane i v případě přetečení při provádění operace.

- **Příklad použití 1**

Example of forward constant measure feed

<kódové okno>

```
            entry
MAIN:       U(01)=U(00)
            if P(00)<=0 then SKIP          'Reverse constant measure feed case "<=" → ">"
            U(01)=U(15)-P(00)
            U(01)=U(01)-U(00)
            ifs U(01)>0                    'Reverse constant measure feed case ">" → "<"
            then
            U(01)=U(00)
            else
            P(00)=U(14)+U(00)
            P(00)=P(00)-2                  ''Reverse constant measure feed case" "-2" → "+2"
            U(01)=U(01)*-1
            end if
SKIP:       P(00)=P(00)+U(01)
            mov P(00) N(00) ACC(0) DEC(0)
            goto MAIN
            end
```

<datové okno>

P(00)=3276800[pls], N(00)=3000[min^{-1}], ACC(0)=1[s], DEC(0)=1[s]
U(15)=7FFFFFFF{pls}, U(14)=80000001{pls}
U(00)=65536[pls] Feed quantity
< Reverse constant measure feed case >
U(14)=7FFFFFFF[pls], U(15)=80000001[pls]

- **Příklad použití 2**

  Nepřímá specifikace polohy a provedení polohové operace.

  <kódové okno>

```
            entry
      MAIN: wait X(11)=1                          ‘čekej dokud X(11)=ON
            U(00)=Xw                              ‘přiřaď data z X()
            U(00)=U(00) and 63                    ‘uspořádej data X() od 0 do 63
            mov P(U(00)) N(00) ACC(0) DEC(0)      ‘operace nájezdu na polohu P(U00))
            wait X(11)=0                          ‘čekej dokud X(11)=OFF
            goto MAIN
            end
```

  <datové okno>

  N(00)=3000[min^{-1}], ACC(0)=1[s], DEC(0)=1[s]

  P(00) až P(63)= nutné nadefinovat předem

PBIAS	název proměnné	rozsah numerické hodnoty	počáteční hodnota	jednotka	velikost	atributy
	posun povelu polohy	-128 to 128	0	pls	–	čtení zápis

- **popis**

  Tento příkaz lze uplatnit, pokud je vyžadováno použití posunu polohy v instrukci "sync" (přičtení určeného počtu pulsů každých 140μs. Posun povelu polohy se využívá při nastavování fázové regulace a řízení synchronizace. Je-li nastavená hodnota kladná, přičítají se pulsy ve směru vpřed.

- **Příklad použití**

  dostavení fáze při řízení synchronizace

  <kódové okno>

```
        entry
        U(00)=100              'hrubé nastavení
        U(01)=10               'jemné nastavení
        sync 2                 'instrukce synchronizace
MAIN:   U(15)=Xw
        U(15)=U(15) and 7
        select case U(15)      'X(02) X(01) X(00)
        case 1                 '   0      0      1      hrubé nastavení vpřed (FWD)
        PBIAS=U(00)
        case 2                 '   0      1      0      hrubé nastavení vzad (REV)
        PBIAS=-1*U(00)
        case 5                 '   1      0      1      jemné doladění vpřed (FWD)
        PBIAS=U(01)
        case 6                 '   1      1      0      jemné doladění vzad (REV)
        PBIAS=-1*U(01)
        case else              '   ostatní               žádné nastavení
        PBIAS=0
        end select
        goto MAIN
        end
```

PFILT	název proměnné	rozsah numerické hodnoty	počáteční hodnota	jednotka	velikost	atributy
	Filtr povelu polohy (Fd-36)	0 to 60000	0	ms	1w bez znaménka	čtení zápis

- **Popis**

  Tento příkaz se používá k nastavení časového primárního filtru povelu polohy v polohové regulaci. Jednotkou nastavení je 1dig=1ms. Nastavení hodnoty filtru tímto příkazem je aktivní ihned a je také zapsáno do parametru Fd-36. Poslední zapsaná hodnota bude uchována i pro opětovné zapnutí sítě. Je-li zadána hodnota 0 je filtr vyřazen.

  Pozn.1: Příkaz může mít prodlevu 10 až 100ms (než se nová hodnota uplatní v regulaci).
  Pozn.2: Při přepisu hodnoty PFILT je servo ve stavu zastaveno a zablokováno (servo lock stop state). Zesílení vnitřní regulace se při provedení instrukce zápisu změny PFILT skokově změní. Pokud použijete příkaz přepisu hodnoty PFILT při běhu motoru, může dojít k poškození nebo zničení stroje vlivem "šoku při přepisu" a pod.
  Pozn.3: Zapíšete-li hodnotu mimo dovolený rozsah, může dojít k chybě provádění E45. Změny je možné provádět pouze v dovoleném rozsahu.

- **Příklad použití**

  <kódové okno>

```
entry
PFILT=10                          'nastav 10ms
mov P(00) N(00) ACC(0) DEC(0)
end
```

  <datové okno>

  P(00)=3276800[pls],, N(00)=3000[min^{-1}], ACC(0)=1[s], DEC(0)=1[s]

POS	název proměnné	rozsah numerické hodnoty	počáteční hodnota	jednotka	velikost	atributy
	aktuální poloha (d-08)	-2^{31} to 2^{31}	0	2^{15} pls/ot.	2w se znaménkem	čtení zápis

- **Popis**

  Tímto příkazem lze číst a zapisovat aktuální polohu (d-08). Jednotkou nastavení je 1dig=2^{-15}ot. Směr vpřed specifikovaný v parametru FA-14 je kladný, směr vzad je záporný. Je-li zvolen v parametru FA-80) typ čidla inC (inkrementální), nedojde ani při hodnotách 7FFFFFFF nebo 80000000 (hexadec.) k přetečení (čítač pracuje dále ve smyčce), což má za následek přesun výchozí polohy. Tato výchozí poloha bude platná i při dalším zapnutí sítě.

  Pozn.1: Dojde-li při provádění k přetečení (pouze u abs. čidla) zobrazí se chyba (E45).
  Pozn.2: Okamžitá poloha se příkazem POS přepíše i v případě kdy je v parametru FA-80 zvoleno absolutní čidlo polohy (AbS)

- **Příklad použití 1**

  Signál dosažení určité polohy

  <kódové okno>

```
        entry
        mov     P(00) N(00) ACC(0) DEC(0);&
MAIN:   ifs     POS>U(00)       na adrese MAIN se porovná aktuální poloha (POS) s
        Then                    hodnotou v U(00) a je-li větší pak je aktivován výstup
        Y(00)=1                 Y(00) = 1.
        else
        Y(00)=0
        end if
        if INP=0 then MAIN
        end
```

  <datové okno>

  P(00)=3276800[pls], N(00)=3000[min^{-1}], ACC(0)=1[s], DEC(0)=1[s]
  U(00)=3276800/2[pls]

SCV	název proměnné	rozsah numerické hodnoty	počáteční hodnota	jednotka	velikost	atributy
	nastavení hloubky S-křivky (Fb-30)	0 to 3	0	–	–	čtení zápis

- **Popis**

Tímto příkazem je možná volba křivek rozběhu a doběhu při provádění polohových instrukcí. Použitím S-křivky je možné snížit nebo odstranit vibrace. Počáteční tvar křivek rozběhu a doběhu je lineární (hodnota 0) k vyšším hodnotám se hloubka S-křivky zvyšuje. Hodnota zadaná tímto příkazem je zapsána i do parametru Fb-30 a bude aktivní i při dalším zapnutí sítě. Následující tabulka přiřazuje hodnotám parametru Fb-30 numerický tvar:

nastavení Fb-30	hodnota SCV
non	0
SHArP	1
rEgLAr	2
LooSE	3

Pozn.1: Změna stavu SCV se provádí ve stavu zastavení a zablokování serva. I když bude přepsána hodnota SCV v průběhu provádění pohybové instrukce, nastavení bude aktivní až při další pohybové instrukci.

Pozn.2: Zapíšete-li hodnotu mimo dovolený rozsah, může nastat chyba provádění (E45). Je nutné používat pouze dovolených hodnot.

- **Příklad použití**

<kódové okno>

```
entry
SCV=1
mov P(00) N(00) ACC(0) DEC(0)
end
```

<datové okno>

P(00)=3276800[pls], N(00)=3000[min^{-1}], ACC(0)=1[s], DEC(0)=1[s]

SFILT	název proměnné	rozsah numerické hodnoty	počáteční hodnota	jednotka	velikost	atributy
	filtr povelu rychlosti (Fd-20)	0 to 60000	0	ms	1w se znaménkem	čtení zápis

- **Popis**

  Tento příkaz umožňuje nastavení hodnoty primárního filtru povelu momentu. Jednotkou nastavení je 1dig=1ms. Zapsání číselné hodnoty je účinné okamžitě a je uloženo do parametru Fd-36. Změněná hodnota filtru bude účinná i při dalším zapnutí sítě. Pokud zapíšete hodnotu filtru 0, je filtr vyřazen.

  Pozn.1: Příkaz může mít prodlevu 10 až 100ms (než se nová hodnota uplatní v regulaci).
  Pozn.2: Při přepisu hodnoty SFILT je servo ve stavu zastaveno a zablokováno (servo lock stop state). Zesílení vnitřní regulace se při provedení instrukce zápisu změny SFILT skokově změní. Pokud použijete příkaz přepisu hodnoty SFILT při běhu motoru, může dojít k poškození nebo zničení stroje vlivem "šoku při přepisu" a pod.
  Pozn.3: Zapíšete-li hodnotu mimo dovolený rozsah, může dojít k chybě provádění E45. Změny je možné provádět pouze v dovoleném rozsahu.

- **Příklad použití**

  <kódové okno>

```
entry
SFILT=10                          'Sets 10ms
mov P(00) N(00) ACC(0) DEC(0)
end
```

  <datové okno>

  P(00)=3276800[pls],, N(00)=3000[min^{-1}], ACC(0)=1[s], DEC(0)=1[s]

SZD	název proměnné	rozsah numerické hodnoty	počáteční hodnota	jednotka	velikost	atributy
	detekce nulové hodnoty	0, 1	–	–	0, 1	čtení

- **Popis**

Tento příkaz umožňuje rozhodnout, zda okamžitá rychlost je nižší než hodnota rychlosti považovaná za nulovou (parametr Fb-22). SZD nabývá hodnot 0 nebo 1 viz následující:

SZD=0: v případě, že okamžitá rychlost je vyšší než hodnota považovaná za nulovou (Fb-22)

1: v případě že okamžitá rychlost je nižší než hodnota považovaná za nulovou (Fb-22)

- **Příklad použití**

<kódové okno>

```
        entry
        mov     P(00) N(00) ACC(0) DEC(0); &        'paralelní provádění
MAIN:   Y(00)=SZD
        if INP=0 then goto MAIN
        end
```

<datové okno>

P(00)=327800[pls], N(00)=100[min^{-1}], N(01)=3000[min^{-1}]

ACC(0)=0.1[s], DEC(0)=0.1[s]

T(00) až T(15)	název proměnné	rozsah numerické hodnoty	počáteční hodnota	jednotka	velikost	atributy
	proměnná povelu momentu	-32768 až 32767	0	%	1w se znaménkem	čtení zápis

- **Popis**

  Tento příkaz umožňuje zadání hodnoty povelu momentu pro pohybové instrukce jako trq nebo tchg. Jednotkou zadání je 1dig=1min^{-1}. Směr vpřed specifikovaný v parametru FA-14 je kladný, směr vzad je záporný.

  Pozn.1: K chybě provádění E45 dojde, pokud nastavíte hodnotu povelu momentu mimo hranici maximálního momentu a použijete jej pro pohybovou operaci. Rozsah numerické hodnoty může být až 2w se znaménkem, pokud použijete pro specifikaci povelu momentu proměnnou.

- **Příklad užití**

  <kódové okno>

```
entry
T(00)=50                ‘50% momentu
trq T(00) N(00) N(01)
wait 10
end
```

  <datové okno>

  N(00)=1000[min^{-1}] , N(01)=1000[min^{-1}]

TFB	název proměnné	rozsah numerické hodnoty	počáteční hodnota	jednotka	velikost	atributy
	výstup momentu	-400 až 400	0	1/50 %	1w se znaménkem	čtení

- **Popis**

  Tento příkaz umožňuje přečtení hodnoty výstupního momentu. Jednotka je 1dig=1/50%. Směr vpřed specifikovaný v parametru FA-14 je kladný, směr vzad je záporný.

- **Příklad použití**

  Následující příklad představuje užití příkazu TFB k ovládání brzdy.

```
<kódové okno>
            entry
            Y(00)=0                          ‘
            mov P(00) N(00) ACC(0) DEC(0);&
BRKOFF:     U(01)= abs TFB
            ifs U(01)<U(00) then BRKOFF
            Y(00)=1                          ‘uvolnění brzdy
BRKON:      Y(00)=SZD
            if INP=1 then BRKON              ‘vyčkej ukončení polohování
            Y(00)=0                          ‘sepni brzdu
            end if
            end

<datové okno>
    P(00)=3276800[pls], U(00)=50[%], N(00)=3000[min-1]
    ACC(0)=1[s], DEC(0)=1[s]
```

TFILT	název proměnné	rozsah numerické hodnoty	počáteční hodnota	jednotka	velikost	atributy
	filtr povelu momentu (Fd-06)	0 až 50000	0	0.01 ms	1w bez znaménka	čtení zápis

- **Popis**

  Tento příkaz se užívá k zadání hodnoty časové konstanty filtru povelu momentu do regulátoru proudu. Jednotkou je 1dig=0.01ms. Hodnota zadaná touto instrukcí je ihned aktivní a je také uložena do parametru Fd-06. proto bude tato hodnota aktivní i při další zapnutí sítě (nedojde k návratu k původní hodnotě). pokud zadáte hodnotu 0 filtr bude vyřazen.

  Pozn.1: Příkaz může mít prodlevu 10 až 100ms (než se nová hodnota uplatní v regulaci).
  Pozn.2: Při přepisu hodnoty TFIL je servo ve stavu zastaveno a zablokováno (servo lock stop state). Zesílení vnitřní regulace se při provedení instrukce zápisu změny TFIL skokově změní. Pokud použijete příkaz přepisu hodnoty TFILT při běhu motoru, může dojít k poškození nebo zničení stroje vlivem "šoku při přepisu" a pod.
  Pozn.3: Zapíšete-li hodnotu mimo dovolený rozsah, může dojít k chybě provádění E45. Změny je možné provádět pouze v dovoleném rozsahu.

- **Příklad použití**

  <kódové okno>

```
entry
TFILT=1000                          ‘nastavení 10ms
mov P(00) N(00) ACC(0) DEC(0)
end
```

  <datové okno>

  P(00)=3276800[pls],, N(00)=3000[min^{-1}], ACC(0)=1[s], DEC(0)=1[s]

TLM TLM(0) až TLM(3)	název proměnné	rozsah numerické hodnoty	počáteční hodnota	jednotka	velikost	atributy
	proměnná omezení momentu (Fd-07 až Fb-10)	0 to max. moment × 50	15000	1/50%	1w bez znaménka	čtení zápis

- **Popis**

Tento příkaz umožňuje nastavení omezení momentu číselnou hodnotou bez znaménka s jednotkou 1dig=1/50%. Vysvětlení TLM až TLM(3) je uvedenou dále.

TLM: hodnota značí omezení momentu pro všechny čtyři kvadranty (par. Fb-07 až Fb-10)
TLM(0): individuální nastavení omezení kladného momentu směru vpřed (parametr Fb-07)
TLM(1): individuální nastavení omezení kladného momentu směru vzad (parametr Fb-08)
TLM(2): individuální nastavení omezení záporného momentu směru vzad (parametr Fb-09)
TLM(3): individuální nastavení omezení záporného momentu směru vpřed (parametr Fb-10)

Zápisem příkazů TLM až TLM(3) jsou jejich hodnoty přeneseny i do parametrů zmíněných výše. Omezení momentu se uplatní okamžitě v dalším prováděcím cyklu po zapsání. Momentové omezení se v pohybové operaci uplatní bez ohledu na rozběhové nebo doběhové časy, nebo omezení rychlosti. Hodnoty zapsané v NLM až NLM(3) jsou brány jako absolutní hodnoty a tak jsou i zapsány do parametrů Fb-07 až Fb-10. Znaménko hodnoty zapsané instrukcí není tedy podstatné.

Pozn1: Pokud provedete instrukce nastavení omezení momentu v průběhu pohybových operací jako „speed“ a „mov“, projeví se tyto okamžitě a může dojít k chybě rychlosti (E84), chybě polohy (E83) nebo chybě překročení rychlosti (E85) ev i jiné.

- **Příklad použití**

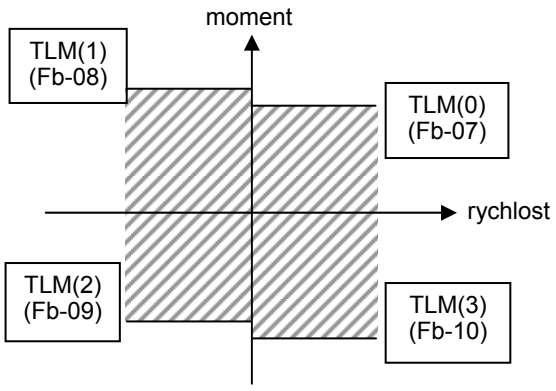
Příklad použití je uveden níže

<kódové okno>

```
        entry
MAIN:   U(00)=XA(0)
        U(00)=U(00)*T(01)
        T(00)=U(00)/10000
        TLM=T(00)
        speed N(01) ACC(0) DEC(0)
        goto MAIN
        end
```

<datové okno>

```
T(01)=15000
```



TRF	název proměnné	rozsah numerické hodnoty	počáteční hodnota	jednotka	velikost	atributy
	okamžitý povel momentu	-400 až 400	0	1/50%	1w se znaménkem	čtení

- **Popis**

  Tento příkaz může zjistit hodnotu povelu monetu v průběhu provádění. Jednotkou je 1dig=1/50%. Směr vpřed specifikovaný v parametru FA-14 je kladný, směr vzad je záporný. Příkaz TRF vám dovolí zjistit stav povelu momentu před uplatněním momentového omezení a filtru povelu momentu.

- **Příklad použití**

  Následující příklad vysvětluje použití TRF pro rozhodnutí o užití brzdy

  <kódové okno>

```
            entry
            Y(00)=0                                 ‘
            mov P(00) N(00) ACC(0) DEC(0);&
BRKOFF:     U(01)= abs TRF
            ifs U(01)<U(00) then BRKOFF
            Y(00)=1                                 ‘uvolni brzdu
BRKON:      Y(00)=SZD
            if INP=1 then BRKON                     ‘vyčkej dokončení polohování
            Y(00)=0                                 ‘sepni brzdu
            end if
            end
```

  <datové okno>

  P(00)=3276800[pls], U(00)=50[%], N(00)=3000[min^{-1}]
  N(00)=3000[min^{-1}], N(01)=3000[min^{-1}], ACC(0)=0[s], DEC(0)=0[s]

U(00) až U(15)	název proměnné	rozsah numerické hodnoty	počáteční hodnota	jednotka	velikost	atributy
	uživatelské proměnné	-2^{31} to 2^{31}	0	–	2w se znaménkem	čtení zápis

- **Popis**

  Tento příkaz umožňuje definovat uživatelské proměnné pro obecné použití. Toto proměnnou můžete použít např. jako dočasnou paměť výsledku aritmetické operace. Bez ohledu na její formu jako poloha nebo rychlost lze ji využít jako proměnnou velikosti 2 slov se znaménkem.
  Pozn.1: Pokud při aritmetické operaci dojde k přetečení nastane chyba provádění E45.

- **Příklad použití**

  Polohování v kruhových souřadnicích specifikovaných tabulkou
  <kódové okno>

```
        entry
        chgORL=X(10)
        U(14)=U(15)/2                   'nastavení poloviny otáčky
        P(00)=0
        ort 0 N(00)  ACC(0) DEC(0)      'dosažení výchozí polohy
MAIN:   wait X(11)=1
        wait X(11)=0
        U(00)=Xw                        'specifikace tabulky hodnotu po hodnotě
        U(00)=U(00) and 511
        ifs U(00)>=U(13)                'omezení na 360^0 a více
        then
        U(00)=U(13)
        end if
        U(01)=U(15)*U(00)
        U(01)=U(01)/360                 'nastavení polohy jedné otáčky
        U(02)=P(00) mod U(15)           'Vytvoří příkaz polohu v poslední otáčce
        ifs U(01)=U(02) then MAIN
        U(01)=U(01)-U(02)               'výpočet množství pohybů
        U(02)= abs U(01)
        P(00)=P(00)+U(01)
        ifs U(02)>U(14)                 'Selects a shortcut
        then
        ifs U(01)>0                     'určení směru otáčení
        then
        P(00)=P(00)-U(15)               'návrat o jednu otáčku
        else
        P(00)=P(00)+U(15)               'posun o jednu otáčku
        end if
        end if
        mov P(00) N(00) ACC(0) DEC(0)
        goto MAIN
        end
```

  <datové okno>

  U(15)=3276800[pls] … počet pulsů čidla na jednu otáčku
  N(00)=3000[min^{-1}], ACC(0)=0.1[s], DEC(0)=0.1[s]

KAPITOLA 7 NESNÁZE, CHYBY A JEJICH PŘÍČINY

Tato kapitola popisuje obsah ochran a programování vhodných zásahů při zobrazení chyby.

7.1 Seznam ochran v programovém provozu 7-2
7.2 Vhodný zásah pří chybě 7-3
7.3 Obsah chyb a zásahy při sestavování programu 7-5

7.1 Seznam ochran v programovém provozu

Následující seznam obsahuje chyby příznačné pro programový provoz.
Popis dalších chyb najdete v uživatelské příručce k servopohonu

p.č.	název funkce	chybové zobrazení	obsah chyby
1	nedovolená chyba instrukce	E43	E43 se zobrazí, pokud je použit kód, který nekoresponduje s instrukcí načítaného programu
2	chyba vnoření	E44	E44 je zobrazena, pokud hloubka vnoření podprogramů překročí úroveň 8
3	chyba provádění	E45	• E45 se zobrazí, pokud cyklus podprogramu nemá začátek jako např. "for" pro instrukci "goto" apod. • E45 se zobrazí, pokud není k dispozici žádná aplikovatelná proměnná při nepřímém určení jako “P(U(xx)) apod. • E45 se zobrazí, pokud je v první pohybové instrukci po spuštění použit tovární formát • E45 se zobrazí, pokud je v pohybové instrukci povel rychlosti 0. • E45 se zobrazí, pokud je v polohové instrukci poloha specifikována nepřímo P(Xn) a souřadnice X(00) až X(11) jsou 0. • E45 se zobrazí je-li povel polohy v polohové instrukci "smov" roven 0 • E45 se zobrazí pokud je omezení rychlosti v momentové instrukci "trq" a "tchg" roven 0 • E45 se zobrazí, pokud dojde při aritmetické operaci k přetečení, nebo podtečení, nebo nastane dělení nulou • E45 se zobrazí, pokud zapisovaná hodnota je mimo dovolený rozsah

7.2 Vhodné akce při chybě

Následující část popisuje vhodný postup při chybě vzniklé v souvislosti s prováděním programu.

Vysvětlení dalších chybových hlášení naleznete v uživatelské příručce servopohonu AD.

číslo chyby	název chyby	příčina	co je potřeba prověřit	doporučený postup
E43	nedovolená chyba instrukce	• servo spuštěno bez načtení programu • program v paměti podprogramu je zničen	Prověřte, zda program přečtený ze servopohonu odpovídá originálnímu vytvořenému programu	Vytvořte znovu program a zapište jej znovu do servopohonu
E44	chyba vnoření	• úroveň vnoření pod-programu překročila 8 • úroveň vnoření v příkaze "for-next" překročila 8 • úroveň vnoření příkazu "if" překročila 8	Prověřte vizuálně úroveň vnoření	pozměňte program tak, aby nedocházelo k překročení dovolené úrovně vnoření 8
E45	chyba provádění	• Žádná použitelná proměnná	• prověřte numerickou proměnnou U(xx).	• omezte rozsah proměnné U(xx) tak aby nepřekročila dovolené hodnoty U(xx), nebo změňte hodnoty přiřazené proměnné U(xx).
		• instrukce "mov" nebo "nchg" má nastaven povel rychlosti 0	• prověřte číselnou hodnotu proměnné N(xx).	• v instrukcích "mov" a "nchg" nemůže být povel rychlosti roven 0
		• V první pohybové instrukci od počátku je použit tovární formát	• prověřte, zda je v prvé pohybové instrukci použit původní formát	• Je potřeba změnit program tak aby byl použit jiný formát
		• Příkaz polohy pro instrukci "mov" je zadán jako P(Xn) a hodnoty X(00) až X(11) jsou 0.	• prověřte stav proměnných X(00) až X(11).	• Je-li povel polohy zadán proměnnou P(Xn), pak je poloha specifikována hodnotou vstupů X(00) - X(11).
		• Povel poloha v instrukci "smov" je 0	• prověřte povel polohy	• program je nastaven tak, že hodnota povelu polohy nemůže být 0
		• při provádění instrukci "trq" nebo "tchq" je nastaveno omezení rychlosti 0	• prověřte nastavení omezení rychlosti	• program je nastaven tak, že omezení rychlosti nemůže být 0
		• při aritmetické operaci došlo k podtečení nebo přetečení, nebo je prováděno dělení nulou	• prověřte, zda při některé operaci může dojít k přetečení, podtečení nebo dělení nulou	• program je nastaven tak, že k přetečení, podtečení, nebo dělení 0 nesmí dojít
		• zapsaná hodnota proměnné je mimo dovolený rozsah	• prověřte zapisovanou proměnnou	• program je nastaven tak, že zapisované hodnoty musí ležet v předepsaných rozsazích.

7.3 Obsah chyb a zásahy při sestavování programu

V následující tabulce jsou uvedeny chyby editace, jejich označení a řešení

číslo chyby	hlášení	obsah (řešení)
1	” No entry at the head of Code Window!!” (žádný počátek v hlavě kódového okna)	v hlavě kódového okna chybí příkaz počátek- entry
2	No end of subroutine exits!!” (neexistuje konec podprogramu)	zdvojení počátku
		počátek podprogramu bez ukončení
		místo end je použit End sub
		není zapsán konec podprogramu
3	”No specification of subroutine name!!” (není zadán název podprogramu)	není zadán název podprogramu
4	”Subroutine name is duplicated!!” (název podprogramu je již použit)	zvolený název podprogramu již existuje
5	”No end sub of subroutine exits!!” (neexistuje konec podprogramu)	není použit end sub.
		‘počátek podprogramu je zdvojený
		podprogram začíná ale nemá konec
		místo end sub je zadán pouze end
6	”An instruction has already exited before the start line of subroutine!!”(instrukce je dříve než počátek podprogramu)	instrukce jiná než “sub” je uvedena za “end”/”end sub”.
7	”Label name is duplicated!!” (zdvojená značka)	jméno značky je použito vícekrát
15	"Parameter of motion instruction is error!!” (chyba parametru polohové instrukce)	chybné zadání paralelního provádění polohových instrukcí
	“Parameter for the position command is error!!” (parametr povelu polohy je chybný)	parametr povelu polohy je chybný
	“Parameter for a speed command is error!!” (parametr povelu rychlosti je chybný)	parametr povelu rychlosti je chybný
	“Parameter for the acceleration time command is error!!”(parametr zadání rozběhu je chybný)	parametr zadání rozběhu je chybný
	“Parameter for the declaration time command is error!!” (parametr zadání doběhu je chybný)	parametr zadání doběhu je chybný
	“Parameter for the torque command is error!!” (parametr povelu momentu je chybný)	parametr povelu momentu je chybný
103	”No associated subroutine name!!” (podprogram udaného názvu neexistuje)	Je zadán název podprogramu, který neexistuje
105	”No associated label name!!” (značka udaného názvu neexistuje)	v instrukci "goto" je zadána značka, která neexistuje

číslo chyby	hlášení	obsah (řešení)
107	”An instruction exists in the position which cannot be compiled!!” (instrukce je v pozici ve které nemůže být sestavena)	pneumonické pole v pravém sloupci instrukce “if(s)”/”then”/”else”/”end if” není nulové
	“Comparison sign is error!!” (chyba porovnání znaménka)	porovnání znaménka ve strukturované instrukci „if“je chybné
	“Comparison data is error!!” (porovnávaná data jsou chybná)	porovnávaná proměnná strukturované instr. if jsou chybná
		porovnávané reálné číslo strukturované instrukce if je mimo rozsah
	“Comparison instruction is error!!” (porovnávaná instrukce je chybná)	instrukce jiná než “then”/”else” je umístěna za instrukcí “if(s)” ve struktuře instrukcí
		instrukce “then”/”else”/”end if” je umístěna za řádkem “then” ve struktuře instrukcí
		instrukce “then”/”else”/”end if” je umístěna za řádkem “else” ve struktuře instrukcí
		instrukce “then” je umístěna před “else”/”end if” za řádkem s instrukcí “then” ve struktuře instrukcí
		instrukce “then”/”else”/”end if” je umístěna za řádkem s instrukcí “else” ve struktuře instrukcí
		instrukce “then”/”else” je umístěna před instrukcí ”end if” za řádkem “else” ve struktuře instrukcí
	“Corresponding instruction!!” (odpovídající instrukce)	instrukce “if(s)” spojená s instrukcí “else” neexistuje
108	“Comparison sign is error!!” (chyba porovnání znaménka)	porovnání znaménka v instrukci „if“ je chybné
	”No associated label name!!” (neexistuje odpovídající značka)	název značky přináležející k instrukci neexistuje
	“Comparison data is error!!” (porovnávaná data jsou chybná)	porovnávaná proměnná instrukce „if“ je chybná
		porovnávané reálné číslo strukturované instrukce if je mimo rozsah
	”No associated label name!!” (neexistuje odpovídající značka)	prověření existence značky ve struktuře „if“
110	”An instruction exists in the position which cannot be compiled!!” (instrukce je v pozici ve které nemůže být sestavena)	pneumonické pole v pravém sloupci instrukcí “for”/”next” není nulové
	“Comparison sign is error!!” (chyba porovnání znaménka)	porovnání znaménka v instrukci „for“ je chybné
	“Start variable is error!!” (počáteční proměnná je chybná)	počáteční proměnná instrukce „for“ je chybná
	“Start value is error!!” (počáteční hodnota je chybná)	počáteční hodnota instrukce „for“ je chybná
	“The end value is error!!” (koneční hodnota je chybná)	koncová hodnota instrukce „for“ je chybná
	“Step value is error!!” (hodnota kroku je chybná)	hodnota kroku ve struktuře „for“ je chybná
	“The for loop instruction is error!!” (instrukce smyčky je chybná)	instrukce “next” je umístěna na řádku za instrukcí “for”
	“No next instruction corresponding with for!!” (není instrukce „next“ k instrukci „for“)	instrukce “next” přináležející k instrukci “for” neexistuje
	“No for instruction corresponding with next!!” (není instrukce „for“ k instrukci „next“)	instrukce “for” přináležející k instrukci “next” neexistuje

..

číslo chyby	hlášení	obsah (řešení)
112	”An instruction exists in the position which cannot be compiled” (instrukce je v pozici ve které nemůže být sestavena)	pneumonické pole v pravém sloupci instrukce “while”/”wend” je nulové
	“Comparison sign is error” (chyba porovnání znaménka)	porovnání znaménka v instrukci „while“ je chybné
	“Comparison data is error” (porovnávaná data jsou chybná)	porovnávaná proměnná v instrukci „while“ je chybná
		reálná číslo v instrukci „while“ je chybné
	“The while loop instruction is error” (smyčka „while“ je chybná)	instrukce “wend” je umístěna na řádku za instrukcí “while”
	“No wend corresponding with while” (není instrukce „wend“ k instrukci „while)	instrukce “wend” náležející k instrukci “while” neexistuje
	“No while instruction corresponding with next” (není instrukce „while“ k instrukci „next“	instrukce “while” náležející k instrukci “wend” neexistuje
113	”An instruction exists in the position which cannot be compiled!!” (instrukce je v pozici ve které nemůže být sestavena)	Pneumonické pole v pravém sloupci instrukcí ”until”/”loop” není nulové
	“Comparison sign is error!!” (chyba porovnání znaménka)	Porovnání znaménka v instrukci "until" je chybné
	“Comparison data is error!!” (porovnávaná data jsou chybná)	porovnávaná proměnná v instrukci „until“ je chybná
		porovnávané reálná číslo v instrukci „until“ je chybné
	“The until loop instruction is error!!” smyčka „until“ je chybná)	instrukce “loop” je umístěna pod instrukci “until”
	“No loop instruction corresponding with until!!” (není instrukce „loop“ k instrukci „until“)	k instrukci “until” neexistuje příslušná instrukce “loop”
	“No until instruction corresponding with loop!!” (není instrukce „until“ k instrukci „loopl“)	k instrukci "loop" neexistuje příslušná instrukce "until"
114	”An instruction exists in the position which cannot be compiled” (instrukce je v pozici ve které nemůže být sestavena)	Pneumonické pole v pravém sloupci instrukcí ”select case”/”case”/”case else”/”end select” není nulové
	“Comparison data is error!!” (porovnávaná data jsou chybná)	porovnávání proměnné v instrukci "select case" je chybné
		porovnávání proměnné v instrukci "case" je chybné
	“Select case instruction is error!!” (zvolená instrukce "case" je chybná)	v strukturované instrukci "if" je umístěna pod řádkem "select case" jiná instrukce než "case"
		v strukturované instrukci "if" je umístěna pod řádkem "case" některá z instrukcí “case”/”case else”/”end select”
		v strukturované instrukci "if" je umístěna pod řádkem "case else" některá z instrukcí “case”/”case else”/”end select
		instrukce “case”/”case else” jsou umístěny v řádku pod instrukcí “case else”
	“No end select instruction corresponding with the select case!!” (žádná instrukce odpoví-dající instrukci "select case")	Neexistuje instrukce "end select” příslušející k instrukci “select case”
115	”No corresponding instruction!!” (žádná odpovídající instrukce)	instrukce “select case” přináležející k instrukci “case” neexistuje
		instrukce “select case” přináležející k instrukci “case else” neexistuje
		instrukce “select case” přináležející k instrukci “end select” neexistuje
116	”No associated subroutine name!!” (žádný přiřazený podprogram)	podprogram odpovídající volanému jménu v instrukci "go sub" neexistuje
117	“Variable is error!!” (chyba proměnné)	proměnná v instrukci "inc" je chybná
118	“Variable is error!!” (chyba proměnné)	proměnná v instrukci "dec" je chybná

číslo chyby	hlášení	obsah (řešení)
123	”The wait time is error!!” (čas prodlevy je chybný)	čas zadaný v instrukci "wait" je chybný
	“Comparison variable is error!!” (porovnávaná proměnná je chybná)	“porovnávaná proměnná v instrukci "wait" je chybná
		“porovnávané reálné číslo v instrukci "wait" je chybné
	“Comparison sign is error!!” (chyba porovnání znaménka)	“porovnání znaménka v instrukci "wait" je chybné
208	”Parameter is error!!” (chyba parametru)	specifikace nahoru/dolů v instrukci "smov je chybná
211	“Parameter is error!!” (chyba parametru)	specifikace posunu rychlosti v instrukci "sync" je je chybná
213	“Parameter is error!!” (chyba parametru)	specifikace režim návratu v instrukci "ort" je chybná
303	”The operator is error!!” (chyba operátoru)	v nastavení kontaktního výstupu je použit jiný operátor než “=”
	“The numerical number is set to the parameter!!” (je nastavena číselná hodnota)	kontaktnímu výstupu je přiřazována číselná hodnota
	“Parameter is error!!” (chyba parametru)	přiřazení proměnné kontaktnímu výstupu je chybné
301	”Variable is error!!” (chyba proměnné)	přiřazení proměnné kontaktnímu vstupu je chybné
	“Operator is error!!” (chyba operátoru)	v nastavení kontaktního vstupu je použit jiný operátor než “=”
305	“Operator is error!!” (chyba operátoru)	v speciálním přiřazení svorek je použit jiný operátor než “=”
	“Parameter is error!!” (chyba parametru)	přiřazení proměnné ke speciálnímu přiřazení svorek je chybné
400	“Variable is error!!” (chyba proměnné)	aritmetická proměnná je chybná
	“Operator is error!!” (chyba operátoru)	operátor je chybný
	“The numerical number is set to the pneumonic!!” (číslo vřazeno do pneumonického řetězce)	v pneumonickém řetězci se vyskytuje reálné číslo
	“The instruction is error!!” (chybná instrukce)	nesprávný znakový řetězec v pneumonickém řetězci
401	”The numerical number is set to the pneumonic string!!” (číslo vřazeno do pneumonického řetězce)	v pneumonickém řetězci se vyskytuje číslo
402	”The instruction is error!!” (chybná instrukce)	v pneumonickém řetězci se vyskytují data, které nejsou ani instrukci ani proměnnou
		V pneumonickém řetězci se vyskytuje parametr bez instrukce
2048	”The level of nesting is too deep” (překročena úroveň vnoření)	úroveň vnoření překročila povolenou hodnotu
2049	”The total step number of intermediate code exceeds 512 lines” (celkový počet kroků překročil 512 řádek)	pořadové číslo kroku překročilo 512 řádek
2050	”Compiler detects the error” (kompilátor našel chybu)	detekce chyby v procesu sestavování programu

Redakční poznámka: Tato publikace je překladem uživatelské příručky firmy HITACHI

č.ADPr01601BX vydané v r 2003

Publikace neprošla obsahovou ani jazykovou úpravou.

AEF, s.r.o
distributor průmyslové elektroniky HITACHI
Pekařská 86
602 00 BRNO
ČESKÁ REPUBLIKA